ENGIE E4800: Data Science Capstone

Second Progress Report

J.P. Morgan: Trading Opportunities

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

24 November 2020

Contents

1 Introduction, Objectives & Data

Financial analysts typically make investment decisions by recognizing patterns in financial stock time series. This project aims to develop data-driven methods to identify such patterns in time series in an unsupervised setting.

Unlike most research in financial time series clustering, we perform multi-scale clustering, identifying patterns which simultaneously reflect short-term variations in the recent past as well as long-term trends in the stock price.

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. We defined chunks of these time series with a long-term scale fixed at 60 days (3 trading months) and a short term scale fixed at 20 days (1 trading month). As this produces a very large amount of time series chunks, we restricted our dataset and randomly picked only several thousands of them.

2 Recap

In the previous progress report, we have implemented different preprocessing methods to capture the multiscale patterns of time series. We also gained some preliminary understanding about our clusters by applying KMeans clustering on the preprocessed data. In this section, we summarize our first approaches and past results.

2.1 Perceptually Important Points (PIP)

Perceptually Important Points (PIP) is a dimensionality reduction method for time series. PIP's strength lies in its ability to preserve the overall shape of the time series by identifying the most noticeable points. 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). 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 a Dynamic Time Warping distance metric. We found that the centroids of each cluster successfully captured the important features of most randomly selected time series from the same cluster. We also found that the cluster means of both smaller (recent past) and larger context (complete time series) embeddings for most clusters were similar to harmonic waves. This is a strong indicator that clustering on a PIP Embedding alone did not catch any complex multiscale patterns, and simply clustered signals according to their long-scale trend. In other words, a naive clustering on PIP embeddings did not yield significantly different results from a KMeans clustering of the default time series without any preprocessing.

2.2 Discrete Cosine Transform (DCT)

As is the case with most time series data, we can view financial data as a finite sequence of values that a signal can take over time. Using this framework, we can apply signal processing algorithms such as Discrete Cosine Transform (DCT) to financial time series data to extract or compress information about a specific stock.

If $(x_k)_{k=1}^N$ is a time series with N points, its DCT coefficients are: $X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}(n+\frac{1}{2})k\right]$

Similarly, we can perfectly retrieve the original signal$(x_k)_{k=1}^N$ from a sequence of DCT coefficients $(X_k)_{k=1}^N$ using the inverse DCT (IDCT): $x_k = \frac{1}{2}X_0 + \sum_{n=1}^{N-1} X_n \cos\left[\frac{\pi}{N}n(k+\frac{1}{2})\right]$

Previously, we found that applying Euclidean distance-based clustering algorithms such as KMeans to all the DCT coefficients directly yielded identical results to clustering on the initial time series.

We also used DCT for dimensionality reduction of a time series signal; while a perfect embedding of the original signal would require the use of all of the DCT coefficients, we were able to compress and partially reconstruct the original signal by using a subset of DCT coefficients. Clustering on this subset of coefficients yielded promising results, but raised issues such as sensitivity to noise and loss of the dimensionality of the original signal.

2.3 Fourier Transform

Fourier transform is an alternative way of expressing a time-domain function by decomposing it into its constituent components and frequencies [1]. This representation relies on the following assumptions:

- The periodicity of the signal
- A sampling frequency F_s 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: $F_{\max} = 2 \cdot F_s$ [2, 3]

Since the Fourier transform coefficients are complex numbers that cannot be directly fed to clustering algorithms, we clustered on two alternative forms of these coefficients. The first one was the concatenation of both real and imaginary parts of the coefficients, and the second was the modulus of the coefficients. We initially discarded the latter as it failed our sanity check [4], but after discussions with our mentors, we later reinstated this method into our research.

We then had to choose an optimal number of Fourier coefficients to represent each time series before clustering. The first coefficient of each Fourier transform accounts for the mean of the signal and can therefore be discarded. Likewise, beyond the 13th coefficient, coefficients are very close to 0 and can be mainly attributed to noise. We therefore selected the coefficients [2,13] to represent each time series.

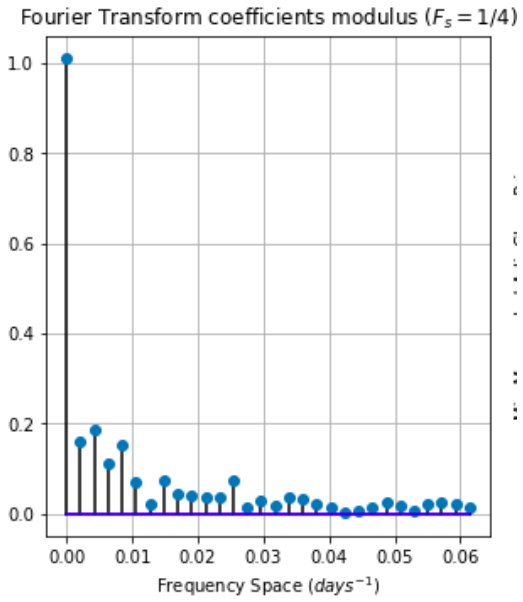

Figure 1: Fourier transform of one time series

2.4 Autoencoders

One 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 to compress and reconstruct the input. These networks consist 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 many autoencoder architectures, from linear layers to LSTM or CNN nonetheless, we found them to perform almost identically. As a consequence we eventually opted for a light architecture composed of a single hidden linear layer. Such a choice presents many advantages: training is faster, the overall model is more interpretable , and yields the same results as more complex architectures.

To perform clustering, we encoded the context scale of 60 days into 20 features which we then concatenated with a short term vector of values over the last 20 days. For each time series, we obtain a resulting multiscale vector of size 40 containing both long-term and short-term scale information.

