

# **Stock Market Dynamics Visualization**

## **ECM3401: Final Report**

Jia Xin Tang Zhi

May 2022

### **Abstract**

This final year project is undertaken with the ultimate aim to explore ways of automating the replication of correlated portfolios across shares with different algorithms and select one to implement into a visualization tool that will enable us to build our portfolio. The stock market dynamics provide an overview of the global economy that acts as a catalyst for stakeholders to participate in order to benefit different sectors financially, mainly by investing in companies that will reward them as well as the economy. These algorithms are evaluated and compared in order to make a well-educated decision on the essential core aims of my project. By the end I will decide whether the program can efficiently analyse the data available from Yahoo Finance in order to provide a quick overview of the financial market state from companies in the S&P 500 US stock prices. This paper will also discuss issues encountered and conclude by evaluating the project using the method I recognized as most effective and suitable.

**I certify that all material in this dissertation which is not my own work has been identified.**

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Summary of Literature Review and Project Specification</b>	<b>3</b>
2.1	Literature Review Summary	3
2.2	Project Specification	4
2.3	Evaluation Criteria	5
<b>3</b>	<b>Design</b>	<b>5</b>
3.1	High-level Architecture	6
3.2	The machine learning workflow	6
<b>4</b>	<b>Development</b>	<b>7</b>
4.1	Collecting and preparing the data	7
4.2	K-Means Clustering	9
4.3	T-Distributed Stochastic Neighbour Embedding (T-SNE)	9
<b>5</b>	<b>Testing</b>	<b>10</b>
5.1	Backtesting	10
5.1.1.	Pipeline API	10
5.2	Elbow method	10
5.2.1	Silhouette coefficient	11
<b>6</b>	<b>Results and Evaluation</b>	<b>11</b>
6.1	Final Product Description	11
6.2	Departures from Original Requirements	11
6.3	Results	12
6.4	Performance	12
6.5	Scopes	12
<b>7</b>	<b>Critical Assessment</b>	<b>13</b>
<b>8</b>	<b>Conclusion &amp; Future Work</b>	<b>14</b>
	<b>References</b>	<b>16</b>
	<b>Appendices</b>	<b>17</b>

# 1 Introduction

In the financial services industry, building a portfolio requires extensive time and effort. The visualisation of portfolio data will demonstrate how critical optimizing portfolio weights and maximizing diversifications for accurate predictions are, and we will also illustrate how machine learning can facilitate more effective portfolio construction by learning hierarchical relationships from the close value matrix. For the purpose of measuring portfolio performance, i.e. evaluating and comparing different strategies and optimising existing ones, I will use instruments that provide insight into their performance relative to our objectives. In general, these metrics are compared to a benchmark that does not represent traditional investment opportunities, such as the S&P 500 index for U.S. equities or the risk-free interest rate for fixed income assets, but rather an overview of the investment universe. The goal of my project will be to review some specific measurements for comparing portfolio results using a variety of approaches to optimize the portfolio performance interaction of a strategy with the market and compute relevant performance statistics [1].

Portfolio management is a concept that should be highlighted and consists of balancing risk-return trade-offs, that is, to minimize the costs associated with operations and maximize profits. In this paper, we introduce k-means clustering as a method of computing and visualizing key features and comparing the performance against other algorithms. Training, testing, and tuning linear models for regression and classification will then be done using cross-validation so that robust out-of-sample performance can be achieved afterwards. Furthermore, these models will be incorporated into the framework for defining and backtesting algorithmic trading strategies. By combining more diverse information into machine learning models, more accurate predictions will be generated. By doing this, they are able to capture more complex patterns than others that are more prominent. Consequently, various approaches have been developed to optimize portfolios, including the application of machine learning (ML) for learning hierarchical relationships among assets, and for treating assets as either complements or substitutes within the portfolio risk profile [1].

## 2 Summary of Literature Review and Project Specification

### 2.1 Literature Review Summary

Throughout my literature review, I explored the field of machine learning (ML) and examined several methods to analyse the financial market situation in order to help investors make an informed trading decision. The aim of this review was to evaluate existing approaches to classify and assess their relevance of those approaches to this particular problem of stock market dynamics to formulate the most appropriate design and back-end logic for my project.

The following section addresses the findings from my literature review are relevant to my objectives, with the reinforcement learning approach for trading being the most suitable due to its ability to produce some analysis without access to explicit training data, as well as the model of a return-maximizing agent in an uncertain, dynamics environment has much common with an investor or a trading strategy that interacts with financial markets [1]. More important, narrow AI for trade based on machine learning uses voluminous amounts of data and sophisticated algorithms to enhance the accuracy of future predictions [2]. When using raw data, machine learning must identify similarities without prior instruction – unsupervised learning. It appears that a reinforcement learning – learning by trial and error – approach can

be applied to the problem of discovering and implementing dynamic trading strategies, in which machine learning algorithms actively choose and even create their own training data [3].

## 2.2 Project Specification

Time and complexity constraints have led to changes in the project specification over the past year. Machine Learning (ML) has remained a key requirement to perform time series clustering. Due to the lack of research material related to the purpose of this study, I approached it with caution. Although the study produced excellent results, no other academic had replicated this concept. The program was redesigned using a different set of features following much more extensive research.

