UNIVERSITY OF LEICESTER

Build a Stock Market Analyzer

Time Series Clustering on FTSE 100 Index

A dissertation report submitted to the ***University of Leicester*** for my Master's degree course in ***'Data Analysis for Business Intelligence'.***

By

Zoeisha Alle

University of Leicester

Abstract:

The global economy centers on the stock market, which presents a rich landscape of investment opportunities and offers a window into the overall financial picture. Yet, due to its intricate nature and sheer unpredictability, the market is no easy ride for investors. Many stock traders have thus turned to something more than mere guesswork to help them navigate the pricing decisions of the stock market—that something is called "analysis." In this research, I aim to employ the technique of time series clustering to the stocks of the FTSE 100 over the course of 2 years, from January 2021 to December 2022.

Investors and data analysts can use the unsupervised learning method of clustering to gain insights into the stock market. By using this technique on stock price data, one can uncover some of the hidden patterns present in stock trends. The research described here involves the daily adjusted close prices of 100 tickers in the FTSE 100 index. These data were obtained using the yfinance package in Python. Principal Component Analysis (PCA) was used to reduce the dimensionality of the dataset and find some of the key features that drive stock price changes.

I've applied three clustering techniques—K Means, Hierarchical Clustering (Agglomerative), and DBSCAN—to create a solution of distinct clusters and then performed a rigorous evaluation of each of these algorithms using a variety of validation metrics and also through a set of visualizations. The results obtained and the appearance of our visualizations suggest that both K Means and Hierarchical Clustering provide adequate solutions. They create distinct clusters that identify stocks with similar important characteristics related to market volatility and performance across time.

From these clusters, investors obtain useful information for creating diversified portfolios and identifying market trends. For data analysts, sharp predictive models and accurate financial forecasts result from the same patterns. Principal components analysis and dendrograms serve as visual aids to showcase the uniqueness of the clusters. The study reinforces the idea that time series clustering is an important tool for investors and analysts, providing precious intelligence on the movements of the FTSE 100 and by pinpointing robust groupings of stocks, this project enables even better investment strategies and financial evaluations that are more strongly supported by data.

Acknowledgements

I would like to sincerely thank all my professors at the University of Leicester for their constant support throughout my postgraduate studies. I am especially grateful to my supervisors, Yanshan Shi and Dr. Larissa Serdukova, for their insightful guidance and direction during my project.

A special mention goes to Professor Evgeny Mirkes, whose teachings in Data Mining, Neural Networks, and Mathematical Modelling ignited my enthusiasm for exploring data and applying it to real-world scenarios.

It has been a privilege to be part of the Mathematical Science Department, where I have honed my skills in Data Analytics. I also feel fortunate to have been part of the 'Citizens of Change' initiative and deeply appreciate the camaraderie of my fellow students and friends from the 'Data Analysis for Business Intelligence' program, who supported me along this journey.

Declaration:

I hereby would like to declare that the information in this document is original, and any information gathered from books, websites or any other sources are clearly mentioned in the references with author names.

Name : Zoeisha Alle

Date : 30-09-2024

Table of Contents

Table of Figures

1.0 Introduction

The stock market is one of the key elements of a free-market economic system, allowing companies to access capital by selling shares and providing investors with the chance of realizing gains through equity ownership. Stock exchanges such as the London Stock Exchange (LSE) serve as a venue where shares can be bought and sold, creating an environment in which company performance, economic trends, and market sentiments converge. The FTSE 100 Index (Footsie) is one of the most important indicators for the financial industry in the UK, and although it is not particularly focused on tech, many technology based companies are represented within its top 100. It is a proxy reflecting the sum of movement in the stock prices of these companies and provides significant insights into market dynamics and economic health.

Stock markets, like the FTSE 100, have a big impact on decision-making over investing. Stock prices and indices are regularly monitored by everyone from individual retail traders to major institutional investors in an attempt to assess risk and possible returns. In a bull market, stocks generally rise as investors anticipate and push through economic growth, so the tell-tale signs of high volumes of falling stock prices scant safer havens asset buyers. Stock prices are volatile and may be affected by macroeconomic numbers, such as gross domestic product (GDP), geopolitical events or development, company news, and performance, which can influence the stock price tremendously. These fluctuations are what investors need to interpret to make logical decisions. These movements typically correlate, sometimes to a high degree of correlation, due to shared external influences, which allows for portfolio optimization and risk management by investors.