2.5 Skipped and Padded Preprocessing

The two other preprocessing methods explored in the previous report were Skipped and Padded values. Our goal was to ensure the capture of multiscale properties of the data by adding two additional scales for each stock price time series of 60 consecutive days. More specifically, the two additional scales were:

- For Skipped values, the first context scale skipped the price every other day, and the second context scale skipped the price every two days. All skipped prices were replaced with 0.
- For Padded values, the first context scale included the 40 most recent stock price data, and the second context scale included the 20 most recent stock price data. All older prices were replaced with 0.

3 Evaluation of previous methods

The next step was to start clustering on the different methods described in the previous section and evaluate which one performs best, while simultaneously optimizing KMeans hyperparameters such as the optimal number of clusters "k". In order to do this, we combined the Silhouette score metric and the elbow method.

3.1 Metric we use

3.1.1 What is silhouette score?

The Silhouette score assesses the quality of a clustering partition by assigning a score which varies between -1 and 1 to each time series. A score close to -1 indicates that the time series has been assigned to the wrong cluster, a score close to 1 indicates that the time series has been assigned to the right cluster, and a score close to 0 indicates overlapping clusters [4]. In our selection, we focus on the Average Silhouette score, i.e. the mean silhouette score over all time series that were assigned to a cluster.

3.1.2 What is elbow method?

The Elbow method consists in displaying a specific metric (e.g. the average silhouette score in our case) as a function of the number of clusters k and selecting the elbow point of the curve as the optimal k [5]. The elbow of the curve is generally defined as the point after which adding an additional cluster no longer improves the specific metric (average silhouette score in our case) significantly.

The elbow method is intuitive, but can be defined in several ways. We chose a mathematical interpretation, defined as follows: we first draw the line between the first and the last values of the metric. We then compute the distance between each of the
orthogonal projection on that line). We then define
furthest from metric.

An example of how we applied this technique is illustrated below, with the case of autoencoders. The furthest point from the line to the silhouette line is at coordinates (7, 0.14), defining our optimal k^* as 7 with a mean silhouette score of 0.14.

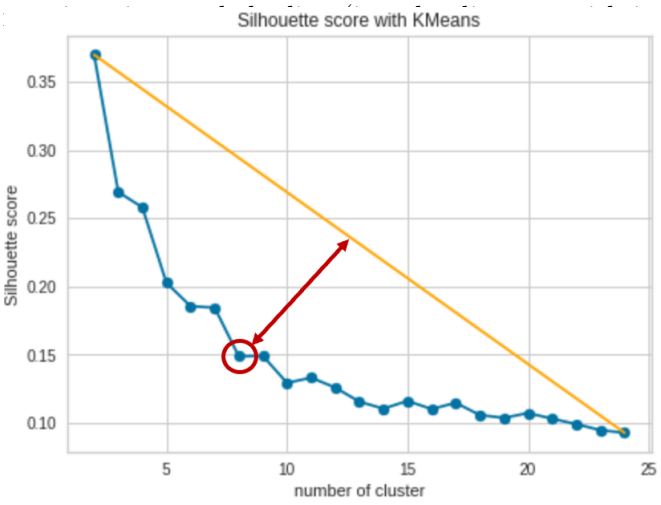

Figure 2: Picking optimal k with silhouette score and elbow method for autoencoders preprocessing

3.2 Performance of Preprocessing Methods

We applied this selection method on all of our preprocessing methods and the results can be found in the table below.

Preprocessing Method	**Optimal k**	**Average S-score for optimal k**
No Preprocessing (Baseline)	6	0.13
PIP Embedding Context Scale	7	0.02
DCT	5	0.16
Fourier Transform (Real)	6	0.17
Fourier Transform (Real + Imaginary)	9	0.14
Autoencoder	7	0.14
Skipped Values	3	0.01
Padded Values	3	0.05

The optimal value for k varies greatly across methods, while the Silhouette scores are overall close to 0. They range between 0.01 (Skipped Values) and 0.17 (Fourier Real Coefficients), which indicates that most time series are in between two clusters.

However, we noticed that DCT and Fourier Transform seemed to perform better than the other preprocessing techniques and we therefore decided to combine them with the other methods with the hope to improve our clustering overall performance.

Regarding Fourier Transform, since using only the modulus of the coefficients performed significantly better than concatenating both real and imaginary parts (0.17 vs 0.14), we will only consider the first method further on.

4 Merging Methods & Evaluation of those Methods

As detailed in the summary table of the previous section, the most promising methods in terms of forming clusters with a high average silhouette score were DCT and Fourier transforms. Therefore, the natural next step to our preprocessing exploration was to combine the "weaker methods" with those more promising techniques. By doing so we created preprocessing pipelines which both captured the multiscale patterns of the data and formed clusters which performed well with respect to our metric.

4.1 DCT smoothing

When using only a subset of the total Discrete Cosine Transform (DCT) coefficients, we found DCT to hold interesting properties for dimensionality reduction.

Notably, we found that varying the number of DCT coefficients used to reduce the dimensions of a time series led to varying silhouette scores. The fewer DCT coefficients were used for clustering, the higher the Silhouette score of the resulting clustered dataset.

This could lead one to mistakenly conclude that keeping a single DCT coefficient would yield the best clustering performance on financial time series. In reality, using fewer coefficients naturally increases the Silhouette score, regardless of whether or not these coefficients capture multi-scale patterns. A clustering algorithm which uses few coefficients

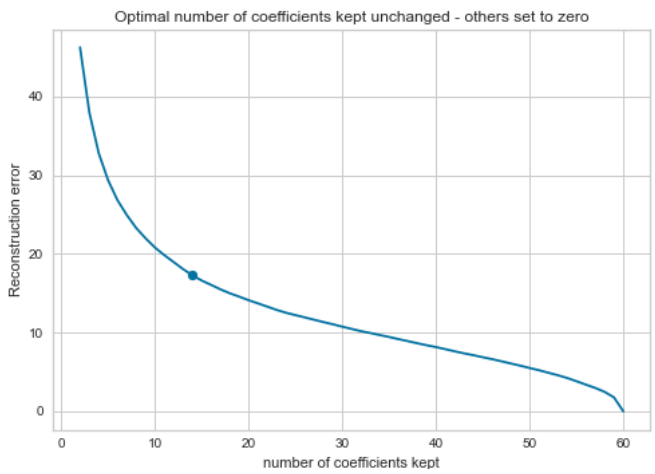

Figure 3: Elbow Method to pick the optimal number of DCT coefficients (optimal nb_coef = 13)

simply because this decreases the Silhouette score could mistakenly discard all information related to a time series' shorter scale and finer details. The resulting clustering algorithm would not cluster time series with similar multi-scale patterns together, and would instead only cluster time series according to their long term trend.