Accordingly, the project specification will no longer contain features such as the differentiation of companies name, or sector separation. As an alternative, our tool uses a classifier to determine the relevance of the selected dataset. During the training of the model, a methodical approach will be integrated to identify the optimal cluster number, and the model will then recognize what trends are the most evident. Listed below are the updated project specifications, reflecting the minor changes made. The project specifications is divided into functional and non-functional requirements.

### 1. Functional requirements

- (a) Import CSV historical dataset from Yahoo Finance: download our time series data.
- (b) Recognise trends/patterns: similar features detection and be able to discover variations in patterns.
- (c) Separate different industries/sectors: dissimilar companies are detected and clustered visually.
- (d) Precision does not imply output accuracy, but at least distinguishing notable differences, as well as making use of real data.
- (e) Performance: processing time should not be longer than 10 seconds. User acceptance (see Appendix C).
- (f) Portfolio visualisation is clear enough for the user to plan tactics and strategies.
- (g) Usability and responsiveness: the UI complexity will be minimised. User-friendly: engaging and easy to use for all kind of investors.

### 2. Non-Functional requirements

- (a) The visual application will have an easy to use and simple design that does not complicate usage of large datasets.
- (b) Portability: the system will operate on MacOS Desktop with M1 chip, as this is the operating system I have access to.

(c) The model must output different scores associated to each cluster in order to group them, but it would not be shown to the user.

(d) Error entering an invalid dataset: error handling.

## **2.3 Evaluation Criteria**

A combination of unit testing, integration testing, and user testing will be used to assess the functional and non-functional requirements of this project. The project will be successful if the software complies with all the following criteria. This will make it a useful and time-saving alternative to standard stock screeners.

- **Variety**

Variety is an important metric to consider when evaluating charts. To ensure variety between stocks, the software will use several types of industries such as consumer discretionary, consumer staples, energies, financials, health care, industrials, utilities and real estate [4]. Assuming that the timeline of companies is a cycle of a year, the minimum interval will be 12 for the months in a year. These values can be changed by the user if they wish.

- **Cost**

Once the software has found a suitable time series, the computational costs should not be expensive and simple to treat by the users. In this case, empty clusters would also be assessed to ascertain that the provided data does not contain any recognisable patterns.

- **Trading experience**

The program will benefit the users in their trading activities whenever they recognise the resulted similarities in the selected stocks and classify the given time series from the platform Yahoo Finance they would like to study or analyse.

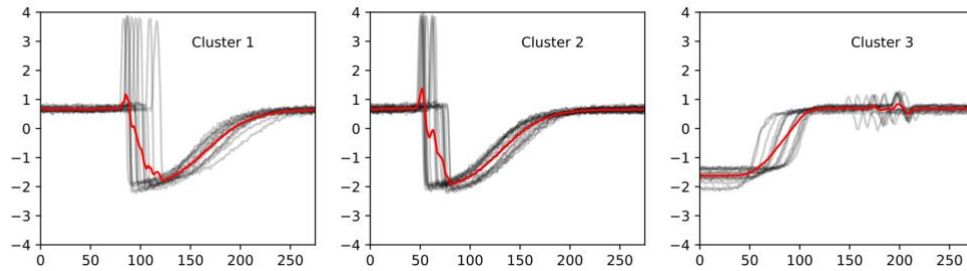
## **3 Design**

At a high level, the design of this project is split into three main parts. Firstly, is the design of the web crawler itself, secondly is the classifier and thirdly is the web application. The design for each of these components have evolved over the year and any modifications from the original design are justified in further sections.

### **3.1 High-level System Architecture**

At a high level, vision-based recognition systems that can autonomously navigate complex historical datasets must successfully develop an association model capable of learning the relationships between different stocks and the corresponding close prices. It is crucial that the model is trained in order to extract the features from the trend that are applicable (for example, those that illustrate its topology and influence) and accurately associate them with the opportune outputs. Therefore, the three main functions of the model can be categorized as follows:

1. **Collecting and preparing the data** – As soon as the data is collected, it is subjected to pre-processing before it is passed on to our selected machine learning algorithms. Tools such as the *elbow* method greatly contribute to identify the desired information. The attributes sampling of inputs in time series can be improved also by using more complex algorithms such as Self-Organising Map (SOM); however, due to time and complexity constraints we discarded this algorithm.
2. **K-Means Clustering** – As shown in *Figure A*, repeated trends are depicted by the optimal number of clusters, each displaying their respective centroids within our time series dataset.



**Figure A.** *K*-means clustering with Euclidean distance.  
The cluster centroids in red capture the shape of the series [5].

3. **T-Distributed Stochastic Neighbour Embedding (t-SNE)** – Lastly, this non-linear algorithm is not limited to hyperplanes and can capture more complex data structures. Nevertheless, it still needs assumptions to reach a solution, but it is an excellent tool for visualizing higher-dimensional data [1].