Furthermore, stock prices can change the overall economic picture. A long-term bull run can boost consumer and business confidence as well as company valuations, in turn pushing the economic scenario forward through increased investment and employment. On the contrary, an extended downtrend can erode investor sentiment and lower corporate investments that eventually trigger economic recessions, as seen in the 2008 Global Financial Crisis. Therefore, it is critical for us to analyze historical stock pricing trends in order to learn about the trend of a market, smartly invest, or predict future economic conditions.

The difficulty and magnitude of stock price data also reduce the effectiveness of traditional analysis approaches. If a similar approach could be taken to group together stocks with a common price behavior, we may discover patterns that are not as apparent in individual stock analysis. Investors use clustering algorithms to discover the stocks that co-move together, thereby unlocking sector-based risks and constructing returns-maximizing portfolios with risk guardrails.

Based on this analysis, the project aims to provide investors with actionable insights that can help them group stocks with similar price trends together. This classification of equity categories may be useful in strategic investment planning to mitigate risks and diversify portfolios. Moreover, this work helps the emerging sector of Financial Data Science by combining sophisticated machine learning concepts with classical market analysis, and in turn will provide a more complete diagnostic on how stock markets behave.

1.1 Problem Statement

Examining stock performance is crucial for making informed investment decisions across different industries and investors need to analyze historical stock price trends to identify patterns that guide future strategies.

Employing Clustering techniques to the time series stock data help group stocks with similar behaviors, offering deeper insights into market dynamics. Therefore, exploring these methods is key to enhancing investment strategies and gaining a better understanding of financial market behavior.

1.2 Objective

The elemental goal of this project is to assess the time-series data (historical stock values) of FTSE 100 stocks so that we can develop methods to specifically classify and ascertain the kind of stock, depending upon their performance records. Especially for this study this research was projected from the following perspectives:

- ➔ To analyze historical stock price data with an intention to recognize performance based mainly on price action, trends, and correlations among major companies.
- ➔ Apply clustering to sort companies that present similar price behaviors and trends over periods of history, displaying linkages in the market.
- ➔ Evaluate the quality of clusters and determine whether they offer insights in a meaningful way about stock performance or even about market dynamics.
- ➔ Share actionable ideas for helping investors make informed investment decisions through learning patterns and relationships in the stock market.

2.0 Literature Review

2.1 Overview of Stock Market Behavior and FTSE 100 Index Composition

Stock markets, such as the FTSE100 index, are inherently dynamic systems where stock prices move up or down due to a multiplicative effect of macroeconomic factors, company performance, investor sentiment, or external events. Every trading day, the levels of each stock are recorded as prices in a time series by databases like Xignite. This is where financial time series data come into play — they tell us how stock prices behaved historically and allow everyone from researchers to investors to obtain insights on trends, volatility, turning points, and moreover in different time periods. In particular, the adjusted close is heavily used as it adjusts for corporate actions such as dividend payments and stock splits to provide a more accurate view of what the stock has actually done throughout history.

The FTSE 100 index is made up of the 100 biggest companies listed on the London Stock Exchange (LSE) by market capitalization. These are from a variety of different industries, offering a broad set of stocks to choose from. This index can inform global investors on the state of the UK economy, we can track the performance of these companies to get decent insights into economic trends and changes in particular sectors.

Below is the list of Tickers within the FTSE 100 including their industry, there are about 40 unique industries.

