POLITECNICO DI TORINO

Master's Degree in
Computer Engineering

Master Thesis

Early Detection of Emotional Issues in High School Students Through Statistical Analysis of Academic Data

Supervisors
prof. Stefano Quer
prof. Ugo Buy, University of Illinois Chicago
prof. Abolfazl Asudeh, University of Illinois Chicago

Candidate
Davide Porello

Academic Year 2022-2023

ACKNOWLEDGMENTS

First, I would like to thank my family and my girlfriend who always supported me during my stay in Chicago. Second, I want to thank my advisor Ugo Buy for his continuous support. Moreover, I thank also Simone, Emilio and Thiru who worked with me on the OSF/CHA Project for allowing me to discuss my ideas with them, which helped me to develop critical thinking. A particular mention is also for my roommates Giulio, Vincenzo, Giuseppe and Simone with whom I have had constructive and pleasant discussions that let me broaden my knowledge. Finally, I want also to thank all the people that I met during my US experience thanks to which I grew as a person.

DP

TABLE OF CONTENTS

CHAPTER			PAGE
1	**INTRODUCTION**		1
	1.1	Motivation	4
2	**BACKGROUND**		5
	2.1	Time series	5
	2.2	Time Series Clustering	6
	2.3	Time Series Representation	8
	2.3.1	Interpolation	8
	2.3.2	Principal Component Analysis	11
	2.4	Distance Metrics	14
	2.4.1	Euclidean Distance	14
	2.4.2	Dynamic Time Warping	15
	2.5	Clustering Prototypes	17
	2.5.1	DTW Barycenter Averaging	17
	2.6	Clustering Algorithms	19
	2.6.1	Partitional Clustering	19
	2.6.1.1	K-means	19
	2.6.1.2	K-medoids	21
	2.6.2	Hierarchical Clustering	22
	2.6.3	Shape-based Clustering	24
	2.6.4	Density-based Clustering	25
	2.6.5	Spectral Clustering	27
	2.7	Evaluation Metrics	29
	2.7.1	Silhouette Index	29
	2.8	Hyperparameter Tuning	31
	2.8.1	Bayesian tuning	32
3	**RELATED WORK**		34
	3.1	Dynamic Time Warping	35
	3.2	Constrained clustering	37
4	**DATA PREPARATION**		39
	4.1	Data Creation	39
	4.1.1	Raw Data	39
	4.1.2	Time Series with the Same Length	41
	4.1.2.1	Old Data Creation Process	41
	4.1.2.2	New Features	42

TABLE OF CONTENTS (continued)

CHAPTER **PAGE**

4.1.2.3 New Data Creation Process . . . 44
4.1.2.4 Interpolation Evaluation . . . 49
4.1.3 Time Series with Different Lengths . . . 54
4.2 Data Preprocessing . . . 56
4.2.1 Data Smoothing . . . 56
4.2.2 Delta Averaging . . . 58

5 METHODOLOGY . . . 60
5.1 Clustering of Time Series with Different Lengths . . . 61
5.1.1 K-means with DTW . . . 62
5.1.2 Classify Clusters . . . 64
5.2 Classical Clustering Algorithms . . . 67
5.2.1 Partitional Clustering . . . 68
5.2.2 Hierarchical Clustering . . . 70
5.2.3 Shape-based Clustering . . . 71
5.2.4 Density-based Clustering . . . 73
5.2.5 Spectral Clustering . . . 74
5.2.6 Classify Clusters . . . 75
5.3 Density-based Clustering with Feature Reduction . . . 77
5.3.1 Principal Component Analysis . . . 78
5.3.2 Density-based Clustering for Outlier Detection . . . 80
5.3.3 Bayesian Tuning . . . 82
5.4 Evaluation Metrics . . . 86

6 RESULTS . . . 91
6.1 Data preprocessing evaluation . . . 93
6.2 Baseline . . . 96
6.3 Final model . . . 109
6.3.1 Fixed Number of Time Series and Expected Dimensions . . . 110
6.3.2 Different Number of Time Series . . . 115
6.3.3 Different Expected Dimensions . . . 119

7 CONCLUSION . . . 124
7.1 Contribution . . . 126
7.2 Future Work . . . 126

CITED LITERATURE . . . 128

VITA . . . 132

LIST OF TABLES

TABLE		PAGE
I	RAW DATA	40
II	DATAFRAME WITH FINAL SCORES	45
III	DATAFRAME WITH TIME SERIES BEFORE INTERPOLATION	47
IV	PARTITIONAL CLUSTERING EXPERIMENTS	69
V	HIERARCHICAL CLUSTERING EXPERIMENTS	71
VI	SPECTRAL CLUSTERING EXPERIMENTS	74
VII	DATA PREPROCESSING EVALUATION	93
VIII	RESULTS OF HIERARCHICAL CLUSTERING	99
IX	RESULTS OF SPECTRAL CLUSTERING	101

LIST OF FIGURES

FIGURE		PAGE
1	Example of “stable” time series.	2
2	Example of “monitor” time series.	2
3	Example of “critical” time series.	3
4	Time series with missing values.	8
5	Time series with linear interpolation.	8
6	Example of linear interpolation	9
7	Principal Component Analysis.	11
8	How Euclidean distance compares time series.	14
9	Euclidean distance one-to-one matching.	14
10	How DTW compares time series.	16
11	DTW warping path.	16
12	Representative time series calculated with DBA vs mean.	18
13	K-means vs K-medoids.	21
14	Dendrogram generated by Hierarchical Agglomerative clustering.	22
15	Core point identified by DBSCAN.	26
16	Bayesian tuning process.	32
17	Number of grades per week.	42
18	Number of students with at least one grade per week.	42
19	Distribution of grades per topic.	44
20	Number of missing values per student.	48
21	Number of missing values per week.	48
22	Interpolation evaluation process.	50
23	Number of missing values per week.	51
24	Time series with missing values.	54
25	Time series with grades shifted to the left.	54
26	Unprocessed time series.	57
27	Time series after moving average with W = 3.	57
28	Unprocessed time series.	57
29	Time series after exponential moving average with $\alpha = 0.5$.	57
30	Unprocessed time series.	59
31	Time series after delta averaging.	59
32	Main workflow.	61
33	DTW with time series of different lengths.	63
34	DBA representative for time series with different lengths.	65
35	Unprocessed time series.	72
36	Normalized time series.	72
37	DBA representative for time series with the same length.	75
38	Density-based Clustering with Feature Reduction.	77

LIST OF FIGURES (continued)

FIGURE **PAGE**

39 Evaluation of data preprocessing with PCA. 79
40 2D representation of the time series. 81
41 Clustering with 10% outliers. 84
42 Clustering with 30% outliers. 84
43 Final clustering in two dimensions. 84
44 Time series with big fluctuations and steep drops. 89
45 Time series with descending trend. 89
46 Table VII Row 2 clustering. 95
47 Table VII Row 4 clustering. 95
48 K-means clustering with time series of different lengths. 97
49 K-means Euclidean clustering. 97
50 K-means DTW clustering. 97
51 K-medoids Euclidean clustering. 98
52 K-medoids DTW clustering. 98
53 K-shape clustering. 98
54 Hierarchical Euclidean clustering. 100
55 Hierarchical DTW clustering. 100
56 DBSCAN Euclidean clustering. 100
57 DBSCAN DTW clustering. 100
58 Spectral clustering. 101
59 Baseline analysis on the dimension of the clusters. 102
60 Baseline analysis of Cohesion and Coupling. 103
61 Baseline analysis of Time Series Accuracy. 105
62 Baseline analysis of Silhouette score. 106
63 Baseline analysis of runtime. 107
64 Model analysis of the dimension of the clusters. 111
65 Model clustering. 111
66 Model analysis of Cohesion and Coupling. 112
67 Model analysis of Time Series Accuracy. 113
68 Model analysis of Silhouette score. 114
69 Model analysis of runtime. 114
70 Analysis of the dimension of the clusters with different input sizes. . 116
71 Clustering with 100 time series. 116
72 Analysis of Cohesion and Coupling with different input sizes. 117
73 Analysis of Time Series Accuracy with different input sizes. 118
74 Analysis of runtime with different input sizes. 119
75 Analysis of the dimension of the clusters with different dimensions. . 121
76 Clustering with 0.3 “critical” dimension and 0.4 “monitor” dimension. 121
77 Analysis of Cohesion and Coupling with different dimensions. 122
78 Analysis of Time Series Accuracy with dimensions. 123

LIST OF ABBREVIATIONS

TS	Time Series
DTW	Dynamic Time Warping
DBA	DTW Barycenter Averaging
PCA	Principal Component Analysis
MSE	Mean Squared Error
DBSCAN	Density Based Spatial Clustering of Applications with Noise
SMBO	Sequential Model-Based Optimization
TPE	Tree of Parzen Estimators
HOA	Hyperparameter Optimization Algorithm
FA	Fluctuations Accuracy
DA	Descending Accuracy
MA	Moving Average
EMA	Exponential Moving Average
UIC	University of Illinois at Chicago

SUMMARY

The goal of this work is to identify markers for the onset of emotional issues, such as depression and anxiety, in high-school-age children. Such markers may involve declining academic performance, truancy, and behavioural issues in schools. The ability to identify markers predicting the onset of emotional issues in school-age children would allow schoolteachers and administrators to notify parents and guardians of the children affected as soon as the markers are detected. This would in turn facilitate early diagnosis and treatment for children who are indeed developing emotional issues.

To accomplish our goal, we applied unsupervised learning methods to partition the input dataset into clusters with similar characteristics with respect to potential markers, such as declining or fluctuating academic performance. Our analysis seeks to cluster students into three classes ("critical", "monitor" and "stable") whose size aligns with statistics available, for instance, from the US Center for Disease Control and Prevention (CDC). The student's academic data are modelled as time series.

Three main clustering approaches are evaluated empirically: (1) clustering of time series with different lengths, (2) classical clustering algorithms and (3) density-based clustering with feature reduction. As it turns out, only the last approach respects our desired cluster size while also achieving good accuracy results with respect to the markers.

Students in clusters exhibiting fluctuations or sharp declines in academic performance are intended to be referred to their school counsellors for further investigation. We are currently

working with a dataset of about 500 students from 5 Catholic high schools under the jurisdiction of the Diocese of Peoria in Peoria, Illinois.

CHAPTER 1

INTRODUCTION

Time series clustering is an essential method utilized in a variety of industries, including banking, healthcare, and environmental monitoring where data are grouped based on their patterns and traits. This is advantageous because it enables us to spot frequent trends, patterns, and anomalies that might be hidden in separate time series. We can learn more about how complex systems behave, find underlying causes for some trends, and improve our forecasts of what will happen next by clustering time series data. Time series clustering is a useful technique for many real-world applications since it can aid in data compression, anomaly detection, and classification of time series data.

The goal of this project is to identify students with possible onset of emotional issues based on their academic data. Our dataset is made of de-identified grades and absence data of about 500 students from 5 Catholic high schools under the jurisdiction of the Diocese of Peoria in Peoria, Illinois. The Institutional Review Board of UIC evaluated this research project in July 2022 and assigned exempt status on June 14, 2022.

From the academic data, we have extracted one time series per student that highlights the trend (grades) of the student during the whole academic year. To classify the students in a meaningful manner, three classes of time series have been defined:

- "stable": this label means that the student had a linear or upward trend during the considered interval; this type of student does not raise concerns (see Figure 1).

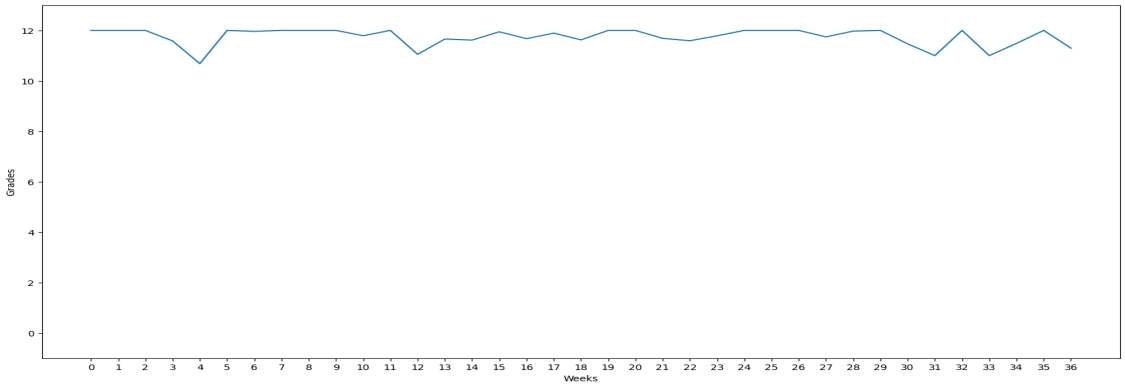

Figure 1: Example of "stable" time series.

- "monitor": this label means that the student had some fluctuations and/or an overall downward trend that is not very marked during the considered interval; this type of student needs to be monitored (see Figure 2).

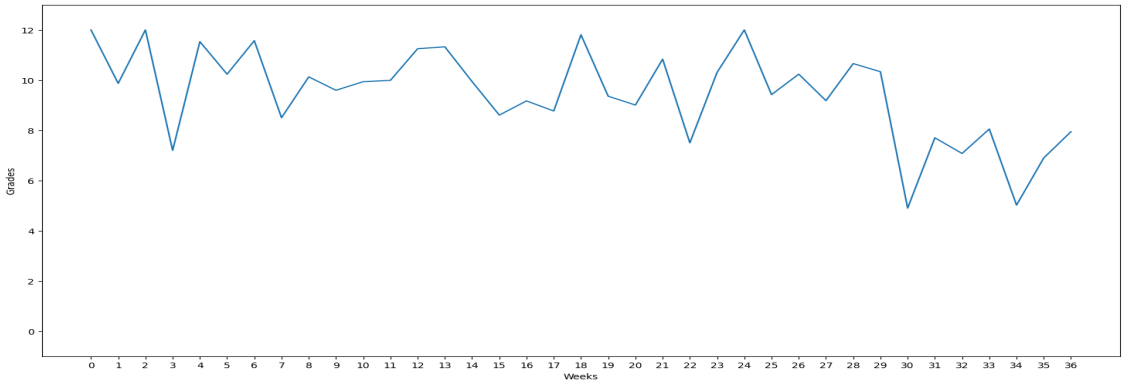

Figure 2: Example of "monitor" time series.

- "critical": this label means that the student had significant fluctuations and/or an obvious downward trend during the considered interval; this type of student is more likely to exhibit emotional issues requiring additional resources (see Figure 3).

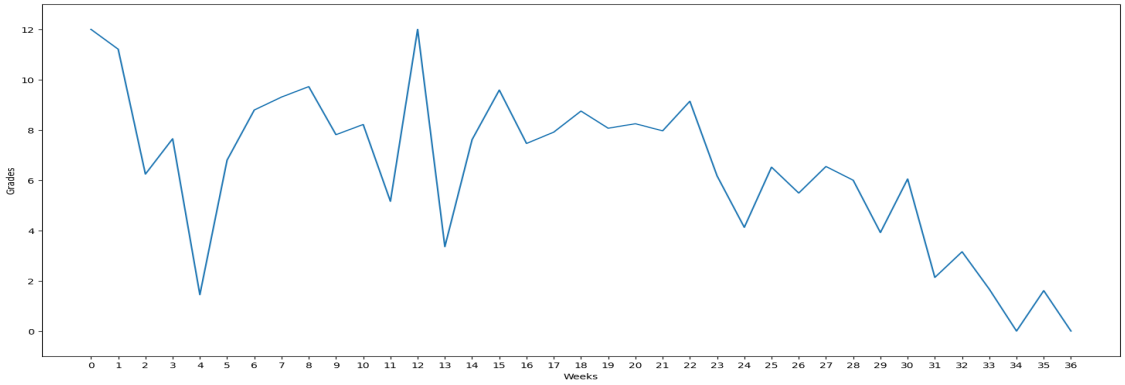

Figure 3: Example of "critical" time series.

The overall goal of the project is to flag at least 10% of "critical" students and 20% of "monitor" students that could be more affected by emotional issues out of the total student population. In the end, the aim is to obtain three clusters of time series corresponding to the three labels distributed as follows: 70% of "stable" time series, 20% of "monitor" time series and 10% of "critical" time series. These percentages are aligned with the statistics available from the US Center for Disease Control and Prevention [1].

To achieve this goal, we implemented and compared three different techniques:

- 1 - Clustering of time series with different lengths (Section 5.1).
- 2 - Classic clustering algorithms (Section 5.2).
- 3 - Density-based clustering with feature reduction (Section 5.3).

Before applying these techniques, the time series need to be created based on the data provided by the Diocese. In Section 4.1, we explain how the time series are created starting from raw data; this process was partly already done by Ishan Choudhary (Section 4.1.2.1). The time series created could not be suitable for clustering purposes since they are very noisy, so data preprocessing techniques are applied to make the time series easier to cluster (Section 4.2).
To our knowledge, there is no previous work related to this work; thus, we developed some specific metrics (Section 5.4) to evaluate the quality of the generated clusters and to compare the different techniques applied (Chapter 6).

1.1 Motivation

The Diocese of Peoria, Illinois, manages several Catholic schools in the territory around Peoria. During the Covid-19 pandemic, the Diocese found that the case of students that showed emotional issues increased significantly. They observed that this development could be linked to the academic performance of the students. For this reason, the Diocese has provided the University of Illinois at Chicago (UIC) with academic data of about 500 students from 5 of their schools to identify classes of students who may require additional resources to cope with emotional issues. The main goal is to early identify the students with persistent drops and/or significant fluctuations in their academic data to detect the onset of emotional issues early on.

CHAPTER 2

BACKGROUND

2.1 Time series

A time series is a sequence of data points collected over time, each of which corresponds to a measurement or observation that was made at a particular moment in time [2]. The data points in a time series are typically recorded at regular intervals, every week in our case.
Time series data might be univariate, which means that only one variable is being tracked across time, such as the grades of a student in the academic year. Alternatively, it can be multivariate, where multiple variables are measured simultaneously over time, such as the grades of a student in the academic year, along with the absences of the student during the academic year. In this work, we use univariate time series considering grades only even if in the future we plan to extend to multivariate time series.

2.2 Time Series Clustering

Time series clustering is a data mining technique that groups together similar time series data based on their patterns and characteristics [3]. Clustering requires looking at data that are ordered in time and finding patterns or connections between different data points. The goal of this technique is to partition the time series into clusters, each of which contains time series that shows similar characteristics.

There are various kinds of time series clustering but, in this work, we only consider whole time series clustering, which means comparing whole time series to find their similarities and differences. As stated in *Time-series clustering – A decade review* [3], whole time series clustering is made of four components: time series representation, distance metrics, clustering representatives and clustering algorithms. The choice of these 4 elements depends on the data and the objective, but it is very important since the quality of the clusters highly depends on the right choice of these components.

Regarding time series representation we have only applied Principal Component Analysis to reduce the dimensionality of the time series in the last technique, in the other techniques we have worked with raw time series. To cluster these time series various clustering algorithms belonging to different categories can be used, such as partitional clustering [4], hierarchical clustering [5], shape-based clustering [6], density-based clustering [7] and spectral clustering [8]. In these algorithms, various distance metrics can be used to calculate the distance between time series; here we consider the most common ones namely Euclidean distance and Dynamic Time Warping (DTW) [9]. Another essential part of time series clustering is finding a representative

or prototype for each cluster and this can be done with various techniques such as a simple mean if the distance metric is Euclidean distance, or DTW Barycenter Averaging [10] if DTW is used as the distance metric.

When the algorithm returns the labels that form the clusters it is important to use some evaluation metrics to assess the quality of the clustering. To do this we can use external or internal indices, depending on whether the ground truth of the time series is available or not. One of the most used and reliable internal evaluation metrics is the Silhouette index [11].

2.3 Time Series Representation

In this work, we will form two types of time series from raw data: time series with different lengths and with the same length. To form time series with the same length it is necessary to use interpolation to fill in missing values that correspond to a student without grades for a certain week. We used Pandas [12] dataframe to store the time series throughout this thesis because Pandas provides integrated and intuitive routines for performing common data manipulations and analyses.

In "clustering of time series with different lengths" (technique 1) and "classical clustering algorithms" (technique 2) data are represented as raw time series while in "density-based clustering with feature reduction" (technique 3) raw time series are processed using PCA to decrease their dimensionality.

2.3.1 Interpolation

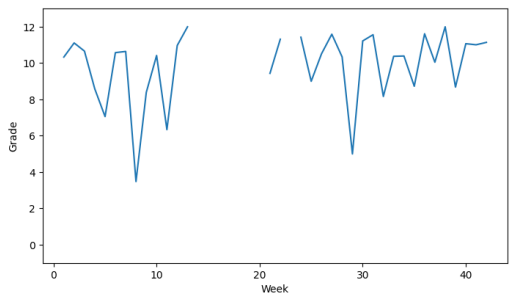

Figure 4: Time series with missing values.

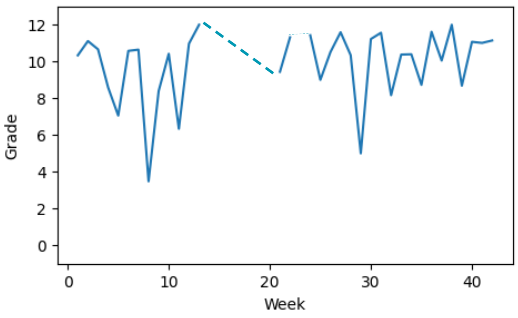

Figure 5: Time series with linear interpolation.

Interpolation of time series means estimating missing values in a time series by using the available data points. Interpolation techniques can be used to fill in the missing values and produce a complete time series from a time series with gaps as shown in Figure 4 and Figure 5. There are several interpolation methods that can be used in time series analysis, we have used the ones offered by the "interpolation" method of Pandas dataframe [13]:

- "linear": the slope of the straight line connecting the two known data points is first calculated, the estimated value at a point between the two known points is then calculated by adding to the value of the first known point, the product of the slope and the distance between the points we want to interpolate as shown in Figure 6.

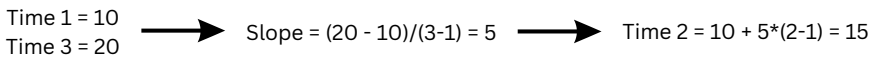

Figure 6: Example of linear interpolation

- "pad": Fill in missing values using previous existing values.
- "backfill": Fill in missing values using subsequent existing values.
- "index": Use the values of an index that is strongly correlated with the time series to estimate the missing values in the time series.
- "nearest": The value of the nearest data point is used as the estimate for the missing value.

- “zero”: Fill missing values with zeros.
- “slinear”: Fit a linear function between the nearest available data points and use this function to estimate the missing values, it differs from linear interpolation since it uses a piecewise linear function (non-continuous function made of several parts).
- “quadratic”: Fit a quadratic function between three adjacent data points and use this function to estimate the missing values.
- “cubic”: Fit a cubic function between four adjacent data points and use this function to estimate the missing values.
- “barycentric”: Use a weighted average of the data points to estimate missing values, with the weights depending on the distance to the interpolation point.
- “polynomial”: Use a polynomial function of a certain degree that passes through the data points to estimate the missing values, it requires specifying the order of the polynomial function.
- “spline”: Use a piecewise polynomial function that is continuous and smooth to estimate the missing values in each sub-interval of the data, it requires specifying the order of the polynomial function.
- “krogh”: Use a piecewise polynomial function to estimate the missing values in each sub-interval of the data, the function can have different degrees for each sub-interval.
- “piecewise_polynomial”: Use a piecewise polynomial function to estimate the missing values in each sub-interval of the data.

- "pchip" (Piecewise Cubic Hermite Interpolating Polynomial): Use a piecewise polynomial function that is monotonic to estimate the missing values in each sub-interval of the data.
- "akima": Use a piecewise cubic function to estimate the missing values in each sub-interval of the data.
- "cubicspline": Use a piecewise cubic function that is continuous and smooth to estimate the missing values in each sub-interval of the data.
- "from_derivatives": Use a piecewise polynomial function constructed in Bernstein basis to estimate the missing values in each sub-interval of the data.

2.3.2 Principal Component Analysis

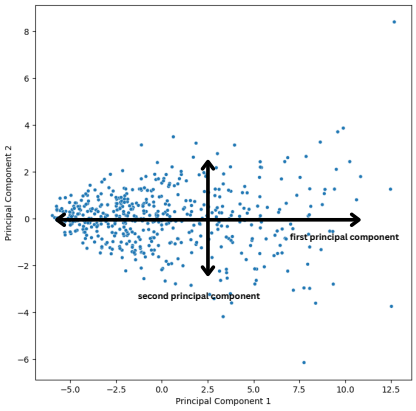

Figure 7: Principal Component Analysis.

Principal Component Analysis (PCA) [14] is a data analysis technique that reduces a large number of potentially correlated variables to a smaller set of variables known as principal components. The main goal of PCA is to keep as much of the data variance while reducing the dimensionality of a data set that contains numerous correlated variables. This reduction is achieved by representing data with principal components (as shown in Figure 7), a new set of uncorrelated and orthogonal variables that are sorted so that the first few retain the majority of the variation present in all the original variables.

Before applying PCA, data must be normalized (mean = 0 and variance = 1) since PCA is a variance-based technique that looks for patterns in the data that capture the majority of the variance. Variables with higher variances will dominate the analysis if the data are not normalized, even if they are not necessarily more informative than other variables. In this way, each variable can contribute equally to the analysis since normalization scales the variables in such a way that they have equal variance. This ensures that the principal components are based on the underlying structure of the data rather than the scale of the individual variables.

The main steps performed by PCA are the following:

1. Normalize the original dataset.
2. Calculate the covariance matrix.
3. Calculate the eigenvalues and eigenvectors of the covariance matrix.
4. Sort the eigenvectors by descending order of corresponding eigenvalues, the first eigenvectors are the ones with the most variance (information).

5. Compute the reduced dataset by multiplying the original dataset with a matrix made of the first k eigenvectors (k specified in input).

6. The reduced dataset will have exactly k dimensions called principal components.

2.4 Distance Metrics

In this work we have used two distance metrics to calculate the distance between time series: Euclidean distance and Dynamic Time Warping. These metrics are the most suitable for time series clustering according to *Time-series clustering – A decade review* [3].