An unsupervised learning approach is implemented to achieve the ability to *learn* how to cluster similar stocks whereby the ‘stock market dynamics’ is presented with data captured from companies within the S&P 500 of multiples industries such as real estate, health care, energies, etc. navigating financial data from which it can learn to imitate recognisable patterns [4].

### 3.2 The machine learning workflow

Predictive solutions and the key stages leading up to their deployment include [1]:

1. Frame the problem, identify a target metric, and define success.
2. Source, clean, and validate the data.
3. Understand your data and generate informative features.
4. Decide on machine learning algorithms.
5. Cross-validation model design: this important knowledge building stage of the system is achieved during the “learning phase” - an iterative process in which the ML Prediction Model is trained, tested, and tuned.
6. Deploy and predict to solve the original problem.

## 4 Development

Development describes the process of deploying the design along with the different technologies used to execute the project at each level. As with the design phase, the development phase was divided into three stages: extraction of data, development of the *k*-

means clustering, and development of t-Distributed Stochastic Neighbour Embedding. Any deviations from the original plan are explained in accordance with their respective sections as there have been some changes since the initial development plan. This project was developed in Python due to the combination of a large standard library and simplicity. Furthermore, Python offers a rich set of tools that can be incorporated into this project, along with an active support community.

Furthermore, data distributions and visualization are fundamental to data science. There are basically two types of visualizations in data science: (a) metric plots and (b) data distribution plots. However, metric plots are a routine aspect of everyday work of a data scientist, while the data visualization algorithms are relatively rare. Among the metric-based visualizations there are Kullback-Leibler (KL) divergence scores between data sets and time series plots about metric values for example. In other words, data visualization enables us to get a better sense of the data and devise better methods, like the t-SNE, even though the metrics alone suffice in most cases [6].

## 4.1 Collecting and preparing the data

The foundation of an Algorithmic Trading (AT) system is building an infrastructure to handle real-time data and a repository of historical data to calculate values of fundamental, technical, and quantitative metrics that scale the pretrade analysis. *Figure B* provides Python pseudo-code for data access from sources, such as Yahoo! Finance, using *ystockquote*, to import Apple ticker data [7]. Historical data used in this project comes from the Yahoo Finance database for the period 2020-21. All companies in the S&P 500 index were retrieved.

```
-> import ystockquote
-> Input ticker
-> Input start date
-> Input end date
-> Store data in variable "data"
```

Actual Code Fragment from Python:

```
import ystockquote
ticker = 'AAPL'
start = '20200101'
end = '20211231'
data = ystockquote.get_historical_prices (ticker, start, end)
```

**Figure B.** Download Yahoo! Finance data in Python [7].

To properly manage this data, a time series decomposition approach is imperative. The breakdown of a time series into systematic and unsystematic components can be utilised as a helpful abstraction for selecting pattern recognition methods [1].

- **Systematic** – Consistent or linear components that can be described and modelled [1] [8].
- **Non-Systematic** – Components that cannot be modeled directly [8].

In time series analysis, three systematic components are usually defined as follows:

- **Level or cycle.** For a time series, the cycle represents its tendency to rise and fall at inconsistent rates. Business cycles are frequently analysed using this components [8] [9].

- **Trend.** It describes the general direction in which the series is moving, i.e. increasing or decreasing values. It can also have a positive or negative trend, but it may not have one at all [8] [9].
- **Seasonality.** Unlike its cycle component, the seasonal component refers to data that rises and falls at consistent rates during a repeating short-term cycle [8] [9].

And it contains one non-systematic component called noise or remainder.

1. **Noise or remainder.** Having removed the trend, cycle, and seasonal components from the time series data, the remainder component is the random fluctuation found in the data that is not accounting for the components above [8] [9].

When dealing with historical data, it is prudent to use a 'seasonally adjusted' time series. This is a time series without the seasonal component, allowing our system to concentrate on detecting the general trend in the data. Furthermore, decomposition of time series also enables us to translate, rotate and rescale the original features to capture any significant variation in the within the seasonal component, so we can investigate what caused the data to move in the way it did. In most cases, we can distinguish between an additive and multiplicative time series from its variation. We have set our model multiplicative in *seasonal\_decompose* since the magnitude of the seasonal component changes over time; otherwise, the series would have been additive or linear [9]. Even though the data are aggregated by month, the period we wish to analyze is by year, which is why we set it to 12 months [10].

## 4.2 K-Means Clustering

This section explores one of the proposed unsupervised machine learning methods that can be used to extract meaningful signals from our data without having to rely on labels for outcome information. Data points are mapping from the code block, grouped according to their similarity regardless of their labels. Similarity is measured by a distance metric, typically measured using Euclidean distance. The complexity comes with the order of the data points. Often, time series clustering can be performed by flattening the series into tables assigning columns to each time index (or aggregation of the series) and constructing clusters of data by splitting them into  $k$  groups and minimizing the total sum of squares for each group [11]. The K-means algorithm is moderately naive as it classifies the data into a user-specified number ( $k$ ) of clusters irrespective of whether  $k$  is the optimal number. Thus, users who use k-means clustering should be able to assess whether the number of clusters they are using is appropriate. The technique we utilised is the *elbow* method (see *Section 5.2*) in order to validate the number of clusters [1] [12].