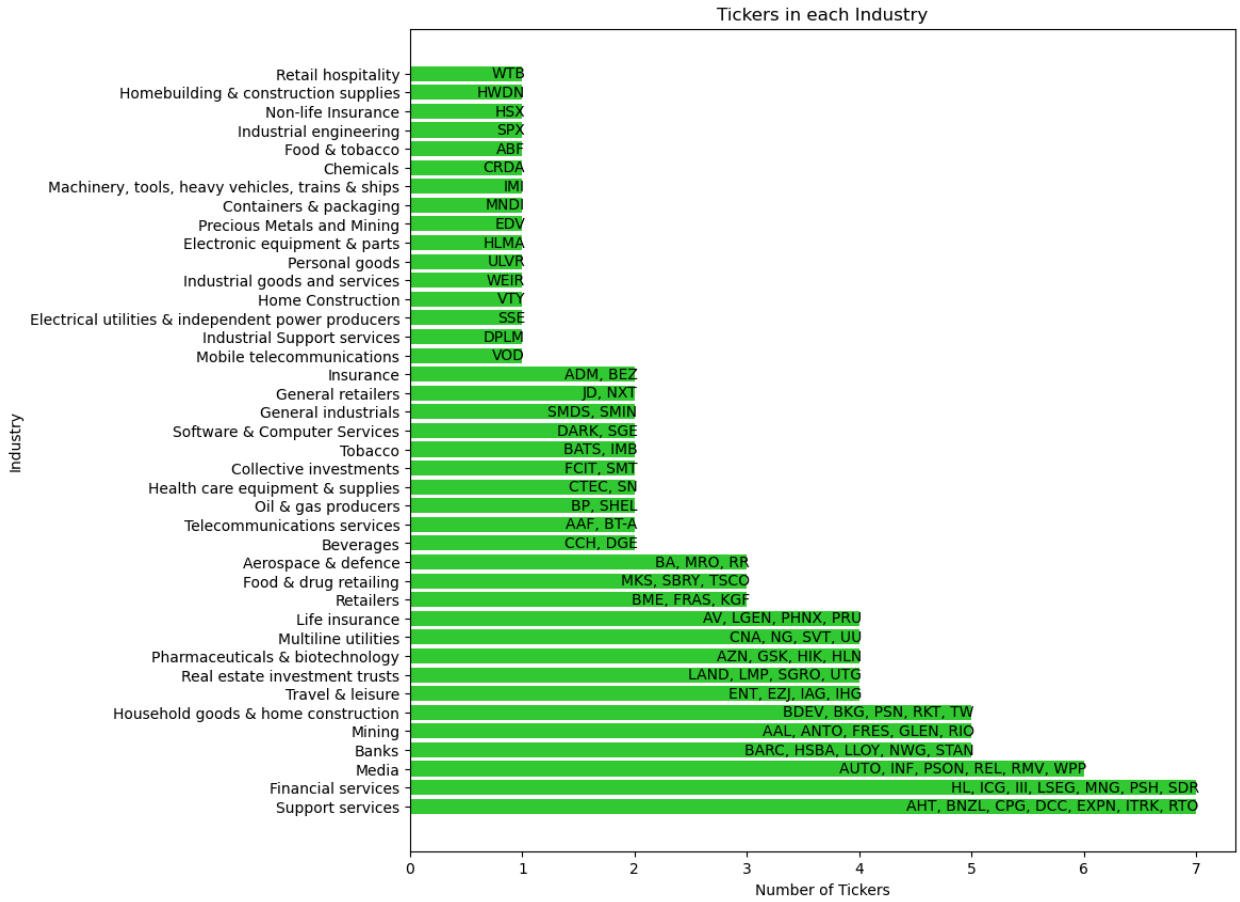


Fig 1: Stocks listed within each sector

Companies within the FTSE 100 are spread across a number of sectors; some notable examples

- ➔ Financials: HSBC, Lloyds Banking Group, and Standard Chartered which collectively represent an area that is central to the index—UK banks.
- ➔ Consumer goods companies: Represented by names such as Unilever or Reckitt Benckiser, are another large component of the index and can affect how it performs based on consumer spending trends.
- ➔ Energy: Given it reflects global energy market trends, notable names such as BP and Royal Dutch Shell are sensitive to both oil prices and geopolitical events.
- ➔ Healthcare: British firms like AstraZeneca and GlaxoSmithKline are key players in the healthcare sector, helping to pioneer new treatments through pharmaceuticals and biotechnology.
- ➔ Tech: The FTSE 100 is becoming more tech-heavy than before, with a greater weighting to stocks like Scottish Mortgage Investment Trust, though still not close to the level of something like NASDAQ.