2.4.1 Euclidean Distance

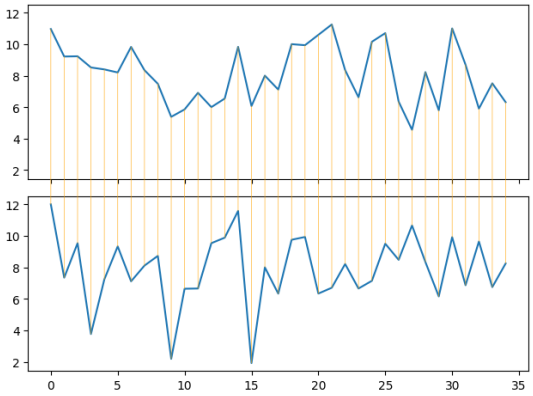

Figure 8: How Euclidean distance compares time series.

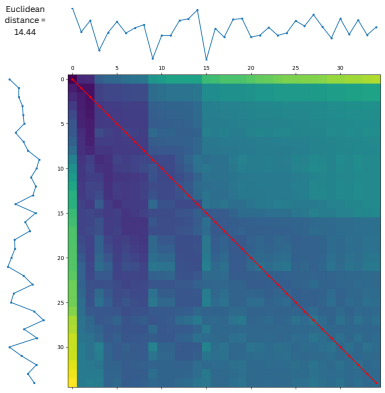

Figure 9: Euclidean distance one-to-one matching.

Euclidean distance is a measure used in many fields to calculate the distance between two data points. When applied to time series, it applies a linear matching between the time series,

comparing each pair of points one-to-one as shown in Figure 8. In Figure 9 we can see the linear path that Euclidean distance creates to compare the corresponding points of the time series. Suppose we have a time series X = (x_1, x_2, x_3) and another time series Y = (y_1, y_2, y_3) the Euclidean distance is computed as following:

$$d(X, Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2} \tag{2.1}$$

This metric is widely used because it is very simple and fast to compute but it has not been designed for time series. If two time series have the same shape but one is shifted a bit Euclidean distance could return a high value even if the two time series are very similar. Moreover, it can not compare time series with different lengths since it is impossible to form a one-to-one matching in that case.

2.4.2 Dynamic Time Warping

Dynamic Time Warping (DTW) [9] uses a dynamic programming approach to align time series so that the distance between them is minimized. The main idea behind DTW, when calculating the distance between two time series, is to shrink and expand different parts of the time series to find their similarities. In this way, we do not have a one-to-one matching between the points of the time series but a point in a time series could be matched with more points in the other time series as shown in Figure 10. The matching between the points is made so that the distance between the time series is minimal and represented by a warping path as shown

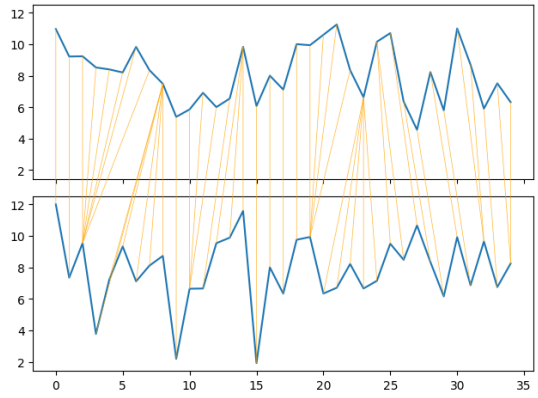

Figure 10: How DTW compares time series.

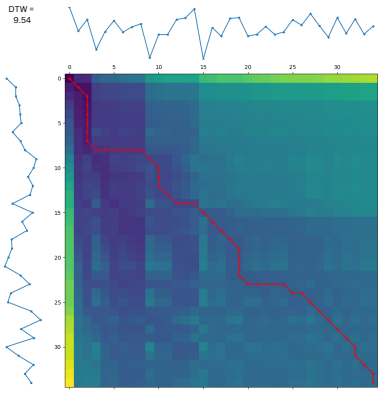

Figure 11: DTW warping path.

in Figure 11. The warping path represents how DTW matches the points of the time series to identify similar sequences. To compute the DTW distance, the differences between each pair of points in the two time series are calculated in a similarity matrix and the warping path is computed finding the shortest path between the top left corner and the bottom right corner of the matrix. DTW is well suited to time series as it returns a distance that is always less or equal than the Euclidean distance; however, it also takes more time to compute.

2.5 Clustering Prototypes

Averaging is an important technique in clustering because it provides a way to calculate the cluster center or centroid. This is especially important in some algorithms such as K-means clustering that use averaging to update the cluster centers at each iteration. The cluster center is a representative point that summarizes the characteristics of the cluster, different clusters can be classified and compared based on the similarities and differences of their centers.
Usually, the cluster center is calculated using an arithmetic mean but, since our data are time series, this method could not be suitable.

2.5.1 DTW Barycenter Averaging

DTW Barycenter Averaging (DBA) [10] is an averaging method used to find an average time series that minimize the DTW distance from the other time series of the cluster. DBA consists of an iterative refinement process where at each iteration two steps are performed:

1. Compute DTW between each time series and the initial average time series to be refined, in order to find associations between coordinates of the average time series and coordinates of the set of time series.
2. Update each coordinate of the average time series as the barycenter (simple mean) of coordinates associated with it during the first step.

DBA starts with an arbitrary time series as a potential average and, with this refinement process, finds the barycenter of the set of time series using DTW.

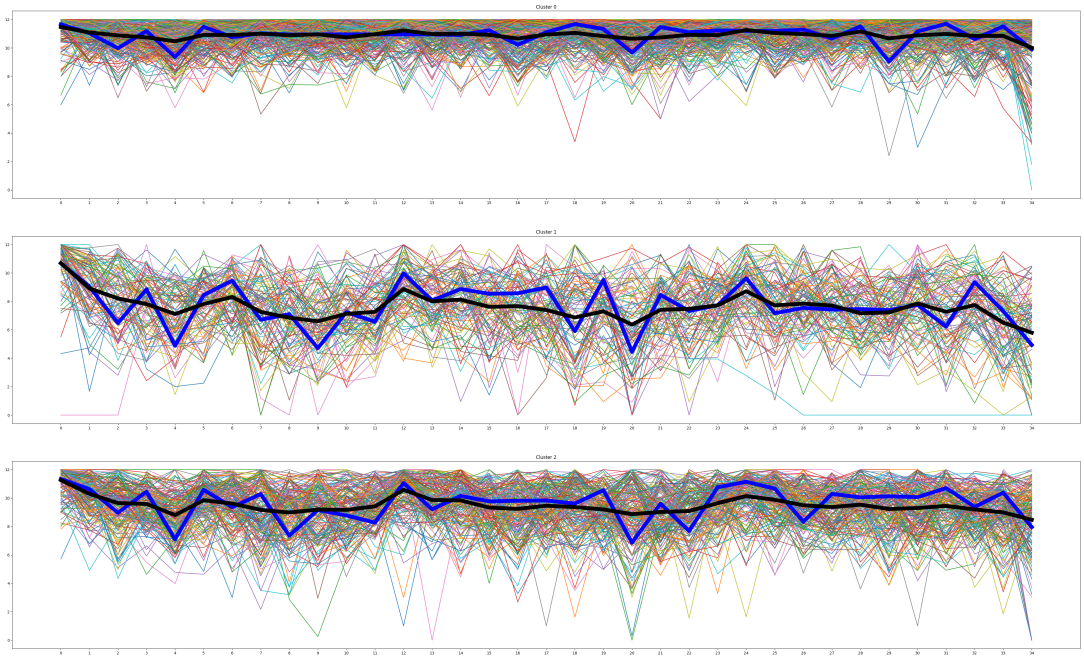

Figure 12: Representative time series calculated with DBA vs mean.

Since we are dealing with time series DBA is more reliable than the arithmetic mean to compute the prototype of a cluster. Indeed, as you can see from Figure 12, the representative time series calculated with DBA (blue lines) and the ones calculated with a simple mean (black lines) do not differ a lot in terms of mean value but they have very different shapes. While the black lines have almost a regular trend, the blue lines have more fluctuations that define better the time series in the cluster.

2.6 Clustering Algorithms

For time series data, a variety of clustering algorithms can be used, some of them are designed specifically for time series while others are widely used algorithms that are also suitable for time series. Depending on the application, these methods can be used to classify time series according to their shape, trend, or other characteristics. The result of the clustering algorithms is represented by labels that specify the cluster number for each time series.

2.6.1 Partitional Clustering

Partitional clustering [4] represents a type of clustering algorithm where the data is partitioned into a predetermined number of clusters, with each data point belonging to exactly one cluster. In other words, partitional clustering separates the data into several distinct, non-overlapping clusters, each of which contains a subset of the data points that are most similar to each other. Partitional clustering algorithms are popular because they are fast, scalable, and easy to implement. However, they have several drawbacks, including their sensitivity to centroid initialization, the requirement to predetermine the number of clusters, and their propensity to converge to a local optimum rather than the global optimum. The two most famous partitional clustering algorithms are K-means and K-medoids.

2.6.1.1 K-means

K-means [15] is the most popular partitional clustering algorithm; it is widely used in many applications. The K-means algorithm starts by initialising K data points (time series in our

case) as cluster centers. For each data point, it finds the closest cluster center and assigns the point to that cluster. When all the points have been assigned, the cluster centers are updated calculating the mean of the data points of each cluster. The process is repeated until the cluster centers are stable as explained in Algorithm 1.

```
input  : K, number of clusters;
D, time series dataset
output: Set of K clusters
Initialization of cluster centers;
do
    for each point p in D do
        Find the nearest cluster center;
        Assign p to the corresponding cluster;
    end
    Update cluster centers calculating the mean of the points in each cluster;
while cluster centers change;
```

Algorithm 1: Pseudocode for K-means algorithm.

The K-means algorithm may form different clusters depending on the distance metric used to calculate the distance between two data points, the averaging method used to update the cluster centers and the initialization of the cluster centers. Indeed, depending on the centroid initialization, K-means could converge to a local minimum instead of finding the optimal solution. A useful technique for cluster center initialization is K-means++ [16], a simple randomized seeding technique that can improve the accuracy of K-means and make it less sensitive to ini-

tialization. Finally, K-means works better with convex clusters while it could return wrong results with concentric distribution and noisy data.

2.6.1.2 K-medoids

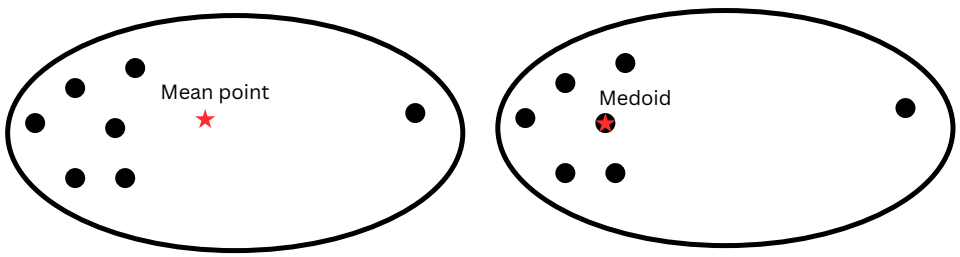

Figure 13: K-means vs K-medoids.

K-medoids [17] is a variant of K-means that is more robust to noises and outliers since, instead of updating the cluster centers with the average point, it uses an actual point called medoid to represent the cluster. The medoid is the most central point in the cluster and it has the lowest distance from all the other points in the cluster, for these reasons it is less sensitive to outliers than the average point calculated by K-means as shown in Figure 13. K-medoids,

unlike K-means, always converges to the optimal solution but it is not scalable to large dataset due to its computational complexity, since it needs to calculate the distance between each pair of data points in the cluster to determine the medoid.

2.6.2 Hierarchical Clustering

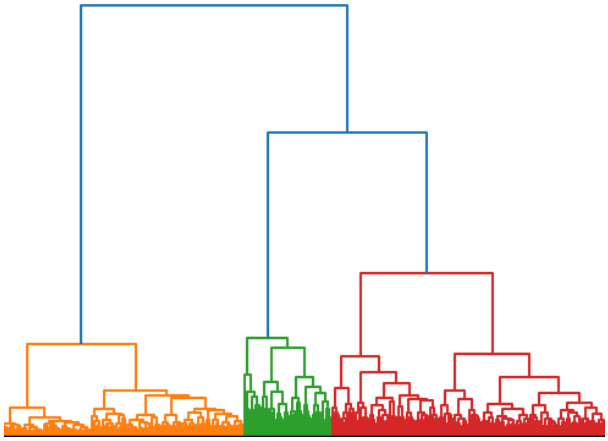

Figure 14: Dendrogram generated by Hierarchical Agglomerative clustering.

Hierarchical clustering [5] is a type of clustering that creates a hierarchy of clusters that can be represented by a dendrogram which is a tree-like diagram that shows the relationships between the clusters. In particular, Agglomerative Hierarchical clustering starts with all the data points belonging to its own cluster and, at each iteration, merges the two closest clusters until only a desired number of clusters is reached. The dendrogram is constructed bottom-up

by aggregating the clusters as shown in Figure 14. This type of clustering does not require specifying in advance the number of clusters but it can be determined after the clustering by looking at the dendrogram and choosing the desired number of clusters.

If we consider Agglomerative Hierarchical clustering, at each iteration the algorithm has to decide which are the two closest clusters to merge. For this reason, a distance metric is used to compute the distance between each pair of data points and, after that, a linkage function is used to identify the two closest clusters. There are many types of linkage functions but the most common are:

- Single linkage: The distance between two clusters is defined as the minimum distance between a point in one cluster and a point in the other cluster.
- Complete linkage: The distance between two clusters is defined as the maximum distance between a point in one cluster and a point in the other cluster.
- Average linkage: The distance between two clusters is defined as the mean distance between a point in one cluster and a point in the other cluster.
- Ward linkage: The distance between two clusters is defined as the sum of the squared deviations between the centroids of the two clusters. The distance between two clusters C_1 and C_2 can be described by the formula

$$d(C_1, C_2) = \frac{|C_1| * |C_2|}{|C_1| + |C_2|} * ||centroid(C_1) - centroid(C_2)||^2 \tag{2.2}$$

where

$$centroid(C_i) = \frac{1}{|C_i|} * \sum_{x \in C_i} x \tag{2.3}$$

The last two functions could be preferred with respect to the first two since they are less sensitive to noise and outliers. In particular, average linkage tends to produce convex clusters with similar sizes and densities while ward linkage tends to produce compact and spherical clusters, and is particularly effective when the clusters have different sizes and variances (the mean distance between time series in the same clusters is very different for each cluster).

2.6.3 Shape-based Clustering

Shape-based clustering is a technique used to group time series data based on their shape or pattern similarity. This method compares time series according to their shapes rather than their exact values, making it possible to spot patterns and trends that might not be obvious through an analysis of raw data. The most representative shape-based clustering algorithm based on time series is called K-shape.

K-shape [6] is a shape-based clustering algorithm for time series data that is invariant to scaling and shifting. It is based on the K-means algorithm with two main differences:

- Shape-based distance: To compute the distance between two time series a shape-based distance is used, it is bounded between 0 and 2 where 0 means perfect similarity. First, a cross-correlation measure is computed shifting left and right a time series while keeping the other fixed, for each shift the distance between the two time series is recorded in the

cross-correlation sequence. The sequence is then normalized and the maximum value, corresponding to the shift for which the two time series are more similar, is used to calculate the final value of the shape-based distance.

- Averaging method: The centroid of each cluster is computed using the shape-based distance such that the sum of squared distances to all the other time series in the cluster is minimized.

K-Shape is particularly well-suited for clustering time series with similar shapes but different scales and shifts, it produces homogeneous and well-separate clusters. One potential limitation of K-shape is that it can be computationally expensive for large datasets since, at each iteration, it requires calculating the cross-correlation measure between each time series and the centroids.

2.6.4 Density-based Clustering

Density-based clustering [7] refers to unsupervised learning approaches that discover well-defined clusters in the data, where a cluster in a data space is defined as a contiguous region of high point density separated from other clusters by contiguous regions of low point density. The data points in the separating regions of low point density that do not belong to any cluster are typically considered outliers. Density-based clustering does not need to know in advance the number of clusters because it will form clusters of arbitrary shapes only identifying the regions of high point density.

Density Based Spatial Clustering of Applications with Noise (DBSCAN) [18] is the most representative density-based clustering algorithm and also one of the first algorithms of this category.

DBSCAN uses the concept of neighbourhood to find core points in the data, and from these it forms the clusters. It takes two parameters as input:

- "eps": Represent the maximum distance between two points for one to be considered in the neighbourhood of the other.
- "min_samples": Represent the minimum number of points in the neighbourhood of a point to be considered as a core point.

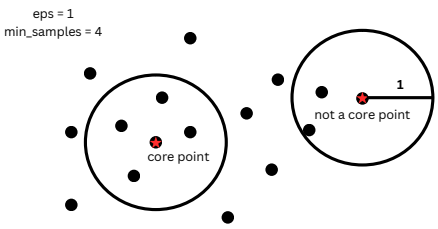

Figure 15: Core point identified by DBSCAN.

A core point is a point that has at least "min_samples" points (included itself) within a circular area of radius "eps" as shown in Figure 15. DBSCAN forms a cluster starting from a core point and adding to the cluster all the points in the neighbourhood of the core point, this operation is repeated for each core point added to the cluster. In the end, the points

not belonging to any cluster are identified as outliers so DBSCAN has the feature of outlier detection.

DBSCAN can form clusters with arbitrary shapes and sizes depending on the values of "eps" and "min_samples", the choice of these parameters can have a significant impact on the resulting clusters. One drawback of DBSCAN is that it is not suitable for high-dimensional data since the density of points could appear more uniform with a high number of dimensions and make it harder to identify regions of high point density.

2.6.5 Spectral Clustering

Spectral clustering [8] is a graph-based clustering technique that performs data reduction from a graph representation of the data and applies classical clustering algorithms to cluster the low-dimensional data. It performs several steps:

1. Construct the similarity matrix that represents the pairwise similarity between the data points.
2. Compute the graph Laplacian matrix, also called affinity matrix, that summarizes the structure of the graph defined by the similarity matrix.
3. Compute the eigenvectors and eigenvalues of the affinity matrix to obtain a low-dimensional representation of the data.
4. Assign labels to the reduced data using a classical clustering algorithm.

The affinity matrix is usually computed using the K-nearest neighbours method that connects two points in the graph only if they are in the K-nearest points of each other.

There are three main methods to assign labels:

- K-means: Use the K-means algorithm to cluster the reduced data.
- Discretize [19]: This approach uses normalized cuts on the graph to cluster the reduced data.
- Cluster qr [20]: Extract clusters directly from the eigenvectors.

Spectral clustering is able to handle non-linearly separable data and discover clusters of arbitrary shapes but it is also very suitable for high-dimensional data since it performs data reduction before clustering. However, it can be computationally expensive and requires specifying the number of clusters upfront.

2.7 Evaluation Metrics

Clustering evaluation metrics are used to assess the quality of the clustering results. There are two categories of evaluation metrics:

- External metrics: Used when the true labels of the data are available, these metrics compare the true labels with the predicted labels to provide a score of the accuracy of the resulting clusters.
- Internal metrics: Used when the true labels of the data are not available, these metrics evaluate the quality of the clustering without any additional information.

Since, in this work, the true labels are not available, only internal metrics can be applied to assess the quality of the clusters. These metrics use different distance measures to calculate how similar data in the same cluster and how different data in different clusters are. As reported in *Understanding of Internal Clustering Validation Measures* [21], the most used and reliable internal evaluation metrics are the Calinski-Harabasz index, I index, Dunn's index, Silhouette index, Davies-Bouldin index, Xie-Beni index, SD validity index and S_Dbw validity index.

2.7.1 Silhouette Index

The Silhouette index [11] is one of the most famous internal evaluation metrics. It measures the closeness of points in the same cluster compared to the distance of points of different clusters. This index consists of two parts:

- "a": The mean distance between a point and all the other points in the same cluster.

- "b": The mean distance between a point and all other points in the nearest cluster.

For each data point, the formulation of the Silhouette is as follows

$$s = \frac{b - a}{max(a, b)} \tag{2.4}$$

The final score is obtained by calculating the mean of the Silhouette score of all the points. The Silhouette score can be calculated using any distance metric; it ranges between -1 and 1 where 1 means dense and well-separated clusters and -1 means that the points are in the wrong clusters.

2.8 Hyperparameter Tuning

Hyperparameter tuning consists of selecting the best set of hyperparameters for a machine learning model to optimize the performance of the model. While parameters are variables that are learned from the data during the training, hyperparameters are set before the training rather than being learned from the data. To tune the hyperparameters it is necessary to try different combinations and evaluate the performance of the model using a loss function to minimize or maximize.

There are different types of hyperparameter tuning:

- Manual search: Try a combination of hyperparameters and manually define the next combination based on the results of the previous one. This approach is time-consuming.
- Grid search: Try all possible combinations of hyperparameters. This approach is impractical when the number of possible combinations is high.
- Random search: Try random combinations of hyperparameters. This approach is feasible but could not give a good set of hyperparameters.
- Bayesian tuning: Utilize Bayesian principles to iterate through different combinations of hyperparameters to find the best result.

Hyperparameter tuning is an important step in building a machine learning model since the choice of hyperparameters can significantly impact the performance of the model. A model with the right combination of hyperparameters can lead to better performance and faster con-

vergence.

2.8.1 Bayesian tuning

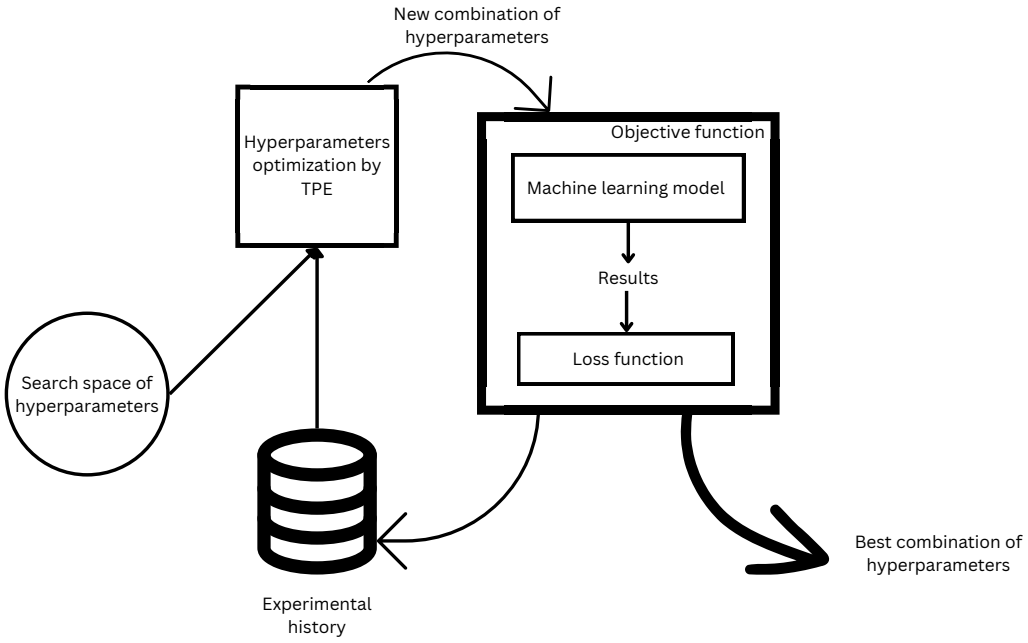

Figure 16: Bayesian tuning process.

Bayesian tuning is a hyperparameter tuning technique that allows finding quickly one of the best sets of hyperparameters without trying all the possible combinations. One of the frameworks that implement Bayesian tuning is Hyperopt [22], a Sequential Model-Based Optimization (SMBO) approach that constructs a model to approximate the performance of the hyperparameters calculated by an objective function. The objective function maps a combination of hyperparameters to a clustering objective by training the model and evaluating the result using a loss function. The optimization procedure is based on the Tree of Parzen Estimators (TPE), a Hyperparameter Optimization Algorithm (HOA) that takes as input the search space of the hyperparameters and the experimental history, and returns which configuration of hyperparameters to try next. With the Bayesian tuning, we try to minimize the loss function by applying to the objective function different combinations of hyperparameters suggested by TPE. The whole process is shown in Figure 16 and, in the end, the best combination of hyperparameters is the one associated with the minimum value of the loss function.

CHAPTER 3

RELATED WORK

The field of time series clustering is very broad and many different approaches have been defined over the last decades. An important consideration about time series clustering that is confirmed in *A benchmark study on time series clustering* [2] is that no single method performs better for all types of datasets, but the right approach depends on the data. Indeed, the task that we would like to accomplish in this work is very domain-specific and strongly depends on the nature of the data. The existing literature mainly refers to general approaches that solve the problem of time series clustering, while only a few works deal with academic data. In this chapter, we present some works related to specific parts of time series clustering that are also used in our work. We start talking about DTW in Section 3.1 and, after that, we discuss constrained clustering algorithms in Section 3.2, which follow an approach similar to what we present in "density-based clustering with feature reduction" (Section 5.3).

3.1 Dynamic Time Warping

In this section, we discuss DTW, the most important distance metric for time series. The main advantage of DTW is that it can expand and shrink time series while comparing them to match similar sequences. This feature makes DTW very effective for time series clustering but its computation has quadratic time and space, so it is limited to relatively small datasets. To sidestep this problem, FastDTW [23] was developed. This method computes an approximation of the optimal warping path of DTW but with linear time and space complexities. To make this approximation, multiple time series are reduced to a lower dimension. After that, a warping path is computed at the lower dimension which is used as an initial guess to find the warping path at a higher dimension. The process is repeated until a warping path for the original dimension of the time series is found.

When computing the warping path, DTW could encounter a point where two sub-paths produce the same distance. When this happens, the algorithm randomly chooses which path to take. In the end, this random choice could lead to a sub-optimal warping path. To resolve this issue, a modification of DTW is proposed in *Modified Dynamic Time Warping for Hierarchical Clustering* [24]; it allows finding the shortest path when equivalent values are encountered in the similarity matrix during the computation of the warping path.

Another disadvantage of DTW is pathological alignments that occur when a single point in a time series is matched with a large subsection of another time series; this could lead to incorrect distance values. Many solutions have been proposed for this problem, and the most effective one is LDTW [25], a variant of DTW that sets a limit on the length of the warping path. With

this simple constraint, LDTW can alleviate the problem of pathological alignments while still allowing the algorithm to find an optimal match between the time series. This algorithm, instead of imposing a strict local constraint on the number of points a point can link to, imposes a softer global constraint that maintains the flexibility of DTW. Another more complex approach is ACDTW [26], which uses two adaptive penalty functions when calculating the warping path. These penalty functions not only resolve the problem of pathological alignments but also allow for a more accurate calculation of the distance between two time series while finding the optimal warping path. Moreover, ACDTW reduces the over-stretching and over-compression of the time series while also avoiding excessive many-to-one or one-to-many matching resulting in a more accurate distance value.