We used a data-driven approach to select DCT coefficients by leveraging the elbow method to pick an optimal number of DCT coefficients ahead of the clustering.
Applying the elbow method to the DCT coefficient selection problem ensured that we kept enough coefficients to accurately represent the multi-scale patterns in signal, while discarding coefficients which did not significantly improve the reconstruction of the original signal.

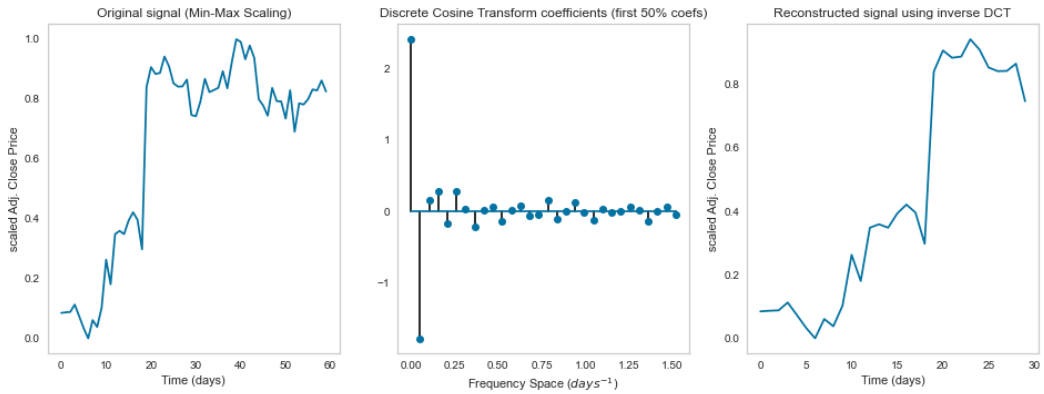

Figure 4: Reconstruction of a time series using only 50% of its DCT coefficients

Using a subset of DCT coefficients raised a new challenge: discarding DCT coefficients reduces the dimensionality of the reconstructed signal, which makes the reconstructed signal impossible to compare to the original. A stock time series of 60 days has an equivalent representation in the frequency space using 60 DCT coefficients. Selecting a subset of the first 50% of coefficients would lead to a reconstructed signal of 30 days (see above figure). While the 30 day reconstructed time series would visually have similar properties to the original 60-day time series, comparing the original and reconstructed signals quantitatively would be challenging, and whether or not multi-scale properties are contained within the reconstructed signal was usually unclear.

To solve this problem, we kept as many of DCT coefficients as there were points in the original time series, and set all the coefficients above the optimal number (identified earlier) to zero instead of removing them altogether.

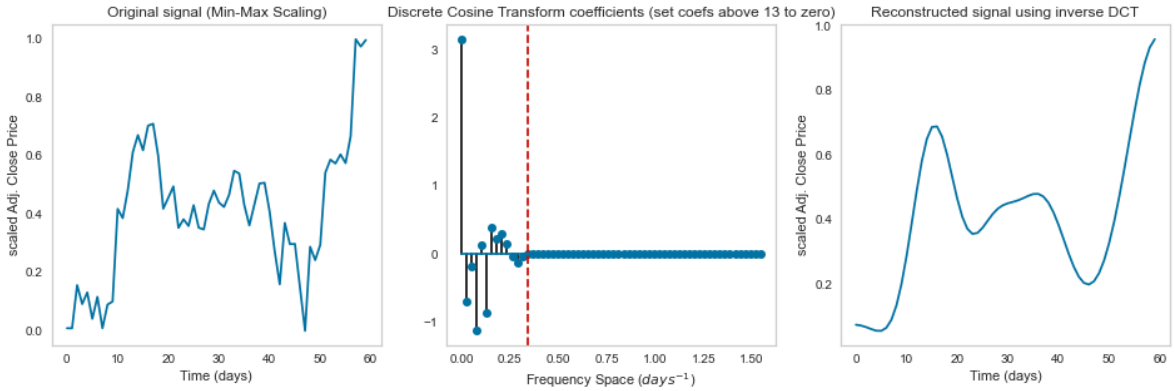

Figure 5: Reconstruction of a time series after setting all coefficients DCT above nb_coef = 13 to zero

This approach led to a reconstructed signal which preserved the dimensionality of the original signal, removed noise from the original signal, and captured multi-scale properties in a way which was easily quantifiable.

We view DCT as a smoothing preprocessing method, which we combined with other techniques ahead of the clustering task.

4.2 DCT & PIP

In this approach, we first applied PIP to reduce the dimension of the context scale from 60 days to 30 most important points. We then used DCT to smoothen the PIP Embedding time series of the context scale. We also experimented with different types of clustering such as KMeans, KMedoids and KMedians with k ranging from 6 to 10. We observed that for each number of clusters k, KMeans clustering gave the highest Average Silhouettes Score. Introducing DCT smoothing with PIP preprocessing increased the Average Silhouette Score from 0.020 to 0.164.