Different underlying market forces at work in each of these sectors tend to make them behave differently. For instance, energy stocks might exhibit a response to changes in oil prices more than, say, the effects of inflation or consumer

sentiment on consumer goods. Grouping these companies by their tendencies to change in price throughout history allows investors to segment stocks not only by sector but also by how they interact with market trends, giving more detailed views of the market.

2.1.1 Importance of Analyzing Financial Time Series

Understanding financial time series data is essential for recognizing market patterns, trends, and irregularities. Appraisal experts and financial researchers who want to forecast stock moves, hedge risks, and make appropriate investment decisions need this analysis. While more common time series analysis methods, such as moving averages, autoregressive models, and volatility estimations (such as GARCH models), have been widely used, clustering allows stocks to be grouped together under conditions of similar price behavior.

2.1.2 Challenges in Financial Time Series

One of the biggest obstacles in studying financial time series is their noisy and non-stationary nature. Stock prices do not follow any single linear path, as they are affected by sudden market events, making it difficult to predict long-term behavior without proper models. This is where clustering algorithms are most useful, as they can help identify groups of stocks that move similarly, allowing investors to create clusters of stocks that respond the same way to market conditions.

2.2 Clustering

2.2.1 Clustering in Stock Markets

Clustering is an unsupervised technique extensively used in financial data analysis for finding natural groupings in data without predefined labels. In stock markets, clustering is especially effective for grouping stocks with similar characteristics, enabling investors to diversify their portfolios, spot sectors with comparable risks, and predict stock performance based on clusters. This is typically done by establishing a "distance" between time series according to some similarity measure and then applying an algorithm that rearranges the stocks.

2.2.2 Application of Time Series Clustering

Cluster analysis has been applied in various studies to financial time series, with differences typically found in the distance measures and clustering algorithms used. For instance,

- ➔ Aggarwal & Yu (2003) explored clustering applications on large financial data and discussed the reduction of dimensionality.
- ➔ Liao (2005) provided a detailed survey on time series clustering algorithms, including their use in finance.
- ➔ Wang et al. (2020) used advanced deep learning methods for clustering, where neural networks were employed to transform stock data into a low-dimensional representation before clustering. These studies indicate that clustering can reveal hidden patterns in financial time series, though the choice of distance measure is crucial.

2.2.3 Distance Measures in Time Series Clustering

Distance measures between two time series in time series clustering govern the similarity, and the choice of metric is particularly important when working with financial data. Below are some commonly used distance measures and their practical application:

1.) **Euclidean Distance:** It is the straight-line distance between points in two time series. By summing the squared differences between the values of two time series at each time point, we can calculate how far apart these two series are. This method works best when the time series are of equivalent lengths and properly aligned.

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Here x_i and y_i represent the stock prices of two different stocks at time point i. The differences are squared to avoid cancellation of positive and negative differences, and the square root ensures that the measure is in the same unit as the original data.

Euclidean distance works well for stocks that follow the same path, particularly in terms of similar magnitudes and timing. However, it breaks down when time shifts occur or when data has widely varying lengths.

2.) Manhattan Distance(Cityblock - Python): Manhattan distance or L1 norm, is the sum of absolute differences between corresponding points for a pair of time series. It differs from Euclidean distance, which squares the differences, making Manhattan distance less sensitive to large outliers or extreme deviations.

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

For each time point i, we take the sum of the absolute differences in the prices of stock 1 and stock 2. Because it uses absolute values, Manhattan distance treats all offsets as equal. This is particularly useful in highly volatile markets, where stock prices can make dramatic spikes or drops, as it provides a better summary of how stocks move out of sync with each other.

3.) Cosine Similarity: It measures how far apart time series are when treated as vectors in a multidimensional space. Instead of focusing on the magnitude of changes, it considers how similar or dissimilar the direction of the time series' movements is

$$Cosine\ Similarity(x, y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \cdot \sqrt{\sum_{i=1}^{n} y_i^2}}$$

In the equation above, the numerator is the dot product of two time series (which serve as placeholders for stock price data), and the denominator is their magnitude. This way, equivalent prices are reduced to make the similarity metric independent of price level. Since cosine similarity helps us put stocks that follow similar trends at different prices over time, it is the perfect metric to group these stocks together. So, for example, if two stocks are of vastly different prices but both moving in the same direction, they will appear to be similar on this scale.