3.2 Constrained clustering

Constrained clustering exploits user-specified constraints to improve clustering accuracy. By incorporating this additional information, the clustering algorithm may prevent incorrect clustering that could result from the intrinsic characteristics of the data. Constraint-based clustering can also enhance the interpretability of the clustering results by enforcing prior knowledge or domain expertise into the clustering process.

Data Clustering with Cluster Size Constraints Using a Modified K-Means Algorithm [27] defined a constrained clustering algorithm that creates clusters of predefined size. In that paper, a modified version of the K-means algorithm integrates the constraints on the size of the clusters. That algorithm uses prior knowledge to initialize the centroids, thereby reducing the probability of converging on a local minimum. Moreover, each point is assigned to the closest cluster center until the size of the cluster is lower or equal to the specified size; otherwise, the point is assigned to the next closest cluster center. This algorithm can satisfy one of the constraints of our task, which is to form clusters of specified dimensions, but it requires prior knowledge to form good clustering and it is not suitable for time series data.

Another type of constraint that could be given as additional information for clustering is instance-level constraints. These are pairwise constraints of two types: *must-link* constraints, which express that two points should be in the same cluster, and *cannot-link* constraints, which express that two points should not be in the same cluster. This information is exploited by constrained K-means [28], a modified version of the K-means algorithm that takes as input a set of *must-link* and *cannot-link* constraints and returns a clustering that satisfies all the constraints.

The main modification of the original algorithm is that it ensures that all the constraints are satisfied when updating the cluster assignments. This algorithm was recently outperformed by *Improved Constrained K-Means Algorithm for Clustering with Domain Knowledge* [29], an improved version of constrained K-means that first uses only *must-link* constraints to form the initial clusters. Subsequently, this method integrates *cannot-link* constraints assigning the remaining data points to the existing clusters. The improved constrained K-means method outperforms the traditional K-means as well as the previous constrained K-means [28].

The same approach of constrained K-means is applied for time series data in COBRAS^{TS} [30] that also integrates some aspects of interactive clustering. COBRAS^{TS} interactively queries the user while performing the clustering to obtain *must-link* and *cannot-link* constraints. It uses these constraints to guide the clustering using spectral clustering with DTW or K-shape. COBRAS^{TS} has shown that a small amount of supervision queries could improve the results of the clustering. Indeed, COBRAS^{TS} has outperformed unsupervised and previous semi-supervised algorithms [28; 29]. It could be very useful to apply this approach to our task, but it can not guarantee that the size of the resulting clusters matches our goal, and it requires an expert to answer the queries.

CHAPTER 4

DATA PREPARATION

4.1 Data Creation

Data creation refers to the process of generating structured data that can be analyzed, interpreted and used to gain insights or solve problems. To guarantee that the data is accurate, dependable, and consistent the quality of data creation is crucial. This is significant because inaccurate conclusions and poor decision-making can result from data of poor quality. To ensure that the data are of high quality and pertinent to the topic at hand, data creation must be done meticulously and systematically.

For this work, the data creation was partially already done previously by Ishan Choudhary who has previously worked on the project. Our contribution has been to review and enhance the data creation process. Moreover, we have implemented new features to make the process more reliable and accurate. These features allow us to create different types of data depending on the needs at hand.

4.1.1 Raw Data

We have created the time series needed for the analysis starting from the grades provided by the diocese in CSV format and then stored in a Pandas dataframe. Each row of the file corresponds to a single grade with the following attributes:

- "person_number": The unique code associated with each student.
- "academic_year": The academic year of the grade, in our case all the grades in the file refer to the academic year 2021/2022.
- "course_category": The topic of the grade, can be one of [""SOC", "ENG", "OTR", "SCI", "LAN", "MTH", "nan"].
- "course_level": The level associated with the grade, can be one of ["REG", "HON", "MOD"].
- "assignment_date": The week number of the grade, ranges from 1 to 53.
- "assignment_type": The type of the grade, can be one of ["OTHR", "TEST", "HOME", "QUIZ", "EXAM"].
- "point_range": The weight of the grade, ranges from 15 to 300.
- "score": The alphabetic score of the grade, ranges from F to A+.
- "location_id": The id of the school, is not useful for the analysis.

The total number of students with at least one grade is 541. In Table I you can see an example of a portion of the data provided by the Diocese.

TABLE I: RAW DATA

person_number	academic_year	assignment_date	assignment_type	point_range	score
1	2122	27	OTHR	15	A+
2	2122	30	OTHR	50	B

4.1.2 Time Series with the Same Length

In this section, we present the process to create time series with the same length that will be used for "classical clustering algorithms" (Section 5.2) and "density-based clustering with feature reduction" (Section 5.3).

4.1.2.1 Old Data Creation Process

The data creation that was previously done is the following:

1. Delete all the rows with a missing value for the "score" attribute.
2. Translate the alphabetic score to a numeric score from 0 (F) to 12 (A+).
3. Calculate the normalized score by multiplying each numeric grade by its "point_range" to weights different grades with different point ranges.
4. For each student, course category and week, calculate the final score as a weighted average (sum of normalized scores divided by the sum of the point ranges).
5. Select all the rows with "assignment_date" between 34 and 47 corresponding to the fall semester.
6. For each student and week, calculate the mean of the final scores of the different course categories.
7. For each student, create a time series with the final scores (one for each week).
8. Interpolate the missing values (corresponding to students that have no grades in a specific week) with the linear interpolation.

4.1.2.2 New Features

To make the process more accurate, we have applied data analysis to decide which data are useful and which are not for our task.

The "assignment_date" is referred to the week number of the year in the raw data but usually, the schools use the week number referred to the academic year (for instance Weeks 1, 2 and 3 of the Fall semester are reported as Weeks 34, 35 and 36 of the year). To address this problem, we have refined the week number according to the academic year (the first week of the academic year will be Week 1). After that, we deleted all the rows with a missing value for the "score" attribute (invalid grades). We have calculated some statistics to decide which weeks to consider for the analysis. In particular, we have calculated the number of grades per week (Figure 17) and the number of students with at least one grade per week (Figure 18).

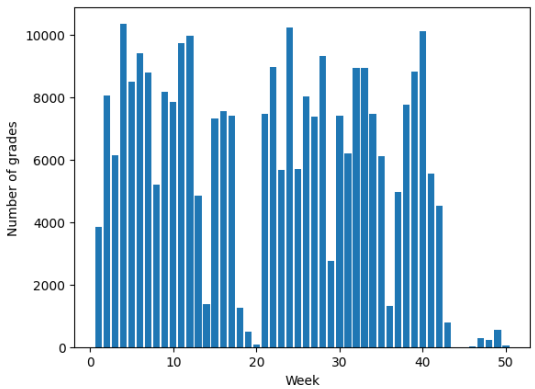

Figure 17: Number of grades per week.

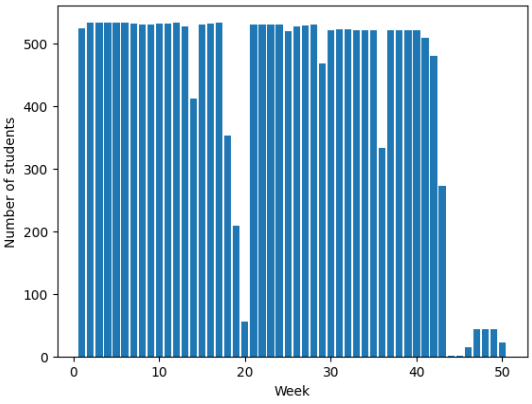

Figure 18: Number of students with at least one grade per week.

As you can see in Figure 17 and Figure 18, four different periods can be clearly identified:

- Fall semester: From Week 1 to Week 17, Thanksgiving could be Week 14.
- Christmas break: Weeks 18, 19 and 20.
- Spring semester: From Week 21 to Week 43, Spring Break could be Week 36.
- Summer break: From Week 44 to Week 53.

The grades submitted during the Christmas break could be associated with the Fall semester but entered late by professors; for this reason, it could be useful to move them to the last week of the fall semester. On the contrary, the grades corresponding to the Summer break could be referred to few students taking classes during summer, so they are not useful for our analysis. Anyway, during the data creation process, there is the possibility to generate the time series for the whole year or to specify an interval of weeks to consider.

Another special case is students that have grades with "MOD" as "course_level", this indicates that a course's content is below grade level for the students enrolled. As we have learned from a psychologist, they have a higher probability to be affected by emotional issues. We have calculated that the percentage of "MOD" grades is about 3% over the total number of grades. Only 82 students over 541 have received at least one "MOD" grade (but these students have also received non "MOD" grades). For these reasons, during the data creation process, there is the possibility to include or not the "MOD" grades.

The last variable that can be chosen during the data creation process is the possibility of creating time series for a specific topic or for all the topics together. Figure 19 shows the

percentage of grades belonging to each topic, the topic “nan” is referred to “MOD” grades discussed previously.

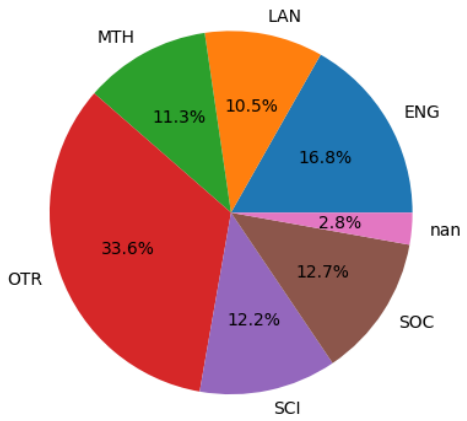

Figure 19: Distribution of grades per topic.

4.1.2.3 New Data Creation Process

With the previous enhancements the new data creation process is the following:

1. Redefine the week number according to the academic year as opposed to the calendar year.
2. Delete all the rows with a missing value for the “score” attribute from the CSV file.
3. Translate the alphabetic score to a numeric score from 0 (F) to 12 (A+).

4. Calculate the normalized score by multiplying each numeric grade by its "point_range" to weights different grades with different point ranges.

$$normalized\_score = numeric\_score * point\_range \tag{4.1}$$

5. If the "MOD" flag is disabled, delete all the "MOD" grades.

6. For each student, course category and week, calculate the final score as a weighted average (sum of normalized scores divided by the sum of the point ranges). A few rows of the result are shown in Table II.

$$final\_score = \frac{\sum normalized\_score}{\sum point\_range} \tag{4.2}$$

TABLE II: DATAFRAME WITH FINAL SCORES

Person Number	Academic Year	Course Category	Assignment Date	Final Score
1	2122	OTR	1	12
1	2122	OTR	2	8.647058824
1	2122	OTR	3	11.65384615
1	2122	OTR	4	5.23943662
1	2122	OTR	5	2.131147541

7. Filter weeks moving grades from the Christmas or Spring break in a previous week that has enough grades. This is done to minimize the missing values, as explained in Algorithm 2.

```
input  : Dataframe with all the weeks
output: Dataframe with filtered weeks only
accepted_weeks = [];
high_threshold = 0.9 * number_of_students;
low_threshold = 0.1 * number_of_students;
for i ← each week of the year do
    if students_per_week [i] > high_threshold then
        Add i to accepted_weeks;
        last_accepted_week = i;
    else if i − 1 in accepted_weeks then
        if students_per_week [i] > low_threshold then
            Add i to accepted_weeks;
            time_series_dataframe ["assignment_date"] = last_accepted_week;
        end
end
time_series_dataframe ← time_series_dataframe where "assignment_date" is in
 accepted_weeks;
```

Algorithm 2: Pseudocode to filter weeks.

8. If specified, select all the rows with "assignment_date" within a specific interval.

9. If specified, select all the rows with "course_category" belonging to a specific topic.

10. For each student, calculate the mean of the final scores of each week.

11. For each student, create a time series with the final scores (one for each week).

TABLE III: DATAFRAME WITH TIME SERIES BEFORE INTERPOLATION

Person Number	0	1	2	3	4	5	6	7	8
1	4.5	3.334	12	6.174	2.335	6.677	3.303	2.821	4.889
2	11.5	11.692	8.56	11.394	9.358	8.869	10.427	3.917	4.259
3	12	9.308	11.576	5.417	0				
4	7.5	10.482	8.494	4.936	3.178	9.81	6.09	4.52	8.733
5	10.8	3.556	6.921	3.047	2.588	5.32	3.709	7.162	2.067

To filter the weeks in Step 7 we could also have filtered the weeks manually but, in this way, we are trying to create a framework reusable in the next years with other academic data. With this data creation process, the aim is not to lose data and to create time series with only a few missing values to interpolate.

If we consider the whole academic year, at this point of the data creation process the dataframe contains 539 time series corresponding to 539 different students. Each of them is made of 35 values corresponding to the final scores of the weeks. The time series created may contain missing values because some students may have not received grades for certain weeks. You can see an example in Row 3 of Table III where there are missing values for Weeks 5 to 8.

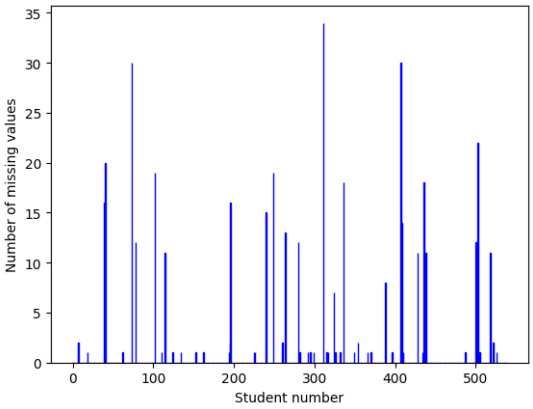

Figure 20: Number of missing values per student.

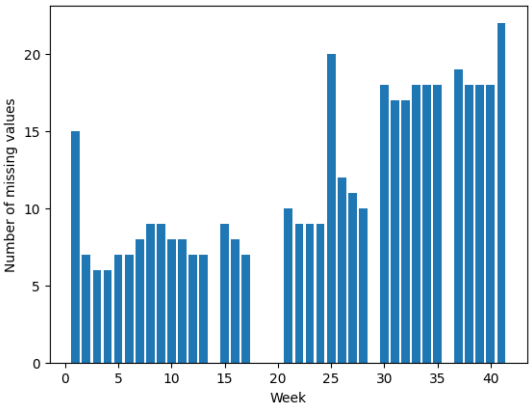

Figure 21: Number of missing values per week.

To analyse better how the missing values are distributed, Figure 20 shows the number of missing values for each student; Figure 21 shows the number of missing values for each week. As you can see there are 419 missing values (around 0.8 missing values per time series) with a few students containing a lot of missing values. Since the interpolation works well with only a few missing values, the students with more than 25% of missing values are discarded. These students must be treated as special cases and analyzed separately. In this way, we obtain 518 time series. After the deletions, there are only 55 missing values remaining in the time series (around 0.1 missing values per time series) distributed over the weeks of the academic year.
At this point, there are a few missing values remaining so interpolation can be applied to the time series to fill all of them. Since there are a lot of different types of interpolation, a specific analysis must be done to discover which method is the best one for our task. You can find this analysis in Section 4.1.2.4.

For the experiments, we decided to use grades referred to the whole year and for all the topics ("MOD" grades included). In the end, we obtained 518 time series, one for each student, each of them made of 35 data points (corresponding to 35 weeks). Each data point is the mean of all the grades received by the student in that week. These time series have all the same length and no missing values, so they are appropriate to be used with "classical clustering algorithms" (Section 5.2) and "density-based clustering with feature reduction" (Section 5.3) that require these characteristics of input data.

4.1.2.4 Interpolation Evaluation

Given the need to fill in the remaining missing values, it is necessary to employ interpolation. The "interpolate" function of Pandas dataframe [13] provides a variety of interpolation methods, as well as other parameters that may influence the accuracy of the interpolation process. The most important parameters selected for the evaluation are the following:

- "method": The Interpolation technique to use. Possible values are ["linear", "pad", "backfill", "index", "nearest", "zero", "slinear", "quadratic", "cubic", "barycentric", "spline", "polynomial", "krogh", "piecewise_polynomial", "pchip", "akima", "cubicspline", "from_derivatives"].
- "order": The order of the interpolation, needed only for "spline" and "polynomial" methods.
- "axis": The axis to interpolate along. "0" to interpolate over rows (vertical interpolation) or "1" to interpolate over columns (horizontal interpolation).

- "limit_area": The restriction on how to fill missing values. "None" means no restriction or "inside" to interpolate only missing values surrounded by valid values.
- "limit_direction": The direction where to fill consecutive missing values. Possible values are ["forward", "backward", "both"].

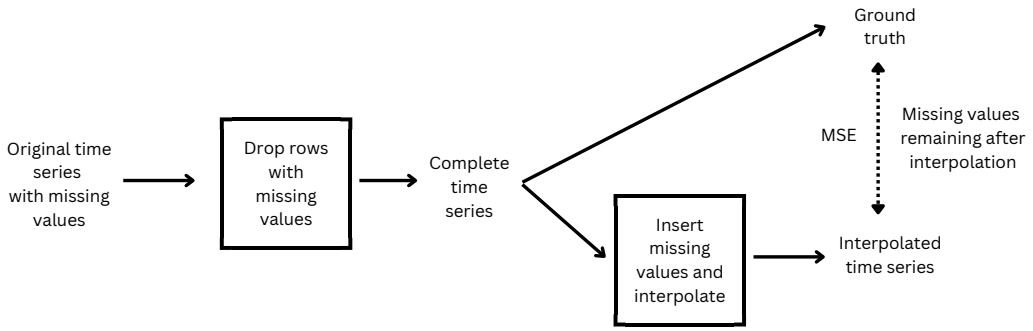

Figure 22: Interpolation evaluation process.

Since the number of possible parameters is reasonable, a grid search can be applied to evaluate all the possible combinations. To evaluate each combination the idea is to construct a ground truth of time series without missing values. From the ground truth, we want to create some time series with missing values to interpolate. In this way, we can evaluate the accuracy of the interpolation by comparing the interpolated time series and the ground truth as shown

in Figure 22.

The process starts from the time series with missing values and performs several steps:

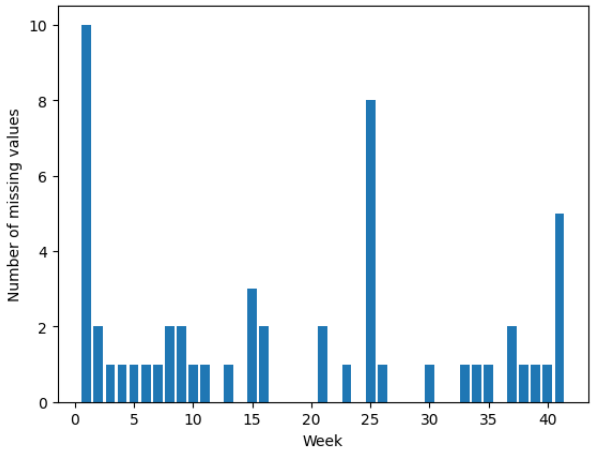

Figure 23: Number of missing values per week.

1. Calculate the number of missing values remaining per week as shown in Figure 23. The total number of missing values is 55 distributed over 36 time series (the other time series do not contain missing values). We can calculate the average number of missing values in the time series with at least one missing value that we will use later (1.53 missing values per time series).

2. Since we want to construct a ground truth, all the time series with missing values are dropped. The remaining 482 time series are complete and these will be our ground truth.

3. From the ground truth we create the time series to interpolate by inserting missing values with the same distribution (Figure 23) and probability (1.53 missing values per time series) of the original time series.
4. Now it is possible to interpolate these newly created time series using all different combinations of parameters specified before.
5. Finally, we evaluate the interpolation by calculating the mean squared error (MSE) between the interpolated time series and the ground truth.
6. Since the interpolation could not fill in all the missing values, we calculate also how many time series still have missing values.

The ideal interpolation method produces time series equal to the ground truth (MSE = 0) and without missing values, indeed we do not want to drop other time series due to remaining missing values after interpolation.

We have evaluated 248 different combinations of parameters and, in the end, the best one has been chosen to be the one with the lowest MSE that fills all the missing values (no remaining missing values in the interpolated time series). The best combination is the following:

- "method": "linear" (linear interpolation)
- "axis": 1 (interpolation along the columns of the dataframe)
- "limit_area": "None" (no fill restrictions)
- "limit_direction": "both" (fill missing values both forward and backward)

This combination is not the best one in terms of MSE but it has a low error (MSE = 0.14, best MSE = 0.09) and allows us to drop no time series after interpolation. In particular, the most important parameter is the "limit_direction" that is set to "both"; it allows the algorithm to interpolate missing values at the beginning and at the end of the time series that usually are difficult to predict.

This combination of parameters is used to interpolate the original time series and create the complete time series used for "classical clustering algorithms" and "density-based clustering with feature reduction".

4.1.3 Time Series with Different Lengths

While "classical clustering algorithms" and "density-based clustering with feature reduction" will take as input time series with the same length, "clustering of time series with different lengths" (Section 5.1) will use time series of different lengths.

The process of creating the time series with different lengths is very similar to the data creation process described in Section 4.1.2.3. The main difference is that there is no need to filter weeks (as explained in Algorithm 2) and to interpolate missing values (Section 4.1.2.4) because the weeks with no grades are discarded and all the following weeks are shifted to the left. In this way, the generated time series respects the chronological order of the grades thereby avoiding the creation of artificial data that could be not very precise.

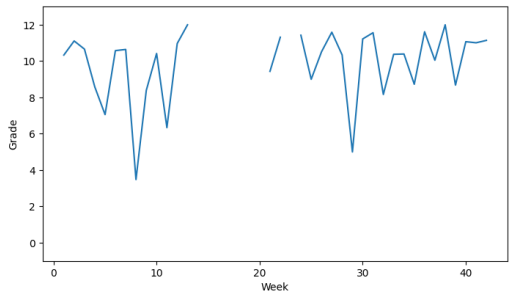

Figure 24: Time series with missing values.

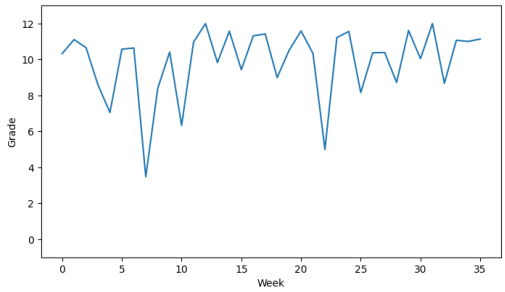

Figure 25: Time series with grades shifted to the left.

As one can see from Figure 24 and Figure 25, if there are missing grades for certain weeks, the following grades are shifted to the left to create a unified time series. The time series constructed in this way will be shorter than the one constructed using interpolation. The length of the final time series depends on the number of weeks without grades and so these time series may have different lengths. The theoretical maximum length is 53 which corresponds to a time series without missing values for each week of the year; this is very unlikely since only a few students have classes also during summer. We have deleted the time series with a length less than half of the maximum length (53 in this case) since it would be difficult to compare two time series with very different lengths.

For the experiments, we decided to use grades referred to the whole year and for all the topics ("MOD" grades included). At the end of this process, we have obtained 521 time series with lengths from 28 weeks to 46 weeks.

4.2 Data Preprocessing

The time series created could not be suitable for clustering purposes since they are quite long (around 35 values) and they present a lot of local fluctuations that could affect the clustering accuracy. For these reasons, two data preprocessing techniques can be applied to the time series before inputting them to the clustering techniques presented later in Chapter 5. The idea to apply these two techniques comes from Ishan Choudhary and Sampath Patel.

4.2.1 Data Smoothing

With the data smoothing our purpose is to flatten each time series to highlight the general trend, discarding the local fluctuations. We have considered two different types of data smoothing:

- Moving average (MA): Given a time series of size N and a specified parameter W (window size) it computes the mean of the first W point of the time series; this mean will be the first point of the smoothed time series. After that, it shifts the window of one position to the right and computes another mean and so on. In the end, we have a smoothed time series with size = N - W + 1. An example is shown in Figure 26 and Figure 27 where each point of the smoothed time series is the mean of W points of the original time series.
- Exponential moving average (EMA): Given a time series of size N and a specified parameter α (smoothing factor), for each point in the original time series it computes the weighted mean of that point and all the previous points, applying more weight to data

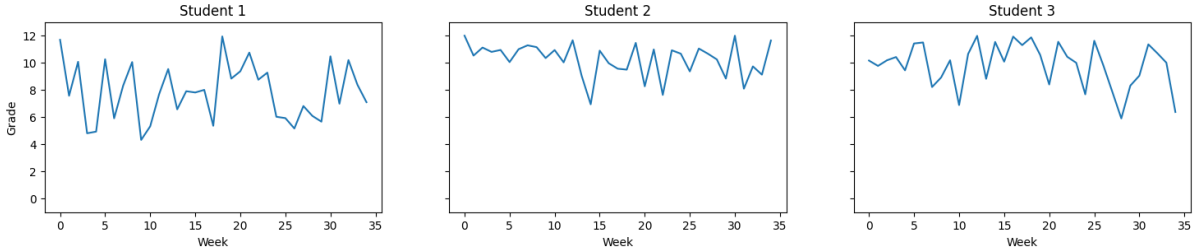

Figure 26: Unprocessed time series.

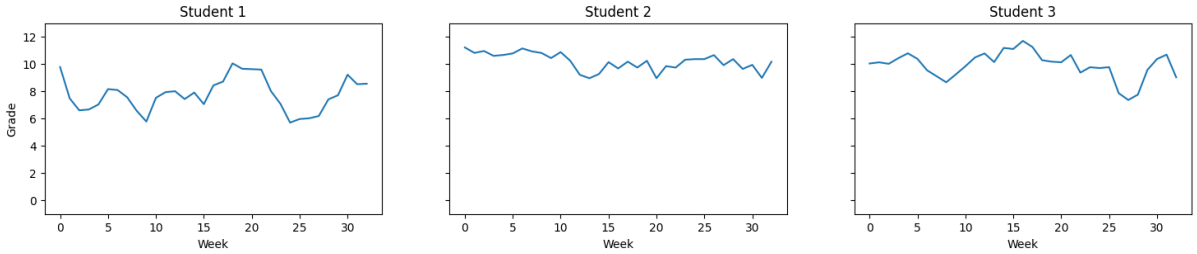

Figure 27: Time series after moving average with W = 3.

