ENGIE E4800: Data Science Capstone

First Progress Report

J.P. Morgan: Trading Opportunities

Romane Goldmuntz, Hritik Jain, Kassiani Papasotiriou,
Amaury Sudrie, Maxime Tchibozo, Vy Tran

23 October 2020

Contents

1 Introduction

1.1 General Overview

Financial analysts typically make investment decisions by recognizing patterns in financial stock time series. These patterns - commonly referred to as chart patterns - are based on visual cues identified with expert domain knowledge, and are largely subjective.

This project aims to develop data-driven methods to identify profitable patterns in time series in an unsupervised setting. We develop these methods under a technical analysis framework, which assumes that all the information relevant to a stock's performance (company fundamentals, market behavior, geopolitical context) is immediately reflected within the stock price, with no need to consider additional exogenous variables.

Unlike most research into financial time series clustering, we perform multi-scale clustering, and identify patterns which simultaneously reflect short-term variations in the recent past as well as long-term trends in the stock price. This project is a collaboration between the Columbia University Data Science Institute (DSI) and J.P. Morgan AI Research.

1.1.1 Problem Definition

We aim to provide a data-driven solution to the following financial Machine Learning problems:

- How can we recognize patterns which separate time series data while capturing complex multi-scale variations (multiple harmonics, non-trivial oscillatory motions) in an unsupervised setting?
- How can we evaluate whether patterns learned through feature engineering and clustering are good indicators of long-term potential profit or short-term risk?

This Capstone report details our advances in data scraping, exploratory data analysis, feature engineering, and developing initial clustering methods. We outline our approach, the main obstacles we have identified so far, and the next steps we will undertake to solve these challenges.

1.1.2 Multi-scale clustering

When analyzing financial markets, it is important to evaluate a stock's performance in the immediate past, while also taking into account its general trend. For example, a financial analyst would make a trading decision based on the value of a stock over the past 5 days, but would likely also take into account its longer-term context (for example whether the stock's behavior over the past month is "bullish" or "bearish"). Finding a balance between the long term and immediate variations of the stock ensures they do not make unprofitable bets, or miss out on brief yet profitable opportunities. The notion of scale is therefore essential for an algorithm to accurately capture properties of stock data which appear intuitive to humans.

To capture the multi-scale properties of the data, we explore data processing methods inspired from Financial Engineering such as Dynamic Time Warping (DTW) and Perceptually Important Points (PIP), from signal processing (fourier transforms) or deep learning (autoencoders) and we also propose a variety of novel methods (exponential weighting, skip sampling, padded sampling).

1.1.3 Filter clustering results for profitable signals

After identifying clusters which appear similar to existing chart patterns, our goal is to filter the clustered time series based on their potential for future profit/loss. Our definition of the profitability of a pattern will be guided by observations from trading practices in the stock market. We will explore approaches to filtering clustered time series for profitable patterns in the second stage of our project.

1.2 Previous Attempt

A J.P. Morgan summer intern previously attempted to identify profitable patterns in time series through unsupervised learning. In their analysis, 50,000 chunks of 50 days of data were randomly sampled using bootstrapping. Different embedding and clustering methods were applied to the subsampled data, but the results obtained for KMeans Clustering did not show particular profitable or unprofitable patterns in each cluster. The centroids of the clusters resembled harmonic waves of different frequencies, which indicates that they cannot capture interesting features of the time series.

To extend this work, we have implemented more sophisticated feature engineering and clustering methods. In addition to Euclidean distance, we are going to use metrics that are tailored to measure distance between time series. We will also look into different ways of evaluating our clusters and decide on the optimal number of clusters that we should use, which will be tackled in the second phase of the project.

2 Data sources and data preparation

In this section, we describe the methods used to prepare the dataset and present the logic behind the data transformations we performed. Our objective is to bring the raw price data into a meaningful format for our clustering analyses.

2.1 Random sampling

Our dataset consists of daily adjusted close prices of the S&P 500 stocks from 1999 to 2020. This data is publicly available using the Yahoo Finance API.

In this project, we are not exploring the full time series, but subsequences of them. A subsequence of size p of a time series X is obtained by selecting p consecutive days inside of X. Depending on the value of p, the number of possible subsequences in our stock can easily reach 1 or 2 millions. Besides, a lot of subsequences tend to be very similar to each other if they are shifted by only a few number of days. For these two reasons, we need to work on a smaller subsample of randomly picked subsequences (typically 1000 or 2000).

To ensure the conditions mentioned above we generate our dataset by randomly selecting subsequences among all available subsequences, regardless of the stock they come from.

2.2 Length of time series

The multiscale aspect of the project requires us to consider two types of time scale: a short-term one and a context one.

- The short-term scale should be up to date and reflect the current behavior of a stock. We chose it to be 20 days which is equivalent to 1 trading month. Beyond this, we consider the information to be obsolete.
- The context scale should explain what is the general behavior of a stock price during the past few months. We picked 60 days or 100 days, equivalent respectively to 3 and 5 trading months, depending on how many days or features we wanted to try a particular method.

2.3 Normalization & Standardization

The need to normalize the time series data arises from the fact that, even though raw stock price data has meaning in the original finance context, under our project's machine learning perspective, unprocessed data would lead to meaningless results. Particularly, a naive approach that directly clusters on the raw data, would group together time series that have the same price scale instead of similar price patterns.

Therefore, in order to account for the aforementioned problem, we use a fixed-window Min-Max Normalization method which will transform our features between [0,1]. We picked this method because we do know or expect that the subsequences in our dataset follow a specific distribution and, additionally, the clustering algorithm that we use (K Means) doesn't make any assumptions about the distribution of our data.

The fixed-window Min-Max Normalization is obtained by applying the following expression to each subsequence v in order to obtain the new value v' [1]:

$$v'_i = \frac{v_i - \min(v)}{\max(v) - \min(v)}$$

The result of applying this transformation to the raw data will result in a transformed dataset where all time series range from [0,1] which is more appropriate for our clustering objective

3 Clustering Pipeline

3.1 KMeans

Several clustering techniques have been established in the unsupervised learning literature. During the first stage of this project, we use the K-means algorithm, which is the most commonly used technique due to its simplicity.