2.3 Clustering Algorithms for Stock Data

A pivotal aspect of structuring clustering algorithms is to spot underlying price patterns by classifying stocks, making it easier for investors to identify some regulation in stock behavior amid large data sets. These algorithms reveal hidden structures, such as how certain stocks move together or react similarly to external market events, allowing for more informed investment strategies. Clustering algorithms classify stocks with shared characteristics, highlighting:

- ➔ **Market Segmentation:** Finding sectors or groups of stocks that move together, allowing us to see how different industries perform in response to economic events.
- ➔ **Risk Management:** If securities with identical price movements are bucketed into one, then an investor can easily mitigate risks by not taking too many of the same style.
- ➔ **Identifying trends in stock prices:** Clustering provides the ability to identify groups of trends that are unrecognizable or intuitively obvious between macroeconomic conditions, stocks, and so forth!

2.4 Algorithms employed in this analysis

2.4.1 K - Means

It is one of the popular clustering algorithms, especially used in financial time series analysis for stock grouping. It creates clusters of best fit for stocks by grouping them into K groups, where K is the number of clusters, and assigns each stock to the cluster closest to the underlying centroid using a certain distance measure (usually Euclidean). It then updates centroids and reassigns stocks iteratively until the clusters stabilize.

Reason to use K-Means:

- ➔ Efficiency: K-Means works well with large datasets, making it ideal for stock market data.
- ➔ Pattern Detection: It uncovers clusters of stocks that trade similarly, helpful for tracking trends and managing risk.
- ➔ Sector Analysis: K-Means can group stocks from the same industry or sector, offering insights into sector-specific movements.

Limitations:

- ➔ Requires the number of clusters (K) to be specified before running.
- ➔ Sensitive to the initial placement of centroids.
- ➔ Assumes clusters are spherical, which may not always represent stock market behavior.

Techniques like the Elbow Method and Silhouette Score are used to optimize the choice of K and evaluate clustering quality.

WCSS: WCSS stands for Within-Cluster Sum of Squares, which defines how compact our clusters are. As the algorithm iterates, the WCSS becomes minimal to ensure that stocks within a cluster are as close as possible to their centroid.

$$WCSS = \sum_{j=1}^{K} \sum_{x_i \epsilon S_j} (d(x_i, c_j))^2$$

Where:

- K is the number of clusters.
- $d(x_i, c_j)$ is the Euclidean distance between stock i and the centroid c_j
- S_j is the set of stocks in cluster j.

K-Means tries to minimize the WCSS, resulting in more compact and well-formed clusters. K-Means was also used in a study by Huang (1998), where it was shown to work effectively for financial data concerning stock prices, proving its efficacy with large datasets and detecting meaningful patterns within.

2.4.2 Hierarchical Clustering

Hierarchical clustering is a powerful technique used to group stocks by their similarity over time, forming a visual representation known as a dendrogram. Unlike the K-Means algorithm, which requires the number of clusters to be predetermined, hierarchical clustering adapts when the ideal number of clusters is unknown, making it particularly advantageous in stock analysis. This clustering can be performed in two main ways:

- ➔ Agglomerative (bottom-up): This method begins by treating each stock as a separate cluster. Gradually, these clusters merge based on similarity, continuing until only one cluster remains.
- ➔ Divisive (top-down): In contrast, divisive clustering starts with all stocks in a single cluster, which is then split recursively into smaller clusters.

Both approaches are useful for revealing the relationships between stocks at different levels of similarity. The dendrogram, in particular, offers investors a clear visualization of how stocks group together over time. This allows them to uncover patterns that might span entire sectors or be specific to individual companies.

Through hierarchical clustering, investors can better understand:

- ➔ Stock correlations: Stocks that exhibit similar price movements over time, often belonging to the same industry or sector.
- ➔ Market segmentations: Clear divisions within the market where stocks naturally form distinct groups based on their price behaviors.

The process begins by calculating the distance or dissimilarity between pairs of stocks, typically based on their price movements. To measure this, analysts can choose from several distance metrics, such as Euclidean distance, Manhattan distance, or Cosine similarity, depending on the nature of the stock data.