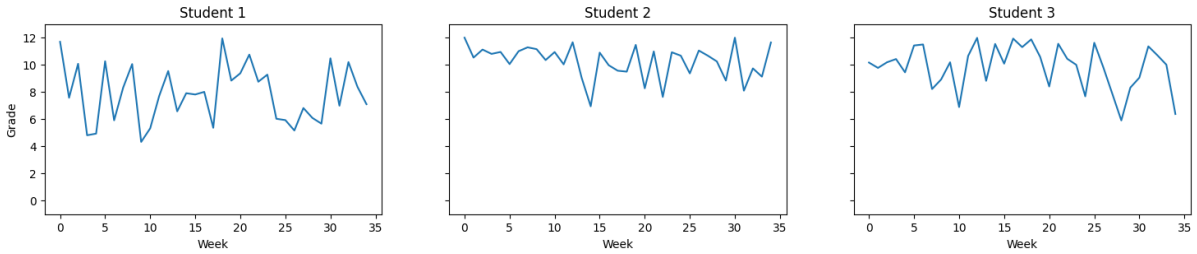

Figure 28: Unprocessed time series.

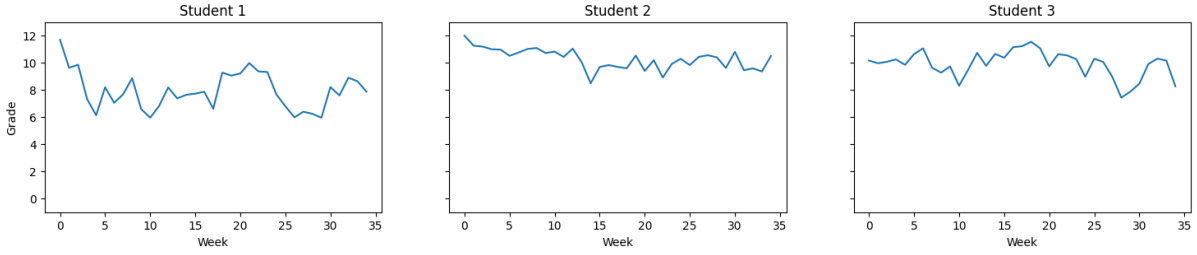

Figure 29: Time series after exponential moving average with $\alpha = 0.5$.

that is more current. If we want to create the smoothed time series y from the time series x, we have the following formulation

$$y_t = (1 - \alpha) * y_{t-1} + \alpha * x_t[31] \tag{4.3}$$

In the end, we have a smoothed time series with size = N. An example is shown in Figure 28 and Figure 29.

If we set a too-high W or too-low α value, we could obtain a too-smoothed time series and we could lose some important fluctuations. Possible values for W are 2 and 3 while for α are 0.5 and 0.75; the values W = 1 and $\alpha = 1$ mean no data smoothing. We will evaluate values of W and α empirically to find optimal values.

4.2.2 Delta Averaging

With the delta averaging, our purpose is to align vertically all the time series so that all have zero mean. In this way, it is possible to differentiate them by their shape and not by their absolute value. With this method, an "A" student will be clustered together with a "C" student since we are not interested in the absolute value but only in the shape of the time series. A possible drawback of this method is that the time series could become very similar to each other and so more difficult to cluster. An example is shown in Figure 30 and Figure 31. We will perform some experiments to discover if this technique is useful for our goal.

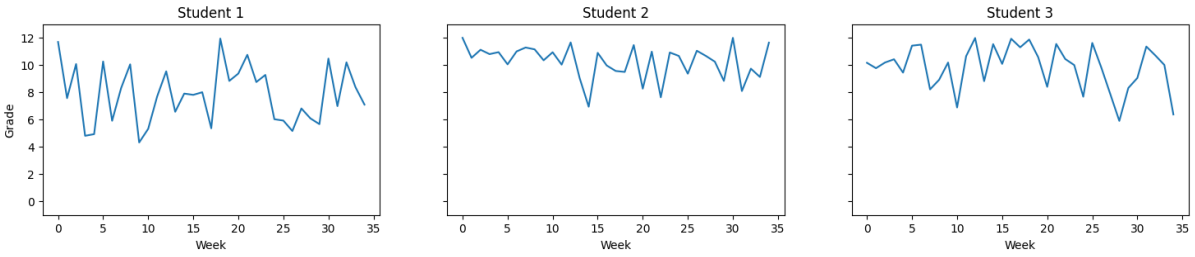

Figure 30: Unprocessed time series.

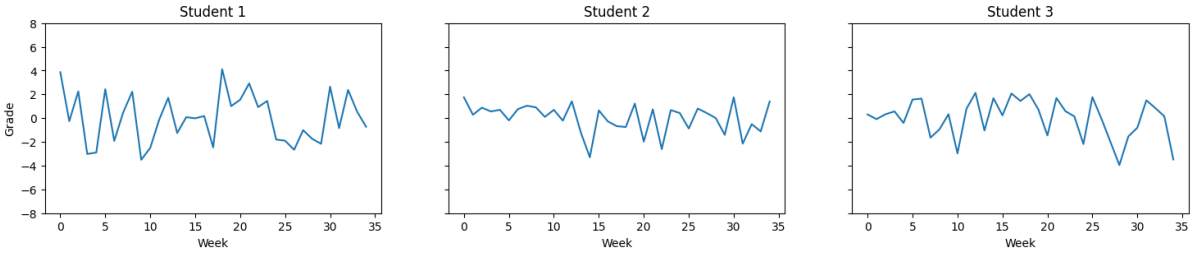

Figure 31: Time series after delta averaging.

CHAPTER 5

METHODOLOGY

In this chapter, we present the three main approaches that we have applied to solve our task and the metrics used to evaluate the clustering results. The goal of these approaches is to create 3 clusters with the following distribution: 70% "stable" students, 20% "monitor" students and 10% "critical" students. The main workflow of these approaches is the following: starting from the original time series (with different lengths for "clustering of time series with different lengths", with the same length for "classic clustering algorithms" and "density-based clustering with feature reduction"), data preprocessing is applied. The preprocessed time series will be given as input to the algorithms to find the label for each time series. These labels will be used to cluster the original time series, the resulting clusters will be evaluated using ad-hoc metrics (Section 5.4).

As you can see from Figure 32, the basic idea behind this pipeline is to preprocess the time series before feeding them into the algorithms to facilitate the clustering. Anyway, in the end, we seek to cluster the original time series 9in such a way that the labels generated by the algorithms are used to cluster the original time series. The resulting clusters are evaluated in terms of size and quality.

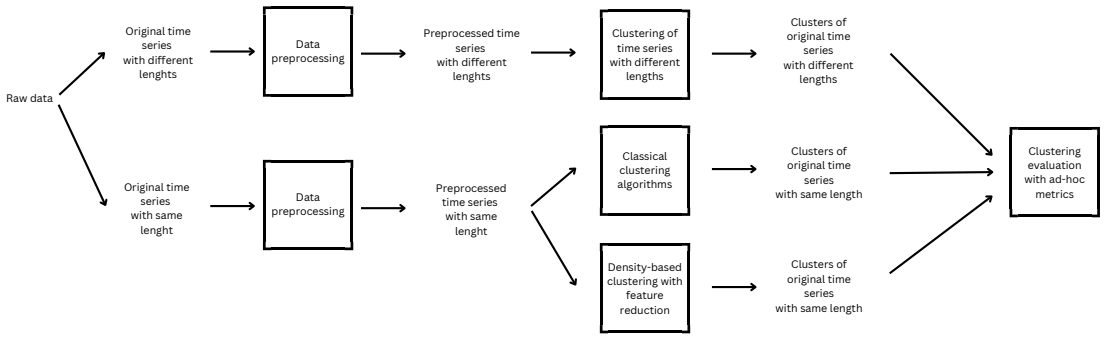

Figure 32: Main workflow.

5.1 Clustering of Time Series with Different Lengths

The first approach applied to our goal is the "clustering of time series with different lengths", in particular using the K-means algorithm. The choice to use time series of different lengths is due to many reasons; first of all, some students could not have grades for a certain week because they were absent or their grades were registered the week after. To sidestep this problem, we decided to use only the data available without interpolating the missing values. In this way, we are more confident about the quality of our data because all data in the time series are derived directly from the raw data and no data is generated artificially by interpolation. The time series used for this approach are created following the process described in Section 4.1.3. Some experiments were performed by applying different data preprocessing techniques to the original time series before inputting them into the K-means algorithm. In this way, we seek to

find which preprocessing technique is the best one for our goal.

5.1.1 K-means with DTW

To cluster time series with different lengths we have used the K-means algorithm because it allows us to specify the number of clusters to create and the distance metric to use. In particular, we have used time series K-means, the variant of the algorithm adapted for time series data.

In the algorithm, when a data point is selected, the distance between the data point and all the cluster centers has to be computed to find the closest cluster center. Usually, the Euclidean distance is used to compute the distance between two data points; it calculates the distances between each pair of corresponding components of the data points. This formulation of the distance supposes that the data points have the same number of components; this is not our case since our data points are time series with different lengths. To address this problem we used Dynamic Time Warping (DTW), a distance metric designed for time series that allows the comparison of data points with different sizes.

DTW computes the distance between two time series, for example A and B, by aligning the time series such that to each element of A corresponds at least an element in B. Since the time series may have different lengths, an element of a time series could correspond to more elements in the other time series. DTW will expand the shorter time series to match it to the longer one. The mapping between the elements of A and B is decided such that the distance is minimized. DTW is very suitable as distance metric for time series clustering since it can match sequences

at different timestamps. Moreover, it is able to deal with time series of different lengths (as we can see in Figure 33), although it can also be used for time series with the same length.

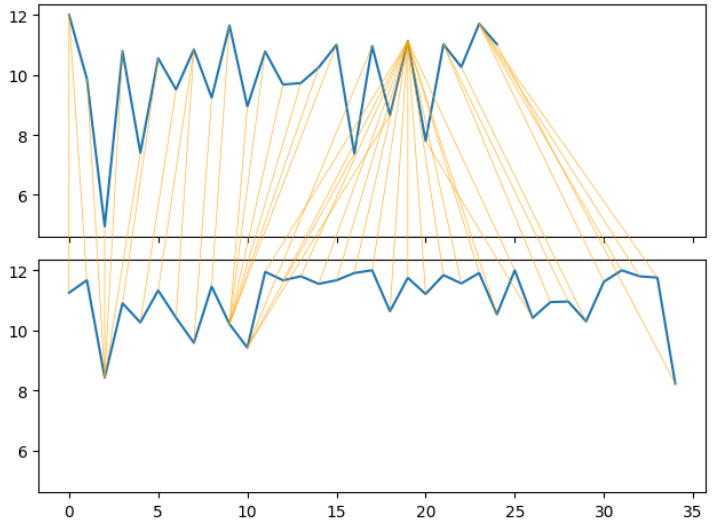

Figure 33: DTW with time series of different lengths.

Using DTW as distance metric, we are able to apply the K-means algorithm to our time series to form 3 clusters. Using DTW the algorithm should be able to group together the time series with common sequences, such as big fluctuations or steep drops. There is no theoretical assurance on the dimension of the clusters formed by K-means because we can not give such information as an input to the algorithm. This could be a problem given our goal to constrain

the dimension of the clusters.

5.1.2 Classify Clusters

After clustering with the K-means algorithm, the problem is to associate the clusters with the classes that we have defined for our task ("critical", "monitor" and "stable"). The "critical" cluster should correspond to time series with significant fluctuations and an evident downward trend. The "monitor" cluster should correspond to time series with a moderate downward trend. The "stable" cluster should correspond to time series with a steady or upward trend. We can postulate that the classes can be distinguished by an average grade of the time series belonging to that class ("critical" will have the lowest mean grade while "stable" will have the highest mean grade).

Since the distance between two time series is computed using DTW, the mean of the time series in each cluster is also computed using DTW to address the problem of time series with different lengths. To generate the time series that represent a whole cluster we have used DTW Barycenter Averaging (DBA), the most robust averaging method based on DTW. DBA is an averaging method used to find an average time series that minimize the DTW distance from the other time series of the cluster.

As you can see from Figure 34, the blue time series are generated using DBA that calculates the average of all the time series belonging to that cluster. The DBA time series will be used as the representatives of each cluster. We could have used the cluster center computed by K-means as the representative of each cluster, but the cluster center is calculated on the preprocessed time

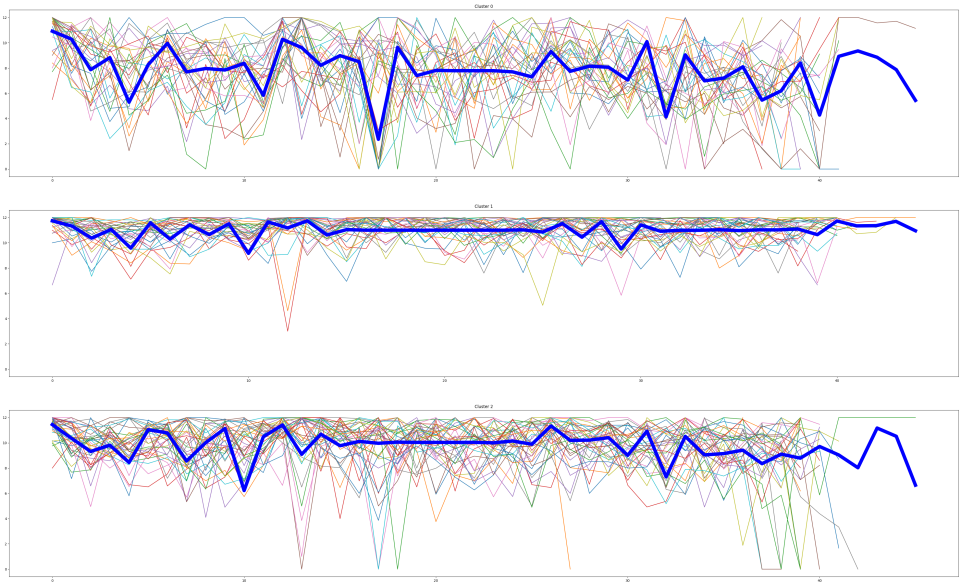

Figure 34: DBA representative for time series with different lengths.

series while we want to cluster the original time series. For this reason, we need to recalculate the cluster centers with DBA on the original time series and use them as representatives of the clusters.

After we calculate the representatives of all the clusters, we can simply compute, for each of them, the mean of the points. After that, we assign the "critical" label to the cluster with the representative with the lowest mean, the "stable" label to the cluster with the representative with the highest mean and the "monitor" label to the remaining cluster. Then we calculate the size of each class and compare it with the expected size. It is unlikely that the values will match the expected dimensions because cannot give any information about the cluster size to the algorithm. To evaluate the quality of the clusters, instead, we have developed two different

metrics discussed in Section 5.4.

5.2 Classical Clustering Algorithms

Another approach to our problem is to cluster students using the state-of-the-art "classical clustering algorithms" for time series. The purpose of clustering using different algorithms is stated in the "Concluding remarks" of *A benchmark study on time series clustering* [2]. We try to create different clustering, assess them and select the one that is more accurate for our goal. Moreover, the same algorithm can be applied using different combinations of parameters and observe how the parameters change the resulting clusters. Different kinds of clustering algorithms should be applied to discover which one outperforms the others:

- Partitional clustering: Decompose the dataset into a set of different clusters given the number of clusters as input. K-means and K-medoids are the algorithms of this kind that we have applied.
- Hierarchical clustering: The objective is to create a hierarchy of clusters. For this kind, we applied Hierarchical Agglomerative clustering.
- Shape-based clustering: Cluster data together based on their shape. We applied K-shape as Shape-based algorithm for time series.
- Density-based clustering: Cluster together regions with high data point density. DBSCAN is the Density-based algorithm that we have used.
- Spectral clustering: Use dimensionality reduction techniques before clustering data in low dimensional space.

These algorithms use different distance metrics to compute the distance between data points and, in most cases, the distance metrics can only deal with data points with the same dimension. Since our data points are time series, we used time series with the same length as input to the algorithms. The times series are created using interpolation as explained in Section 4.1.2.3. For each algorithm, some experiments will be performed by applying different data preprocessing techniques to the original time series before inputting them into the algorithms. In this way, we seek to find which preprocessing technique is most appropriate for our goal. If available, the variant of the algorithm adapted for time series data is used (e.g. time series K-means).

5.2.1 Partitional Clustering

Partitional clustering is the most common clustering technique. It has not been designed to work with time series but it is also suitable for this task. Given the number of clusters K in input, it partitions the data in K disjoint clusters such that each cluster contains at least one point and each point belongs to one cluster. This technique allows us to specify the number of clusters; this is an advantage for our goal since we are bound to create exactly 3 clusters. Moreover, it is possible to specify the distance metric to use so that we can evaluate which one is the best for our goal.

K-means and K-medoids are the most used Partitional algorithms; we tried both. The two algorithms are very similar, they only differ in how the cluster centers are updated. In the K-means algorithm, the cluster centers are updated calculating the mean of the points in each cluster. In the K-medoids algorithm, the cluster centers are updated calculating the medoid of

each cluster. The medoid is an actual point of the cluster with the minimum sum of distances to other points in the cluster, for this reason K-medoids is more robust to outliers than K-means.

TABLE IV: PARTITIONAL CLUSTERING EXPERIMENTS

Algorithm	Number of clusters	Distance metric	Averaging method	Initialization
K-means	3	DTW	DBA	K-means++
K-means	3	Euclidean	Mean	K-means++
K-medoids	3	DTW	DBA	K-means++
K-medoids	3	Euclidean	Mean	K-means++

As we show in from Table IV, we have performed some experiments both with DTW and Euclidean distance as distance metrics; the number of clusters is fixed at 3. In the K-means algorithm, DBA is used as averaging method when DTW is the distance metric while a simple mean is used as averaging method with Euclidean distance as distance metric. In K-medoids, the medoid computation is always used as averaging method. For both algorithms, K-means++ (a simple randomized seeding technique that can improve the accuracy of K-means and make it less sensitive to initialization) is used as the initialization method of the cluster centers.
We expect DTW to outperform Euclidean distance since the first has been designed specifically to align time series that show common sequences, even if they are at different times. In this way, it should return a lower distance than Euclidean distance that compares the time series

without taking into account the time dimension.

5.2.2 Hierarchical Clustering

Hierarchical clustering generates a hierarchy of clusters. In particular, in this work, we have applied a Hierarchical Agglomerative algorithm that uses a bottom-up approach to construct the hierarchy. In the beginning, each data point is a different cluster and, at each iteration, the two closest clusters (identified using certain criteria) are merged until there is only one cluster left.

Before applying Hierarchical clustering we need to calculate the distance matrix that contains the distances between each couple of data points. To compute the distance matrix we have considered two distance metrics: Euclidean distance and DTW. After that, the linkage matrix is computed starting from the distance matrix based on certain criteria: single, complete, average and ward. The linkage matrix will be used by the algorithm to decide which clusters merge at each step. One of the characteristics of Hierarchical clustering is that you do not have to specify the number of clusters but you can extract how many clusters you need from a so-called dendrogram. The dendrogram is a tree-like diagram that shows the relationships between the clusters.

We have performed some experiments with this algorithm extracting always 3 clusters from the dendrogram but changing the distance metric and the linkage criteria as you can see from Table V. Also in this case we expect that DTW outperforms Euclidean distance as distance metric since the first has been designed specifically for time series. Regarding linkage criteria,

TABLE V: HIERARCHICAL CLUSTERING EXPERIMENTS

Number of clusters	Distance metric	Linkage criteria
3	DTW	Single
3	DTW	Complete
3	DTW	Average
3	DTW	Ward
3	Euclidean	Single
3	Euclidean	Complete
3	Euclidean	Average
3	Euclidean	Ward

the average and ward methods are the more promising because are less affected by noise; moreover, they consider all the points in each cluster when merging two clusters.

5.2.3 Shape-based Clustering

Shape-based clustering relies on the idea to group together data points with the same shape and, for this reason, it is particularly suitable for time series clustering. The most famous Shape-based clustering algorithm is K-shape, a K-means variant that uses a Shape-based measure to compute the distance between two time series. This measure is able to consider the shape of the time series and recognize time series that exhibit similar sequences.

K-shape takes as input parameters only the number of clusters to produce and the time series. For this algorithm, there are no parameters to choose so only one experiment with 3 clusters can be performed. One characteristic of this algorithm is that it uses internally a normalized cross-correlation measure in the Shape-based measure. The cross-correlation is able to align two time

series thereby achieving shift-invariance; however, this method requires that the time series be normalized (i.e., with mean = 0 and variance = 1) before giving them as input to the algorithm. Normalizing the time series is like applying delta averaging and data smoothing together. This could make it more difficult to cluster the time series since they become very similar to each other as you can see in Figure 35 and Figure 36. Beyond the normalization problem, K-shape is invariant to scaling and shifting. This could be a problem since the "critical" and "monitor" time series have lower grades with respect to the "stable" ones so K-shape could have difficulties distinguishing the clusters.

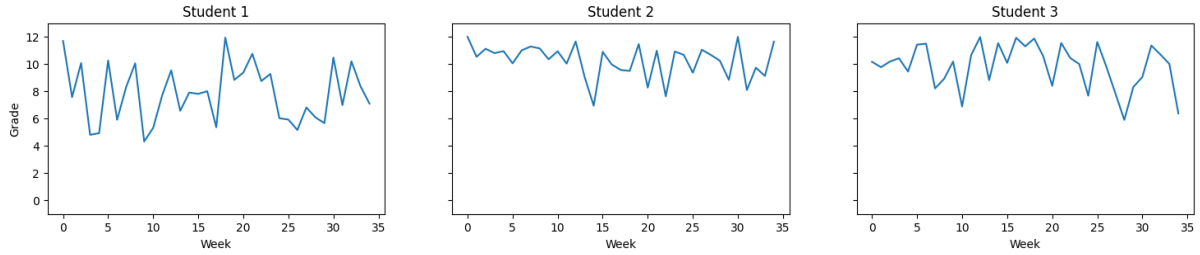

Figure 35: Unprocessed time series.

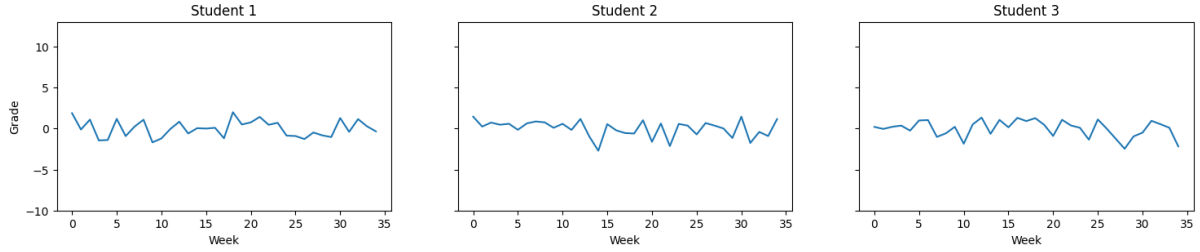

Figure 36: Normalized time series.

5.2.4 Density-based Clustering

Density-based clustering seeks to cluster regions of high data point density that are separated by regions of low data point density. This type of clustering generates clusters with arbitrary shapes that do not assure within-cluster similarity because it only relies on the concept of density region. This type of clustering has not been designed for time series but it can be applied to time series clustering. However, it is difficult to visualize how it works since the dimensionality is high. The number of clusters is not needed in input because it is decided by the algorithm.

We have applied DBSCAN, one of the most used Density-based clustering algorithms. This algorithm relies on the concept of neighbourhood to find core points and from these points form the clusters. The core points are defined using "eps" and "min_samples" values. By varying these parameters we can change significantly the shape and size of the clusters. Another feature of DBSCAN is that it is able to identify outliers during the clustering. This could be very useful for our goal since the students that we are trying to identify could correspond to outliers.

We have performed some experiments with DBSCAN using both Euclidean distance and DTW as distance metrics, we have also changed the values of "eps" and "min_samples" to form 3 clusters with high time series density. Also in this case, we expect DTW to outperform Euclidean distance. The main problem with this algorithm is the high dimensionality of our time series, which could make it difficult to find regions of high point density.

5.2.5 Spectral Clustering

Spectral clustering performs data reduction before applying a clustering algorithm to the low dimensional representation of the data to form the clusters. The low dimensional representation is obtained by calculating the eigenvectors of the graph Laplacian matrix (affinity matrix) derived from the similarity graph. This technique is very useful to find non-convex clusters but it is memory and time-consuming. Nevertheless, we expect that it will perform well toward our goal due to the high dimensionality of our data. Indeed, clustering a low-dimensional representation would make it easier to find well-separated clusters.

We have performed some experiments with Spectral clustering to form 3 clusters. We have applied K-nearest neighbours to construct the affinity matrix and different strategies to assign labels in the low dimensional space as one can see in Table VI. K-means uses the K-means algorithm to cluster the reduced data, Discretize uses normalized cuts on the graph to cluster the reduced data while Cluster qr extracts clusters directly from the eigenvectors.

TABLE VI: SPECTRAL CLUSTERING EXPERIMENTS

Number of clusters	Affinity matrix	Assign labels
3	K-nearest neighbors	K-means
3	K-nearest neighbors	Discretize
3	K-nearest neighbors	Cluster qr

5.2.6 Classify Clusters

After generating the cluster with one of the classical algorithms we associate the clusters with the classes that we have defined for our task ("critical", "monitor" and "stable").

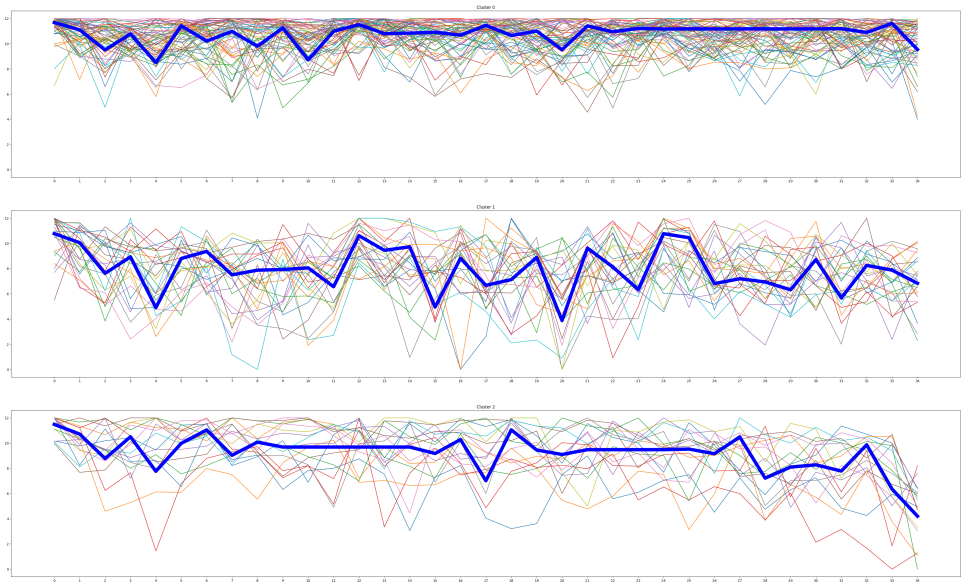

Figure 37: DBA representative for time series with the same length.