K-means is an iterative algorithm that tries to partition a dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. In our case, one data point corresponds to a randomly sampled subsequence belonging to a stock price's time series, as explained in Section 2. K-means tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster [2]. Next we will discuss two distance metrics that we used for time series clustering using K-means.

3.1.1 Euclidean distance

The most common way to calculate the distance between two points is the Euclidean distance. For two points a and b, this metric assumes that they are the furthest vertices of a right triangle and calculates the length of the hypotenuse of the triangle. This is how most people who are not familiar with math would define distance on a map. The suitability of using Euclidean distance as a metric largely depends on the problem. It has the advantage that most optimization methods are designed with Euclidean distance in mind and the computational costs can be well constrained.

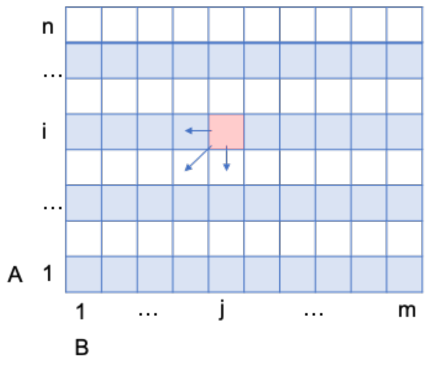
In general, for two points p and q given by Cartesian coordinates in an n-dimensional Euclidean space, the distance can be calculated as:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2}$$

3.1.2 DTW

Dynamic Time Warping (DTW) is a technique used to measure similarity between two time series. Unlike Euclidean distance, it can capture the similarity of two time series even when they are out of sync in terms of time and speed. For example, Euclidean distance function will produce a poor similarity score between two audio recordings of a person speaking at different speeds [3].

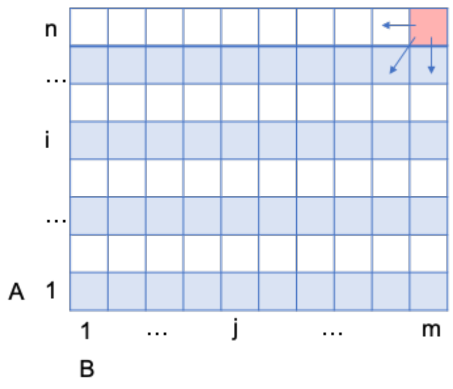
Assume that we have 2 time series A and B with lengths n and m, respectively, and that we want to calculate the similarity between A and B. DTW will first calculate the Euclidean distance between each pair of time points (i, j) with $i = 1, \ldots, n$ and $j = 1, \ldots, m$ using Dynamic Programming, then fill in every possible value in this table:



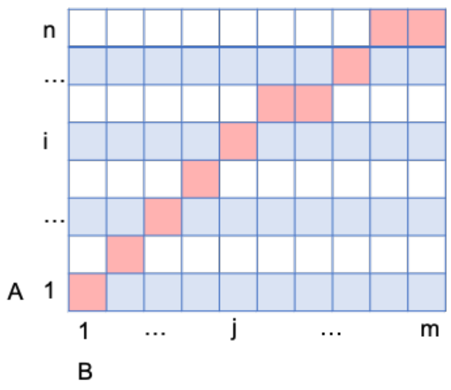
Use Dynamic Programming to fill in D(i,j) in this table

$$D(i,j) = D(i,j) + min \begin{cases} D(i\text{-}1,j) \\ D(i,j\text{-}1) \\ D(i\text{-}1,j\text{-}1) \end{cases}$$

After every value in the above table is filled in, we'll find the Warp Path from point (m, n) back to point $(1, 1)$ by first choosing the smallest value around point (m,n) and then progress to point $(1, 1)$ using the same technique. After the Warp Path is found, the similarity between time series A and B will be the sum of all the values belonging to the Warp Path [4].



Find Warp Path starting from point (m,n) going to point (1,1) by choosing the smallest value



Sum of all the values in the Warped Path will be the similarity between A and B

When it comes to averaging sequences of time series that are grouped together using DTW, an appropriate averaging method is very important. We'll be using DTW Barycenter Averaging (DBA) algorithm to find the centroid of each cluster.

DBA starts with some random guess about the centroid of the cluster, which can be the medoid or a random time series that belongs to the cluster. At the same time, DBA uses DTW to find the optimal alignment between the random centroid and each of the time series in the same cluster. As a result of DTW, each point p in the random centroid will be "tied" to more than one points of each time series. Then for each point p, calculate the Barycenter c from all the points that are tied to p, and update p to c. We'll repeat the process until convergence criteria are met. At the end, we'll have a centroid that best represents the time series belonging to that cluster [5].

3.2 Multiscale

3.2.1 Skipped Values

Our first approach to capture multiscale patterns in time series is for each data point of 60 consecutive stock price values to include two context scales. The first context scale will skip the price every alternative day, replacing the price of that day by 0, and the other will skip the price every two days, replacing the two prices by 0 as well. By adding these two dimensions we give more weight to the points that capture a general trend instead of the day-to-day price fluctuations. This will lead to a (60,3) matrix representation, each column accounting for one scale and each row accounting of a day. The result is depicted in Figure 1.

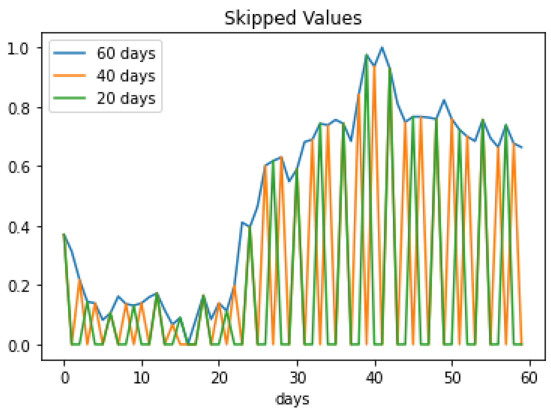


Figure 1: Skipped Values Datapoint

After performing K means clustering with k=10 and distance metric DTW, we obtained the following clusters.