2.4.2.1 Linkage Methods for Hierarchical Clustering

The linkage methods define the distance between two clusters and there are four common linkage methods

1.) **Single Linkage ("Minimum")**: This method links the two clusters that have the least distance between them in terms of their closest members. It connects clusters only if they are the nearest to each other, which often creates long, chain-like clusters.

$$d(C_i, C_j) = \min_{p \in C_i, q \in C_j} d(p, q)$$

From the above formula, C_i and C_j are clusters.

$p\ and\ q$ are points (i.e., stock price vectors) in clusters C_i and C_j, respectively.

$d(p, q)$ is the distance (often Euclidean) between point p in cluster C_i and point q in cluster C_j

2.) **Complete Linkage ("Maximum") :** Complete linkage, on the other hand, calculates the largest distance between any two points in the clusters. It usually leads to more tightly packed clusters but is sensitive to outliers.

$$d(C_i, C_j) = \max_{x \in C_i, y \in C_j} d(x, y)$$

3.) **Average Linkage**: This method will compute the average pairwise distance between all points in two clusters, helping to achieve more balanced clusters.

$$d(C_i,\ C_j) = \frac{1}{|C_i||C_j|} \sum_{x \epsilon C_i, y \epsilon C_j} d(x, y)$$

Where $|C_i|\ and\ |C_j|$ are the number of stocks in clusters C_i and C_j, respectively.

4.) **Ward's Method (Variance Minimization) :** It minimizes the increase in variance when two independent clusters are merged. It is popular because it can produce cost-effective clusters that are mostly spherical.

$$d(C_i,\ C_j) = \frac{|C_i||C_j|}{|C_i|+|C_j|} \left|\left| \overline{C}_i - \overline{C}_j \right|\right|^2$$

From the formula, $\overline{C_i}\ and\ \overline{C_j}$ represent centroids of clusters $C_i\ and\ C_j$

$\left|C_i\right| and \left|C_j\right|$ are the sizes of the clusters

$\left\|\overline{C_i} - \overline{C_j}\right\|$ is the Euclidean distance between the centroids of the two clusters.

2.4.2.2 Dendrogram Construction

The algorithm merges the clusters step-by-step after computing the distances and its application with a given linkage method. The process can be visualized with a dendrogram that illustrates which stocks or clusters amalgamate at what level of similarity. The distance at which the clusters are merged (called a height) is represented along the height of branches.

Cutting the Dendrogram: After determining how far apart clusters are, the next step is to "cut" the dendrogram at some level to create clusters based on a desired similarity threshold. Where the cut is taken determines the number of clusters, allowing flexibility in selecting fewer clusters.

In a study by Müller et al., to classify stocks based on their movements, a cluster analysis described by Patil and Kamble (2010) was performed using hierarchical clustering. These groups of stocks, which showed similar patterns in various market conditions, were highlighted on a dendrogram, providing useful insights for investors seeking to diversify or select correlated assets.

2.4.3 DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

It is a clustering algorithm that separates data points based on density, making it robust for clusters with arbitrary shapes and noise (outliers) in the dataset. This is especially ideal for stock data, where typical cluster patterns might not accurately predict market movements, while also accounting for noise (irregular price changes on a daily basis).

Key Parameters of DBSCAN:

- ε **(Eps)**: The distance that delineates a data point's neighborhood. A cluster is thought to consist of all points that are within this distance from a core point.

$$N_\varepsilon(p) = \{q \epsilon D | d(p,q) \leq \varepsilon\}$$

  Where $N_\varepsilon(p)$ is the set of points within distance ε of point p, $d(p,q)$ is the distance between points *p and q, and D* is the dataset.

- **Min Samples**: The minimum number of points required to form a **dense region**. A point is considered a **core point** if it has at least points within its ε - neighborhood.

The task of this algorithm is to identify which points have at least neighbors within their ε - radius. These are referred to as core points. A point is reachable from a core point if it lies within the ϵ-radius. Points that cannot be connected to any core point due to low density are marked as noise.

