ENGIE E4800: Data Science Capstone

Final Report

J.P. Morgan: Trading Opportunities

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

18 December 2020

Contents

1 Introduction

1.1 Motivation, Background & Business Problem

In financial trading, technical analysis aims to evaluate investments and identify opportunities using only a stock's price and volume data. It isn't immediately obvious why this type of analysis may have predictive value, especially when compared with fundamental analysis. Fundamental analysis looks at a company's financial reports, the state of the economy, and industry trends to determine whether the 'true' or assessed value of a stock reflects its current traded value. If the current price is below the assessed price, the stock price will increase, and vice versa if the assessed price is lower than the current price. On the other hand, in a technical analysis view, all known fundamentals of a company's business are instantaneously factored into the stock price. Thus, there is no need to explore the economic conditions of a company – everything is reflected in the price.

Under a technical analysis assumption, predictions are made by identifying patterns which are empirically "known" to lead to a predetermined outcome. For example, the "head and shoulders" pattern is a reversal sign, indicating a change from bull to bear trend. However, the presence of multiple patterns in a given time series adds complexity to a technical analyst's task of predicting stock behaviors. A pattern may be 70% head and shoulders, and 20% channel up, for example.

Relying on known patterns then, can induce subjectivity into technical analysis. In this capstone project, we sought to best utilize our backgrounds in data science to approach technical analysis in an objective and a data-driven way.

1.2 Problem Statement

The primary question we aim to answer is, can we identify meaningful patterns in stock data using unsupervised learning, given the *multiscale* nature of these financial time series?

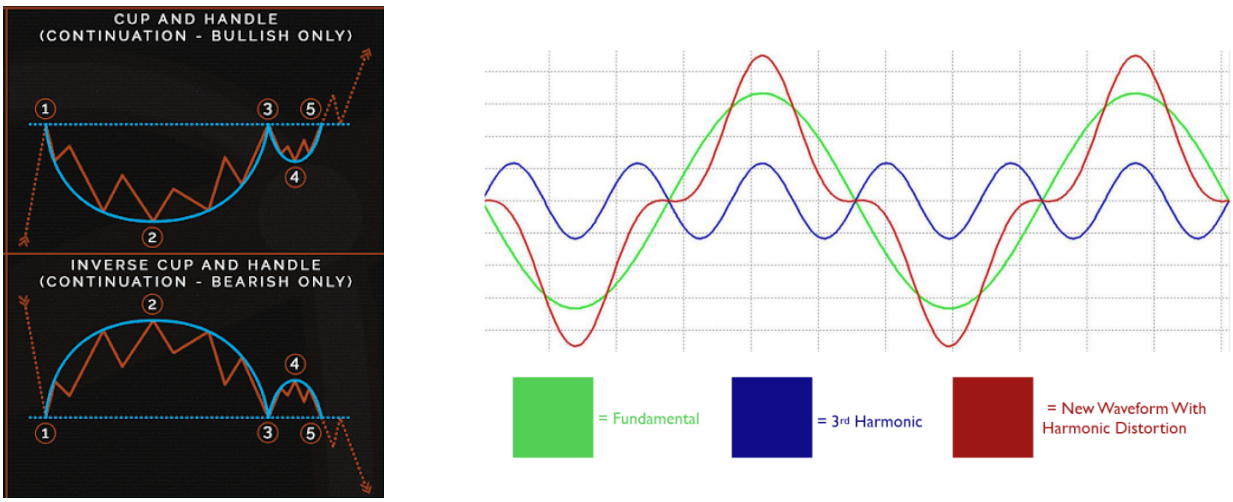

Figure 1: Multi-scale clustering: **Left**: a) Cup and handle chart pattern. **Right**: b) Harmonics

Financial time series are multiscale in nature. When analyzing the stock market, 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. As an illustration, consider the cup and handle chart patterns shown in Fig 1 a). It consists of consecutive arcs of different sizes, that is, the same pattern occurs in two different timescales.

The longer timescale would show the general trend of a stock over the past few months, while the shorter timescale would reflect the current behavior, or most recent behavior of a stock. The two time scales could very well be overlapping. It then becomes necessary to view financial time series also in terms of multiple harmonics that make up a sine wave, as depicted in Fig 1 b). Finding a balance between the long term and immediate variations of the stock ensures that traders 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.

1.3 Existing Work & Literature Review

A body of research supports the idea that leveraging chart patterns in technical analysis has useful predictive value. For example, for a bull flag chart pattern, two separate papers studied methods of pattern matching and quantifying how well a given stock chart matches the bull flag chart pattern. The first of these papers, published in 2002, identified trading rules based on how well the stock chart matched a bull flag and found these trading rules to be effective on out-of-sample examples [1]. The second, published in 2007, focuses on two stock market indices - Nasdaq Composite Index (NASDAQ) and Taiwan Weighted Index (TWI). This second study found that technical trading rules correctly predict the direction of market changes. It also found that matches with the bull flag pattern were correlated with higher returns [2]. In considering the bull flag chart pattern alone then, these two separate research papers find that one can make a profit, on average, by leveraging chart patterns.

A more thorough overview of technical analysis chart patterns is provided in a paper by Andrew W. Lo, Harry Mamaysky, and Jiang Wang [3]. The researchers examine ten different chart patterns in-depth, and demonstrate how to recognize the patterns quantitatively.

Today, most research into technical analysis chart patterns addresses either the evaluation of their usefulness in trading or the application of new technologies to recognize chart patterns.

We took our lead from a JPMorgan research intern who used various clustering algorithms to cluster the time series of adjusted close prices. It was observed that the mean time series in these clusters are best characterized by simple harmonic functions. He further found that filtering for the stock's sector or profitability did not add any predictive power to the clustering results.

1.4 Overall Approach

We break the multiscale pattern detection problem down into three subproblems. First, how can we gather data which has multiscale properties? We use real data from the S&P 500 Index over the past twenty years, and also generate artificial data to help us test our algorithms. Secondly, how do we extract multiscale features from this data? To this end, we explore a variety of methods from financial engineering, physics, and machine learning such as Discrete Cosine Transforms, Fourier Transform, Autoencoders, Perceptually Important Points (PIP), and Padded & Skipped Sampling. We cluster the features extracted through these methods using a KMeans algorithm and evaluate the clustering results with appropriate metrics. We then combine the methods that are found to produce high quality clusters to obtain complete clustering pipelines. Finally, to verify that our methods are capturing multiscale patterns, we look into the PIP Permutation Entropy in the resulting clusters, and the Shannon Entropy of Conditional Probability Distributions of independently clustered long and short time series.