The same approach presented in Section 5.1.2 is applied here with DBA used to calculate the representative of each cluster. In this case, since the time series have the same length, we could have used a simple mean over all the time series in each cluster to calculate the representative of the cluster. However, we decided to stick with DBA for a fair comparison of the results obtained here with the results obtained by clustering of time series with different lengths

(Section 5.1). As you can see from Figure 37, the blue time series are generated using DBA which calculates the DTW average of all the time series belonging to that cluster.

When we calculate the representatives of all the clusters using DBA, we can simply compute, for each of them, the mean of the points. After that, we can assign the "critical" label to the cluster with the representative with the lowest mean, the "stable" label to the cluster with the representative with the highest mean and the "monitor" label to the remaining cluster. After we assign the classes to the clusters generated by the algorithm, we calculate the size of each class and compare it with the desired sizes. It is unlikely that the dimensions of the clusters will match their expected dimension because we cannot give any information about the cluster size to the algorithm. To evaluate the quality of the clusters, we have developed two different metrics which will be presented in Section 5.4.

5.3 Density-based Clustering with Feature Reduction

The last approach that we present has been specifically designed for our goal. This approach combines the idea of Spectral and Density-based clustering with hyperparameter tuning. From Spectral clustering, it takes the idea to reduce the dimensionality of the data before inputting them into a clustering algorithm. The data reduction in this case is performed with Principal Component Analysis (PCA). The link with Density-based clustering is because DBSCAN is used as clustering algorithm. Indeed, DBSCAN can identify outliers and it is the only algorithm for which we can tune the input parameters ("eps" and "min_samples") to force the size of the resulting clusters.

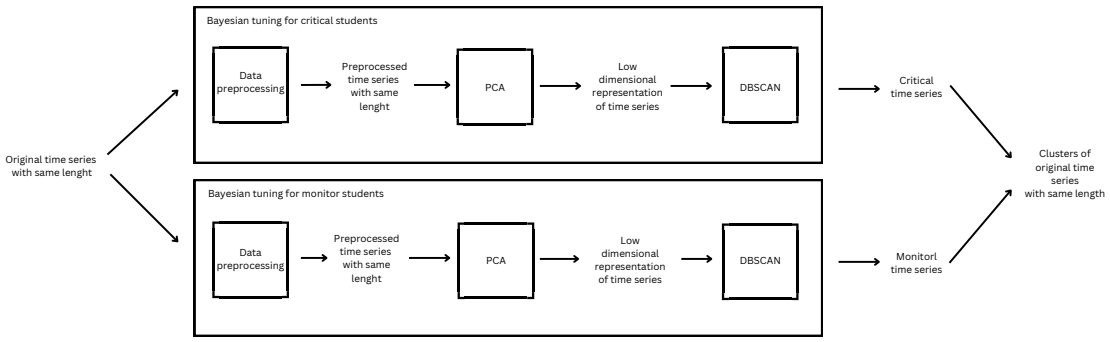

Figure 38: Density-based Clustering with Feature Reduction.

Figure 38 shows the same model applied twice, the first one to identify "critical" time series and the second one to identify "monitor" time series. The model is made of three main components:

- PCA: Feature reduction technique used to downscale the time series to a lower dimension.
- DBSCAN: Density-based clustering algorithm used to cluster the low dimensional data points through outlier detection.
- Bayesian Tuning: The whole process is encapsulated inside an optimizer that uses Bayesian principles to tune the hyperparameters to reach the desired cluster size.

The major constraints for our goal are finding time series with big fluctuations and descending trends and the size of the resulting cluster. The first constraint is resolved using DBSCAN to identify outliers in the data points. The second is accomplished by performing fine-tuning on the hyperparameters of the model. This tuning process is feasible due to PCA which reduces the dimensionality of the data and makes the clustering phase quick.

5.3.1 Principal Component Analysis

PCA is a dimensionality reduction technique that is able to extract information from a high-dimensional space by projecting it into a low-dimensional sub-space. PCA breaks down the data into principal components that hold most of the variance (information) of the data. For this reason, it is important to apply normalization (mean = 0 and variance = 1) before performing PCA. Each principal component represents a percentage of the total variation captured from the

data. We can represent the original data by considering only the first N principal components that hold the major part of the information.

The main reason to apply a data reduction technique in this context is to make the time series more suitable for clustering. Moreover, the clustering applied to low-dimensional data is much faster than applied to the original data. In this way, it is feasible to use a Bayesian tuning that repeats the clustering phase multiple times to find the best combination of hyperparameters.

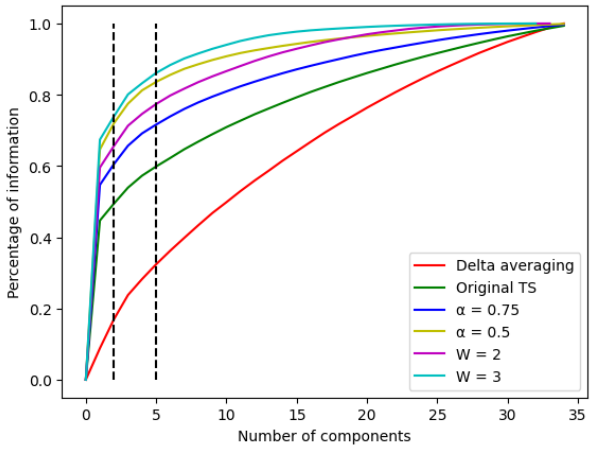

Figure 39: Evaluation of data preprocessing with PCA.

An important step is to decide the number of principal components that will represent the low-dimensional data. We want that our reduced data have low dimensionality to speed up the clustering phase and allow us to tune the hyperparameters. At the same time, the percentage of variance in the reduced data should be high enough to ensure a good clustering quality. To

make this decision, in Figure 39 we show the percentage of variance contained in the reduced data with respect to the number of principal components considered. Each line represents a different data preprocessing technique applied to the time series.

As shown in Figure 39, if we apply delta averaging (red line) to the time series we lose variance in the first principal components with respect to the original time series (green line) without data preprocessing. However, if we apply data smoothing the percentage of variance in the first principal components increases. For these reasons, a good choice for the number of principal components could be between 2 and 5 (or more) to ensure a percentage of variance higher than 60% applying data smoothing to the data.

5.3.2 Density-based Clustering for Outlier Detection

After applying PCA, the time series has been transformed into low-dimensional points that can be more easily clustered using classical clustering algorithms. If we apply PCA with only 2 components to the original time series we obtain a dataset of two-dimensional points that can be represented on a Cartesian plane as shown in Figure 40.

Figure 40 shows the distribution of the data in two dimensions making it possible to identify a region of high-density points surrounded by other points that can be identified as outliers. The motivation for using Density-based clustering and, in particular, DBSCAN comes from the distribution of the data. With DBSCAN our aim is to identify 2 clusters, one big cluster that encloses the high-density region and another cluster of the remaining outliers. DBSCAN is really suitable in this situation since it is able to identify outliers while clustering the data.

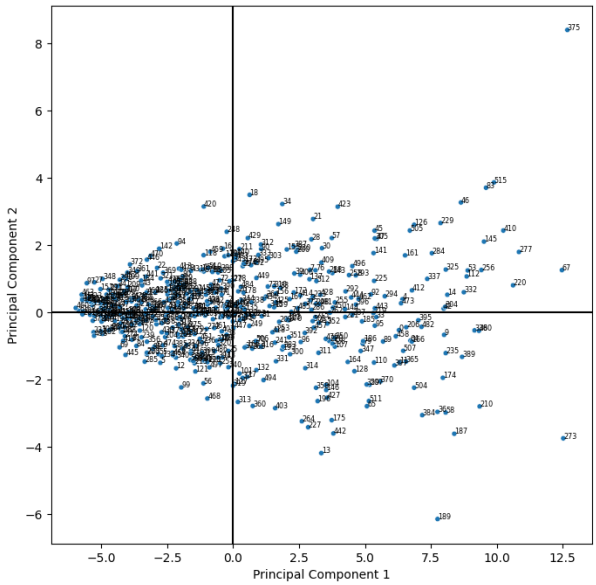

Figure 40: 2D representation of the time series.

Moreover, we can change the size and the shape of the generated clusters by changing the parameters “eps” and “min_samples that are used to determine the core points.

Since the principal components of PCA capture most of the variance, the time series with similar trends will be placed closer in the low dimension while different time series (such as with big fluctuations or descending trends) will be placed more spaced out. If we translate this characteristic to the distribution of the data shown in Figure 40, we can assume that the high-density region is made of time series that should not be flagged because they are very similar to each other. On the contrary, outliers are points differing from the others, meaning that they could represent time series with big fluctuations or descending trends that we want to flag. The problem of flagging “critical” and “monitor” time series becomes finding the right

amount of outliers using DBSCAN on low-dimensional data.

5.3.3 Bayesian Tuning

The most challenging constraint in this work is to form clusters with a specific size, which means that we have to find a way to decide the dimension of the clusters and not leave the decision to the clustering algorithm. For this purpose, DBSCAN is the most suitable clustering algorithm because it has two parameters ("eps" and "min_samples") that control the number and the size of the resulting clusters. The other algorithms presented in Section 5.2 do not have this important feature.

To form clusters with specific sizes we need to tune the two parameters of DBSCAN but also the number of principal components to consider for PCA. We have noticed that, if we fix the number of components of PCA, it could be difficult to find the right size of the clusters. To tune these hyperparameters a grid search is unfeasible since the combination of the hyperparameters is too high; moreover, a random search is not reliable.

To optimize hyperparameters we have used Bayesian tuning, which constructs a model to approximate the performance of the hyperparameters. Through an objective function, the model maps a combination of hyperparameters to a loss function that we would like to optimize (in our case the size of the clusters). The hyperparameters that we want to optimize are the following:

- Number of components of PCA: This will determine the size of the low-dimensional data. Possible values are from 2 to 5.

- "eps": Parameter of DBSCAN that represents the maximum distance between two points for one to be considered in the neighbourhood of the other. The range of possible values depends on the experiment but usually falls between 0.5 and 5.
- "min_samples": Parameter of DBSCAN that represents the number of samples in the neighbourhood of a point to be considered a core point. The range of possible values depends on the experiment but usually falls between 2 and 20.

The model, at each step, takes as input the search space of the hyperparameters and the experimental history, and returns which set of hyperparameters to try next. In this way, the objective function is executed a few times until the loss function is minimized. This process is feasible due to the low dimensionality of the data which makes the clustering quick. If we had applied Bayesian tuning on the time series without data reduction, it would have been unfeasible due to the time taken by computing the objective function. Moreover, DBSCAN could struggle to form good clusters due to the high dimensionality. The operations performed in the objective function are explained in Algorithm 3.

In the objective function we are evaluating if, given a set of hyperparameters, we can form two clusters, one with high density and one with the outliers. In that case, we calculate the difference between the size of the outliers and the desired size that represents our loss function. Note that, if the desired size is greater than the actual size of the outliers, the loss function is maximized (return infinite) since we want the size of the outliers to be greater or equal to the desired size.

input : Original time series, percentage of outliers to find, hyperparameters
output: Original time series corresponding to outliers

Apply data preprocessing techniques to time_series;
Normalize the values of time_series;
Apply PCA given number_of_components;
Apply DBSCAN to the low-dimensional data given eps and min_samples;
if *number of clusters == 2* **then**
    target = number of time_series * percentage_outliers;
    if *target > number of outliers* **then**
        return inf;
    else
        return number of outliers − target;
    end
else
    return inf;
end

Algorithm 3: Pseudocode of the objective function.

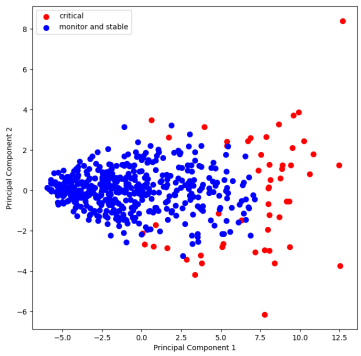

Figure 41: Clustering with 10% outliers.

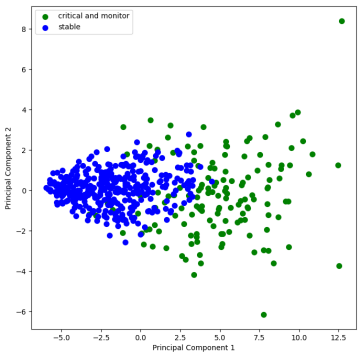

Figure 42: Clustering with 30% outliers.

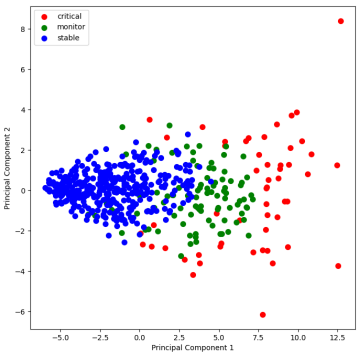

Figure 43: Final clustering in two dimensions.

To discern between “critical” and “monitor” students the whole tuning process is repeated two times. The first one to identify at least 10% of points as outliers that correspond to “critical” students (Figure 41). After that we calculate the exact number of “critical” time series identified and the process is done again to identify at least other 20% of points as outliers (Figure 42). In this way, we are enlarging the cluster of outliers to identify also the “monitor” students. The green cluster of Figure 42 contains both “critical” and “monitor” time series so, from these, we delete the “critical” time series flagged before and we obtain the “monitor” time series that are at least 20% over the total. In (Figure 43) you can see the final clustering of reduced data. This technique can be adapted to every percentage of “critical” and “monitor” students that we want to flag and could be also extended to more classes.

5.4 Evaluation Metrics

To evaluate the resulting clusters we are only allowed to use internal metrics since we do not have the true labels of the data. The internal metrics evaluate if the clusters are dense and well-separated. These characteristics are not suited to our goal since we seek to form 3 clusters, two of which will not be dense and well-separated. Indeed, the "critical" and "monitor" clusters will be made of time series very different from each other because students could show big fluctuations or descending sequences at different times during the year.

Among the internal metrics, we have considered only the Silhouette score for our analysis since it is simple and bounded. In this way, we can compare our results with a well-known metric. The Silhouette is composed of two scores, the closeness of points in the same cluster and the distance of points of different clusters; both values are computed using DTW as distance metric because it is suitable for time series. The Silhouette score is bounded between -1 for incorrect clustering and +1 for highly dense clustering; we can know how good is our clustering by evaluating how close is the score to 1.

Since classical internal metrics are not suitable for our task, we have developed two other metrics to evaluate the quality of the clusters:

1. Cohesion and Coupling: For each data point, we calculate the distance between the point and the other points in the same cluster (Cohesion within clusters) and the distance between the point and the points in the other two clusters (Coupling among clusters). After that, we calculate the mean of Cohesion and Coupling of each point in the same cluster and we obtain the Cohesion and Coupling for each cluster. In the end, we make

the reciprocal to obtain the final Cohesion and Coupling for each cluster as explained in Algorithm 4. Since this computation could be expensive in time we have used FastDTW [23] as distance metric, an approximated version of DTW that has linear complexity in time and space.

```
input  : Time series and labels of the clustering
output: Cohesion and Coupling score for each cluster
Cohesion = [];
Coupling = [];
for l ← each different label do
    X = [];
    Y = []; for i ← from 0 to length(time_series)-1 do
        if labels [i] == l then
            for j ← from i to length(time_series) do
                if labels [j] == l then
                    Append to X FastDTW(time_series [i], time_series [j]);
                end
            end
            for j ← from 0 to length(time_series) do
                if labels [j] ! = l then
                    Append to Y FastDTW(time_series [i], time_series [j]);
                end
            end
        end
    end
    Append to Cohesion the mean of X;
    Append to Coupling the mean of Y;
end
Cohesion = 1/Cohesion;
Coupling = 1/Coupling;
```

Algorithm 4: Pseudocode to calculate Cohesion and Coupling.

The idea behind this metric is very similar to Silhouette. Indeed, Cohesion represents the closeness of points in the same cluster, so we want it to be high (i.e., with low distances). Coupling represents the distance of points of different clusters, so we want it to be low (i.e., with high distances). In the end, we will have six scores, two scores for each cluster. For Cohesion, we expect that the "critical" cluster has the lowest score because it should be made of time series very different from each other. The "stable" cluster, instead, should have the highest score since it is made of similar time series. For Coupling, we expect that the "monitor" cluster has the highest score since it should be made of time series in the middle between "critical" and "stable". Instead, the other two clusters should have a lower score since they are made of time series different from the other clusters.

2. Time Series Accuracy: Since our goal is to identify time series with big fluctuations and descending trends, we need to find a way to verify if the "critical" and "monitor" time series that we are flagging have these characteristics. The idea is to find algorithmically the time series that we would like to flag and verify if these time series are present in the "critical" and "monitor" clusters that the algorithm has formed. First of all, we calculate the number of time series present in the "critical" and "monitor" clusters (N for example). After that, we calculate, for each time series, a number that depends on the characteristics that we want to evaluate (e.g. big fluctuations or descending trends). We sort the time series based on the numbers calculated and we check how many of the first N time series are present in the "critical" or "monitor" cluster. This measure is highly affected by the size of the "critical" and "monitor" clusters. Indeed, if they contain a lot of time series,

it is easier for the time series that we want to flag to be present in these clusters. We aim to obtain a high Time Series Accuracy while controlling the size of the clusters.

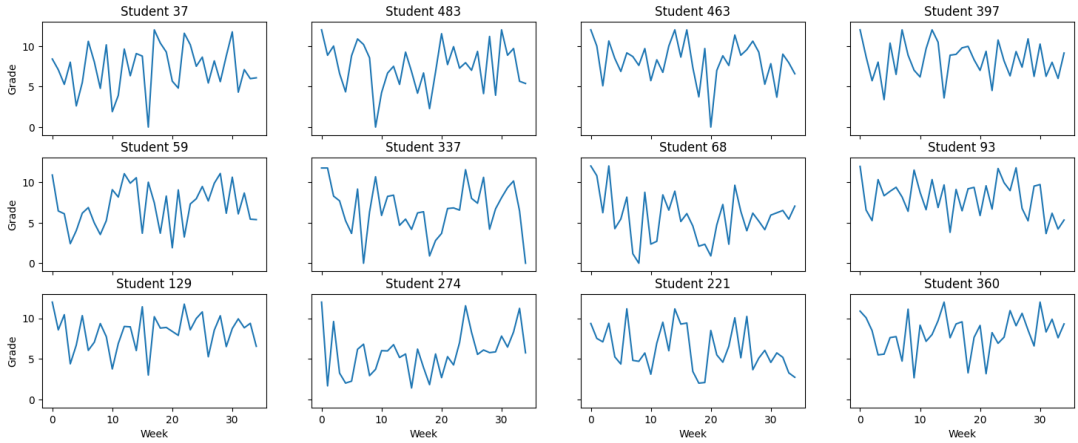

Figure 44: Time series with big fluctuations and steep drops.

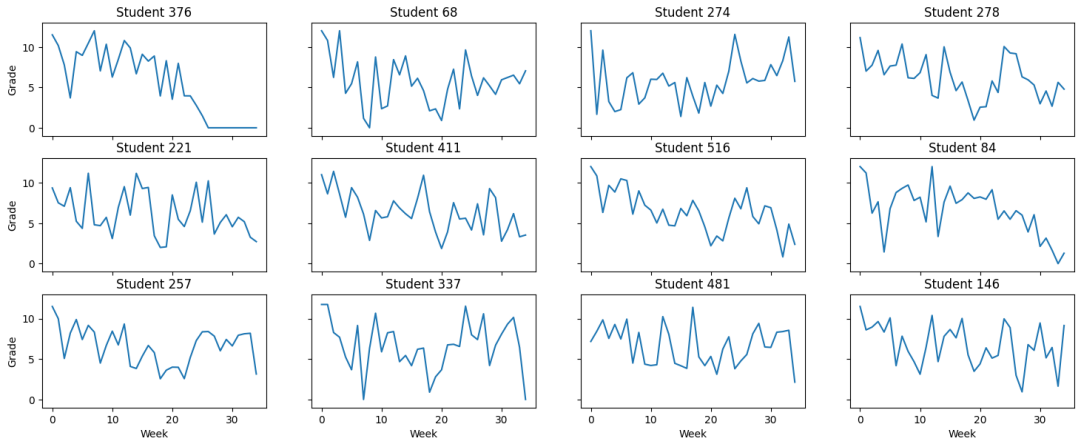

Figure 45: Time series with descending trend.

To find the time series with big fluctuations we calculate, for each time series, the sum of the differences between two consecutive points A and B if B $<$ A. After that, we calculate if the time series with the highest sums are present in the "critical" or "monitor" cluster and we obtain the "fluctuations accuracy" (FA). In this way, we want to identify time series with big fluctuations and steep drops as shown in Figure 44.

To find the time series with decreasing trend we calculate, for each time series, the mean of the values of the time series. After that, we calculate if the time series with the lowest means are present in the "critical" or "monitor" cluster and we obtain the "descending accuracy" (DA). In this way, we want to identify time series with an overall descending trend as shown in Figure 45.

CHAPTER 6

RESULTS

In this chapter, we present the results of the experiments performed on our three techniques. In particular, "clustering of time series with different lengths" and "classic clustering algorithms" will be used as a baseline (Section 6.2) while "density-based clustering with feature reduction" is the final model that we really want to evaluate (Section 6.3). Our goal is to achieve a Time Series accuracy similar to the baseline while controlling the size of the clusters. For each technique, we have performed different experiments using different combinations of parameters specific to the algorithm used. To compare these techniques we have evaluated different aspects:

- Size of the clusters: We compute the size of the "critical" and "monitor" clusters to compare them with the expected values.
- Cohesion and Coupling: We compute the Cohesion and Coupling values for each cluster. We want Cohesion to be high while Coupling to be low.
- Time Series Accuracy: For each clustering, we compute the "fluctuations accuracy" (FA) and "descending accuracy" (DA) to assess the quality of the clusters. We want both values to be close to 1.
- Silhouette: For each clustering, we compute the Silhouette score even if this metric may or may not be suitable for our task. We want it to be close to 1.

- Runtime: For each clustering, we compute the runtime of the algorithm. This measure is not really fundamental but it could give an idea of the time needed to produce the results.

Our goal is quite ambitious since our time series are quite long and potentially include multiple fluctuations. Our algorithms are challenged to form dense and well-separated clusters. Moreover, the sequences that we want to identify (e.g. big fluctuation and descending trends) could appear at different locations in the time series so it could be not very easy to cluster these time series.

6.1 Data preprocessing evaluation

To evaluate the three different techniques fairly, it is important to fix the data preprocessing techniques applied to the times series before clustering. To do so we have performed some experiments applying different data preprocessing techniques to the time series. After that, we have clustered the time series using K-means with both DTW and Euclidean distance as distance metrics. The preprocessing techniques evaluated are the following: no data preprocessing, delta averaging, data smoothing by Exponential Moving Average (EMA) with $\alpha = 0.5$ and data smoothing by Moving Average (MA) with W = 3.

TABLE VII: DATA PREPROCESSING EVALUATION

n.	Delta Av.	Data Sm.	W/α	Distance Metric	FA	DA	Silhouette	Runtime
1	False	no		DTW	0.89	0.94	0.21	9.38
2	True	no		DTW	0.83	0.81	0.24	8.32
3	False	EMA	0.5	DTW	0.90	0.95	0.19	19.55
4	**False**	**MA**	**3**	**DTW**	**0.91**	**0.96**	**0.16**	**10.35**
5	False	no		euclidean	0.89	0.99	0.21	1.96
6	True	no		euclidean	0.61	0.62	0.16	1.5
7	False	EMA	0.5	euclidean	0.89	0.98	0.21	2.22
8	**False**	**MA**	**3**	**euclidean**	**0.89**	**0.99**	**0.21**	**1.85**

Table VII shows the experiments performed to evaluate the data preprocessing techniques. These techniques are evaluated based on FA which represents the accuracy for time series with big fluctuations, DA which represents the accuracy for time series with descending trends. In

Table VII are also reported the Silhouette score and the runtime of the algorithm for each data preprocessing technique.

The technique with the lowest values of FA and DA is delta averaging (Rows 2 and 6). This is due to the fact that the time series with delta averaging all have zero mean, so they are very similar and difficult to distinguish. If we apply delta averaging we obtain lower accuracies than without applying any preprocessing technique (Rows 1 and 5). The technique with the highest values of FA and DA is data smoothing by MA (Rows 4 and 8), this technique has accuracies little higher than data smoothing by EMA (Rows 3 and 7).

If we consider the findings in Figure 39 and the results in Table VII, the only two possible data preprocessing techniques are data smoothing by MA or EMA. EMA has the advantage that it does not change the size of the time series after smoothing while MA shortens the time series depending on the window size. The biggest advantage of MA is that it allows for the lowest runtime of the algorithms since the time series are more smoothed and simpler to handle. In the end, MA with W = 3 has been chosen as the best data preprocessing technique. we will apply this combination to the time series before clustering them with all the three techniques that we are going to evaluate.

Another aspect that arises from the previous experiments is that the Silhouette score is not suitable for our task, indeed the score does not always give an accurate evaluation of the clustering. Looking at Rows 2 and 4 of Table VII, we can see that Row 2 has a higher Silhouette score than Row 4 but lower values of FA and DA. If we base our analysis on the Silhouette score the clustering produced by the configuration of Row 2 would thought to be better than

the one produced by Row 4 since the Silhouette score is closer to 1. If, instead, we observe the results of the clustering of Row 2 in Figure 46 and Row 4 in Figure 47 we can see that the second clustering is better. The blues lines in the figures are the representatives of the clusters calculated using DBA. In Figure 47 the clusters are better separated and, in particular, the "stable" and "monitor" clusters are more homogeneous. For this reason, when analysing the results of the algorithms, we will trust more the FA and DA values than the Silhouette score. The Silhouette score measures the closeness of points in the same cluster but the "critical" and "monitor" clusters that we aim to create should contain time series that could not be similar.

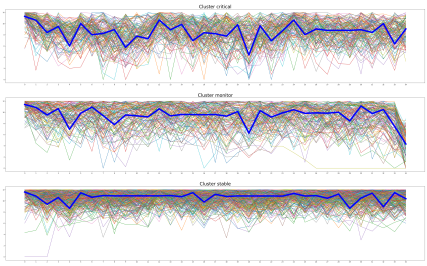

Figure 46: Table VII Row 2 clustering.