2.5 Evaluation of Clustering Methods

It is very important that the formed clusters are meaningful and of high quality, which is why we need to evaluate how good our clustering results are. Internal metrics typically evaluate how well-formed the clusters are, quantifying cohesion and separation, which is applicable for clustering stock time series data.

Key Metrics:

1.) **Silhouette Score:** Evaluates the similarity of data points within the same cluster with respect to different clusters. The metric ranges from -1 (bad clustering) to +1 (well-defined clusters).

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

Where, $a(i)$ is the average distance between point i and all other points in the same cluster.

$b(i)$ is the average distance between point i and all other points in the nearest cluster.

2.) **Davies-Bouldin Index (DBI):** It is the average "similarity ratio" of each cluster to the most similar cluster. Smaller is better—lower DBI indicates better clustering.

$$DBI = \frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right)$$

Where, k is the Number of clusters

σ_i is the Average distance between each point in cluster i and the cluster centroid c_i

$d(c_i, c_j)$ is the distance between centroids $c_i \; and \; c_j$

3.) **Calinski-Harabasz Index (Variance Ratio Criterion):** The ratio of the total between-cluster dispersion to the within-cluster dispersion is measured by this metric. Better clustering is indicated by a higher value.

$$CH = \frac{Tr(B_k)}{Tr(W_k)} * \frac{n-k}{k-1}$$

From the above formula, n is the Total number of data points

k = Number of clusters

$Tr(B_k)$ = Trace of the between-cluster scatter matrix

$Tr(W_k)$ = Trace of the within-cluster scatter matrix

These evaluation metrics provide insights into the cohesion within clusters and separation between clusters. A combination of these metrics, such as Silhouette Score, DBI, and CH Index, along with the assessment of intra-cluster cohesion and inter-cluster separation, ensures a comprehensive evaluation of clustering quality, guiding improvements in algorithms or parameter choices.

2.6 Dimensionality Reduction Technique

2.6.1 Principal Component Analysis (PCA): An Overview

PCA is a popular method for dimensionality reduction across the landscape of analysis without losing important pieces of information. Concrete examples include financial time series like stock prices from the FTSE 100 index, which are clearly high-dimensional data (i.e., with 100 stocks over hundreds of days). The challenge of analyzing and clustering such datasets comes from the noise, redundancy, and computational complexity they present.

PCA projects the original dataset into a new set of orthogonal variables known as principal components that correspond to the directions of maximum variance in the data. The first principal component explains the most variance, and subsequent components capture the remaining variance in orthogonal directions.

Advantages of Stock Market Clustering using PCA:

- ➔ **Dimensionality Reduction**: Index data with around 100 dimensions can be reduced to fewer components, simplifying the clustering process and making interpretation easier.
- ➔ **Enhances Visualization**: PCA can visualize the clustering of stocks by reducing them to 2D or even 3D space, showing patterns difficult to see in higher dimensions.
- ➔ **Filters Out Noise**: By focusing on the components that explain the most variance, PCA filters out noise or less important fluctuations in stock prices.
- ➔ **Faster Clustering**: Clustering algorithms like K-Means and Hierarchical Clustering run faster when data is reduced.

Covariance Matrix in PCA

PCA works by analyzing the covariance between different features (in this case, stock prices). The covariance matrix helps to understand how two variables vary together. The principal components are the eigenvectors of this covariance matrix.

$$Cov(X) = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \mu)(X_i - \mu)^T$$

From the above formula, X_i is the i^{th} data point (vector of stock prices for different companies),

μ is the mean of the dataset, and

n is the number of data points.

Once the covariance matrix is calculated, PCA performs **eigenvalue decomposition** on it. The eigenvectors of the covariance matrix are the principal components, and the corresponding eigenvalues represent the amount of variance explained by each component.

Eigenvalue decomposition:

This involves calculating the covariance matrix of a dataset and finding its eigenvalues and eigenvectors. By sorting the eigenvalues in descending order and selecting the top eigenvectors, PCA reduces the dataset's dimensionality, preserving the most significant variance and simplifying analysis while retaining the most critical information.

$$Cov(X)W = \lambda W$$

Where W represents the eigenvectors (principal components),
λ represents the eigenvalues (variance explained by each principal component).