2 Methods

2.1 Data

2.1.1 Real Data

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

- 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 the general behavior of a stock price during the past few months. We set it to 60 days, which is equivalent to 3 months. Beyond that period, stock prices are not considered accurate enough to help us make investing decisions.

As this produces a very large amount of time series chunks, we restricted our dataset and randomly picked only several thousands of them.

2.1.2 Synthetic Data

To ensure the data used throughout our analysis actually contained multiscale properties, we chose to also generate data ourselves. A 'synthetic' data generating process allowed us to create time series which combined short-scale patterns (high frequency variations) and long term trends (low frequency variations). We used a signal processing approach leveraging Discrete Cosine Transform (DCT - [Appendix]) coefficients to control the scale of patterns which appeared in the signal. Low-frequency DCT coefficients in the DCT frequency domain determine long scale trends and high-frequency DCT coefficients determine short-scale patterns in the temporal domain.

We generated synthetic data by randomly selecting one coefficient in the long scale and one or more coefficients in the short scale. Once these coefficients were selected, we transformed the coefficients from the frequency domain back to the time domain to create a new time series.

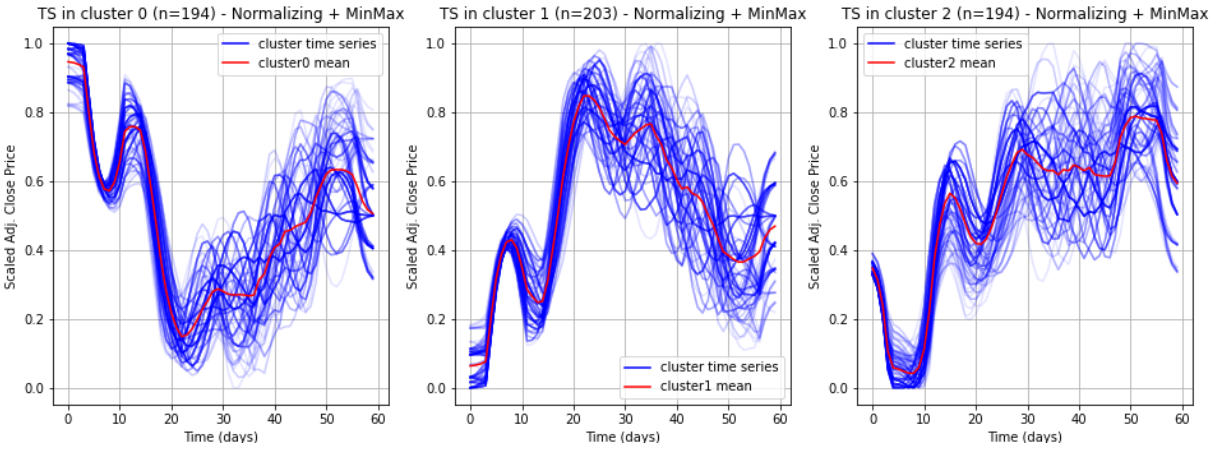

Figure 2: Examples of clustering on synthetic data.

2.2 Pre-processing

Our data preprocessing explorations led us to combine initially separate methods in order to construct better performing clusters. In this section we will present the key data transformations applied by each combination of methods to raw time series data.

2.2.1 DCT & Skipped Values

The first step of the "DCT & skipped" preprocessing method consisted in passing all time series through a smoothing filter (DCT) which removed noise for the data points. More details on this filter can be found in the [Appendix].

Next, in order to ensure the capture of multiscale properties of the data, we added two additional scales for each stock price time series of 60 consecutive days to form a "3-scale matrix". The first context scale skips the price of a stock every other day, and the second context scale keeps the price every two days, and skips other days. All skipped prices were replaced with 0.

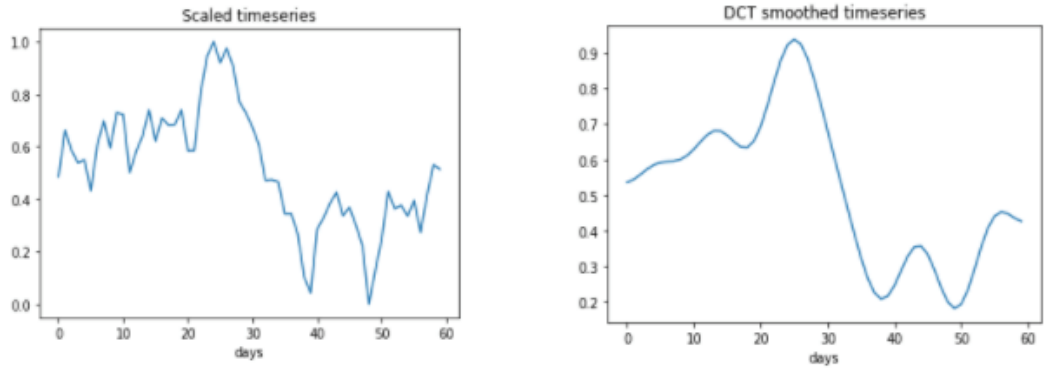

Figure 3: **Left**: Example of unprocessed time series. **Right**: DCT smoothed time series.

$$\textbf{Skipped} = \begin{array}{c} \begin{matrix} 0 & 1 & 2 & 3 & 4 & .. & 59 \end{matrix} \\ \begin{bmatrix} 0.54 & 0.55 & 0.56 & 0.57 & 0.59 & .. & 0.43 \\ 0.54 & 0 & 0.56 & 0 & 0.59 & .. & 0 \\ 0.54 & 0 & 0 & 0.57 & 0 & .. & 0 \end{bmatrix} \end{array} \begin{matrix} \\ 60 \\ 40 \\ 20 \end{matrix}$$

Figure 4: Skipped values representation of a data point.

2.2.2 DCT & Padded Values

In a similar approach to the above section, the first step of this preprocessing method was to pass all time series through a DCT smoothing filter. We then added two additional scales for each stock price time series of 60 consecutive days. The first context scale included the 40 most recent days of stock price data, and the second context scale included the 20 most recent days of stock price data. All older prices were replaced with 0.

$$\textbf{Padded} = \begin{array}{c} \begin{matrix} 0 & 1 & .. & 19 & 20 & .. & 59 \end{matrix} \\ \begin{bmatrix} 0.54 & 0.55 & .. & 0.65 & 0.69 & .. & 0.43 \\ 0 & 0 & .. & 0 & 0.69 & .. & 0.43 \\ 0 & 0 & .. & 0 & 0 & .. & 0.43 \end{bmatrix} \end{array} \begin{matrix} \\ 60 \\ 40 \\ 20 \end{matrix}$$

Figure 5: Example Padded values datapoint.