Combinations	$k = 10$	$k = 9$	$k = 8$	$k = 7$	$k = 6$
('MinMaxScaler', 'KMeans')	0.1498	0.1431	0.1365	0.1609	0.1640
('MinMaxScaler', 'KMedoids')	0.0991	0.0966	0.1121	0.1253	0.1519
('MinMaxScaler', 'KMedians')	0.1228	0.1309	0.1411	0.1521	0.1551

For a more detailed analysis, we also took a closer look at the Silhouette scores of each individual time series within each cluster when KMeans clustering was applied to the PIP Embedding with 6 clusters. We found that most time series were correctly assigned to their cluster. A few observations were assigned to the wrong clusters or were in between clusters.

4.3 DCT & Autoencoders

In section 2.4, we saw that autoencoders only performed marginally better than the no-preprocessing pipeline. In this section, we changed the overall method by first smoothing the time series using DCT before applying scaling.

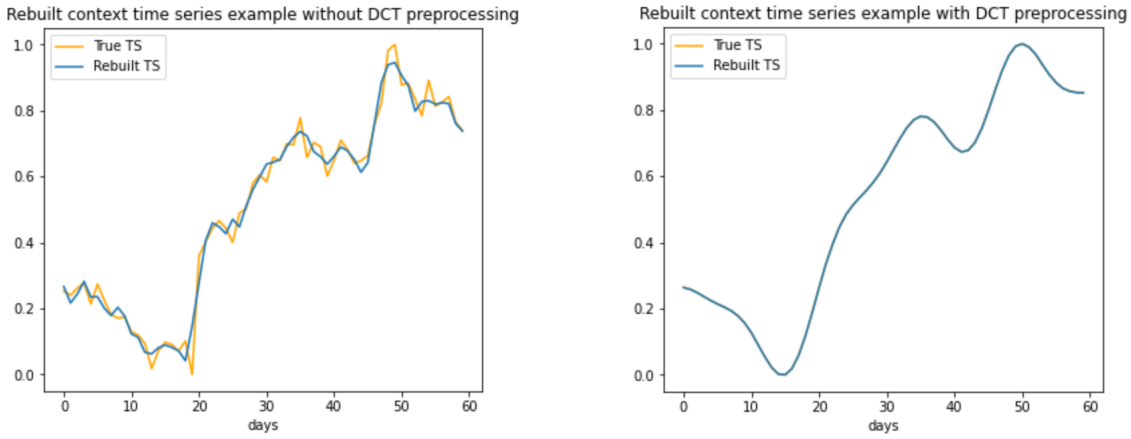

Figure 6: Reconstruction example with and without DCT preprocessing

Autoencoders tend to smoothen time series, which explains why applying DCT smoothing leads the autoencoder reconstruction to be almost perfect (see Fig 6). After training the neural network, the average Mean Square Error (MSE) between the reconstruction and the original time series was 1.10^{-3} while the average MSE between the reconstruction and the DCT smoothed signal was 1.10^{-6}. To alleviate notations, only raw estimates are given.

We then performed KMeans clustering following the same process described in section 2.4 and successfully improved the average Silhouette score from 0.14 to

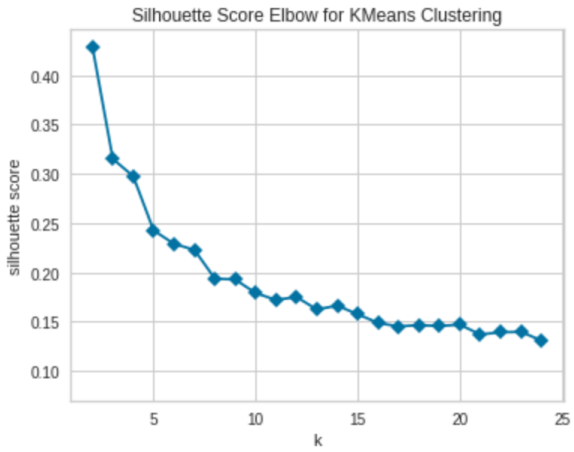

Figure 7: Silhouette Score for Autoencoders. The optimal value of 0.24 is achieved with $k^* = 7$.

0.24. Moreover, by increasing the number of clusters, the silhouette score seems to converge around 0.15, which is still better than the optimal value found without DCT (see Fig7).

4.4 DCT & Skipped and Padded Processing

For this preprocessing pipeline we combined DCT smoothing with the Skipped and Padded values methods.The rest of the preprocessing which added the two context scales for the Skipped and Padded methods stayed the same.

The next step in our analysis was to find the optimal number of clusters for each of these methods. We computed the Average Silhouette Score for a range of values for k. We used KMeans clustering with a Dynamic Time Warping (DTW) metric, and therefore also used DTW as the distance metric when evaluating the Silhouette Score.

For 2000 DCT-Skipped values we found the optimal number of clusters to be $k^* = 6$ with an average DTW-Silhouette $s^* = 0.132$.

For 2000 DCT-Padded values we found the optimal number of clusters to be $k^* = 6$ with an average DTW-Silhouette $s^* = 0.07$.