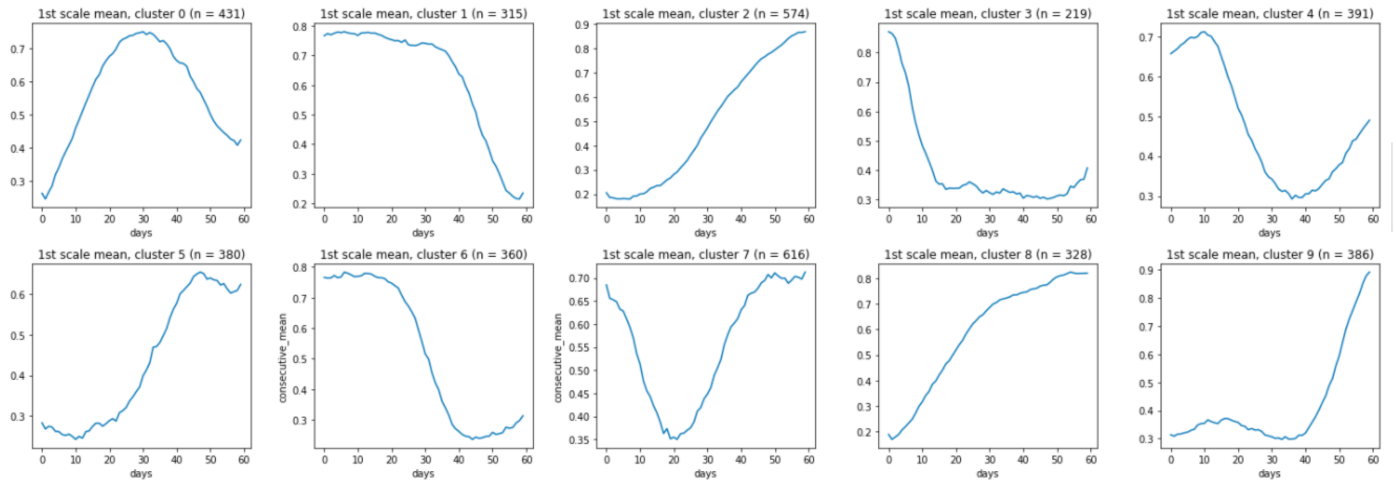


Figure 2: Cluster results (mean of all first scales within each cluster) for Skipped Values Method. We omitted the two context dimensions for visualization purposes.

Figure 2 shows the mean of the first scale of all time series within a cluster. More specifically, all first scale values of cluster i on day j were averaged and the result is plotted above.

This visualization enabled us to make the following observations: the number of time series within each cluster is around 300-400 for most of the clusters, with the exception of cluster 3, which contains around 200 time series, cluster 2 and cluster 7, which both contains about 600 observations. We can also notice that clusters 2 and 3's first scale averages, along with the ones of cluster 1 and cluster 8, do not lead to harmonic signals but have a clear upward trend (cluster 2, cluster 8) or downward trend (cluster 1, cluster 3).

3.2.2 Padded Values

Similar to the Skipped Values approach, the Padded Values method also includes two context scales for each time series. The first context scale includes the 40 most recent stock price data, replacing the 20 oldest prices by 0s, and the second scale includes the 20 most recent stock price data, replacing the 40 oldest prices by 0s. By adding these two dimensions we give more weight to the most recent 40 and 20 days of the timeseries. The resulting (60,3) data point is depicted in Figure 3.

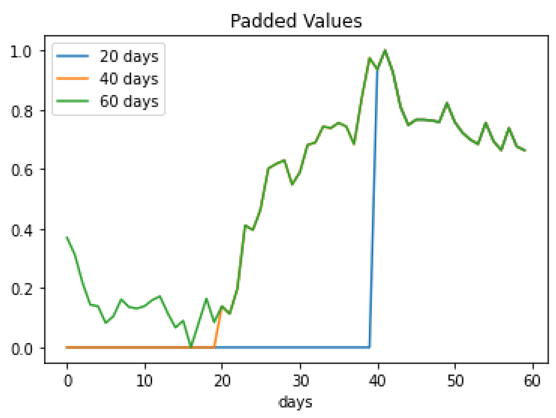


Figure 3: Padded Values Datapoint

After performing K means clustering with $k = 10$ and distance metric DTW, we obtained the following clusters.

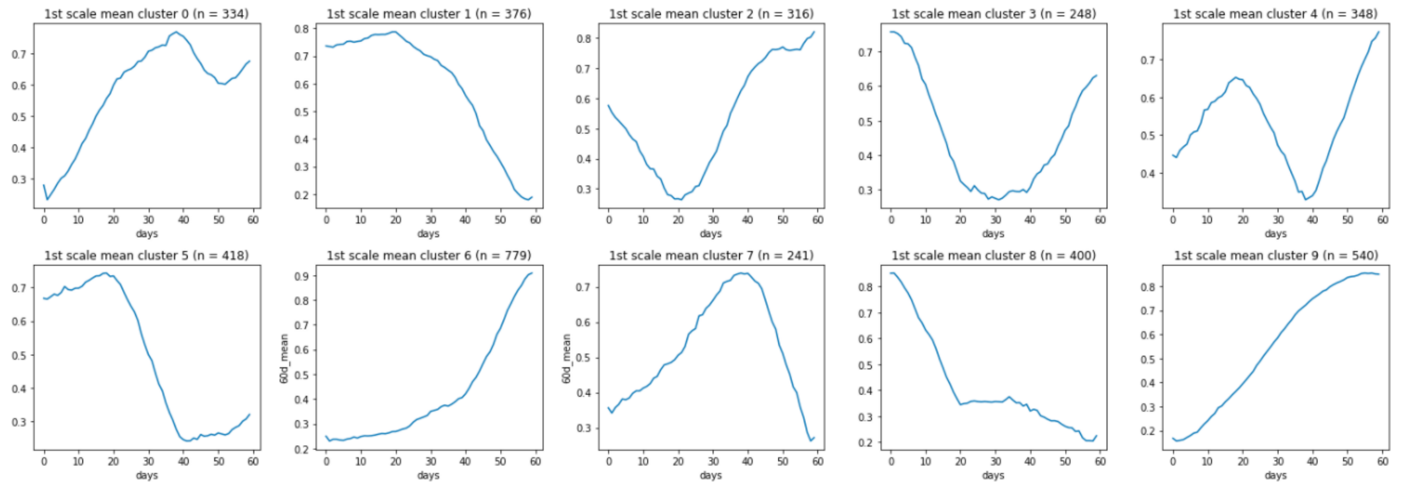


Figure 4: Cluster results (mean of all first scales within each cluster) for Padded Values Method. We omitted the two context dimensions for visualization purposes.