2.2.3 Fourier & Skipped Values

We also decided to combine the 3-scale matrix explained in 2.2.1 with Fourier transforms, a method that transposes a signal from the time domain to the frequency domain [Appendix].

For this step however, we do not perform DCT prior to obtaining the matrix as we want the Fourier transform to capture all the information contained in the data. We therefore create the 3 scales from the initial time series and take the Fourier transform of each of the scales. The resulting matrix is a 3×60 matrix with 3 rows of Fourier coefficients.

The last step for this preprocessing method is to restrict the number of Fourier coefficients for each scale. Beyond a certain point, the Fourier coefficients can be attributed to noise rather than valuable information in the time series, and must be removed. The optimal number of Fourier coefficients to keep was set to 11 after an extensive visual analysis.

2.2.4 Fourier & Padded Values

This preprocessing method is very similar to the "Fourier + skipped" approach from section 2.2.3, with the only difference that the Fourier transforms were applied to the 3-scaled "padded" context matrix explained in section 2.2.2, instead of the "skipped" matrix detailed in section 2.2.1.

2.2.5 DCT & Autoencoders

Another pipeline that we explored used Autoencoders as a data preprocessing step. Autoencoders are neural networks trained to reconstruct the input by reducing the inner dimension of hidden layers. The hidden layers can then be used as compressed representations of the input. We use autoencoders to extract features of the long term scale of a time series. As shown in the pipeline (Fig 6), we first smooth the signal using DCT, then train an autoencoder to extract features.

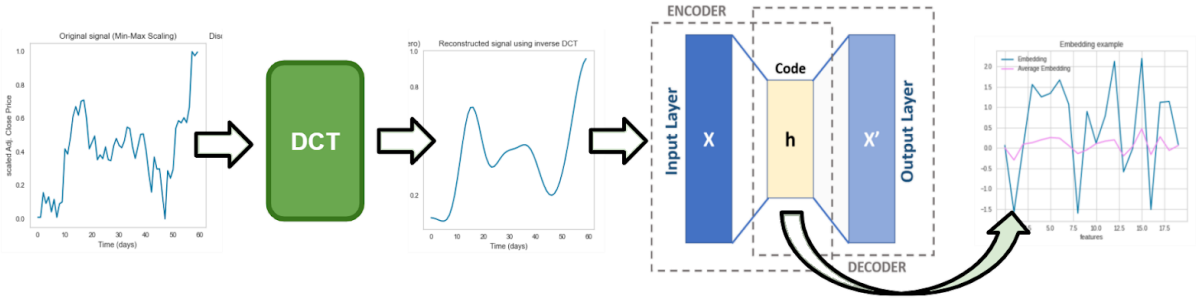

Figure 6: Autoencoder preprocessing pipeline

2.2.6 DCT & PIP Embedding

The last preprocessing method that we explored made use of the concept of Perceptually Important Points (PIP). This is a method for dimensionality reduction of time series which iteratively extracts the most important points from a human observer perspective. The first two most important points (PIPs) are the first and last points of the time series. The next PIP (third PIP) will be the point in the time series with maximum vertical distance from the first two PIPs. The fourth PIP will then be the point in the time series with maximum distance to its two adjacent PIP's. We repeat this process until the time series is reduced to the number of data points we want to keep for clustering.

The formula for vertical distance function to extract PIP from time series is the following [4]:

$$VD(p_3, p_c) = |y_3 - y_c| = \left| \left(y_1 + (y_2 - y_1) \cdot \frac{x_c - x_1}{x_2 - x_1} \right) - y_3 \right|$$

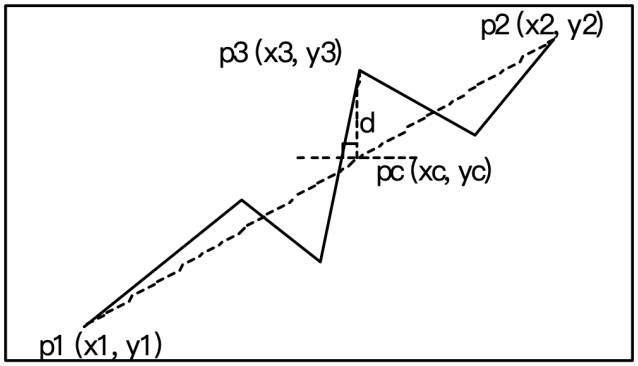

Figure 7: Vertical distance metric

In this approach, we use PIP to extract the 30 most important points from a scaled 60-day time series. The PIP Embedding is then smoothed out by removing noise DCT coefficients.

2.3 Analytic Methods

Once our preprocessing methods had been defined, we needed to perform clustering in order to capture the multiscale patterns within our time series. We used a KMeans clustering algorithm and optimized the number of clusters "k^*" for each of our preprocessing methods.

2.3.1 KMeans

The KMeans clustering algorithm is often used to detect existing patterns within time series [5, 6]. Since the purpose of our research is to create clusters which captures multiscale patterns in time series, this characteristic seemed particularly well-suited to our task.

However, the main drawback of KMeans was the need to determine the number of clusters beforehand and to input it into the algorithm. We detail the method used to optimize the number of clusters "k" for each preprocessing method in the next section.

2.3.2 Optimization of the Number of Clusters

In order to optimize the number of clusters, we combined the Silhouette Score with the Elbow Method.

The Silhouette Score is a metric that accounts for the inter and intra cluster distance for each cluster [See Appendix]. This means that the more distinct and compact the clusters are, the better, and the higher the Silhouette Score will be.

However, in the case of time series, the Silhouette Score is a decreasing function of the number of clusters. For this reason, we need to find a balance between maintaining a high score without discarding too many clusters. The Elbow Method helped us both achieve this goal and define the optimal number of cluster "k" for each preprocessing method.

2.4 Evaluation

The natural next step to our analysis was to define and use some evaluation metrics to measure how well our clusters are performing and how well they are capturing multi scale patterns. In the previous section, we mentioned using the Silhouette Score to find the optimal number of clusters in each method. In this section, we will further define some concepts that are very important to understand multiscale cluster evaluation.

2.4.1 Permutation Entropy for Multiscale Evaluation

Permutation Entropy is a method which measures the complexity of a given time series. We will describe the general Permutation Entropy measure, how we adapted it, and how combining it with the PIP algorithm allowed us to evaluate a wide range of time series.