Following decomposition and production of our matrix, i.e., an array of arrays of stocks, the open-source SciPy library is used to compute successive clusters using our *k\_means(k)* function on Numpy arrays. *Section 6.3* presents results, with each subfigure depicting series from a given cluster and their centroid (in red). The *cluster.vq* module implements k-means clustering by providing a vector quantization tool. By quantizing vectors, we can reduce distortion and improve accuracy. Usually, the distortion is calculated by finding the Euclidean distance between the centroid and each vector. In the *kmeans2()* method, convergence is not checked by threshold values, but rather by more parameters that determine how centroids are initialized, how empty clusters are dealt with, and whether the input matrix contains only finite numbers. Based on the vector, this method returns centroids and the clusters to which it belongs [13]



[14]. Using this approach has the advantage of being fairly simple to apply, flexible, and works well with large datasets, but it has the disadvantage of having to define the number of centroids manually, being vulnerable to errors, and relying on the initial value of each centroid [15].

### **4.3 T-Distributed Stochastic Neighbour Embedding (T-SNE)**

With t-SNE, the local structure of the data is not only captured but also preserve the global structures of the data like clusters. By preserving the neighbourhood structure of a high-dimensional dataset, t-SNE aims to address some of the problems associated with stochastic neighbour embedding (SNE). Over the potential neighbours of each point, a Gaussian probability distribution is defined centered on that point; the SNE algorithm aims to minimize the difference between probability distributions in the higher dimension and the lower dimension [16].

In the field of machine learning, t-SNE has become a popular technique for analyzing high-dimensional data because it creates two-dimensional maps from data with hundreds or thousands of dimensions. As a result of its versatility, t-SNE has become an increasingly popular dimensionality-reduction algorithm. Unfortunately, its flexibility makes it difficult to interpret. The algorithm adjusts the visualizations out of sight. Yet, by studying how t-SNE behaves in simple cases, you can develop a sense for what is happening [17].

An asymmetric probability matrix is created using the perplexity, a method used in information theory to measure how well a probability distribution predicts a sample. The impact of parameters on embedding for t-SNE is meaningful. It is important to select the right value of perplexity as it balances the local and global aspects of the dataset. A very high perplexity will cause clusters to merge into a single big cluster, while a low value will produce many close clusters that are meaningless. Considering our case scenario, I chose perplexity 75 as it provided a good representation of the global geometry [16].

As a summary, t-SNE is not only effective for non-linear datasets, but it also outperforms other non-linear algorithms. The t-SNE algorithm, which has a tendency to stall at local optima and is slow due to the closest neighbour search queries [16], will be badly affected by intrinsic dimensions greater than 2-3. In these cases, dimensionality reduction will modify the data with the purpose of representing the original information with fewer features, allowing improved predictions. Finally, time series cross-validation with scikit-learn makes the time-series nature of the data imply that cross-validation produces a situation where data from the future will be used to predict data from the past. This is unrealistic at best and data snooping at worst, to the extent that future data reflects past events [1].

## **5 Testing**

### **5.1 Backtesting**

In backtesting, an algorithm is simulated based on historic data in an effort to produce performance results that are generalizable to new market settings. Apart from the general uncertainty around forecasts in ever-changing markets, certain aspects of implementation can also affect the result, increasing the chance of mistaking patterns that hold in-sample for patterns that hold out-of-sample [1]. Jansen's book explains the end-to-end process of designing and testing a ML model and backtest evaluations.

Backtesting a strategy with data survivorship bias can be risky because it can inflate the historical performance of the strategy. Other common pitfalls are related to how the backtest programme is written, or, more fundamentally, storage formats and data types [1] [18].

### 5.1.1 Pipeline API

The Pipeline API simplifies the calculations of factors for a variety of securities using historical data. Pipeline significantly improves efficiency because it maintains an event-driven architecture while vectorizing factor computation where possible [1]. However, during the testing phase, we did not go into great detail.

## 5.2 Elbow method

In the *elbow* method, the ideal value of  $k$  is determined by the sum of squared distances (SSE), which is computed based on the distance between the data points and their clusters. Ideally, the SSE will flatten out at a certain value of  $k$ , at which point we will see an inflection point. Visually, this graph resembles an “elbow”, and thus the name of the method [19].

A feature of this procedure is that it runs  $k$ -means clustering for a range of values of  $k$  (say, from 1 to 10), calculating for each value of  $k$  the SSE. Then, using a line chart, we plot the SSE for every value of  $k$ . The “elbow” on the line chart is the  $k$  value that leads to the best SSE. As we increase  $k$ , the SSE tends to decrease toward 0, since each data point is its own cluster, so there is no error between it and the centroid of its cluster. Hence our goal is to choose a small value of  $k$  that has a low SSE, and there is a point at which increasing  $k$  can produce diminishing returns [12]. The output graph shown in *Figure C* indicates how many centroids should be chosen to achieve better clustering. Based on this graph, it is clear that if the number of centroid is 3, then we will have a better chance of fine clustering. This curve aids in deciding between computational expense and knowledge gain from a dataset [15].