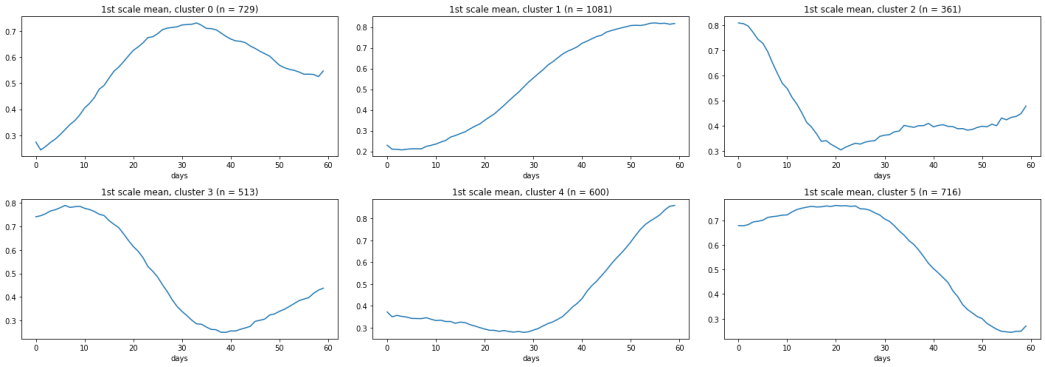

Figure 8: Per day Average time series Clusters for DCT-Skipped values data points

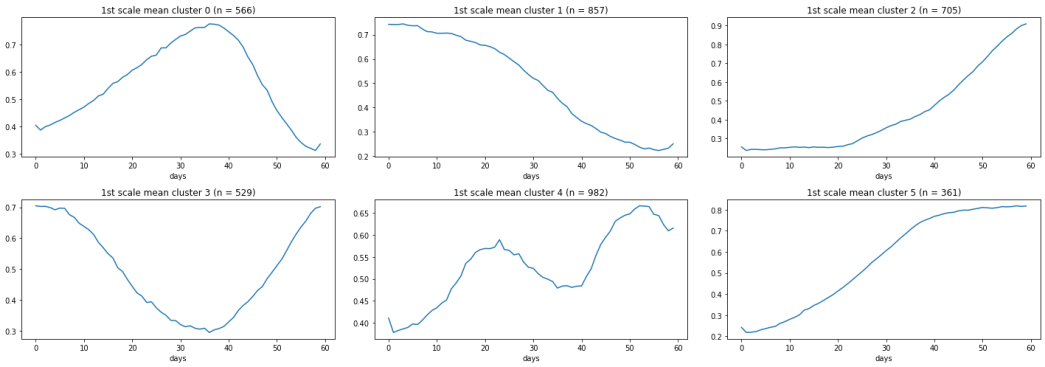

Figure 9: Per day Average time series Clusters for Padded values data points

In the above figures, we calculated the mean time series contained in several clusters for either method. In both cases, we found that time series were grouped into clusters evenly, and the cluster means captured simple harmonic patterns.

4.5 Fourier & Skipped and Padded Processing

Similarly to the previous subsection, we combined Fourier transform preprocessing with the Skipped and Padded values methods. The additional step was to perform a Fourier transform to each raw time series, eliminate the lower frequencies (higher Fourier coefficients) and keep only the real part of the Fourier

signal. We did not transform the Fourier coefficients back to time domain.

We calculated the optimal number of clusters using KMeans clustering and the Silhouette score. Note that in this case we used the Euclidean distance instead of DTW for both the clustering and the metric.

For 4000 Fourier-Skipped values we calculated the optimal number of clusters to be $k^* = 6$ with an average Silhouette Score $s^* = 0.146$.

For 4000 Fourier- Padded values we calculated the optimal number of clusters to be $k^* = 6$ with an average Silhouette Score $s^* = 0.146$.

Compared to the previous preprocessing pipelines, this method resulted in higher Silhouette scores overall and per time series-per cluster.

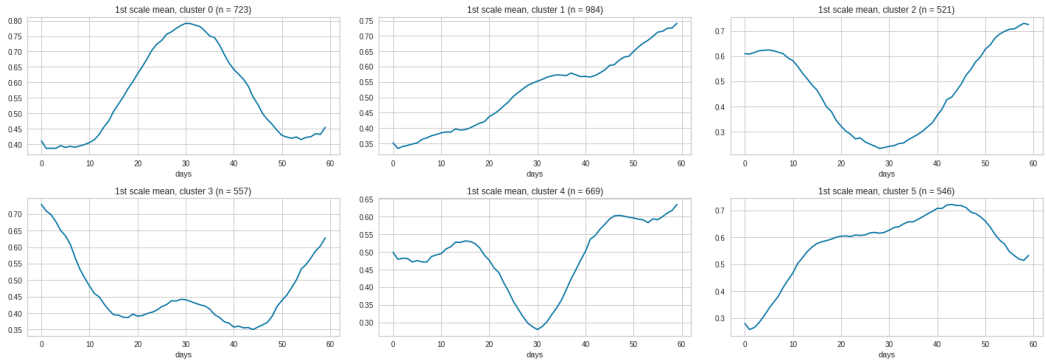

Figure 10: Per day Average time series Clusters for Fourier-Skipped values data points

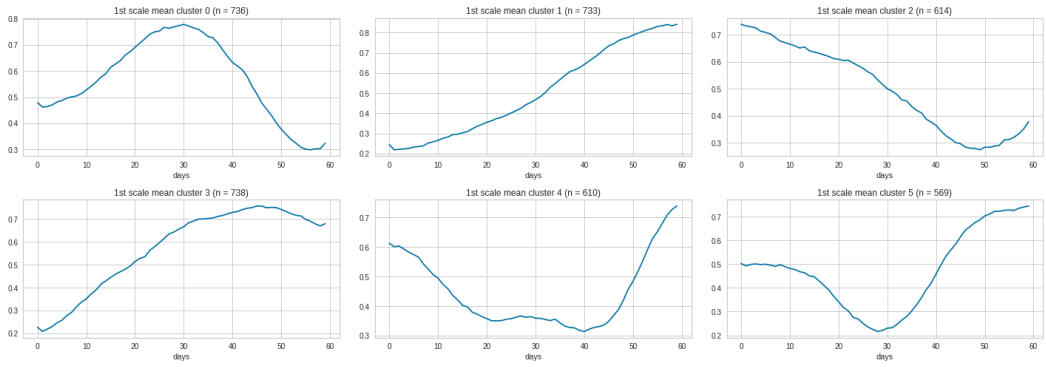

Figure 11: Per day Average time series Clusters for Fourier-Padded values data points

In the above two figures, we can see that the two clustering methods are capturing more complex patterns in each cluster mean. For instance, in the first graph for both methods we can recognize a "head and shoulders" pattern.