Figure 4 follows the same reasoning as Figure 2 (for the Skipped Value Method) and shows the mean of the first scale of all time series within a cluster.

This visualization enabled us to make the following observations: the number of time series within each cluster is again pretty balanced, with around 200-400 for most of the clusters, with the exception of cluster 9 (540 time series) and cluster 6 (780 time series). We can also observe a clear upward trend in cluster 6 and cluster 9, and a clear downward trend for cluster 1 and cluster 8.

3.2.3 PIP Embedding

Another approach to create multi scale subsequences can be achieved by using Perceptually Important Points (PIP) which is a dimensionality reduction method of time series. PIP's strength lies in its ability to preserve the overall shape of the time series by identifying the most noticeable points [6].

For a subsequence of 60 consecutive stock prices, we want to capture both the context scale (60 days) and the shorter scale (20 most recent days). If we want to capsule this information into a data point, we would need to reduce the 60 day point to a 20 day point to match their dimensions. Additionally, we want to perform this dimensionality reduction and preserve as much information as possible. This can be achieved using PIP. (Figure 5a.)

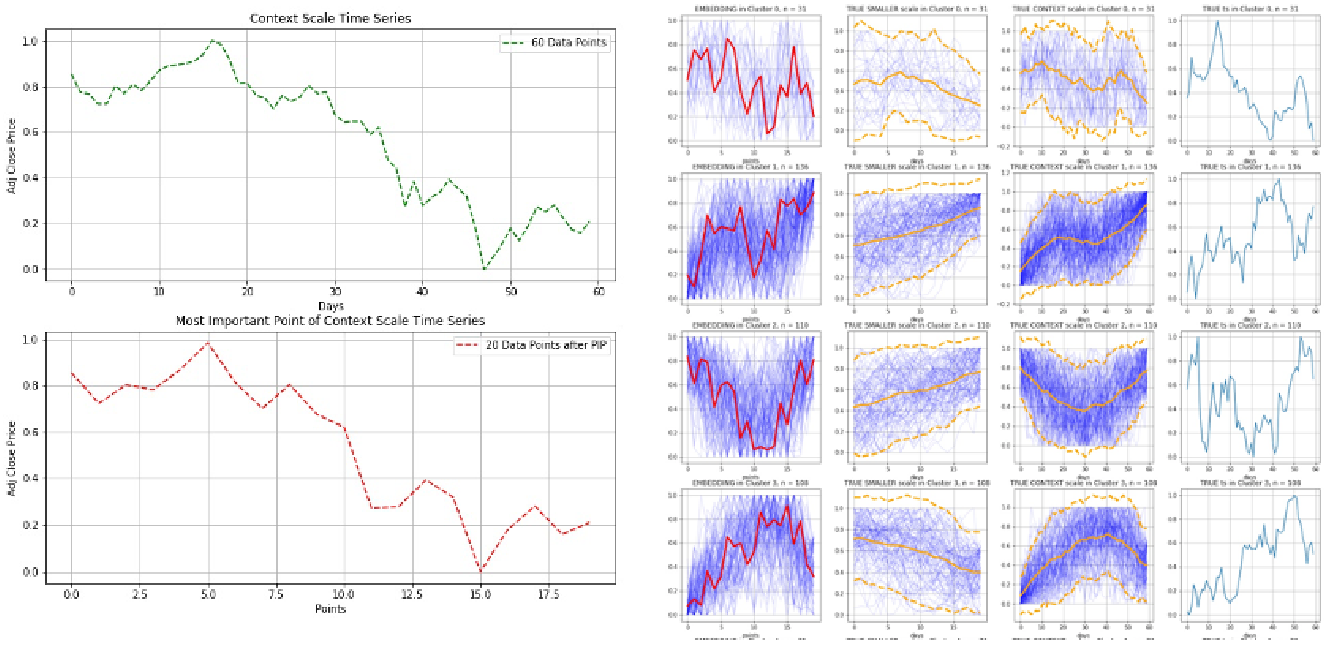


Figure 5: a(Left): Top figure is a subsequence of shape (1,60), Bottom figure is the same datapoint reduced to a dimension of (1,20) using PIP. b(Right): Results of the first 4 clusters on PIP Embeddings

Since PIP can capture the most important points of a time series, we want to see if clustering on PIP Embeddings only is sufficient to catch both important patterns on the context and the short term scales. We applied KMeans Clustering with 10 clusters using Dynamic Time Warping as our metric.

Figure 5b. shows the results for the first 4 clusters. The first vertical set of plots are what we cluster on, which are the PIP Embedding of length 20 obtained from the context scale time series. The red line is the centroid found using the DBA algorithm. The second and third plots are the true smaller and context scale time series that correspond to the embeddings of the cluster. Final plot is one example of the context scale time series belonging to that cluster.

We can see that the centroid can capture most of the important features of a time series chosen randomly from the cluster. However, the means of the smaller and the context scales associated with the embeddings in one cluster look very similar to harmonic waves. As a result, we are not catching any complex patterns by clustering on the PIP Embedding only.

3.2.4 Autoencoder Embedding

Another way to match the dimensions of the short-term and context scale is to use autoencoder neural networks. The particularity of these networks is that the output should be equal to the input. In other words these networks are learning the input. These networks are composed of an encoder which reduces the dimension and a decoder which rebuilds the input from the embedding. Once the autoencoder is trained we just need to use the encoder to generate the embedding of the desired length.

We tried several architectures such as fully connected linear layers or LSTM. They mainly perform the same, and we will present the result of the architecture with one hidden layer. As the context scale is 60 days, we reduce the dimension to get 20 features. We then concatenate with the short term scale in order to perform clustering. The dataset used was composed of 2000 time series. Below, in Figure X we show the results after using KMeans on 10 clusters. For visualization purposes only 5 out of 10 are displayed.