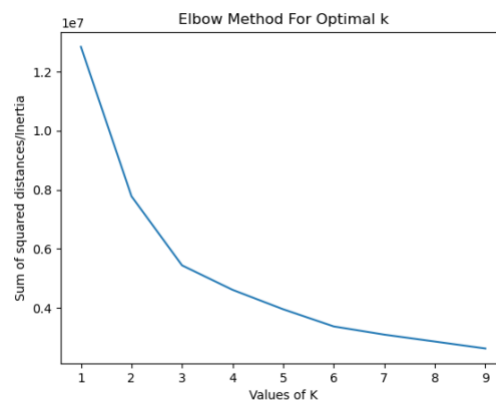


Figure C: A curve to identify the optimal number of clusters.

Nevertheless, the elbow method is not always successful, especially when the data is not well clustered. In cases we see a fairly smooth curve, we might investigate another method for determining the optimal  $k$ , such as computing silhouette scores, or we might reconsider whether clustering is the best way to proceed with our data [12].

### 5.2.1 Silhouette coefficient

The *silhouette coefficient* quantifies how similar a data point is within a cluster (cohesion) to other clusters (separation). We firstly choose a range of  $k$  values (1 to 10 for instance) for each value of  $k$ , and finally plot the *silhouette scores*. Figure D is the equation for calculating the *silhouette coefficient* for a specific data point. In which [20]:

- $S(i)$  is the silhouette coefficient of the data point  $i$ .
- $a(i)$  represents the average distance between  $i$  and all other data points in the cluster to which  $i$  belongs.
- $b(i)$  represents the average distance between  $i$  and all clusters to which  $i$  does not belong.

$$S(i) = \frac{b(i) - a(i)}{\max \{a(i), b(i)\}}$$

**Figure D.** *Silhouette coefficient* equation [20].

The primary distinction between *elbow* and *silhouette coefficient* is that *elbow* only computes the Euclidean distance, whereas *silhouette* considers variables such as skewness, variance, high-low differences, and so on. Because of its calculation simplicity, *elbow* is better suited than *silhouette score* for datasets with a smaller size or time complexity. Scores for silhouette analysis appear to have an advantage over the *elbow* method because they allow you to evaluate clusters based on multiple criteria, and it is very likely that the most optimal number of clusters in K-means is determined [21].

## 6 Results and Evaluation

### 6.1 Final Product Description

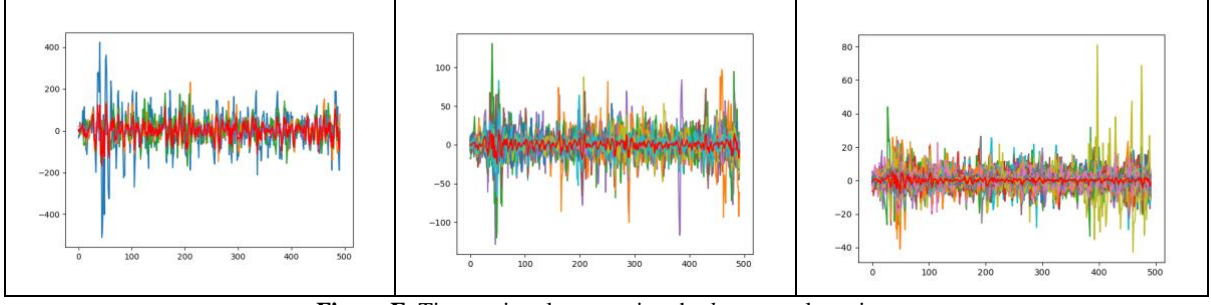
This project is built on the implementation of a stock screener that is distinct from others such as Finviz [22]. It provides a creative way to visualise the capital markets dynamics in order to construct our portfolios.

### 6.2 Departures from Original Requirements

Due to time and complexity constraints, one of the suggested techniques, the SOM algorithm, was eliminated. I did not integrate platforms like *TensorFlow* [23] or *Keras* [24] because they were not essential for the main core of my project.

### 6.3 Results

The fitness for purpose of this project is achieved partially victoriously. In Section 5.2, we concluded that the optimality in the  $k$ -means clustering is found when we set  $k=3$ . For this reason, I obtained three clusters as follows:

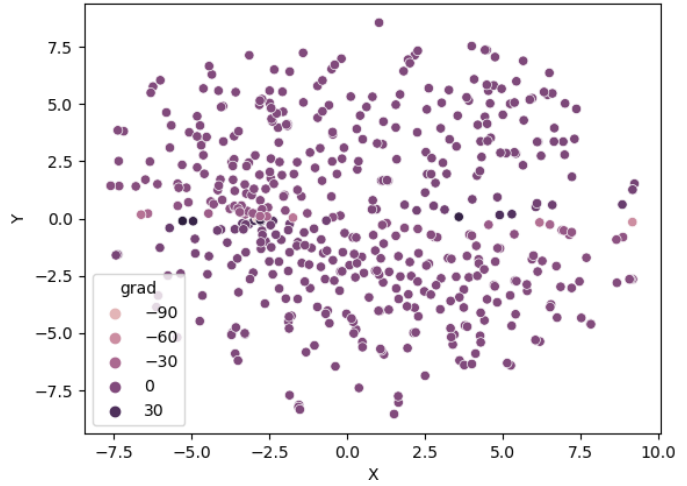


**Figure E.** Time series clusters using the  $k$ -means clustering.

Figure E shows discernible trends that are shared by similar companies, in which the red centroid is the centre of the cluster. It is somewhat the average of the cluster.