2.4.1.1 General Permutation Entropy

Permutation Entropy aims to classify motifs inside a time series using a sliding window. For instance, if we were to represent a time series using only 3 points, we could be interested in knowing whether the points are constantly increasing or showing an increase-decrease pattern. Permutation Entropy aims to help us better visualize such patterns. The following figure (Fig 8) exhibits all the possible permutations of any three values and thus all possible 3-points patterns that can appear inside the condensed representation of a time series.

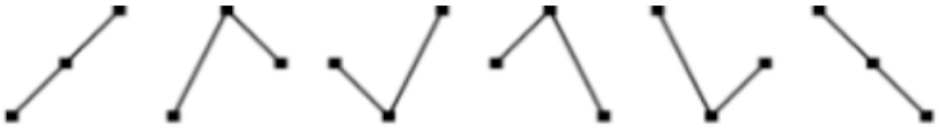

Figure 8: 3-Points Patterns: a) (1,2,3) b) (1,3,2) c) (2,1,3) d)(2,3,4) e) (3,1,2) f) (3,2,1)

A sequence of n real numbers can lead to $n!$ different orderings (permutations). We can associate one of the $n!$ permutations to a given set of values $(x_1, \ldots, x_n)$ by ordering it. To choose which permutation use use to represent the values of a time series, we use the following process:

- Pick three (or n) consecutive data points of the time series (x_1, x_2, x_3)
- Order the values: $x_{t_1} \leqslant x_{t_2} \leqslant x_{t_3}$
- Generate the permutation by reindexing the time indices (t_1, t_2, t_3)
- Build the histogram of patterns

The result of this process is shown Fig 9 where we point the result of the first sliding window. When the values are already ordered, the resulting permutation is the identity one: $(1, 2, 3)$. While we might want to use more than 3 motif points, the factorial function unfortunately increases very quickly (e.g $5! = 120, 10! > 10^6$). As a consequence interpretability becomes impossible for large values of n and the resulting histogram flattens patterns to a uniform distribution.

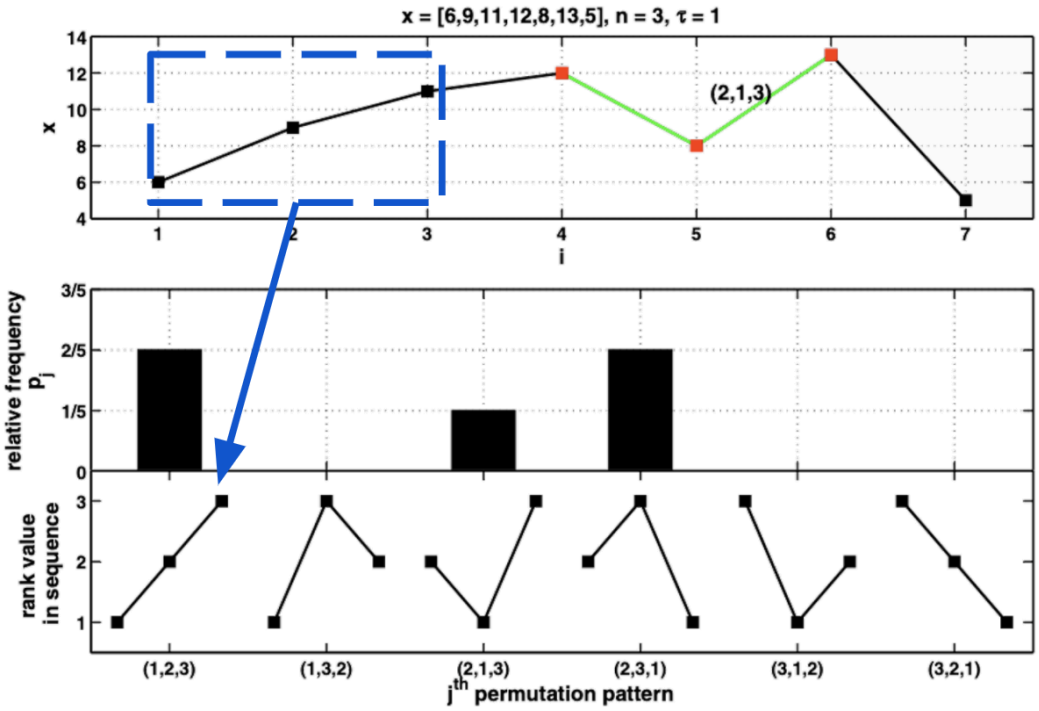

Figure 9: Illustration of the Permutation Entropy process. The result of the first sliding window is highlighted in blue.

2.4.1.2 PIP Permutation Entropy

To adapt permutation entropy from a single time series to clusters of multiple time series, we use the PIP algorithm. PIP extracts 3 points which we use to perform a motif analysis. We use the previous process, this time using PIP to extract data points instead of a sliding window.

We once again recognize patterns in time series by associating each point to a pattern produced by its 3-PIP transformation. The motifs presented Fig 8 also show all the possible permutations of any three values and thus all possible 3-PIP patterns that can appear for each time series.

We are interested in detecting these motifs in the long scale and in the short scale for all raw time series in each cluster. By doing so, we can see the major trends for each scale while getting a sense of how well our clustering pipeline is performing.

In order to detect one of these patterns in the long scale, over 60-days for example, we can reduce each raw time series using 3-PIP. Note that the data points, after the 3-PIP transformation, will belong to one of these patterns.

In order to detect one of these patterns in the short scale, we can reduce each 60-day time series using a 5-PIP transformation. Note that after the 5-PIP transformation, the time series will have 5 points, therefore in order to "uncover" which of the above patterns appears in the short scale we can look at a sliding window of length 3.

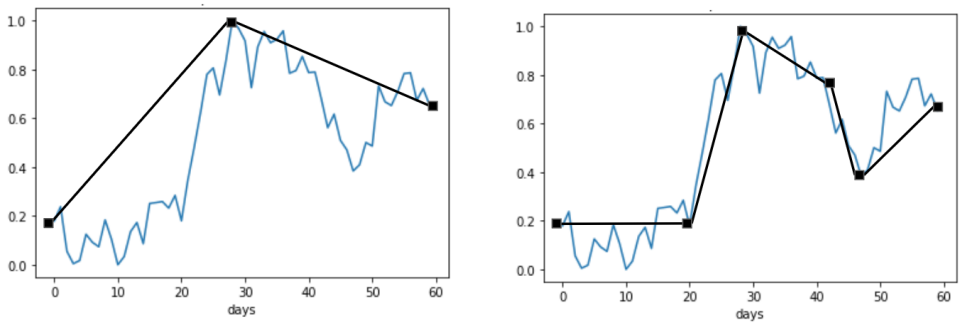

Figure 10: **Left**: 3-PIP time series **Right**: 5-PIP time series

As you can see on the figure above (Fig 10), the long term pattern is (1,3,2) and for the 5-PIP transformation the 3 sliding windows exhibit the following patterns; w1: (2,1,3), w2: (1,3,2) w3 (3,1,2).