4.6 Summary of Combined Methods

A summary of the aforementioned methods can be found in the following table:

Preprocessing Method	**Optimal k**	**Average S-score for optimal k**
DCT - Autoencoders	7	0.240
DCT - PIP Embedding Only	6	0.164
DCT-Skipped Values	6	0.132
DCT-Padded Values	6	0.070
Fourier Transform-Skipped Values	6	0.146
Fourier Transform-Padded Values	6	0.146

From the above table, we can see that combining the DCT preprocessing and the Autoencoders method yields the highest overall Silhouette score with an optimal number of clusters $k = 7$.

5 Multi-scale evaluation

The primary goal of this project is to mine multiscale patterns from financial time series. This includes non-trivial oscillatory motions with multiple harmonics, which would be indicative of both short-term trends in the recent past, as well as trends over the long term.

We developed two different methods to quantitatively evaluate the effectiveness of our methods in capturing multiscale patterns: *conditional probability distributions of independently clustered time series* and *PIP permutation entropy*.

5.1 Conditional probabilities - DCT & Independent (long-short, fine-coarse) Clustering

If we modify the durations or resolutions (sampling frequency) of time series of stock prices and we perform a new round of clustering, what is the conditional probability that the time series of a given stock assigned to a particular cluster in the first round of clustering gets assigned to a particular cluster in the second round? To answer this question, we independently perform clustering on two sets of time series of adjusted close prices that differ in 1) the duration (long or short) - over the last three months versus those over just the past one month, and 2) the resolution (fine or coarse) - prices from each day over the past three months versus prices taken every nth day. Consequently, we obtain two independent sets of clusters for each of the two cases.

For clustering based on two different durations, the longer time period of three months translates to about 60 days of stock price history, and the shorter time period of one month is taken to be 20 days. The finer resolution of sampling the time series is one day (the lowest possible) and the coarse resolution is $n = 3$ days. We preprocessed the time series before the conditional probability analyses as follows. Each time series of adjusted close prices, either long or short, or fine or coarse, was first transformed into a time series of percentage change in the price relative to the first time point in the time series. Min-max scaling was applied to further transform the time series. Finally, the transformed time series were smoothed using DCT as a low pass filter before performing KMeans clustering.

5.1.1 Long vs Short Time Series

The number of clusters for KMeans clustering on the longer time series was chosen to be $k^* = 7$ with the elbow method, and for the shorter time series, $k^* = 8$. The silhouette scores were 0.15 and 0.145 respectively.

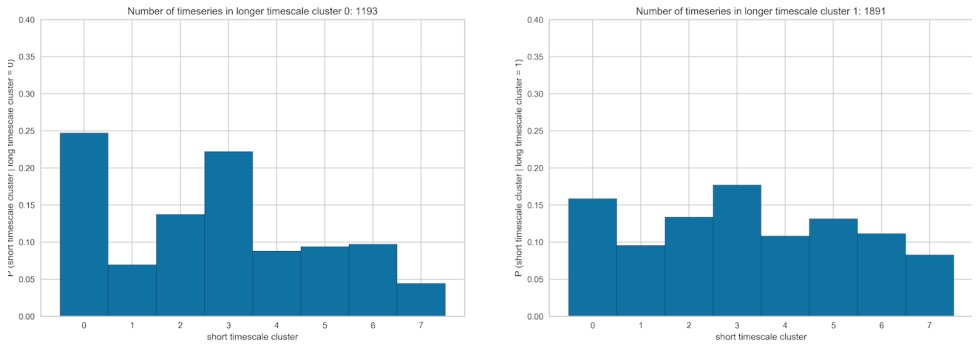

Figure 12: Conditional probabilities of the longer duration time series clustered in clusters 0 and 1 going into various clusters when a shorter duration is considered: P(short duration cluster | long duration cluster = i)

From the figure 12, we see that the longer time series clustered in cluster 3 get clustered into cluster 0 with a high probability when taken in a shorter duration, and the behavior is reciprocated in the cluster 0 of shorter time series (figure 13). This observation is indicative of a sustained trend (eg. increasing) in the stock price's time series which is dominant even in a short time period. A closer inspection of the clustered time series confirms that it is indeed the case. For the other clusters however, the conditional probability distributions are rather uniform, which points to the multiscale phenomena that we aim to capture.

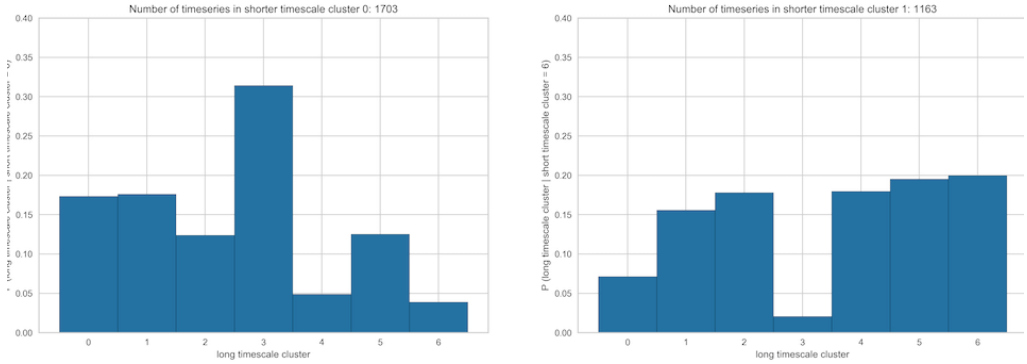

Figure 13: Conditional probabilities of the shorted duration time series clustered in clusters 2 and 3 going into various clusters when a longer duration is considered: P(long duration cluster | short duration cluster $= i$)

5.1.2 Fine vs Coarse Time Series

The optimal number of clusters for KMeans clustering on the finer time series was chosen to be $k^* = 7$ with the elbow method, and for the coarse time series, $k^* = 6$. The silhouette scores were 0.149 and 0.157 respectively.