A high-level overview of the t-SNE methodology is as follows:

In this example, Figure F performs the same function as  $k$ -means clustering. Clusters are not easily distinguished from one another. The legend, on the other hand, shows that there are five distinct clusters. Again, perplexity=75 provides the best global geometry among other values tested, and verbose=1 allows us to see the computation performance of the program.



**Figure F.** Clusters using t-SNE.

## 6.4 Performance

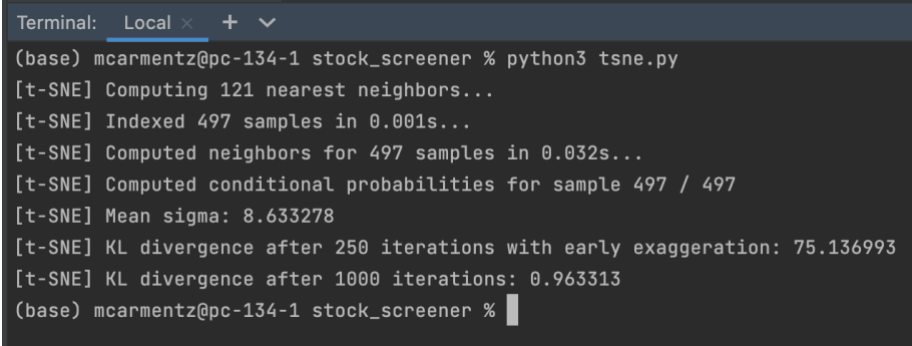
Cross-validation is a technique where a training sample is used to generate the prediction; and a validation sample to confirm that the prediction is correct. The technique is used to assess the algorithm's predictive performance. This study uses a specific dataset, which would have been better showing other periods or removing certain parts of the information to reach a more accurate conclusion.  $K$ -means clustering achieves most of the functional and non-functional requirements from Section 2.2 by producing the visualizations we wanted.

In the context of the t-SNE algorithm we must mention the Kullback-Leibler divergence (KL divergence), whose origins are based on mathematical probability theory and information theory and is popular in the data mining literature. This is a measurement is a comparison between in two probabilistic distributions over the same variable  $x$  or measure. Since KL divergence is not a metric measure, it cannot determine a distance measure, only the "distance" between two distributions [25]. Figure G reflect this equation as follows:

$$D_{KL}(p(x)||q(x)) = \int_{-\infty}^{\infty} p(x) \ln \frac{p(x)}{q(x)} dx$$

**Figure G.** KL equation [25].

According to our performance functional requirement, clusters in the *t\_sne.py* successfully converges in less than 1 seconds for 497/497 conditional probabilities samples. *Figure H* shows the output computations for the *tsne()* function.

A terminal window titled 'Terminal: Local' with a dark background and light text. It shows the execution of a Python script named 'tsne.py'. The output consists of several lines of status messages from the 't-SNE' module, including computing nearest neighbors, indexing samples, computing neighbors and conditional probabilities, and reporting the mean sigma and KL divergence at different iteration points. The prompt '(base) mcarmentz@pc-134-1 stock\_screener %' is visible at the top and bottom of the terminal output.

```
Terminal: Local × + ∨
(base) mcarmentz@pc-134-1 stock_screener % python3 tsne.py
[t-SNE] Computing 121 nearest neighbors...
[t-SNE] Indexed 497 samples in 0.001s...
[t-SNE] Computed neighbors for 497 samples in 0.032s...
[t-SNE] Computed conditional probabilities for sample 497 / 497
[t-SNE] Mean sigma: 8.633278
[t-SNE] KL divergence after 250 iterations with early exaggeration: 75.136993
[t-SNE] KL divergence after 1000 iterations: 0.963313
(base) mcarmentz@pc-134-1 stock_screener %
```

**Figure H.** *tsne.py* logging outputs.

## 6.5 Scopes

Although this project mainly focuses on historical capital time series, it would not work for other types of datasets like linear datasets; we could also adjust assets available with historical prices at minute, hour, daily, weekly, monthly or even yearly intervals. In addition to this, unforeseeable aspects like inflation rates, financial crises, wars, or COVID-19 may have played a role that is not encompassed by standard trends, for which I have accommodated it as a normal pattern.

Some limitations of the *k*-means clustering are that user must manually define the number of centroids, which is prone to errors, and relying on each centroid's initial value; and for the *t*-SNE are that this algorithm does not guarantee to converge to a global optimum of its cost function.