By looking at the histograms of patterns that time series clusters contain, we can infer which are the strongest patterns in the long scale and in the short scale (within each of the three windows). This metric can detect whether a general increase pattern (with 3-PIP points) is a sum of smaller increasing patterns, or if instead it is the result of a spiky signal ending with an increase (with 5-PIP points).

An ideal clustering pipeline would create clusters which exhibit one major motif on the long scale and one major pattern for each of the short scales for all clusters. A failed clustering pipeline would include motifs with nearly uniform frequency in the long and short scale.

2.4.2 Conditional Probability for Multiscale Evaluation

In this section, we investigate the predictive power of our clustering models, or equivalently, how short-term trends relate to long-term trends from the past.

Consider one set of time series of stock prices for 60 consecutive days, and a second set consisting of time series of the last 20 days of the longer time series in the first set. We cluster these two sets of time series independently and examine the probability distributions of cluster assignments for the short-term time series in any of the given long-term time series clusters.

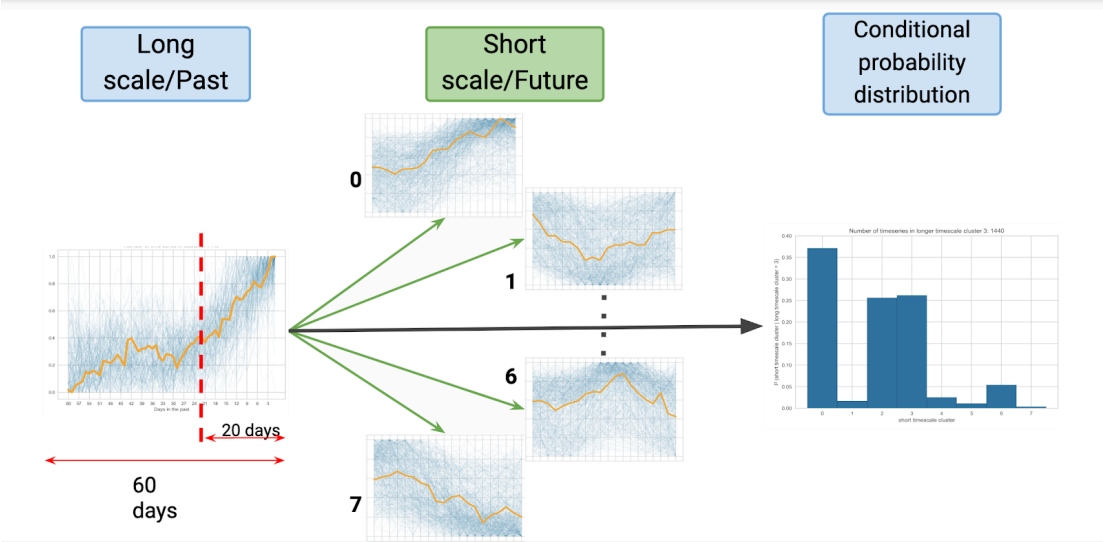

Figure 11: Conditional Probability Distribution Analysis for multiscale evaluation

This process is illustrated in Fig 11. On the left we have one of the long-term time series clusters, which exhibits an increasing trend. The short-term time series that belong to these clusters could go into 7 different clusters in this clustering with different probabilities. As we would expect, we find that that cluster 0, which also shows an increasing trend in the short term, has the highest probability, while cluster 7, which shows a decreasing trend, has the lowest probability. The time periods in these two sets could also be separated in time. For example, given a set of time series of 80 days, we can obtain two different sets of time series by splitting the original time series at a fixed point. We follow this procedure for the different clustering pipelines to obtain the corresponding conditional probability distributions. Fig12 below shows some of the conditional probability distributions from the Autoencoders pipeline.

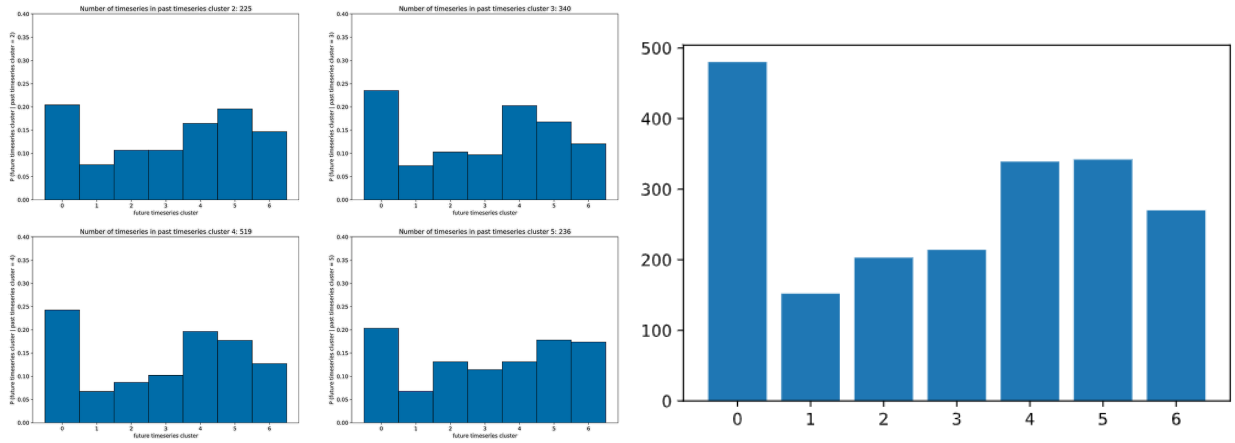

Figure 12: **Left**: Conditional probability distribution of future cluster assignments for 4 (total 7) of the past time series clusters obtained from the Autoencoder pipeline. **Right**: Histogram of populations of the future time series clusters

2.5 Results

In this section we will present our clustering evaluation results based on the metrics that were introduced in section 2.3.2. Moreover we illustrate the logic behind multiscale evaluation and analyze the results of two methods.

2.5.1 Clustering Evaluation

By using the average silhouette score and the elbow method (section 2.3.2) , we have gathered the results of the optimal number of clusters k^* and the optimal average silhouette score for each preprocessing method.

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 $k^* = 7$.

2.5.2 Multiscale Evaluation with Permutation Entropy Results

In this section we present how to use PIP Permutation Entropy (Section 2.4.1) in order to examine the multiscale properties captured by our clusters.