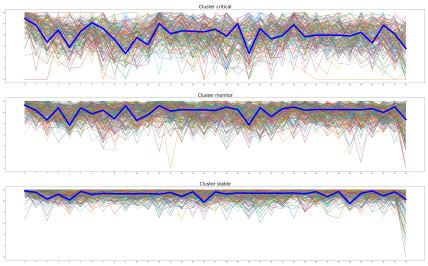

Figure 47: Table VII Row 4 clustering.

6.2 Baseline

In this section, we present the results of the experiments for the first two techniques: "clustering of time series with different lengths" (Section 5.1) and "classical clustering algorithms" (Section 5.2). These techniques will be used as a baseline to see the level of accuracy achieved by state-of-the-art algorithms for time series clustering. Obviously, these techniques can not satisfy the constraint on the dimension of the resulting clusters since we cannot specify such constraints.

About "clustering of time series with different lengths" only K-means with DTW as distance metric can be applied since this is the only way to deal with this type of time series. In "classical clustering algorithms", we have tried six algorithms with different combinations of parameters. When possible, both Euclidean distance and DTW were used since they are the distance metrics best suited to compare time series.

For each algorithm, if it allows for more combinations of parameters, only the combination that gives the best clustering in terms of Time Series Accuracy has been considered for the analysis. In particular, if the algorithm allows us to choose the distance metric, the best combination both for Euclidean distance and DTW is considered. For each algorithm, we show the best combination of parameters and the clustering obtained with that combination:

- Clustering of time series with different lengths: K-means with DTW has been used. The resulting clusters are shown in Figure 48.

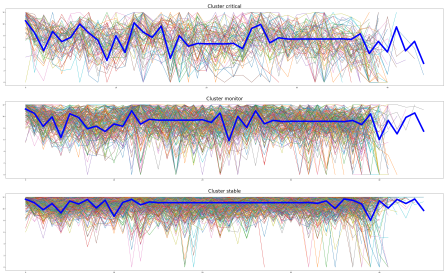

Figure 48: K-means clustering with time series of different lengths.

- K-means: We have experimented with both Euclidean distance and DTW. The resulting clusters are shown in Figure 49 and Figure 50.

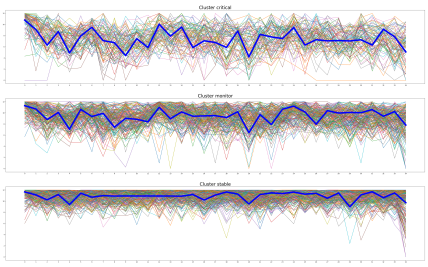

Figure 49: K-means Euclidean clustering.

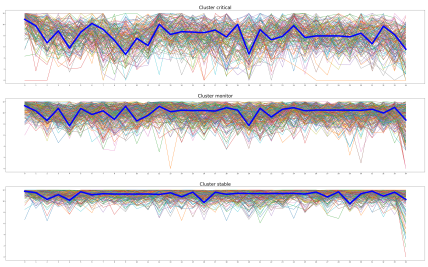

Figure 50: K-means DTW clustering.

- K-medoids: We have experimented with both Euclidean distance and DTW. The resulting clusters are shown in Figure 51 and Figure 52.

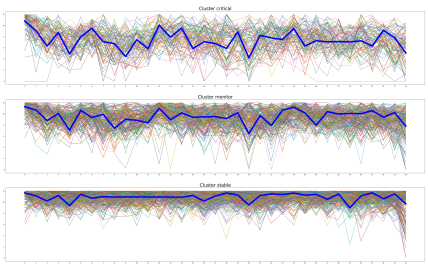

Figure 51: K-medoids Euclidean clustering.

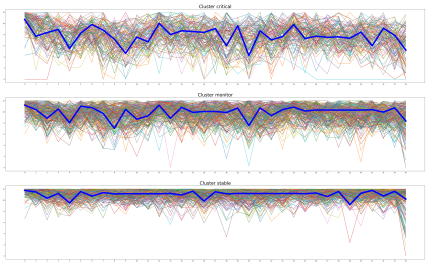

Figure 52: K-medoids DTW clustering.

- K-shape: Here we can not choose the distance metric so only one combination of parameters is possible. The resulting clusters are shown in Figure 53.

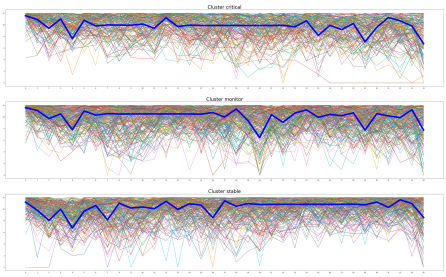

Figure 53: K-shape clustering.

- Hierarchical clustering: We have experimented with both Euclidean distance and DTW, and also with different linkage types. The results are reported in Table VIII. The results show that only the ward linkage is able to give a reasonable size for the "critical" and "monitor" clusters while the other methods construct very unbalanced dendrograms. Indeed, "critical" and "monitor" clusters contain very few time series and all the others are

in the "stable" cluster. This is probably because ward linkage is less sensitive to outliers and it is suitable when the clusters have different sizes and variances, as in this case. Moreover, note that the clustering produced by ward linkage has high accuracy (FA and DA) and is more balanced than the others but it has a lower Silhouette score. This fact confirms our intuition that the Silhouette score is not suitable for our goal and should not be trusted when evaluating the clustering.

TABLE VIII: RESULTS OF HIERARCHICAL CLUSTERING

Distance Metric	Linkage	Size Critical	Size Monitor	FA	DA	Silhouette
euclidean	single	0.001	0.001	0.000	0.500	0.350
euclidean	complete	0.001	0.187	0.704	0.898	0.358
euclidean	average	0.001	0.046	0.360	0.840	0.393
euclidean	**ward**	**0.145**	**0.454**	**0.906**	**0.971**	**0.199**
DTW	single	0.001	0.001	0.000	0.500	0.480
DTW	complete	0.001	0.062	0.364	0.939	0.394
DTW	average	0.001	0.064	0.382	0.941	0.393
DTW	**ward**	**0.050**	**0.297**	**0.828**	**0.967**	**0.304**

In the end, ward linkage has been selected with both Euclidean distance and DTW as distance metrics, the resulting clusters are shown in Figure 54 and Figure 55.

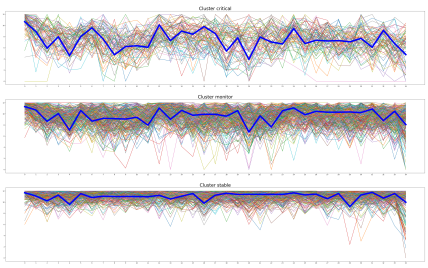

Figure 54: Hierarchical Euclidean clustering.

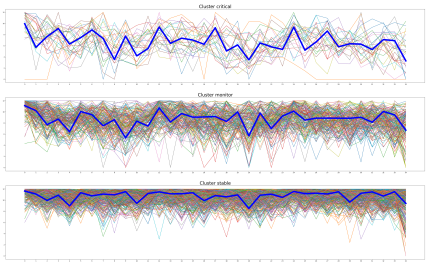

Figure 55: Hierarchical DTW clustering.

- DBSCAN: We have experimented with both Euclidean distance and DTW, we have also tried different combinations of "eps" and "min_samples". This algorithm is not suitable to deal with time series data, indeed it struggles to find exactly 3 clusters. This is due to the dimensionality; indeed, the time series seem to have a uniform distribution in the high-dimensional space and the algorithm is not able to find regions of high-density points. For this reason, almost every point is identified as an outlier that is classified in the "critical" cluster. The resulting clusters are not satisfactory and are shown in Figure 56 and Figure 57.

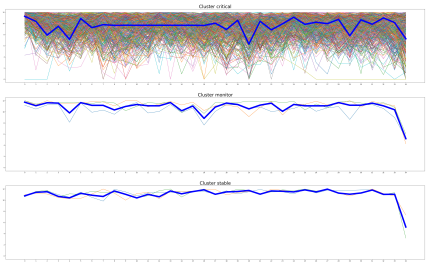

Figure 56: DBSCAN Euclidean clustering.

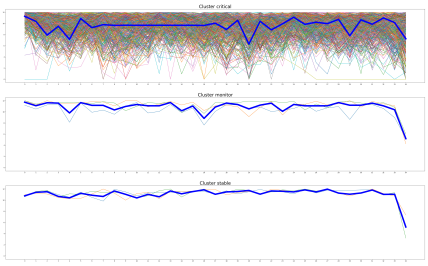

Figure 57: DBSCAN DTW clustering.

- Spectral: We have done a few experiments using different methods to assign the labels to the reduced data, the results are reported in Table IX. If we look at the dimensions of the clusters and the accuracy these methods are almost equivalent.

TABLE IX: RESULTS OF SPECTRAL CLUSTERING

Affinity matrix	Assign labels	Size Critical	Size Monitor	FA	DA	Silhouette
K-nearest neighbors	**K-means**	**0.293**	**0.490**	**0.956**	**0.993**	**0.140**
K-nearest neighbors	discretize	0.305	0.459	0.957	0.990	0.143
K-nearest neighbors	cluster qr	0.286	0.492	0.955	0.990	0.145

In the end, the K-means method has been selected to assign the labels since it is the method that we are most familiar with. The resulting clusters are shown in Figure 58.

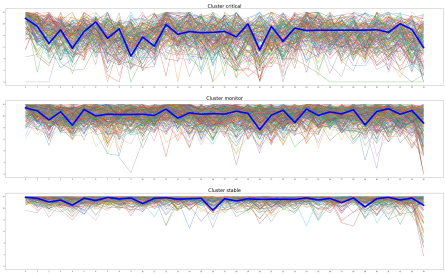

Figure 58: Spectral clustering.

Now that we have shown the best results for each clustering, it is necessary to analyse and compare them to discover which algorithm performs better and to define a baseline for our final

model. The first aspect to analyse is the size of the resulting clusters since the main constraint of our problem is to create a "critical" cluster with at least 10% of students and a "monitor" cluster with at least 20% of students.

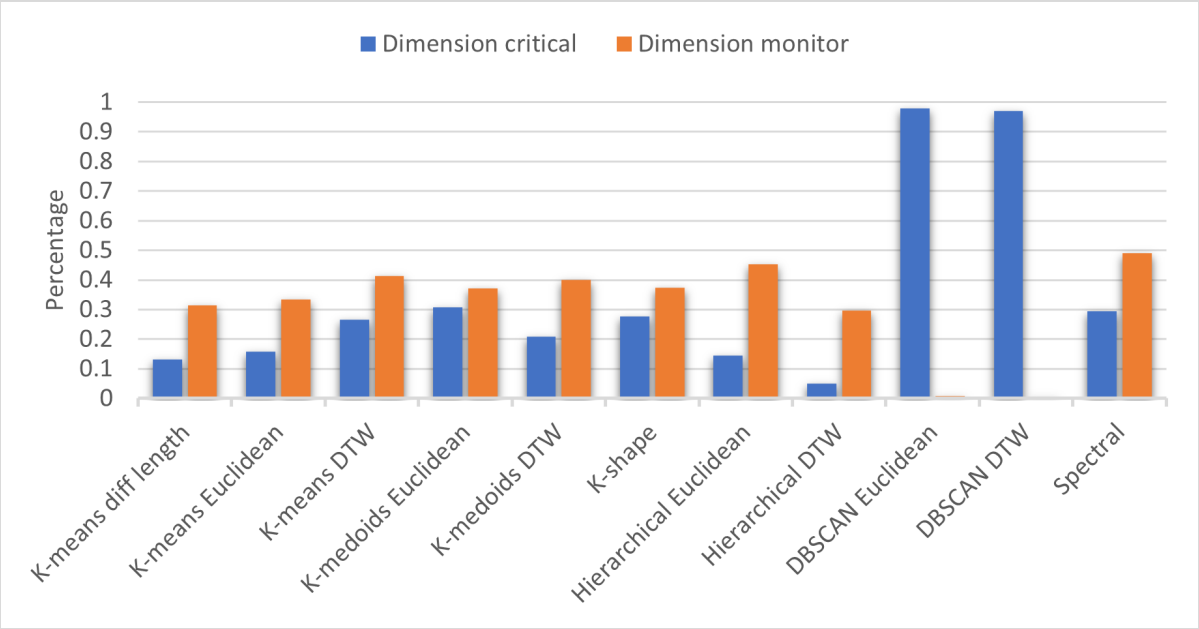

Figure 59: Baseline analysis on the dimension of the clusters.

As shown in Figure 59, the dimensions of the resulting clusters are similar for all the algorithms analyzed. In particular, the "critical" cluster contains between 10% and 20% of time series while the "monitor" cluster contains between 30% and 40% of time series. The only exception is DBSCAN which is not able to find regions of high-density points, both with Eu-

clidean and DTW as distance metrics. Indeed, it classifies almost all the time series as outliers that are inserted in the "critical" cluster as you can see from Figure 56 and Figure 57.

The dimensions of the clusters are decided automatically by the algorithm depending on the data. For almost all the algorithms the constraint is satisfied since the "critical" and "monitor" clusters contain, respectively, more than 10% and 20% of time series but we seek to find dimensions closer to our target. Moreover, these dimensions are fixed so if, in the future, we would like to change the target dimensions of the clusters these algorithms could not satisfy the constraint.

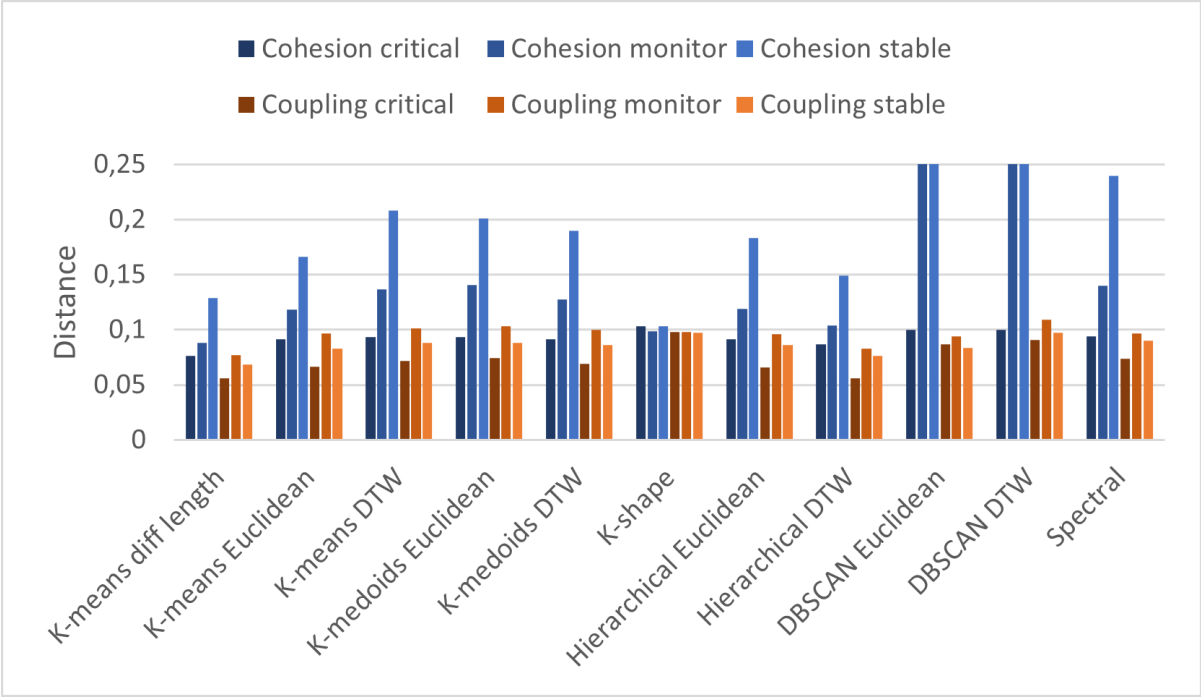

Figure 60: Baseline analysis of Cohesion and Coupling.

The second very important aspect to evaluate is the quality of the clusters. To do this we have calculated, for each algorithm, the Cohesion and Coupling in Figure 60 and Time Series Accuracy in Figure 61. As you can observe from Figure 60, almost every algorithm has a descending trend for the Cohesion values with the "critical" cluster that has the lowest value while the "stable" cluster has the highest value. This means that "critical" and "monitor" clusters are made of time series with different shapes that have a higher mean distance, "stable" cluster instead is made of similar time series that has a lower mean distance. With respect to Coupling, almost every algorithm has the highest value associated with the "monitor" cluster and the lowest value associated with the "critical" cluster. This is because the "monitor" cluster consists of time series in the middle between the "stable" time series (with linear trends) and the "critical" time series (with big fluctuations and descending trends).

Also for these measures, DBSCAN does not perform well since almost all the time series are in the "critical" cluster. Beyond DBSCAN, also K-shape does not perform well since the Cohesion and Coupling of all the clusters are almost the same. This is due to the fact that the K-shape needs to normalize the time series before clustering. This normalization flattens the time series and makes it difficult to classify them into well-formed clusters. Indeed, if we look at Figure 53, the three clusters are very similar; clearly, some time series are in the wrong cluster. On the contrary, the other algorithms show the values expected.

Looking at Figure 61 we can note that all the algorithms, except K-shape, show a high accuracy (around 90%) both for FA and DA, this means that almost every time series with big fluctuations or descending trends have been classified in the "monitor" or "critical" cluster.

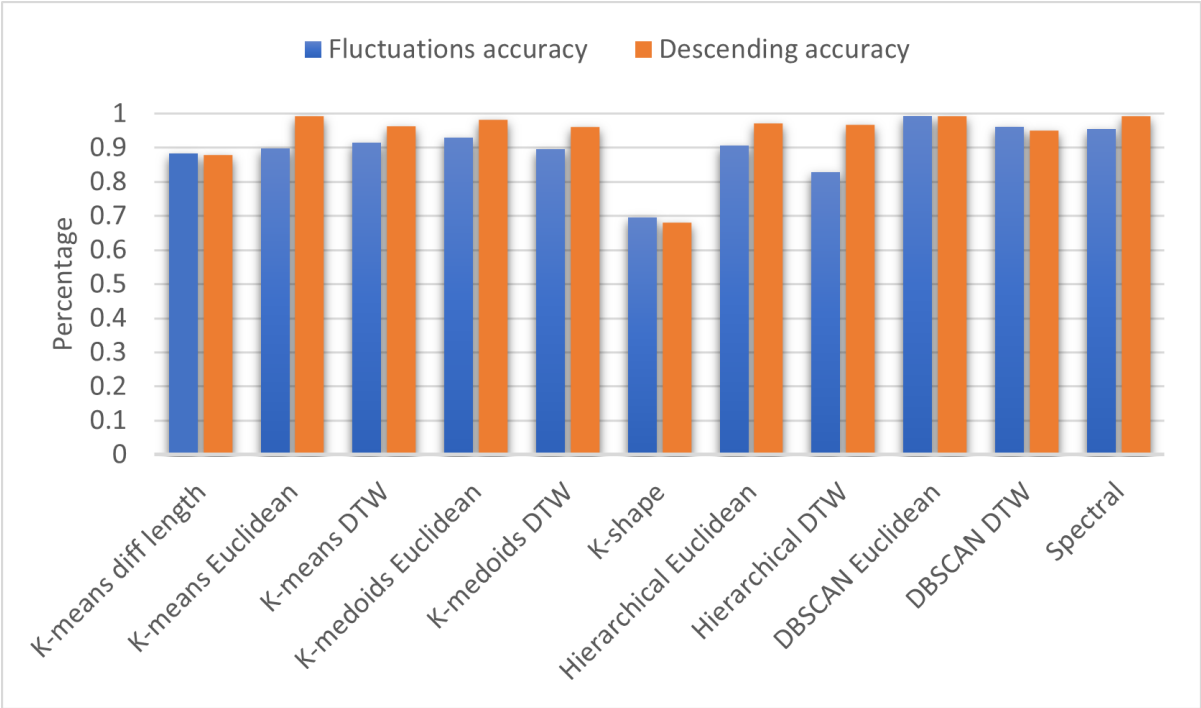

Figure 61: Baseline analysis of Time Series Accuracy.

K-shape, instead, has FA ad DA values around 60% so around 40% of the time series that we would like to flag have been classified in the "stable" cluster. It is also important to note that there is not any improvement when using DTW instead of using Euclidean distance. This is probably because our time series are very noisy and also because they could have common sequences at very distant instants of time that not even DTW is able to match.

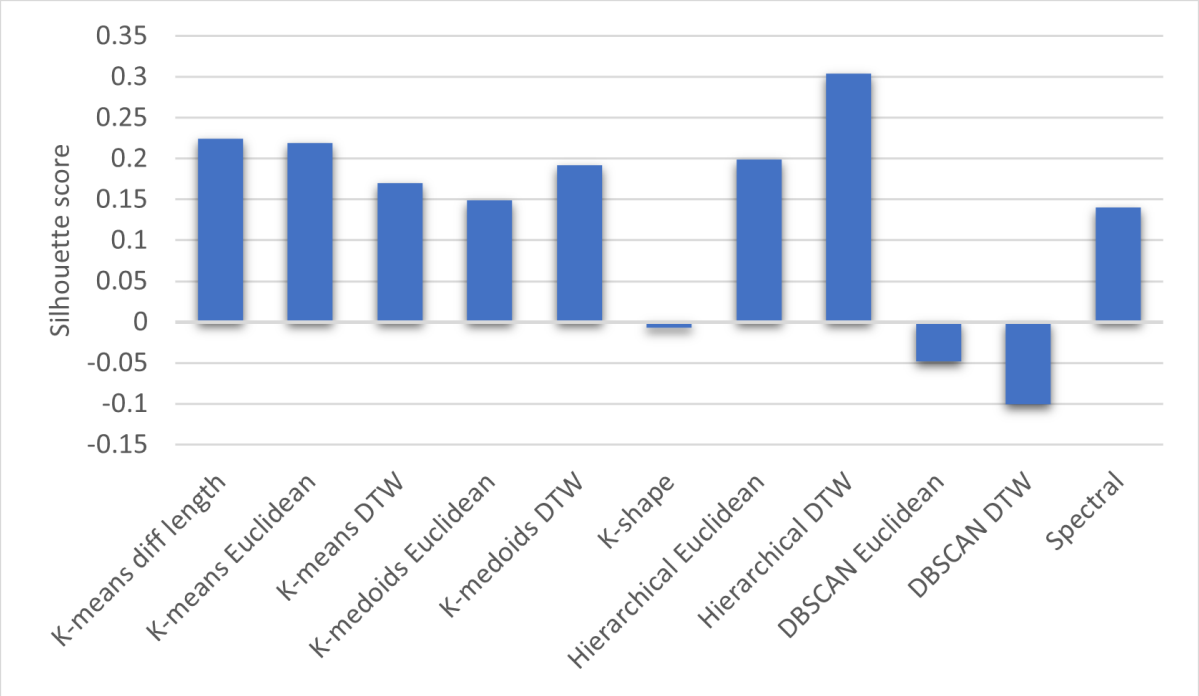

Figure 62: Baseline analysis of Silhouette score.

Another measure that we can consider for evaluation is the Silhouette score. The Silhouette could be useful to know if some algorithms have scored very differently from others which could highlight bad clustering. As you can see from Figure 62, K-shape and DBSCAN are the only 2 algorithms with a negative Silhouette score, this fact supports our previous observations and confirms that these two algorithms are the worst for our goal.

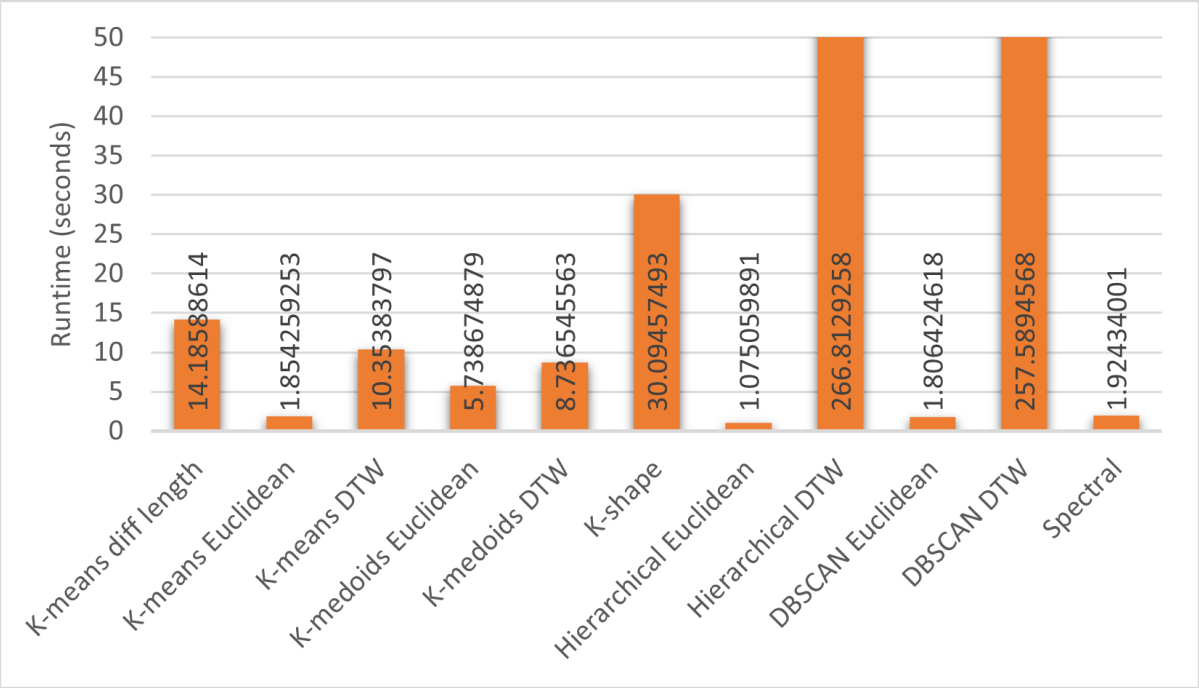

Figure 63: Baseline analysis of runtime.

The last and less important aspect that we have evaluated is the runtime of the algorithms that are shown in Figure 63. From this plot, we can see that the algorithms that use DTW have a higher runtime than their version with Euclidean distance. In particular, Hierarchical with DTW and DBSCAN with DTW have a runtime of more than 4 minutes because they need to compute the distance between each pair of points which is very expensive.