## 7 Critical Assessment

In spite of a couple areas I should have handled differently and a couple areas for improvement, I was satisfied overall with the results as all three main criteria are successfully achieved in *Section 2.3*. Besides creating a useful tool that meets the main objectives laid out at the beginning of my project and resolves the problem that prompted my choice of this topic, gained first-hand experience with a wide range of software development methodologies as well as gained valuable insight into the exciting area of machine learning.

On the other hand, I have obtained this helpful model, which I will undoubtedly apply in my future financial operations. Although the programme is not perfect, my intention is to address bugs and produce a better version of this stock screener.

Lastly, I conducted user testing survey through a User Acceptance Form (see *Appendices A*), in which five computer science students answered:

- 60% of the students replied it was easy to navigate through the program application layout .
- 20% of people understood the information.
- 100% of the group thought some information was missing, mainly because there was not a user guide on how to use it. It was a bit confusing as not everything was labelled.
- 20% have encountered trouble establishing a trading decision.

Improvements I have been suggested are making the user interface friendlier applying different shades of the colours red and green to indicate good or bad trades.

## 8 Conclusion & Future Work

We began by presenting the explosive growth of digital data as well as the rise of machine learning as a strategic tool in the realm of investment and trading. In response to the emergence of global business and technology trends beyond finance, this dynamic is more likely to persist than stall or reverse in the near future, as many investment firms are just starting to leverage artificial intelligence tools, and employees are refining the relevant skills and business processes towards value creation by exploiting these new opportunities. The coming years will likely see numerous fascinating developments for the application of ML to trading, including the automation of the machine learning process, the generation of synthetic training data, and quantum computing. Jansen propose several providers that are designed to simplify ML workflows. The *H2O.ai* platform, for instance, combines cloud computing with machine learning automation, which allows users to explore patterns in data using thousands of potential models, with interfaces in Python, R, and Java. A service such as *Dataiku* assists analysts and engineers in exploring, prototyping, and building their own data applications; or a platform such as *Datarobot*, which automates the model development process that allows a rapid building and deployment of predictive models in the cloud or on-premises [1].

Throughout this project, we discussed several libraries from the Python ecosystem. Python has evolved into the language of choice for data science and machine learning. In recent years, *NumPy* and *SciPy* are among the open-source libraries that have continued to grow and diversify, and they are built on the robust core of scientific computing libraries. As a result of the popular *pandas* library, Python has been the go-to tool for data science, as well as *scikit-learn*, one of the most popular support libraries for modern, specialized ML, and that often integrates with workflow automation tools like Pipeline (see *Section 5.1.1*). Additionally, there are several companies that leads open-source initiatives in order to build and extended Python ecosystem. For example, the quantitative hedge fund *TwoSigma* or *Bloomberg*, providing quantitative analysis tools to the Jupyter Notebook environment and facilitating an interactive analysis of financial data [1].

On balance, according to the general theory of  $k$ -means clustering, the data points in various clusters would be dissimilar to one another, and this can be optimized using the *elbow* method. It has advantages such as scaling to large datasets, converging always, and is often (but not always) faster than other clustering methods like hierarchical clustering. However, it has shortcomings such as clustering imbalanced data, choosing  $k$  manually, relying on initial assumptions, and scaling with high dimensions [19]. By using the *elbow* method, we predicted that 3 centroids would improve the clustering, so I selected this number of clusters, which had improved information gain, and obtained better clusters [15].

In my future work, I would have liked to have completed the third algorithm proposed, Self-Organising Map (SOM). Unfortunately, due to the time and complexity limitations, I was unable to pursue this option. Further exploration will include the use of *TensorFlow* while creating some visualisations [26]. Other research may even take into account sentiment analysis, technical indicators, and news data will be examined in addition to other types of data, such as fundamental indicators, and social media data from a variety of social media platforms such as Reddit or Twitter. This project has given me valuable experience in software development that will help me in my future career, as well as a useful tool to use in making my investment decisions. As a whole, even though the second algorithm works, it did not meet the objectives I had planned, I consider this project to be a modest success.