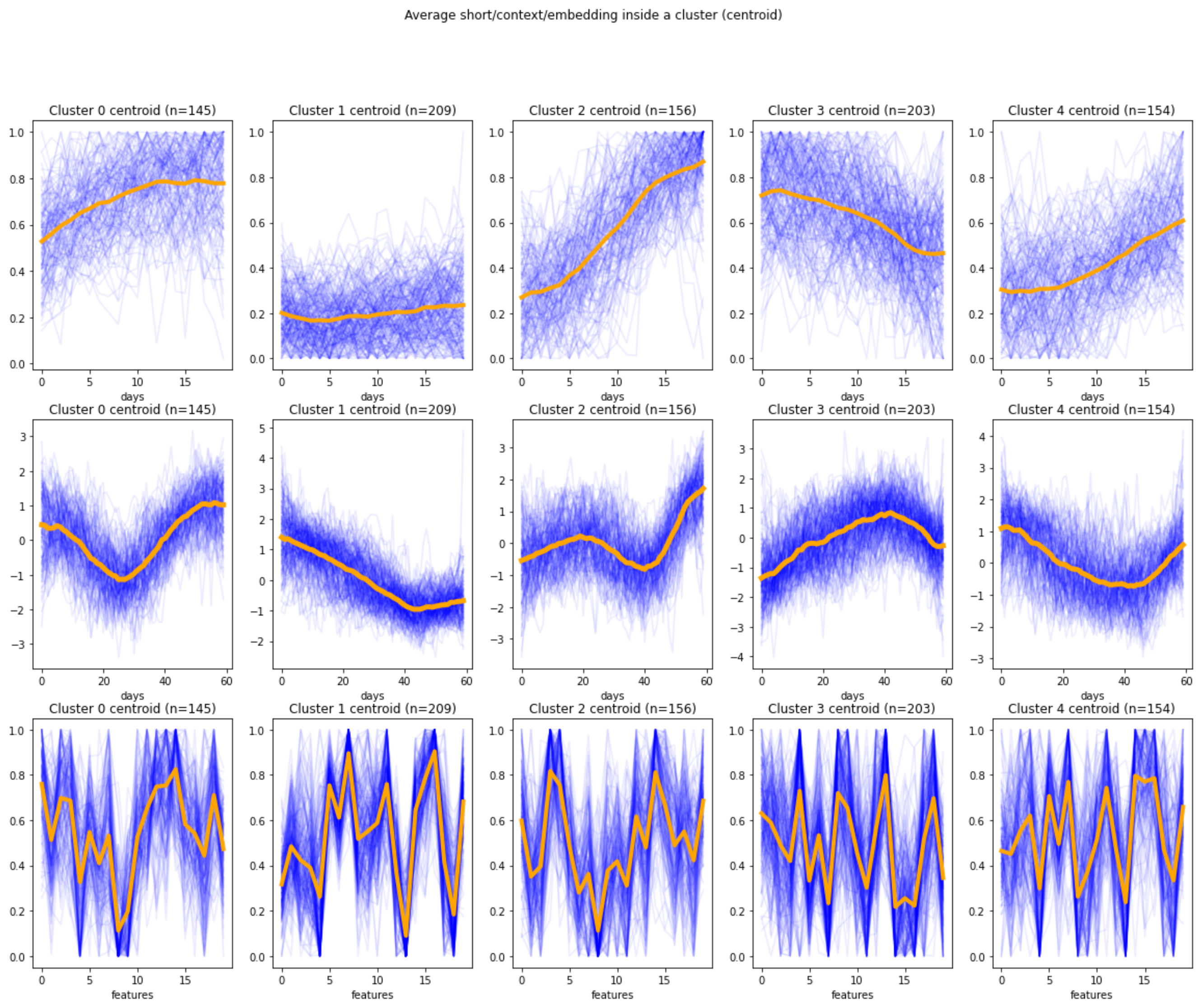


Figure 6: 5 clusters out of 10 by clustering on the concatenation of the short-term scale and the embedding features.

On the Figure 6 above, the first row represents the average short-term time series with its alpha plot for each cluster. The second row accounts for the same variable for context scale, and the third one shows embedding features.

The short-term scale row shows linear centroids. However, the alpha plot is highly spread around each centroid and we are therefore not catching any complex pattern.

Looking at the embedding clustering row, it seems that KMeans is actually clustering on the embedding. The alpha plot is more compressed around the centroid curve. Nonetheless, looking at the context scale row, the conclusions are the same as the short-term one. In other words, even if we catch patterns in the context embedding it is not sufficient to catch patterns in the context scale time series.

3.2.5 Weighting time series

Another approach that we have pursued to achieve the representation of multiple time scales in our clustering analyses is the idea of weighting the different points in a time series nonuniformly.

From the formulation of Euclidean distance explained in 3.1.1, it is clear that all time points [1, n] will be given the same importance when calculating the distance between any two time series p and q. However, there are situations where this is not desirable, especially with time series data. It is frequent that with this kind of data the most recent values should be given higher weights, as they are most important for the analysis. This is indeed the case with stock price time series - it is common that investors want to know which stocks are correlated, but the recent behavior of the stocks is certainly more important for them than what happened, for instance, one year ago. In order to take this into account, we will define a weighted measure of correlation between two time series, updating the previous equation by weighting the different time points as:

$$d(p,q) = \sqrt{w_1(p_1-q_1)^2 + w_2(p_2-q_2)^2 + \cdots + w_i(p_i-q_i)^2 + \cdots + w_n(p_n-q_n)^2}$$

In our analysis we use the progression of weights as $w_j = \alpha^j$, for a suitable choice of α (slightly above 1). Our observations fell in line with what we expected with our weighting scheme. The most recent days in history are weighted significantly higher, and the time series clustered together tend to be very close to each other in the first few days. The variance in the prices increases as we go further back in the history.

We consider two different approaches to incorporate a longer timescale context as well as shorter timescale context when clustering the different time series. The idea is that two stocks may display a very similar trend (for example, a sharp increase) in a short span of time, one week for instance, but this may or may not be indicative of a true similarity of their trends. A comparison of over a longer timescale increases our confidence in terms of future performance predictions and investment targets.

In our first approach, we increasingly weigh the oldest time points in addition to weighting the most recent time points. That is the weights decrease as we go back in history, but then increase to get the influence of the oldest prices in the available time series.