6.3 Final model

In this section, we present the results for the final model implemented by "density-based clustering with feature reduction" (Section 5.3). With this model we aim to satisfy all the constraints of our goal: identify 3 well-separated clusters where the "critical" cluster should contain at least 10% of time series with significant fluctuations and/or evident downward trend, the "monitor" cluster should contain at least 20% of time series with some fluctuations and/or an overall downward trend and the "stable" cluster should contain the rest of time series. In this model, differently from the previous two techniques, we give as input to the algorithm the size of the "critical" and "monitor" clusters. In this way, we should obtain clusters with precise dimensions. Moreover, with this approach, we can modify the desired size of the clusters to adapt them to future needs.

To evaluate in a general way the model we have performed three types of experiments:

1. Cluster all the time series to find at least 10% of "critical" students and at least 20% of "monitor" students that is the actual goal of this work (Section 6.3.1).
2. Cluster subsets of the time series to find at least 10% of "critical" students and at least 20% of "monitor" students. This is done to discover if the algorithm works well with datasets of different dimensions (Section 6.3.2).
3. Cluster all the time series specifying different dimensions for the "critical" and "monitor" clusters. This is done to discover if the model is able to form clusters with arbitrary dimensions (Section 6.3.3).

For the first type of experiment, the algorithm has been executed five times since it is not deterministic but the result could differ between different runs. Before each experiment, we need to define the search space of the hyperparameters; in particular, we need to decide the discrete range of values for the number of components of PCA and the two parameters of DBSCAN ("eps" and "min_samples"). The result of the algorithm depends on the dimension of the search space and on the maximum number of trials that the Bayesian tuning can use to find the best combination of hyperparameters. It is obvious that, if the search space is large, the Bayesian tuning could need more trials to find a good combination of hyperparameters. An advantage of this method is that, if the results that we obtain are not satisfactory, we could simply enlarge the search space so that there are more possible combinations of hyperparameters. Another solution could be to increase the number of trials. Usually, the Bayesian tuning is able to find a good combination by trying only 1/10 of all possible combinations in the search space; this allows a reasonable runtime of the algorithm.

6.3.1 Fixed Number of Time Series and Expected Dimensions

In Figure 64 we can see the dimensions of the "critical" and "monitor" clusters for the five runs of the algorithm. The target values (10% for "critical" and 20% for "monitor") are marked by black horizontal lines. We want that the dimensions of the clusters are as close as possible to the targets but not lower. For all the runs, the algorithm has been able to find clusters with dimensions very close to the expected values. As you can see for Figure 64, the results are not equal but they differ a little bit; this is because the Bayesian tuning could

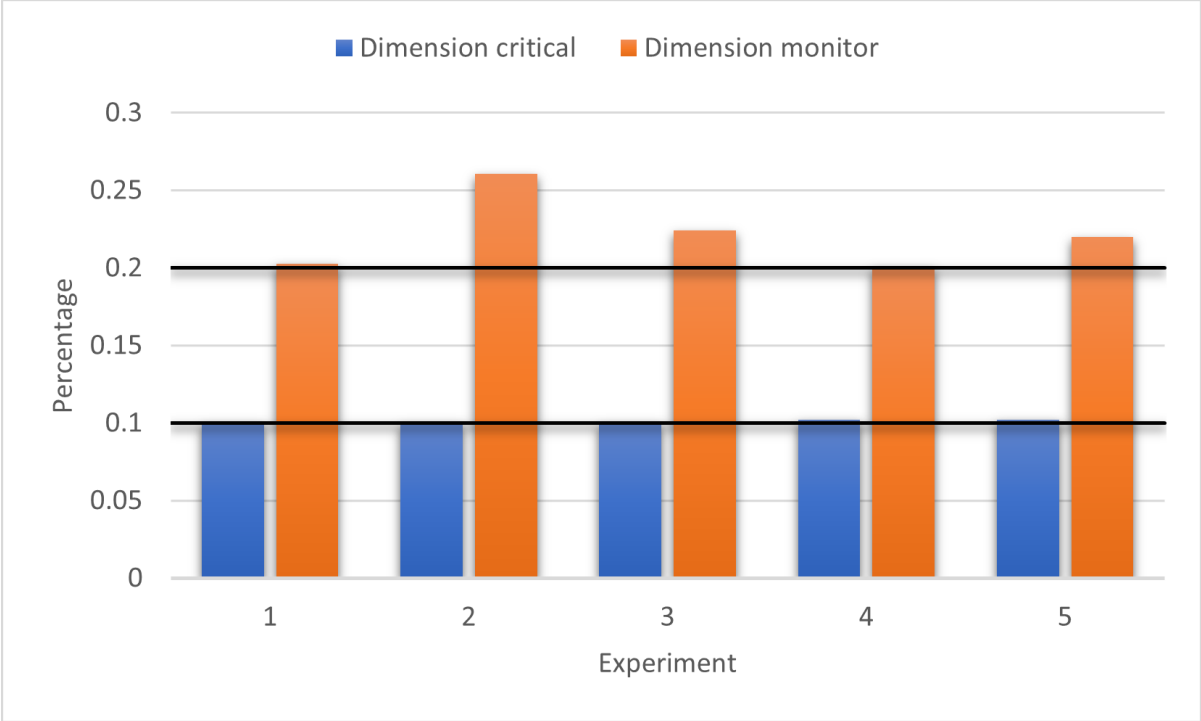

Figure 64: Model analysis of the dimension of the clusters.

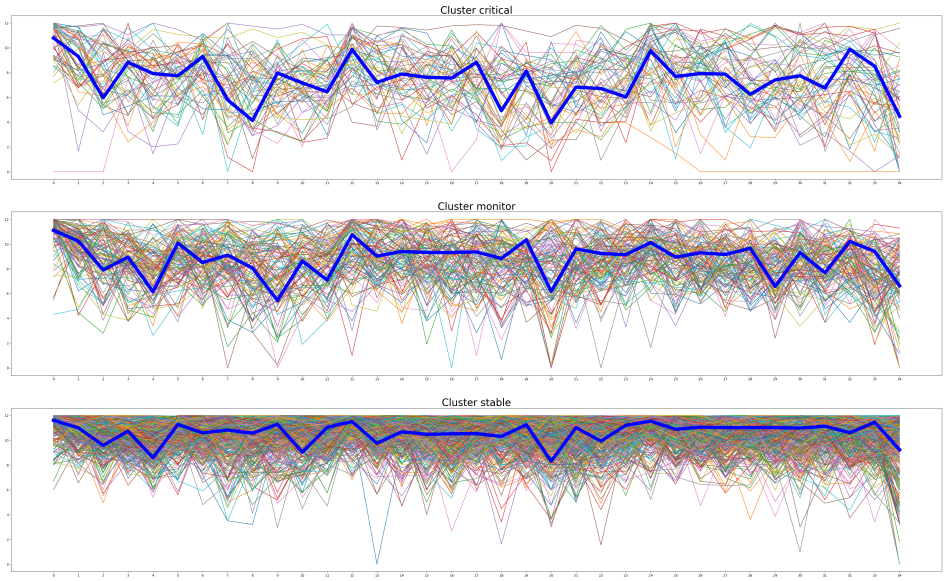

Figure 65: Model clustering.

choose a different combination of hyperparameters at each run of the algorithm. An example of clustering produced by the model is shown in Figure 65, here we can note that the clusters are well-separated. In particular, the "stable" cluster is very dense with time series that has a high quote and a trend almost linear.

[htbp]

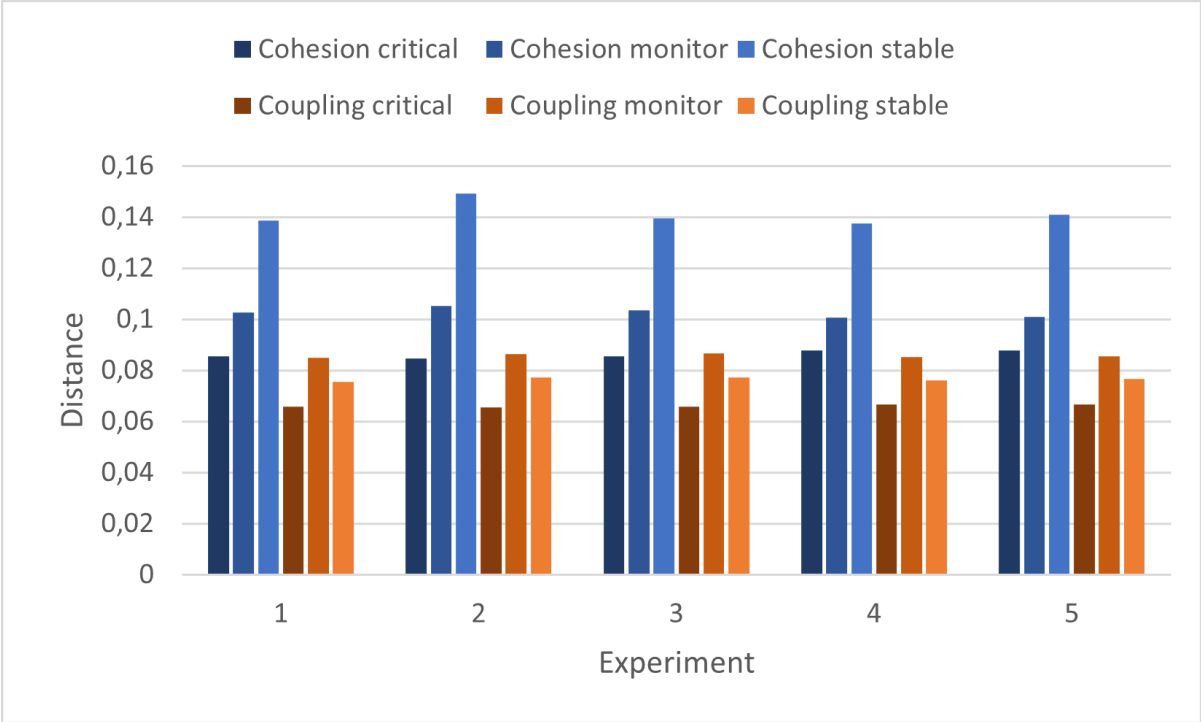

Figure 66: Model analysis of Cohesion and Coupling.

In Figure 66 it is shown the Cohesion and Coupling values for each experiment. Their trend mirror the values expected. Cohesion values have an ascending trend while Coupling values have an inverse parabolic trend with the highest value associated with the "monitor" cluster and the lowest value associated with the "critical" cluster.

[htbp]

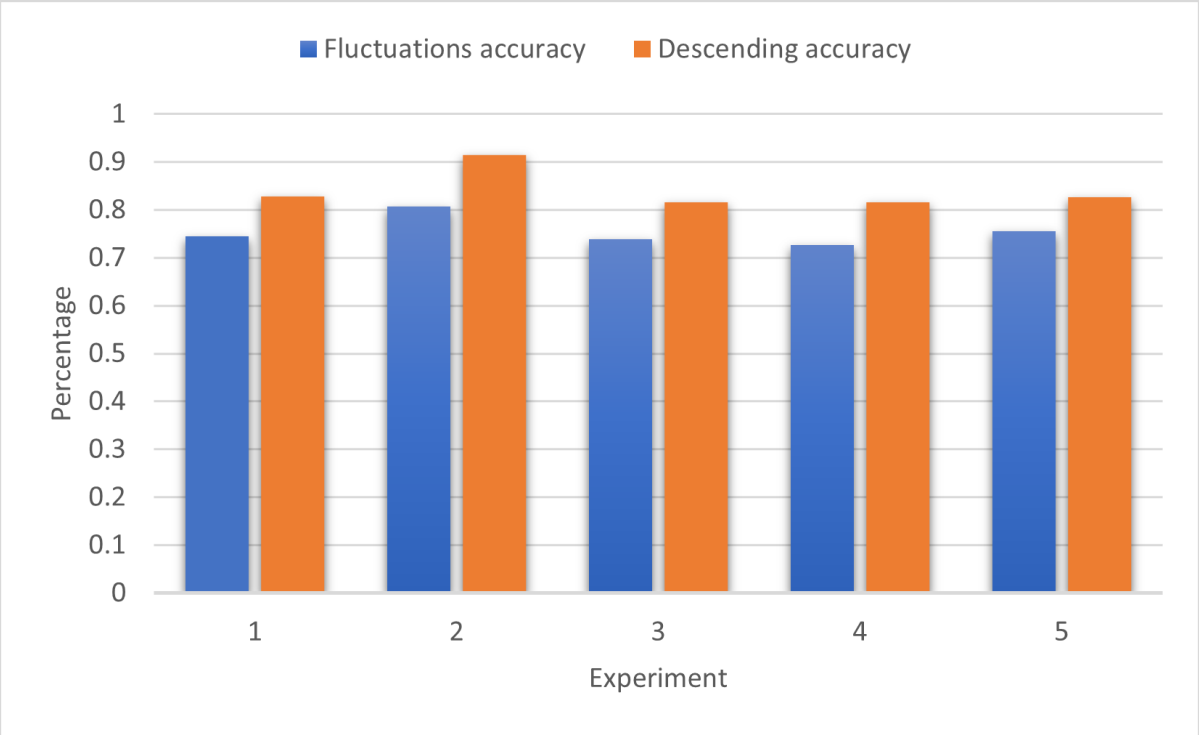

Figure 67: Model analysis of Time Series Accuracy.

In Figure 67 we evaluate the Time Series Accuracy and we can observe that FA is above 70% and DA is above 80%. These accuracies are lower than the highest values of the baseline (Figure 61) because these values highly depend on the dimension of the "critical" and "monitor" clusters. As you can see in Figure 59 those algorithms form bigger clusters and so it is obvious that they can reach higher accuracies. Moreover, we can note in Figure 67 that experiment 2 has the highest accuracy; this is because experiment 2 has formed the clusters with the largest sizes as we can see from Figure 64.

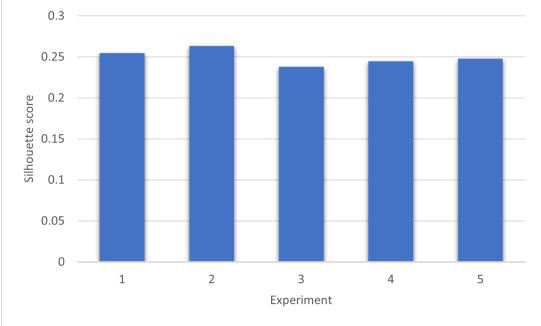

Figure 68: Model analysis of Silhouette score.

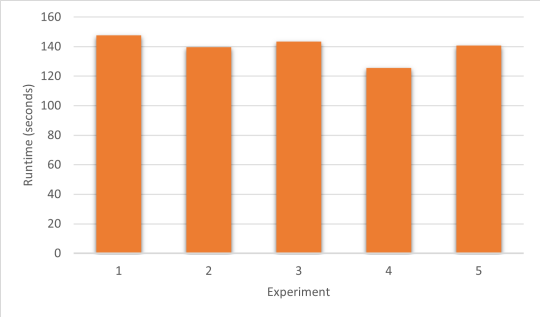

Figure 69: Model analysis of runtime.

Figure 68 shows the Silhouette scores that are almost the same for all the experiments and comparable to the values of the baseline (Figure 62). Figure 69, instead, shows the runtime of the algorithm for each experiment. The average runtime is about 130 seconds (more than 2 minutes). This value could seem very high if compared to the lowest values of the baseline (Figure 63). On the contrary, it is not high if we consider that the Bayesian tuning performs 100 trials to discover the best combination of hyperparameters over the above 1000 possible combinations in the search space. The runtime, indeed, depends on the size of the search space because the TPE needs more time to decide the next combination of hyperparameters to try if the search space is bigger. Moreover, it also depends on the number of trials that the Bayesian tuning can use to find the optimal set of hyperparameters.

6.3.2 Different Number of Time Series

In this section, we have performed different experiments with the model fixing the percentage of "critical" and "monitor" students to flag (respectively 10% and 20%) but changing the number of times series in input. This is done to discover if the algorithm has the same performance with datasets of different dimensions. The number of time series in input for each experiment is specified on the x-axis of the following plots.

In Figure 70 we can see the dimensions of the "critical" and "monitor" clusters created by the algorithm with respect to the number of time series in input. It is possible to observe that the algorithm can create clusters with dimensions very close to the values expected independently from the number of time series in input. This is very important because it means that the model can be used also with datasets of different dimensions and it is not only specific to our dataset. In Figure 71 you can find an example of clustering produced by the model for 100 time series. Here, with respect to Figure 65, it is possible to appreciate the clustering better and to see the differences between the 3 clusters. In the "critical" cluster there are time series with low grades and very big fluctuations, the "monitor" cluster contains time series with little higher grades and frequent fluctuations and in the "stable" cluster we can find time series with high grades and only a few fluctuations.

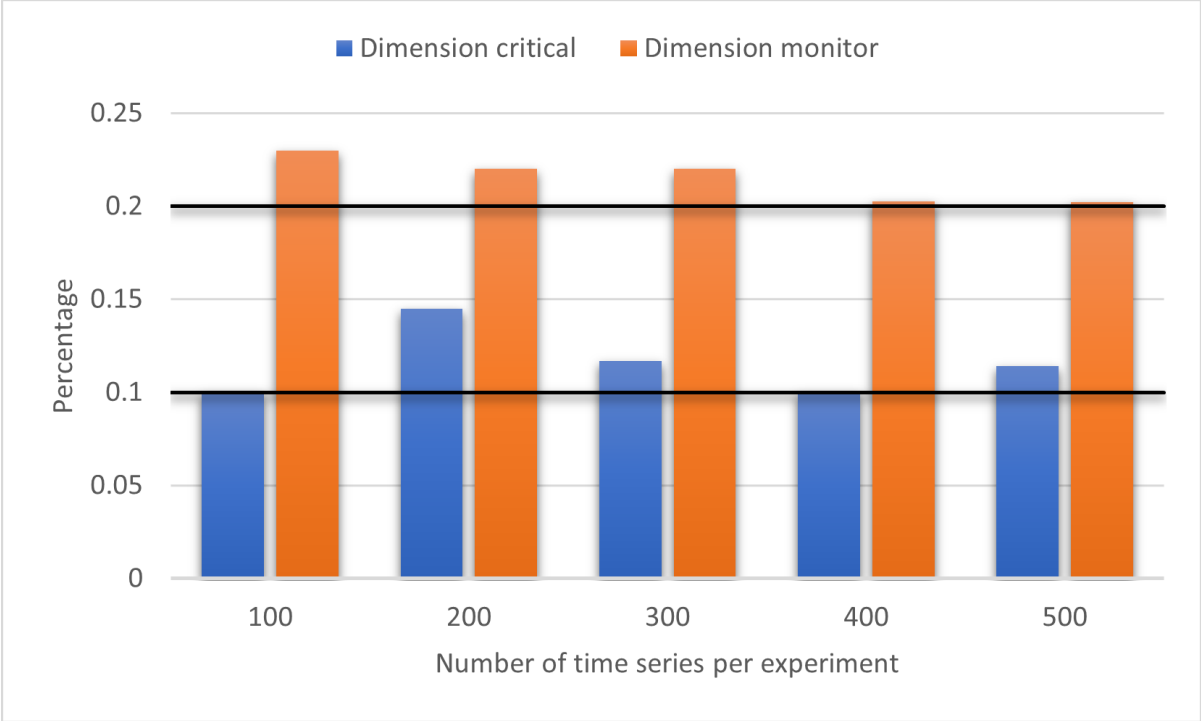

Figure 70: Analysis of the dimension of the clusters with different input sizes.

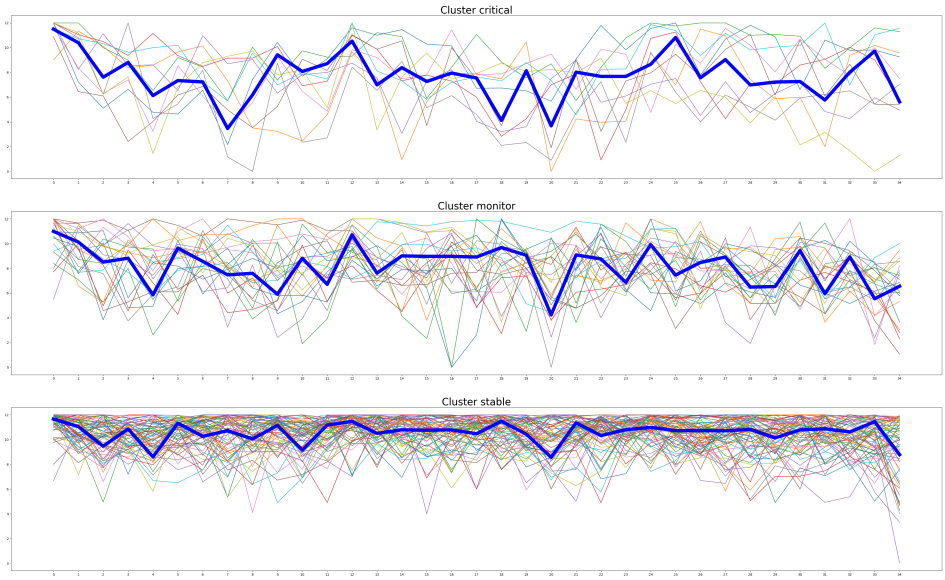

Figure 71: Clustering with 100 time series.

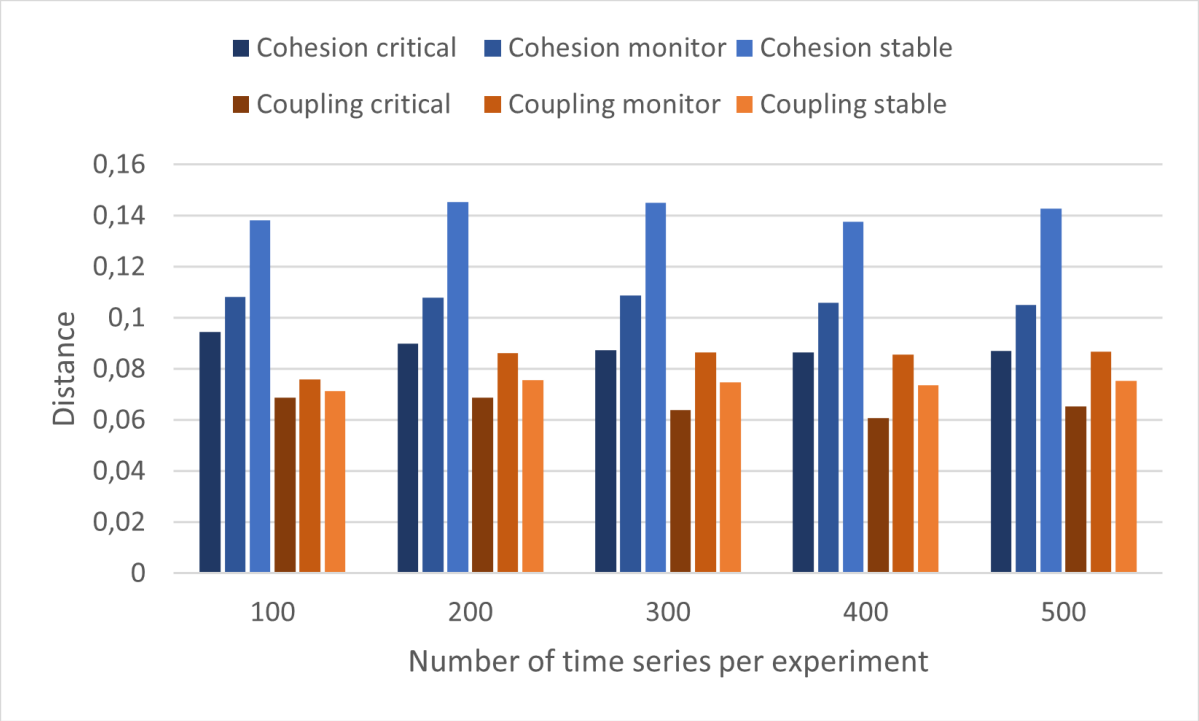

Figure 72: Analysis of Cohesion and Coupling with different input sizes.

In Figure 72, we have calculated the Cohesion and Coupling distances with respect to the number of time series in input. It is possible to see that these values are distributed as expected and correspond to the ones obtained in Figure 66. This means that the mean distance between the time series in the same cluster (Cohesion) and the mean distance between the time series of different clusters (Coupling) do not change with different input sizes.

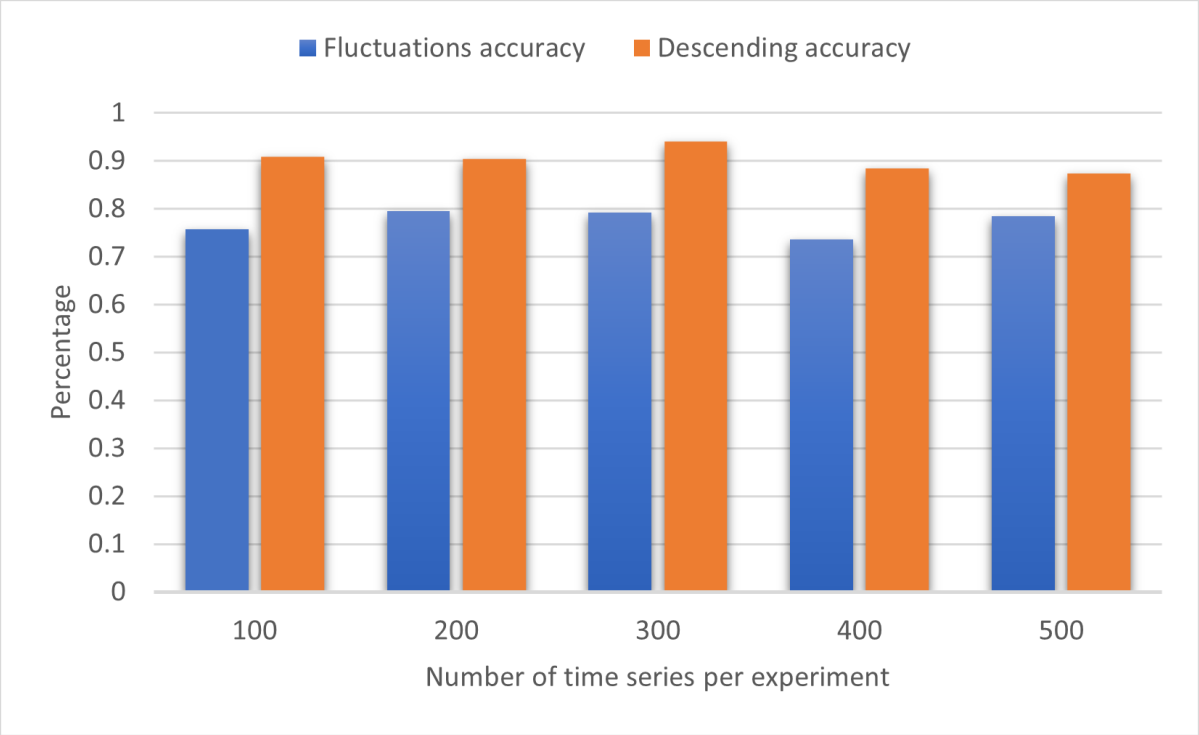

Figure 73: Analysis of Time Series Accuracy with different input sizes.