## References

- [1] S. Jansen. 2020. “Machine Learning for Algorithmic Trading: Predictive models to extract signals from market and alternative data for systematic trading strategies with Python (2<sup>nd</sup> Edition)”. *Expert Insight*. Birmingham.
- [2] J.P. Meltzer. 2018. “The impact of artificial intelligence on international trade”. *Brookings Institute*. Retrieved from [https://www.hinrichfoundation.com/media/2bxltgzf/meltzerai-and-trade\\_final.pdf](https://www.hinrichfoundation.com/media/2bxltgzf/meltzerai-and-trade_final.pdf)
- [3] G. Ritter. 2017. “Machine Learning for Trading”. *SNNR*. Retrieved March 8, 2022 from [https://papers.ssrn.com/sol3/papers.cfm?abstract\\_id=3015609](https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3015609).
- [4] Barchart. 2022. Retrieved from <https://www.barchart.com/stocks/indices/sp-sector/industrials?viewName=main>.
- [5] Tsllearn. 2017. “Time Series Clustering”. Retrieved January 8, 2022 from [https://tslearn.readthedocs.io/en/stable/user\\_guide/clustering.html](https://tslearn.readthedocs.io/en/stable/user_guide/clustering.html).
- [6] ScienceDirect. 2022. “Kullback–Leibler Divergence”. *Elsevier*. Retrieved from <https://www.sciencedirect.com/topics/mathematics/kullback-leibler-divergence>
- [7] P. Treleaven, M. Galas, and V. Lalchand. 2013. “Algorithmic Trading Review”. *Commun. ACM* 56, 11 (November 2013), 76-85. Retrieved from <https://doi.org/10.1145/2500117>
- [8] J. Brownlee. 2017. “How to Decompose Time Series Data into Trend and Seasonality”. *Machine Learning Mastery*. Retrieved January 4, 2022 from <https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality/>.
- [9] A. Plummer. 2020. “Different Types of Time Series Decomposition”. Retrieved January 3, 2022 from <https://towardsdatascience.com/different-types-of-time-series-decomposition-396c09f92693>.
- [10] B. Bonaros. 2021. “Time Series Decomposition In Python”. Retrieved January 7, 2022 from: <https://towardsdatascience.com/time-series-decomposition-in-python-8acac385a5b2>.
- [11] A. Amidon. 2020. “How to Apply K-means Clustering to Time Series Data”. *Towards Data Science*. Retrieved January 9, 2022 from <https://towardsdatascience.com/how-to-apply-k-means-clustering-to-time-series-data-28d04a8f7da3>.
- [12] R. Gove. 2022. “Using the elbow method to determine the optimal number of clusters for k-means clustering”. Retrieved February 24, 2022 from <https://gist.github.com/rpgove/0060ff3b656618e9136b>.
- [13] GeeksforGeeks. 2022. “K- means clustering with SciPy”. Retrieved from <https://www.geeksforgeeks.org/k-means-clustering-with-scipy/>.
- [14] The SciPy community. 2022. “Scipy.cluster.vq.kmeans”. Retrieved from <https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.vq.kmeans.html>



- [15] Y. Chauhan. 2020. “K Means clustering with python code explained”. *Towards Data Science*. Retrieved February 10, 2022 from <https://towardsdatascience.com/k-means-clustering-with-python-code-explained-5a792bd19548>.
- [16] Asdspal. 2022. “Tsne.ipynb”. *GitHub*. Retrieved 23, Marh from <https://github.com/asdspal/dimRed/blob/master/tsne.ipynb>.
- [17] Distill. 2022. “How to Use t-SNE Effectively”. Retrieved February 6, 2022 from <https://distill.pub/2016/misread-tsne/>
- [18] E. P. Chan. 2021. “Quantitative Trading: How to Build Your Own Algorithmic Trading Business (2<sup>nd</sup> Edition)”. *Wiley*. New Jersey.
- [19] Vatsal. 2021. “K-Means Explained”. *Towards Data Science*. Retrieved April 17, 2022 from <https://towardsdatascience.com/k-means-explained-10349949bd10>.
- [20] A. Banerji. 2021. “K-Mean: Getting The Optimal Number Of Clusters”. *Analytics Vidhya*. Retrieved April 17, 2022 from <https://www.analyticsvidhya.com/blog/2021/05/k-mean-getting-the-optimal-number-of-clusters/>.
- [21] A. Kumar. 2021. “Elbow Method vs Silhouette Score – Which is Better?”. *Data Analytics*. Retrieved April 1, 2022 from <https://vitalflux.com/elbow-method-silhouette-score-which-better/#:~:text=The%20major%20difference%20between%20elbow,high%2Dlow%20differences%2C%20etc>.
- [22] Finviz. 2007. “Fviz: financial visualizations”. Retrieved from [https://finviz.com/map.ashx?t=sec\\_all&st=w1](https://finviz.com/map.ashx?t=sec_all&st=w1).
- [23] TensorFlow. 2021. “An end-to-end open source machine learning platform”. Retrieved from <https://www.tensorflow.org/>.
- [24] Keras. 2021. “Deep learning API written in Python”. Retrieved from <https://keras.io/>.
- [25] S. Shukueian. 2022. “Divergence”. Retrieved March 15, 2022 from <https://shukueian.medium.com/divergence-14e8c60a70e2>.
- [26] J. D. Seo. “Implementing T-SNE in Tensorflow [ Manual Back Prop in TF ]”. *Towards Data Science*. Retrieved May 1, 2022 from <https://towardsdatascience.com/implementing-t-sne-in-tensorflow-manual-back-prop-in-tf-b08c21ee8e06>.

## Appendices

### A User Acceptance Form

## Stock Screener User Acceptance Testing

Q1) Were you able to navigate easily using the layout of the program application?

Yes ☐ No ☐

Q2) Were you able to clearly understand the information?

Yes ☐ No ☐

Q3) Was all the information you needed available? If not, please describe what information is missing?

Yes ☐ No ☐

.....

.....

Q4) Did you have trouble establishing a trading decision?

Yes ☐ No ☐

Q5) What was your experience with analysing the results of the program?

.....

.....

Q6) What improvements could be made to the application in regard to its usage?

.....

.....