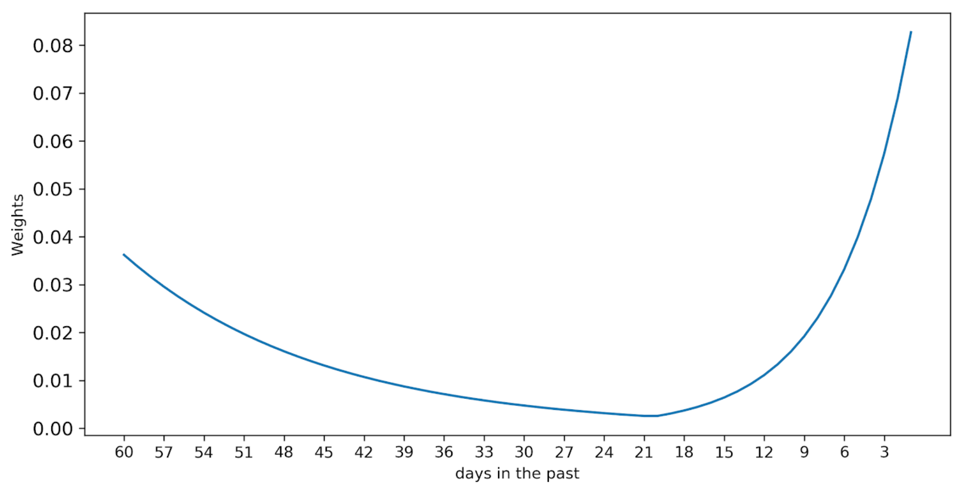
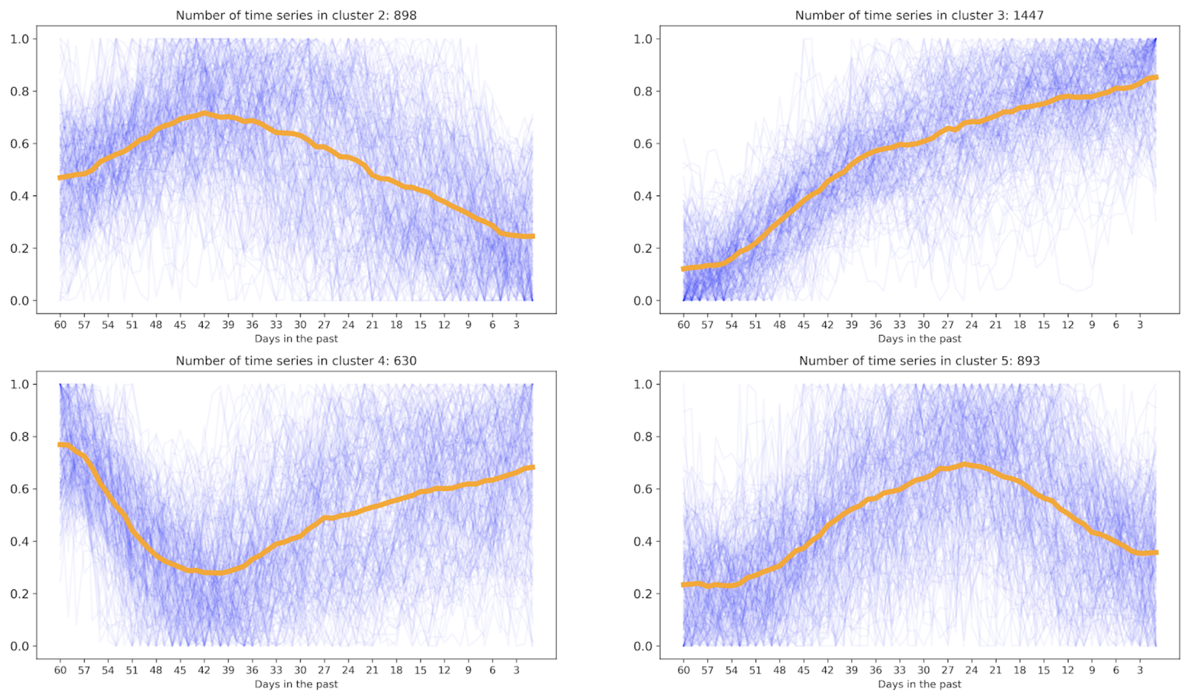


Figure 7: α for the forward progression of weights (recent past) is set to 1.2, while that for the backward progression of weights is set 1.07. The weights are such that the oldest stock price is weighted about half as much of the latest stock price. The length of the time series on the longer scale is 60 days.

The figure below shows 4 of 10 sample clusters obtained from this weighted clustering approach. As anticipated, the time series clustered together tend to be close to each other on the tails, while having larger variance in the middle.



We found that it is hard to assess the utility of these clustering results in the context of future predictions. In the second approach, we will perform two rounds of clustering in a nested manner. First round will group the time series based on their similarity over a longer timescale, followed by a second round to further cluster time series based on similarity on a much shorter timescale within each of these initial clusters.

3.3 Fourier Transforms

Fourier transform is an alternative way of expressing a time-domain function by decomposing it into its constituent components and frequencies [7]. This representation relies on the following assumptions:

- The periodicity of the signal
- A sampling frequency Fs such as $1/T = F_s/N$ where T is the period of the signal and N is the number of samples
- The Nyquist-Shannon theorem: $F_{\max} = 2 \cdot F_s$ [8].

We decided to try to cluster the Fourier transforms of time series instead of the time series themselves. Before clustering the Fourier transforms, we decided to implement a sanity check to assess the quality of our new representations. This sanity check consists in taking the inverse Fourier transform of the time series to make sure that we can retrieve the original signal. In Figure 8 below, the upper-left graph represents an original time series and the upper-middle graph its Fourier transform coefficients. The sanity check results are shown in the upper-right graph and we can see that the original signal is perfectly reconstructed.

However, Fourier transforms are complex numbers and some clustering algorithms such as K-Means do not accept complex data as input, and we therefore had to find a way around it. The first solution that we tried was clustering only the modulus of the complex numbers. This processing is represented in the second line of Figure 8, and we can see that the sanity check fails and that the original signal cannot be retrieved.

We then decided to go with an alternative method called Discrete Cosine Transform, which is further explained in the next section.