Another assurance on the quality of the generated clusters is shown in Figure 73 where we have calculated the FA and DA values for each experiment with different input sizes. Also in this case, we can note that the accuracies are similar to the values of Figure 67. We can conclude that the quality of the clusters remains almost the same, even changing the size of the time series in input.

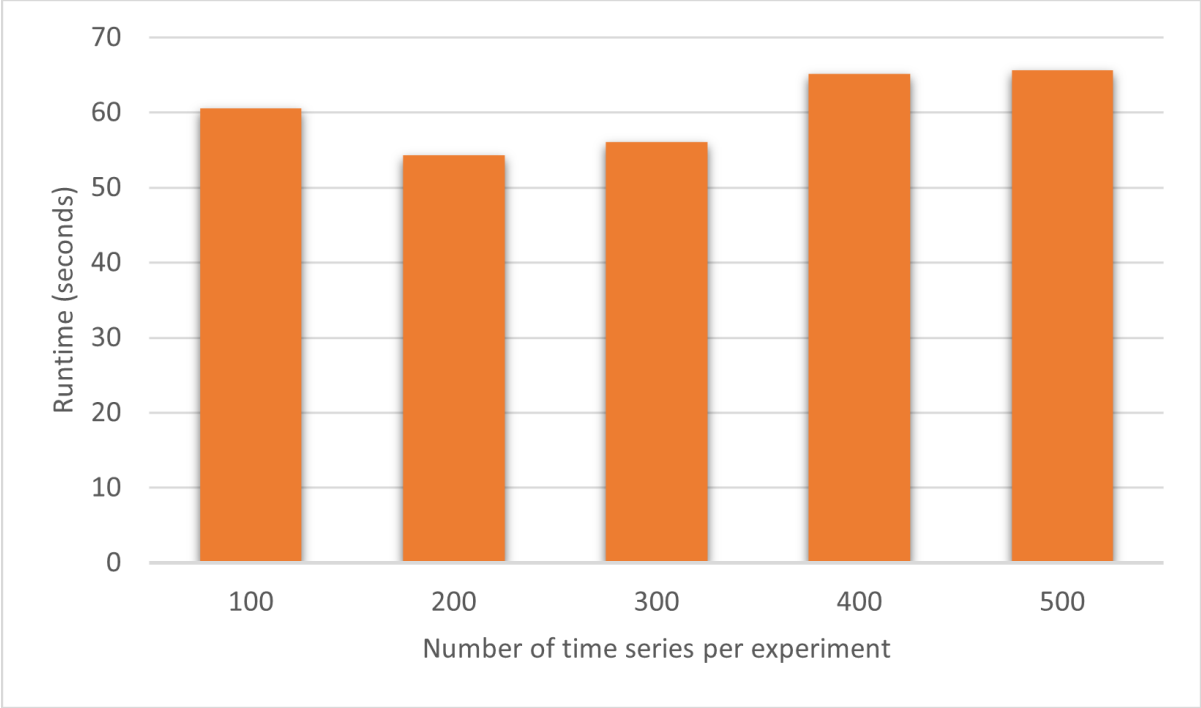

Figure 74: Analysis of runtime with different input sizes.

An interesting aspect is shown in Figure 74 where we can see the runtime of the algorithm with respect to the number of time series in input. Here it is possible to see that the runtime is constant between the experiments. This is because the runtime depends mainly on the size of the search space and on the number of trials that the Bayesian tuning is allowed to perform; it is only partly affected by the input size.

6.3.3 Different Expected Dimensions

In this section, we have performed different experiments with the model using all the time series but changing the expected dimensions of the "critical" and "monitor" clusters. This is

done to discover if the algorithm can create clusters with arbitrary sizes. For each experiment, the expected dimensions of the clusters are specified on the x-axis of the following plots.

In Figure 75, we can see the dimensions of the "critical" and "monitor" clusters created by the algorithm with respect to different expected dimensions of the clusters. It is possible to observe that the algorithm can create clusters with dimensions very close to the expected values for all the experiments. This is very important because it means that the model can be used to create clusters of arbitrary dimensions, so it could be suitable for a lot of other applications. In Figure 76, you can find an example of clustering produced by the model with 30% of "critical" time series and 40% of "monitor" time series. Here is possible to observe that the clusters are dense and very well separated even if the specified dimensions are high.

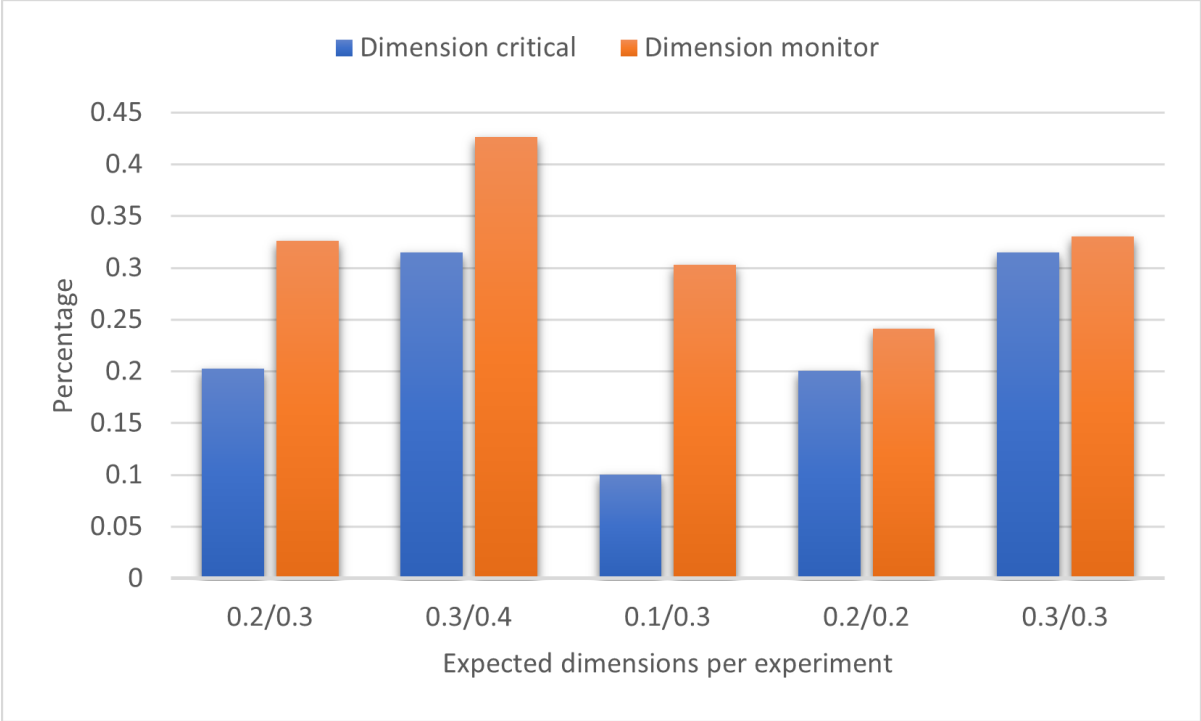

Figure 75: Analysis of the dimension of the clusters with different dimensions.

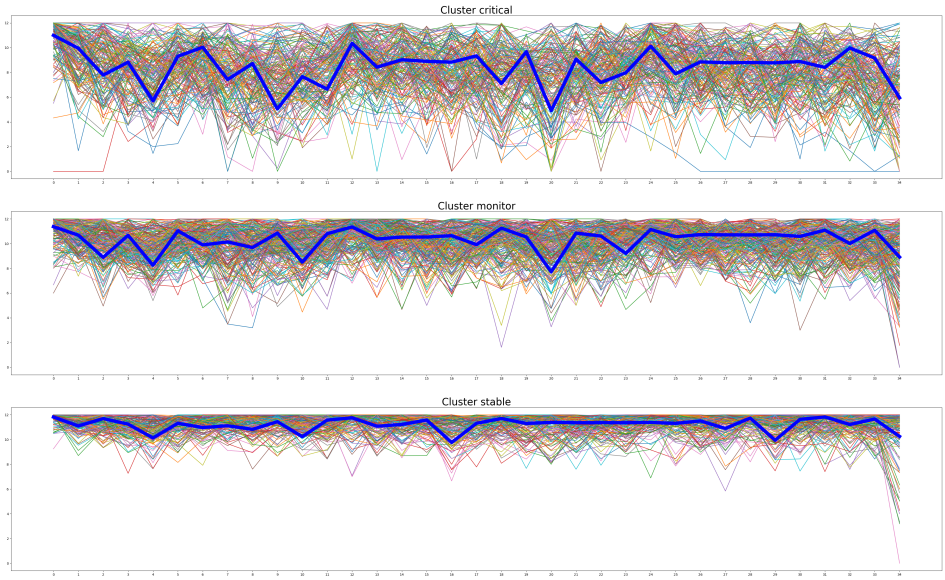

Figure 76: Clustering with 0.3 “critical” dimension and 0.4 “monitor” dimension.

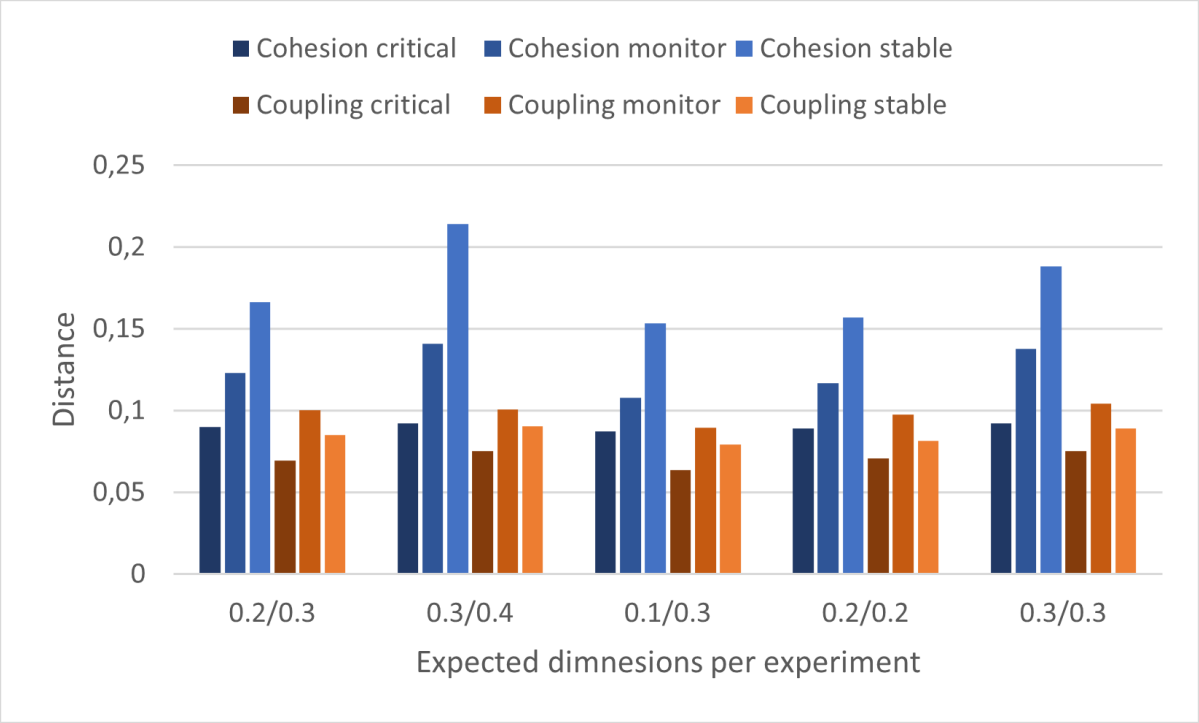

Figure 77: Analysis of Cohesion and Coupling with different dimensions.

In Figure 77, we have calculated the Cohesion and Coupling distances with respect to different expected dimensions of the clusters. The values of Cohesion and Coupling are distributed as expected and correspond to the ones obtained in Figure 66. This means that the clusters maintain the same characteristics while changing their dimensions.

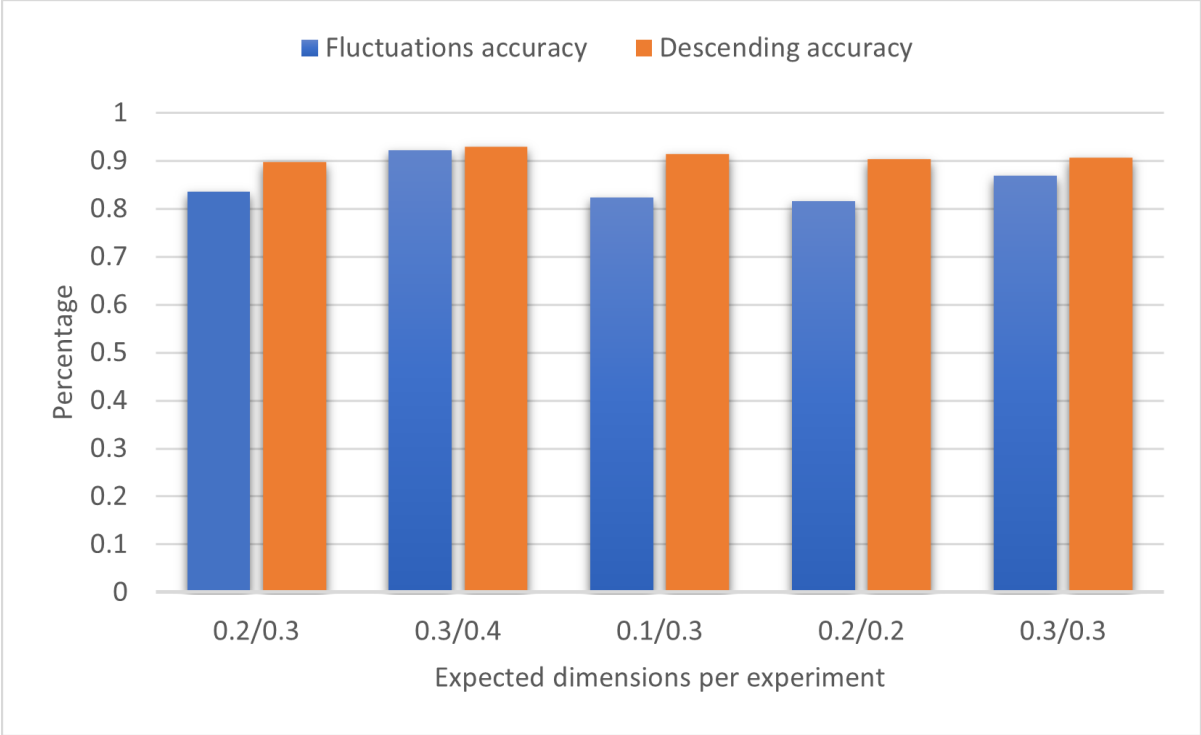

Figure 78: Analysis of Time Series Accuracy with dimensions.

In Figure 78, we can see the FA and DA values for each experiment with respect to different expected dimensions of the clusters. In this case, we can note that the accuracies are higher than the values achieved in Figure 67. This is because the "critical" and "monitor" clusters are bigger; indeed, the FA and DA values are highly affected by the dimensions of the clusters. After this analysis, we can conclude that the algorithm can create "critical" and "monitor" clusters of arbitrary size while keeping the same quality of the generated clusters.

CHAPTER 7

CONCLUSION

We faced the problem of classifying students into three different classes ("critical", "monitor" and "stable") through clustering of time series representing their academic data to detect the possible onset of emotional issues. The "critical" and "monitor" clusters that we want to create are characterized by specific dimensions and time series with special markers such as significant fluctuations and descending trends.

We considered three different techniques to classify these time series: "clustering of time series with different lengths", "classical clustering algorithms" and "density-based clustering with feature reduction". For the first technique, we have created time series with different lengths putting together the available grades. The other two techniques use time series with the same length that we have created interpolating grades for the missing weeks. We have used the "linear" interpolation method after an accurate analysis of all the available interpolation methods. Before clustering the time series are preprocessed using data smoothing, and in particular moving average with window size = 3. This is the data preprocessing technique that gives the best results.

In the first technique, time series with different lengths are classified using the K-means algorithm with DTW. In the second technique, we have applied several clustering algorithms to classify the time series in 3 different clusters using both Euclidean distance and DTW as distance metrics. For each algorithm, we have found the best combination of parameters that

gives the best clustering.

In the third technique, we have used Principal Component Analysis (PCA) to reduce the size of the times series before clustering. With PCA, similar time series in the high-dimensional space correspond to points with similar principal components in the low-dimensional space that creates a zone of high density, while time series different from the others correspond to isolated points in low dimensions. The reduced time series are clustered using DBSCAN to identify two clusters, one high-density region of similar points and the cluster of the outliers that correspond to students that we would like to flag. Bayesian tuning is used around PCA and DBSCAN to tune the hyperparameters and find clusters with specific sizes. The process is repeated twice, first to find at least 10% of "critical" time series and second to find at least 20% of "monitor" time series.

The first two techniques represent a baseline to be compared with the third technique that incorporates the size constraints. We have evaluated all the techniques by calculating the size of the clusters and using ad-hoc metrics to evaluate the quality of the clustering with respect to the markers. It turns out that the final model can satisfy the size constraints of the clusters while achieving an accuracy comparable to the baseline. This model has shown similar performance, also varying the number of the time series in input and the expected dimensions of the resulting clusters.

We believe that our work could help the schools to identify students that are developing emotional issues or even predict the onset of these emotional issues by analyzing the students' academic data. In this way, the parents or schoolteachers of students flagged as "critical" or

"monitor" could be notified and the students could access to appropriate school resources.

7.1 Contribution

The main contributions of our work include:

- Module to produce time series with specific characteristics (interval of time, topic), both of different lengths or with the same length using interpolation.
- Module to evaluate the interpolation methods and choose the one more suitable to the nature of the data.
- Ad-hoc metrics to evaluate the quality of the clusters and the accuracy of the algorithm to identify time series with markers.
- An algorithm that can find clusters of arbitrary size specified in input with high quality and accuracy with respect to the markers.

7.2 Future Work

In the future, we plan to improve the effectiveness of our algorithm in the detection and prevention of the onset of emotional issues by considering more than one time series per student. Indeed, we aim to extend our analysis by taking into account, for each student, a time series for each topic and a time series for the absences. We also plan to evaluate in a more accurate way the performance of our algorithm using some data labelled by a psychologist. Moreover, an interesting improvement to this work could be achieved by applying an evolutionary clustering

[32] approach to our algorithm to produce a clustering for each timestamp. Comparing these clustering, it could be possible to define an interval in which a student is flagged as “critical” or “monitor”. In this way, it could be possible to flag students in real time and allow for early detection of emotional issues by evaluating how the clusters have changed at different timestamps.

CITED LITERATURE

1. Centers for disease control and prevention - cdc, Apr 2023.

2. Javed, A., Lee, B. S., and Rizzo, D. M.: A benchmark study on time series clustering. Machine Learning with Applications, 1:100001, September 2020.

3. Aghabozorgi, S., Seyed Shirkhorshidi, A., and Ying Wah, T.: Time-series clustering – a decade review. Information Systems, 53:16–38, October 2015.

4. Jin, X. and Han, J.: Partitional clustering. In Encyclopedia of Machine Learning, eds. C. Sammut and G. I. Webb, pages 766–766. Boston, MA, Springer US, 2010.

5. Nielsen, F.: Hierarchical Clustering. In Introduction to HPC with MPI for Data Science, ed. F. Nielsen, Undergraduate Topics in Computer Science, pages 195–211. Cham, Springer International Publishing, 2016.

6. Paparrizos, J. and Gravano, L.: k-shape: Efficient and accurate clustering of time series. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, SIGMOD '15, pages 1855–1870, New York, NY, USA, May 2015. Association for Computing Machinery.

7. Sander, J.: Density-based clustering. In Encyclopedia of Machine Learning and Data Mining, eds. C. Sammut and G. I. Webb, pages 1–5. Boston, MA, Springer US, 2016.

8. von Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing, 17(4):395–416, December 2007.

9. Berndt, D. J. and Clifford, J.: Using dynamic time warping to find patterns in time series. In Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, AAAIWS'94, page 359–370, Seattle, WA, Jul 1994. AAAI Press.

10. Petitjean, F., Ketterlin, A., and Gançarski, P.: A global averaging method for dynamic time warping, with applications to clustering. Pattern Recognition, 44(3):678–693, Mar 2011.

CITED LITERATURE (continued)

11. Rousseeuw, P. J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics, 20:53–65, November 1987.

12. Mckinney, W.: pandas: a foundational python library for data analysis and statistics. Python High Performance Science Computer, Jan 2011.

13. pandas.dataframe.interpolate — pandas 2.0.1 documentation.

14. Abdi, H. and Williams, L. J.: Principal component analysis. WIREs Computational Statistics, 2(4):433–459, 2010.

15. Jin, X. and Han, J.: K-means clustering. In Encyclopedia of Machine Learning, eds. C. Sammut and G. I. Webb, pages 563–564. Boston, MA, Springer US, 2010.

16. Arthur, D. and Vassilvitskii, S.: k-means++: the advantages of careful seeding. In Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, SODA '07, pages 1027–1035, USA, January 2007. Society for Industrial and Applied Mathematics.

17. Jin, X. and Han, J.: K-medoids clustering. In Encyclopedia of Machine Learning, eds. C. Sammut and G. I. Webb, pages 564–565. Boston, MA, Springer US, 2010.

18. Ester, M., Kriegel, H.-P., Sander, J., and Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96, pages 226–231, Portland, Oregon, August 1996. AAAI Press.

19. Yu and Shi: Multiclass spectral clustering. In Proceedings Ninth IEEE International Conference on Computer Vision, pages 313–319 vol.1, October 2003.

20. Damle, A., Minden, V., and Ying, L.: Simple, direct and efficient multi-way spectral clustering. Information and Inference: A Journal of the IMA, 8(1):181–203, March 2019.

21. Liu, Y., Li, Z., Xiong, H., Gao, X., and Wu, J.: Understanding of internal clustering validation measures. In 2010 IEEE International Conference on Data Mining, page 911–916, Dec 2010.

CITED LITERATURE (continued)

22. Bergstra, J., Yamins, D., and Cox, D. D.: Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. In Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28, ICML'13, pages I–115–I–123, Atlanta, GA, USA, June 2013. JMLR.org.

23. Salvador, S. and Chan, P.: Toward accurate dynamic time warping in linear time and space. Intelligent Data Analysis, 11(5):561–580, Jan 2007.

24. Sammour, M., Othman, Z. A., Rus, A. M. M., and Mohamed, R.: Modified dynamic time warping for hierarchical clustering. International Journal on Advanced Science, Engineering and Information Technology, 9(5):1481–1487, October 2019.

25. Zhang, Z., Tavenard, R., Bailly, A., Tang, X., Tang, P., and Corpetti, T.: Dynamic time warping under limited warping path length. Information Sciences, 393:91–107, July 2017.

26. Li, H., Liu, J., Yang, Z., Liu, W., Wu, K., and Wan, Y.: Adaptively constrained dynamic time warping for time series classification and clustering. Information Sciences, 534, May 2020.

27. Ganganath, N., Cheng, C.-T., and Tse, C. K.: Data clustering with cluster size constraints using a modified k-means algorithm. In 2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, pages 158–161, October 2014.

28. Wagstaff, K., Cardie, C., Rogers, S., and Schrödl, S.: Constrained k-means clustering with background knowledge. In Proceedings of the Eighteenth International Conference on Machine Learning, ICML '01, pages 577–584, San Francisco, CA, USA, June 2001. Morgan Kaufmann Publishers Inc.

29. Huang, P., Yao, P., Hao, Z., Peng, H., and Guo, L.: Improved constrained k-means algorithm for clustering with domain knowledge. Mathematics, 9(19):2390, January 2021.

30. Van Craenendonck, T., Meert, W., Dumančić, S., and Blockeel, H.: Cobrasts: A new approach to semi-supervised clustering of time series. In Discovery Science, eds. L. Soldatova, J. Vanschoren, G. Papadopoulos, and M. Ceci, Lecture Notes in Computer Science, pages 179–193, Cham, 2018. Springer International Publishing.

CITED LITERATURE (continued)

31. pandas.dataframe.ewm — pandas 2.0.2 documentation.

32. Chakrabarti, D., Kumar, R., and Tomkins, A.: Evolutionary clustering. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '06, pages 554–560, New York, NY, USA, August 2006. Association for Computing Machinery.

VITA

NAME	Davide Porello

EDUCATION

	Master of Science in Computer Science, University of Illinois at Chicago, May 2023, USA
	Master of Science in Computer Engineering, Jul 2023, Politecnico di Torino, Italy
	Bachelor's Degree in Computer Engineering, Jul 2018, Politecnico di Torino, Italy

LANGUAGE SKILLS

Italian	Native speaker
English	Professional working proficiency
	2021 - IELTS examination (7.0/9)
	A.Y. 2022/23 One year of study abroad in Chicago, Illinois
	A.Y. 2021/22. Lessons and exams attended exclusively in English

SCHOLARSHIPS

Spring 2023	Graduate Hourly Employee (15 hours/week) with monthly stipend
Fall 2022	Italian scholarship for TOP-UIC students

TECHNICAL SKILLS

Basic level	Data Analysis, Linux OS
Average level	Machine Learning, Software testing
Advanced level	Algorithm, Data Structures, Software Development

VITA (continued)

WORK EXPERIENCE AND PROJECTS

2023	SneakerScanner website: develop and host a website that reunites all the sneaker offers from different sites in a simple web interface leading the users to the original website for purchase, implemented frontend in React and backend using serverless and scheduled functions to retrieve data from third-party APIs and notify users if the price falls below a specified threshold, Implemented database using Supabase (SQL DB) with Redis as cache.
	Android applications: develop Android applications to learn the basic principles of Android using Android Studio and following OOP design patterns.
2022	Detection and evaluation of bias in machine learning: implement the approach described in the "Detection and Evaluation of Machine Learning Bias" research paper applying C++ parallelization techniques and measuring their performance, implementation of machine learning models (linear and polynomial regression) from scratch, evaluation of the bias using alternation function and KL divergence metric, implementation of concurrent k-fold cross validation using C++ ThreadPool.
	StudyPlan website: develop a client-server website to organize the study plan of the university, implemented frontend in React and backend using Node and Express, the server exposes some APIs that the client uses for CRUD operations.