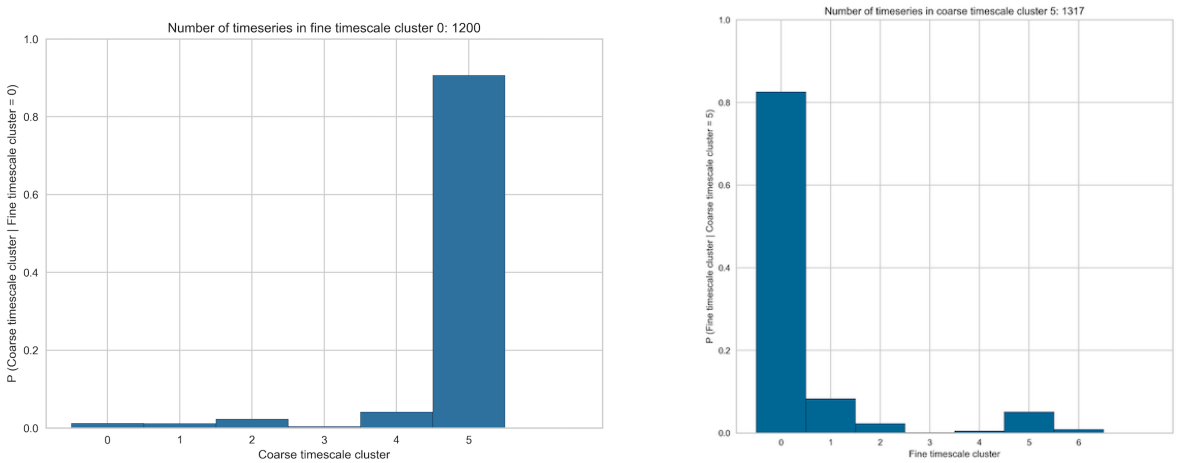

Figure 14: (left) P(coarse time series cluster | fine time series cluster $= 0$); (right) P(fine time series cluster | coarse time series cluster $= 5$)

We found that there is a very high overlap between the time series in a fine time series cluster with a particular overall trend and those in a coarse time series cluster that displays a similar trend. Therefore, the observations with conditional probability distributions analyses based on the resolution of time series are consistent with what one would expect, and do not provide any new insights into the clusters.

The observations from this analysis shed light on the multiscale patterns intrinsic to financial time series. We hope to utilize these ideas in analyses aimed at filtering for profitable patterns in the future.

5.2 PIP Permutation Entropy

Permutation entropy is a measure of signal (time series) complexity based on comparing neighboring values of each point and mapping them to ordered patterns [6].

Consider the time series $\{X_t\}_{t=1}^{T}$ and its time-delay shifted embedding representation $X_j^{m,\tau} = \{X_j, X_{j+\tau}, ..., X_{j+(m-1)}$ for $j = 1, 2, ..., T-(m-1)\tau$, where m and τ denote respectively, the embedding dimension and time delay. To compute its permutation entropy, each of the $N = T-(m-1)\tau$ subvectors is assigned a single motif out of $m!$ possible motifs (representing each unique possible ordering of m different objects). The Permutation Entropy is then defined as the Shannon entropy of the $m!$ distinct symbols $\{\pi_i^{m,\tau}\}_{i=1}^{m!}$:

$$H(m,\tau) = -\sum_{i:\pi_i^{m,\tau}\in\Pi} p(\pi_i^{m,\tau}) \log(\pi_i^{m,\tau})$$

with $p(\pi_i^{m,\tau})$ is defined as:

$$p(\pi_i^{m,\tau}) = \frac{||\{j : j \leq N, type(X_j^{m,\tau}) = \pi_i^{m,\tau}\}||}{N}$$

where $type(\cdot)$ denotes the map from pattern space to symbol space and $||\cdot||$ denotes the cardinality of a set.

We start out with the first three PIP points and observe which motif they contain using the histogram of three-point motifs within each cluster (and later across clusters). For 3 PIP points, there are $3! = 6$ different possible motifs. We use three PIP points to broadly describe the overall trend of the time series, and expect most time series within a cluster to exhibit the same one or two main motifs. By comparing the long term motifs distribution with the one from the short term scale we hoped to start capturing the multiscale phenomena: An increasing trend in the long term might end with an increasing trend in the short term as well or a decreasing one. That is said, it is not currently a fully thorough metric of the multiscale phenomena but rather and indicator of it.