For our pipeline evaluation, we create the Permutation Entropy histograms and examine which motifs prevail in each cluster and for each scale. We will present the PE histograms of two methods; DCT- Autoencoders and DCT-Skipped. For all methods the complete Permutation Entropy histograms can be found in the [Appendix].

2.5.2.1 DCT & Skipped Values

In order to offer another intuitive way of quickly seeing how clusters capture multiscale patterns we will examine clusters that are the same in the long scale but differ in the short scale.

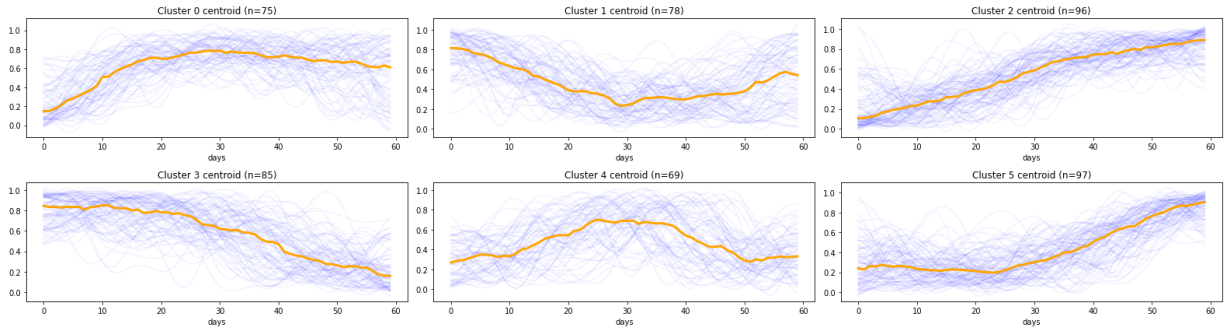

Figure 13: DCT-Skipped clusters

By observing cluster 0 and cluster 4 one can see that they exhibit similar long scale patterns. In order to verify this observation for each cluster we compute the 3-PIP points Permutation Entropy histogram and verify that most of the time series in the long scale in these clusters follow the (1,3,2) pattern.

On the other hand, when we compute the most frequent patterns in the short scale we see that they differ. As we can see in the figure below, window 1 of cluster 0 the most frequent pattern is increasing (1,2,3) whereas in cluster 4 it is decreasing and then increasing (2,1,3) and so on. Thus even though these clusters are the same in the long scale patterns, they are different in the short scale.

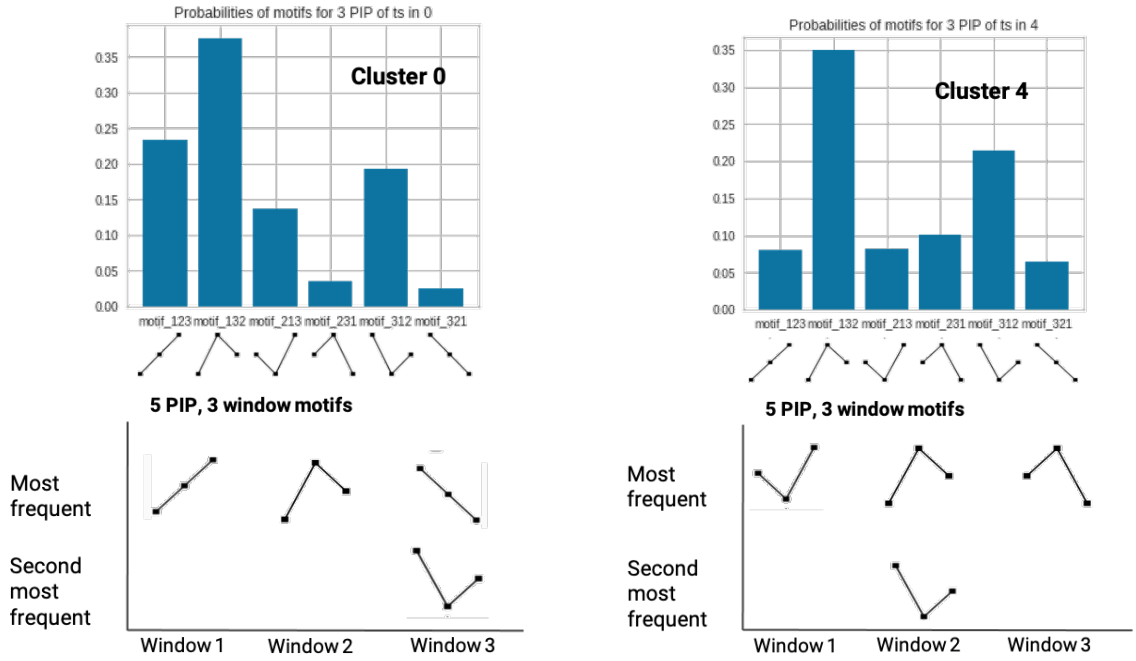

Figure 14: **Top**: PE histogram of 3-PIP long scale is the same for cluster 0 and 4, **Bottom**: Short scale sliding window of simplified PE histograms exhibit different short term patterns

For the three window plots in the above figure, we have presented the most popular trends for each window. We have also included the second most popular trend if its frequency is less than 10% different from the first one.

Therefore, this clustering pipeline successfully separated those two methods according to their multiscale trends.

2.5.2.2 DCT & Autoencoders

Similarly, in the DCT- Autoencoder method, if we look at two clusters that exhibit similar long scale patterns, cluster 5 and cluster 9 in the below figure, we can quickly see that even though those clusters exhibit similar long scale Permutation Entropy histograms, the short term trends differ.

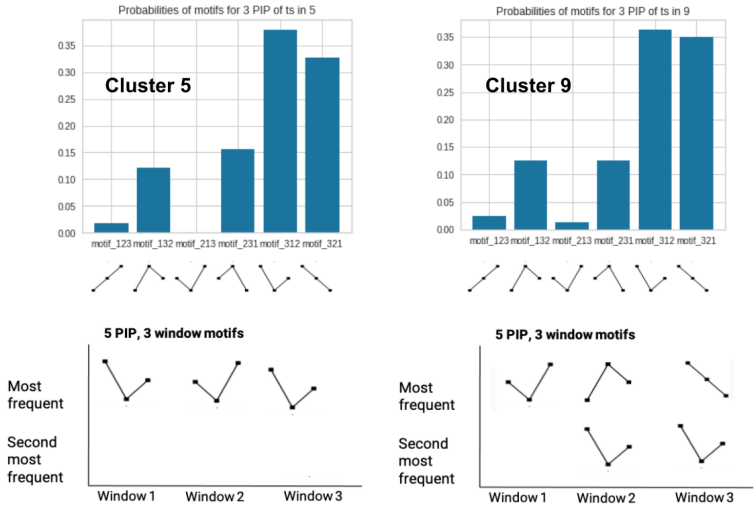

Figure 15: **Top**: PE histogram of 3-PIP long scale is the same for cluster 5 and 9, **Bottom**: Short scale sliding window of simplified PE histograms exhibit different short term patterns

Therefore, this clustering pipeline successfully separated those two methods according to their multiscale trends.