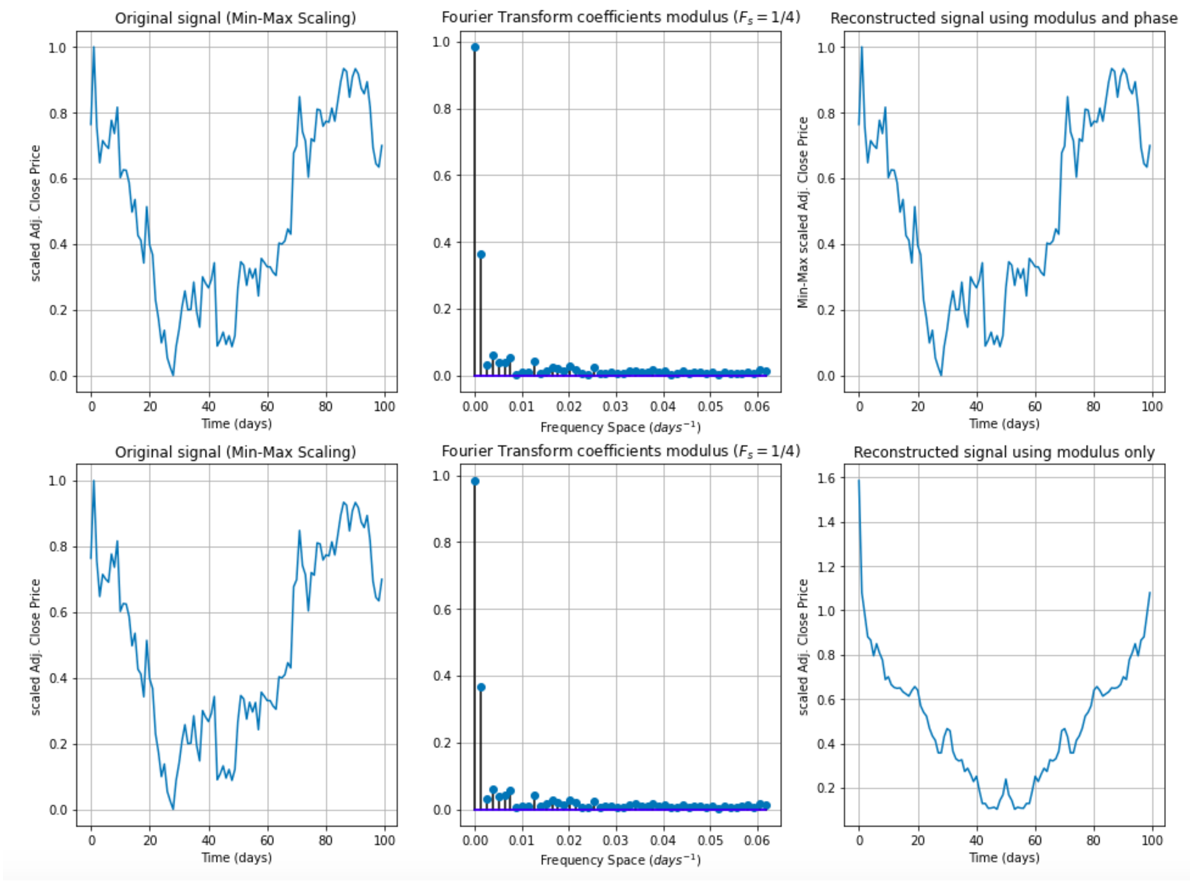


Figure 8: Two financial time series with their Fourier transform coefficients

3.4 Discrete Cosine Transform

Discrete Cosine Transforms (DCT) are a family of Fourier Transforms (FT) which use only real-valued coefficients, making them more naturally suited for clustering than the complex coefficients of FT. The DCT also alleviates issues which limit FT, such as sensitivity to boundary discontinuities.

The DCT is invertible, meaning a signal can be represented using only its DCT coefficients without loss of information (lossless compression), and has powerful properties such as linearity and symmetry. From a Machine Learning perspective, we view the DCT as an embedding transformation to be applied to the stock data ahead of an unsupervised clustering task. We applied KMeans clustering to the DCT coefficients directly, in the hope that clustering in the DCT space would group financial time series together based on multi-scale properties.

We found that KMeans clustering on DCT coefficients captured interesting properties of stock market data (Fig.9. multiple oscillations: cluster #0, stock rallies: cluster #2, rebounds: cluster #3). We believe signal processing transformations such as DCT are promising and worthy of more in-depth analysis.

Like FT, DCT are subject to limitations, most notably their sensitivity to noise and reliance on the assumption that signals are periodic. In the future, we aim to use a smaller number of FT and DCT coefficients to remove noise, as well as explore signal processing embedding methods such as Discrete Wavelet Transforms (DWT), which have translation, scale and dilation-invariance properties which could be well-suited to financial Time Series clustering.

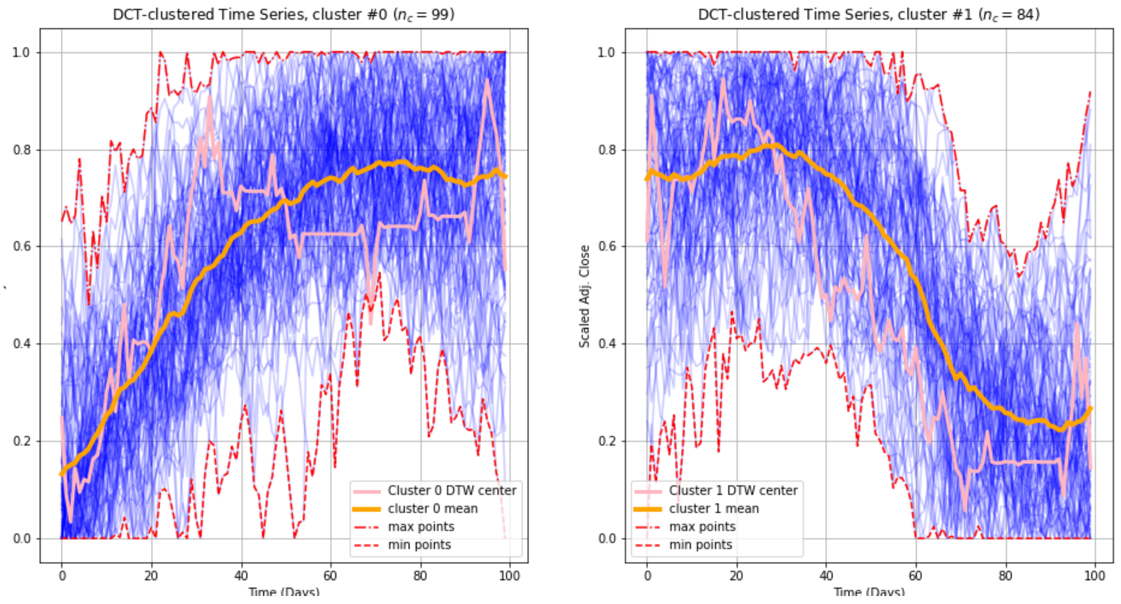


Figure 9: KMeans clustering of DCT coefficient embeddings

4 Next steps

4.1 More Evaluation Metrics

4.1.1 Qualitative metrics

Qualitative metrics are the first tool to feel the quality of our clustering result. Qualitative metrics are visual tools to evaluate the performance of the clustering strategy. There are mainly two such tools we used:

- The shape of the centroid. If the centroid curve is mainly flatten it gives no interesting trend or patterns. Having linear or harmonic trends shows the lack of more complex patterns. Finally a spiky centroid might be able to catch unusual patterns we should look into.
- The alpha plot around the centroid shows how representative is the centroid for this cluster. An alpha plot spread around the centroid shows the inability of this cluster to catch complex patterns. However, a compressed alpha plot proves that the centroid actually explains all the time series inside that cluster.

4.1.2 Quantitative metrics

We need a quantitative metric to evaluate the quality of our clusters and to make strategic decisions regarding data processing or clustering algorithms. The metric we decided to settle with in a first instance is called Silhouette Score and is computed as following:

$$S_i = \frac{B_i - A_i}{\max(A_i, B_i)}$$

Where

- S_i is the Silhouette Score for time series i
- A_i is the mean intra-cluster distance, i.e. the mean of all the distances between time series i and all the other time series within the same cluster
- B_i is the mean inter-cluster distance, i.e. the mean of all the distances between time series i and all the other time series from other clusters.

The Silhouette Score varies between -1 and 1. A score of -1 means that the time series has been wrongly assigned to its current cluster and should belong to a different one, a score of 1 means that we are confident the time series belongs to its assigned cluster, and a score close to 0 means that the time series is in between two clusters [9].
We are planning to apply this score in three different ways.

First, for every combination of data processing and clustering method, we want to compute the mean Silhouette Score over all time series. This will give us a first assessment of how well our pipeline is performing on our dataset.

Second, we want to use this mean Silhouette Score to optimize relevant parameters in the selected clustering method. For example, to optimize the number of cluster k for K-Means algorithm.

Finally, we wrote a function that computes the percentage of time series within each cluster that received a score above a certain threshold. We are thinking about applying this function to assess the quality of the clustering by, for example, computing the percentage of time series within each cluster that received a silhouette score above 0.5. Another application for this function would be to consider the time series with a poor silhouette score as noise and to remove them from their respective clusters to design more accurate visualizations.

4.2 Alternative Clustering Methods

4.2.1 Optimization of number of clusters

So far in our analysis we have not implemented a robust method to pick k. Therefore, one of the improvements that we would like to integrate in our processing pipeline is to find the appropriate number of clusters k for the patterns using an appropriate metric. Below are some methods that we will explore in order to achieve this [10]:

- The "Elbow" Method [10]: in which the sum of squares at each number of clusters is calculated and graphed, and the user looks for a change of slope from steep to shallow (an elbow) to determine the optimal number of clusters.
- The Gap Statistic [11]: that compares the total within intra-cluster variation for different values of k with their expected values under null reference distribution of the data.
- The Silhouette Method [12]: that compares the silhouette score for each value of k as explained in the previous section.
- Calinski-Harabasz Index [13]: which is the ratio of the sum of between-clusters dispersion and of inter-cluster dispersion for all clusters (where dispersion is defined as the sum of distances squared).
- Davies-Bouldin Index [14]: signifies the average 'similarity' between clusters, where the similarity is a measure that compares the distance between clusters with the size of the clusters themselves

4.2.2 Other algorithms

Apart from optimizing the number of k, we would also like to explore other unsupervised clustering algorithms. Some of which are:

- Hierarchical Clustering (Agglomerative/Divisive) [15]: where the merges or splits of the data happens in a greedy manner.
- K-Shape Clustering [16]: a centroid-based clustering algorithm that can preserve the shapes of time-series sequences.
- DBSCAN [17]: which is an algorithm views clusters as areas of high density separated by areas of low density
- Kernel k-means [18] is an extension of the standard k-means clustering algorithm that identifies nonlinearly separable clusters

References

[1] S. Garcìa and et al, "Data preprocessing in data mining," *Springer*, 2015.

[2] D. Imad, "K-means clustering: Algorithm, applications, evaluation methods, and drawbacks," *Towards datascience*, 2018.

[3] S. Kyaagba, "Dynamic time warping with time series," *Medium*, 2018.

[4] T. Elena, "Dynamic time warping algorithm," 2017.

[5] P. François, K. Alain, and G. Pierre, "A global averaging method for dynamic time warping, with applications to clustering," *Pattern Recognition*, vol. 44, no. 3, pp. 678 – 693, 2011.

[6] T. Fu, Y. Hung, and F. Chung, "Improvement algorithms of perceptually important point identification for time series data mining," pp. 11–15, 2017.

[7] "Fourier transform," *Deep AI.*

[8] O. Khairul, "Deconstructing time series using fourier transform," *Medium*, 2020.

[9] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53 – 65, 1987.

[10] F. Pedregosa and et al, "Scikit-learn: Machine learning in python," *JMLR*, vol. 12, pp. 2825–2830, 2011.

[11] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a dataset via the gap statistic," vol. 63, pp. 411–423, 2000.

[12] "Selecting the number of clusters with silhouette analysis on kmeans clustering," *Scikit-learn documentation.*

[13] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics*, vol. 3, no. 1, pp. 1–27, 1974.

[14] "Davies-bouldin score," *Scikit-learn documentation.*

[15] "Hierarchical clustering," *Wikipedia.*

[16] J. Paparrizos and L. Gravano, "K-shape: Efficient and accurate clustering of time series," vol. 45, no. 1, 2016.

[17] "Dbscan," *Scikit-learn documentation.*

[18] G. Tzortzis and A. Likas, "The global kernel k-means clustering algorithm," pp. 1977–1984, 2008.