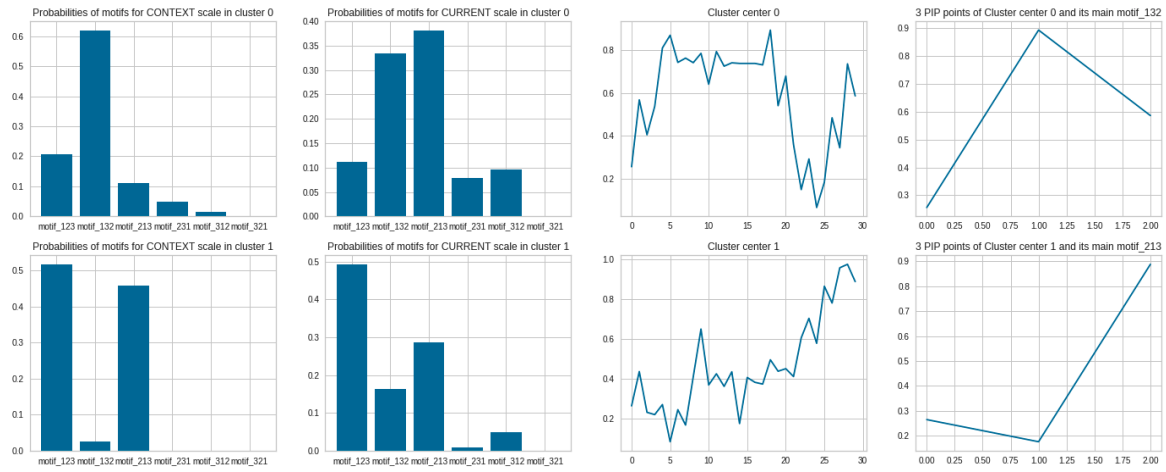

Figure 15: Demonstration of Permutation Entropy analysis on clustering

In Figure 15 above, the distributions of 3 PIP-motifs contained in every context scale (=long scale) time series within each cluster is presented in the first column. The probabilities of motifs of 3 PIPs in the corresponding shorter scale time series is shown in the second column. The third column shows the cluster centers of the associated clusters. These cluster centers are then represented by 3 PIPs in the fourth column. The main motif for cluster 0, for example, is (1,3,2), which is consistent with the highest frequencies of the motifs in the first column. The current or shorter scale, however, has largely different motifs probability distributions than the context scale. The plot shows that the context scale motifs are not necessarily the same as the short-term scale one, which could indicate different patterns are captured on multiple scales.

6 Conclusion and Next Steps

This section concludes our work on preprocessing and modelling for the analysis of multiscale financial time series. Over the past months, we have developed algorithms to preprocess the data (DCT smoothing

and Fourier Transform denoising) and extract informative multi-scale features from financial time series (Skipped & Padded values, PIP, Autoencoders and Weighted Sampling). In parallel, we have explored different clustering algorithms (KMeans, KMedian, KMedioids) while using data-driven methods to identify the optimal combinations of parameters in order to perform an informative clustering. We have also developed tools (Silhouette Score, Conditional Probabilities, PIP Permutation Entropy) to visualize and measure the performance of the preprocessing, feature extraction and clustering algorithms.

The final phase of our project will consist in the development of a robust data-driven method to evaluate whether or not our metrics capture multi-scale properties.

6.1 Synthetic Data

To check whether the multi-scale algorithms work as we would expect, we can generate artificial (synthetic) data and "control" for multi-scaleness: we generate two similar datasets of time series with multiscale properties in one dataset and none in the other. We then measure whether the scores of each of our methods differ significantly from one dataset to the other. This would offer a new objective way to compare metrics and methods and select the ones which best capture the differences between the two datasets.

To generate synthetic data, we use an interesting property of DCT coefficients. Each coefficient corresponds to a periodic signal with a fixed frequency. By assigning nonzero values to specific coefficients, we can control the scale of patterns which appear in the signal.

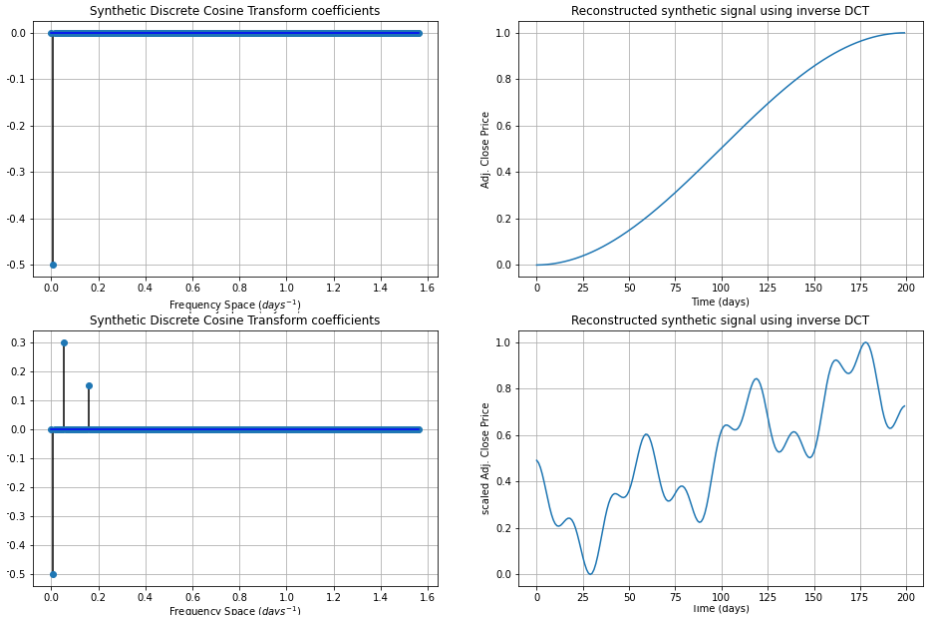

Figure 16: Superimposing patterns from multiple scales by adding DCT coefficients.

In the DCT frequency domain:

- The first DCT coefficients are for low-frequency patterns, i.e long scale trends.
- The middle coefficients are for high-frequency patterns, i.e short scale trends.
- The last coefficients are for noise.

We can then generate synthetic data by randomly selecting one coefficient in the long scale (first coefficients) and one or more coefficients in the short scale (following coefficients).

Using this process to generate synthetic time series, we will evaluate each method objectively, compare it to a baseline method of no preprocessing, and quantify the added value of using the method in measuring multiscale properties on the real time series dataset.

References

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

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

[3] "Using a digitizer for time-domain measurements," *National Instruments Corporation.*

[4] G. Romane, J. Hritik, P. Kassiani, S. Amaury, T. Maxime, and T. Vy, "Engie e4800: Data science capstone, first progress report, trading opportunities," 2020.

[5] Mudgalvivek, "Machine learning : Clustering : Elbow method," *Medium*, 2020.

[6] F. Bilal, C. Badong, K. Andreas, and P. José, "Weighted-permutation entropy: A complexity measure for time series incorporating amplitude information," *Physical Review.*