2.5.3 Conditional Probability & Shannon Entropy Results

We described the procedure of obtaining the conditional probability histograms for different clustering pipelines in section 2.4.2. The more non-uniform and peaky the conditional probability histograms are, with most of the time series grouping into few clusters, the higher the predictive power of the given clustering pipeline is. To quantitatively pursue this idea, we find the Shannon entropy for each of the conditional probability distributions. A more non-uniform distribution would have a lower Shannon entropy and vice versa.

$$\eta(X) = \frac{H}{H_{max}} = -\sum_{i=1}^{n} \frac{p(x_i)\log_b(p(x_i))}{\log_b(n)}$$

Figure 16: Expression for normalized shannon entropy; n = number of clusters

The table below shows the shannon entropy for two of our clustering pipelines, along with a vanilla KMeans pipeline that serves as a baseline. The entropy values are normalized for a fair comparison across different pipelines that could have different numbers of clusters, and further averaged over all the clusters.

Pipeline	**40/40**	**50/30**	**60/20**
KMeans	0.978	0.975	0.986
PIP Embedding	0.972	0.993	0.982
Autoencoders	0.962	0.949	0.978

Table 1: Mean normalized Shannon entropy across clusters obtained using different pipelines on 80 days time series with split as X/Y

As we expected, the mean normalized entropy numbers in the table above indicate that the Autoencoders pipeline has a relatively higher predictive power when compared to a vanilla KMeans pipeline and the PIP embeddings pipeline. The core idea underlying this analysis can also be used to filter for profitable patterns in stock time series data in the future.

3 Discussion

3.1 Summary

Our collaboration with J.P. Morgan AI Research was structured around three key questions. How can we extract or generate multiscale data? How can we capture informative features from this data? Finally, how can we qualitatively and quantitatively assess whether our feature extraction methods capture multiscale patterns?

Individually, each of these questions proved to be fascinating engineering challenges, requiring a combination of ingenuity, creativity, and diligent research. Collectively, these questions allowed us to draw strength from our pluridisciplinary backgrounds, combining approaches from signal processing, financial engineering, machine learning and statistics to approach a financial problem from a Data Science perspective.

The data used for our analyses consisted of time series from the S&P 500, segmented into 60-day chunks and scaled between 0 and 1. In parallel, we generated artificial data by leveraging Discrete Cosine Transforms, ensuring each time series contained multiscale patterns.

We extracted features from this data using established techniques including autoencoders and Perceptually Important Points, while also contributing our own methods such as skipped values and

padded sampling. Notably, we combined the aforementioned techniques with a Fourier Transform-based preprocessing to maximize metrics indicative of a high-quality clustering. Clustering was performed using KMeans, and our use of the Silhouette score clustering metric optimized by the elbow method allowed us to compare all feature extraction methods on a level playing field.

Crucially, some of our most significant findings were related to the evaluation of methods and pipelines. We devised two frameworks to evaluate the multiscale feature extraction capabilities of our algorithms: "Conditional Distribution" and "PIP Permutation Entropy". The first evaluation method allowed us to objectively measure the capabilities of our pipelines in predicting the future behavior of stock data. The second evaluation method provided a much-needed tool to visualize contents of clusters through intuitive and intelligible representations.

The pipelines and tools we created throughout this project were extensively documented to allow researchers to build on our findings and expand applications of multiscale clustering in financial time series.

In the future, these tools will help improve operational efficiency and data-driven decision-making at the Bank.

3.2 Conclusion & Take Home Messages

Our research opens new directions for multiscale analysis and evaluation of financial time series. While some heuristics such as silhouette score and visualizations initially helped us check the quality of our clustering, we finally created our own evaluation metrics which better captured the multiscale phenomena. To the best of our knowledge, this is the first study to use Permutation Entropy to evaluate multiscale phenomena in time series.

We explored many preprocessing approaches to cluster time series, from signal processing (Fourier Transform and DCT) to embeddings (autoencoders and PIP) and more custom methods (Skipped and Padded). Our analysis led us to the conclusion that ensembling different methods yields better clusters. Specifically, we found that combining different preprocessing methods, such as DCT and Autoencoders for example, led to higher quality clustering than using these methods separately.

However, our results varied significantly across methods and clusters: some methods seemed to capture multiscale for specific clusters but not for all of them, and other methods didn't capture multiscale for any of the clusters.

3.3 Future Work

The next steps are twofold. On the one hand, we encourage fine-tuning the existing pipelines we built in order to fully capture the multiscale aspect of the time series. On the other hand, as a finance oriented project we encourage introducing a notion of profitability or volatility of a given time series.

In terms of profitability, we may want to classify clusters according to whether or not they contain profitable patterns. In other words, we may want to detect clusters which provide likely trading opportunities and prevent us from investing in risky stocks. Furthermore, a notion of profitability could help predict future stock behavior, thus facilitating the identification of trading opportunities.

As previously mentioned, combining different preprocessing methods led to a higher quality clustering than using these methods separately. As a consequence, further combinations of prepro-

cessing methods or further fine-tuning of hyper-parameters foreach pipeline could be explored. Assessing the output of the evaluation metrics for these new combinations may lead the clustering algorithm to better capture all the multiscale phenomena of our time series.

3.4 Ethical Considerations

Throughout the project, data was either collected using publicly available APIs or synthetically generated.

While no ethical considerations were identified in the direct scope of our project, we believe methods of unsupervised clustering of time series in finance merit particular scrutiny in the long term. The fact that this project employed unsupervised methods suggests it could potentially replace traders who currently perform the pattern detection themselves. As a consequence, it could be problematic if no humans were present to supervise or understand the machine's decision-making, particularly in case it made a succession of wrong decisions.

For instance, more machines making unsupervised trading decisions might cause more flash crashes or other behaviors which humans cannot really explain. The behavior of a machine trading again itself is frighteningly unpredictable.

References

[1] W. Leigh, N. Modani, R. Purvis, and T. Roberts, "Stock market trading rule discovery using technical charting heuristics," *Expert Systems with Applications*, vol. 23, no. 2, pp. 155 – 159, 2002.

[2] J.-L. Wang and S.-H. Chan, "Stock market trading rule discovery using pattern recognition and technical analysis," *Expert Systems with Applications*, vol. 33, no. 2, pp. 304 – 315, 2007.

[3] A. W. Lo, H. Mamaysky, and J. Wang, "Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation," *NBER Working Paper*, 2000.

[4] T.-C. Fu, F.-L. Chung, V. Ng, and R. Luk, "Pattern discovery from stock time series using self-organizing maps," *Workshop Notes of KDD2001 Workshop on Temporal Data Mining*, 01 2001.

[5] A. Dotis, "Why use k-means for time series data? (part one)," *influxdata*, 2018.

[6] A. Dotis, "Why use k-means for time series data? (part two)," *Medium*, 2018.

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

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

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

[10] 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.

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

4 Appendix

4.1 Key Concepts

4.1.1 Fourier Transform

The 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 = Fs/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, 9].

For some of our preprocessing methods, we applied the Fourier transforms on time series before performing the clustering.

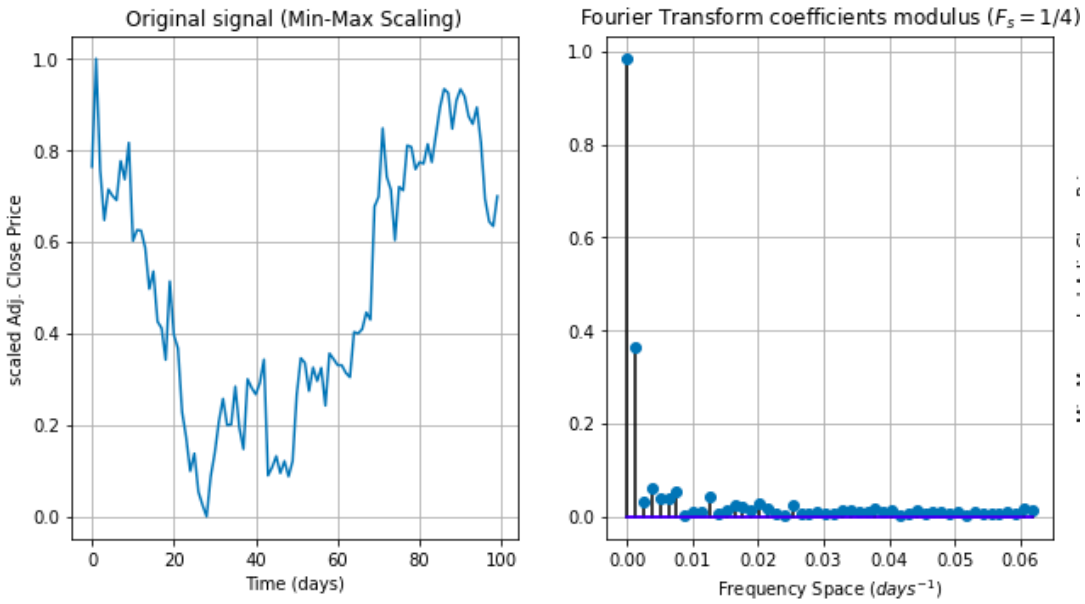

Figure 17: Time Series in the time domain and its Fourier transform (frequency domain)

4.1.2 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. DCT offers an equivalent representation of time series by considering each signal as a superposition of harmonic patterns of different frequencies, rather than points in time.

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. This result can be seen as a consequence of Parseval's Theorem, here applied to DCT, showing Fourier Transform to be a euclidean isometry.

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.

4.1.3 Silhouette Score

The Silhouette Score is a metric that is used to evaluate the quality of clustering. It 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 [10].

4.1.4 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 [11]. 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 metric points and the line (i.e. the distance with its orthogonal projection on that line). We then define the elbow point as the point at which the line is the furthest from metric.

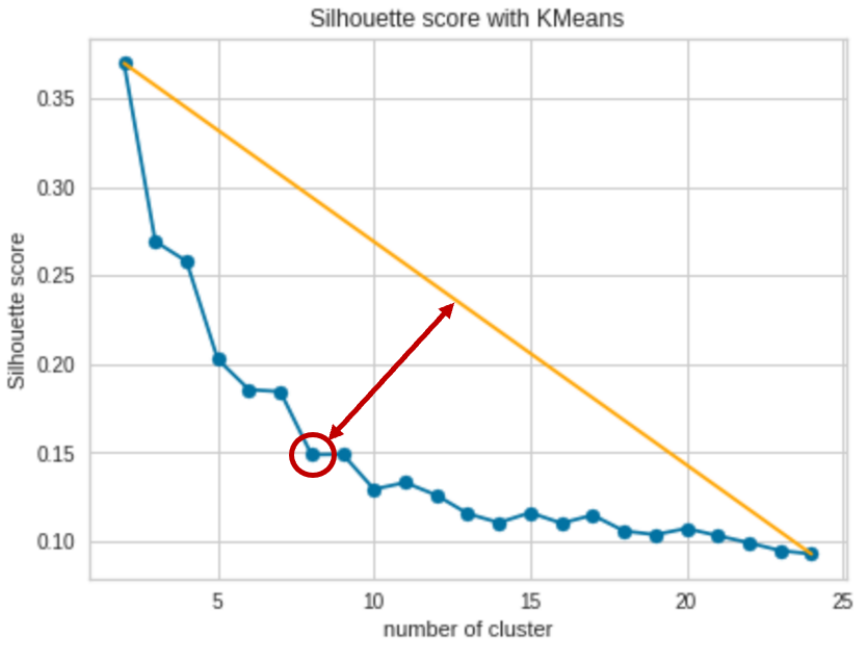

Figure 18: Picking optimal k with silhouette score and elbow method

4.2 Permutation Entropy Results for Multiscale Evaluation

4.2.1 DCT & Skipped Values

Figure 19: PE histogram for DCT-Skipped; First column is long scale, last three are the three short scale windows

4.2.2 DCT & Padded Values

Figure 20: PE histogram for DCT-Padded; First column is long scale, last three are the three short scale windows

4.2.3 Fourier & Skipped Values

Figure 21: PE histogram for Fourier-Skipped; First column is long scale, last three are the three short scale windows

4.2.4 Fourier & Padded Values

Figure 22: PE histogram for Fourier-Skipped; First column is long scale, last three are the three short scale windows

4.2.5 DCT & Autoencoders

Figure 23: PE histogram for Autoencoders Embedding; First column is long scale, last three are the three short scale windows

4.2.6 DCT & PIP Embedding

Figure 24: PE histogram for PIP Embedding; First column is long scale, last three are the three short scale windows