3.0 Methodology

The methodology details how historical stock prices of the FTSE 100 index are analyzed using a systematic approach in identifying patterns in stock price movements using clustering techniques

3.1 Data Collection

The dataset we used for this analysis has historical daily adjusted closing prices of FTSE 100 index companies from January 2021 through December 2022. The yfinance Python package was used to retrieve the data efficiently. Having a broader snapshot of how stocks performed in this time period is important to understanding the trends and patterns.

Reasons for the selection of the time period: This observation period was chosen to examine market conditions during the COVID-19 pandemic, which significantly influenced the economy and global stock markets.

Key factors include:

- ➔ Brexit Transition: Initial trading disruption arising from new trade agreements and regulations.
- ➔ Inflation Surge: The rise in inflation generated fears of higher interest rates and weakened stock valuations.
- ➔ Russia-Ukraine Conflict: Geopolitical tensions resurfaced, causing energy prices to spike and market instability.
- ➔ Central Bank Policies: A shift towards tightening monetary policy influenced investor sentiment and stock prices.
- ➔ Tech Sector Adjustments: Variations in valuations and operations came under scrutiny.
- ➔ Climate Events and ESG Focus: Increased investor interest in sustainable companies with low exposure to climate-related risks affected stock performance across sectors.

3.2 Data Preprocessing

Before analysis, the data was preprocessed as follows:

3.2.1 Handling Missing Values:

Out of 100 tickers, two had not been included in the FTSE 100 index at the start of the given time period.

1.) **Dark.L (Darktrace, a cybersecurity technology company)** introduced to the FTSE 100 index in 04/2021.
2.) **HLN (Haleon, a pharmaceutical company**) added to the index at the end of the observation period in 07/2022.

The second ticker, HLN, has over 90% missing data, making it a significant challenge. In such cases, using fabricated data to fill these gaps can lead to inaccurate representation of market conditions, resulting in misleading conclusions and poor-quality clustering results. Therefore, HLN is excluded from the analysis.

For the remaining tickers, including Dark.L, I used more reliable techniques to handle missing data. Specifically, I employed linear interpolation, which fills missing values by averaging the values before and after gaps, preserving the data structure. Afterward, I applied the BFill (Backfill) method to fill remaining missing values with the last observed value. This combination of techniques ensures a more complete and accurate dataset, improving the clustering analysis quality.

3.2.2 Normalization:

After handling the missing values, the data underwent normalization using the StandardScaler, which standardizes features by removing the mean and adjusting them to have a unit variance. This method modifies the data so that each feature has a mean of zero and a standard deviation of one, ensuring that all features have an equal impact on the clustering process.

Normalization is especially crucial when dealing with financial time series data, like stock prices, as different stocks can exhibit significantly different price ranges. For instance, one stock might fluctuate between £10 and £100, while another could range from £1,000 to £10,000. Without normalization, clustering algorithms might be unduly influenced by features with larger values, resulting in biased outcomes and potentially misleading cluster formations.

By normalizing the data, we minimize the bias introduced by these differing ranges. This allows clustering algorithms, such as K-Means and hierarchical clustering, to function on a more equitable basis, facilitating the identification of patterns and the grouping of similar time series with greater accuracy. Consequently, the clusters generated are more likely to be meaningful, representing genuine relationships in the data rather than mere scale artifacts. In summary, normalization is an essential step in preparing financial time series data for clustering analysis, enhancing the reliability and interpretability of the results.

3.2.3 Pre - Analysis(Descriptive Analysis):

Descriptive analysis is a very important first step that we perform before applying the clustering algorithm, because in this case it provides a statistical exploration of the main characteristics of our set. This flow includes data summarization and visualization to mine patterns or discover trends and outliers that help us learn more about the distribution, central tendencies (like mean and median), variability (like range & standard deviation), etc. Further, descriptive analysis assists in detecting one or more outliers like inconsistencies, missing data points required to be fixed before follow-up analytics. Finally, this high-level view allows analysts to know how to deal with those issues in the context of the underlying structure of the data and throttle them such that the dataset must be cleaned, normalized and noise free. This in turns improves the data quality which is much needed to enhance reliability and clustering algorithms accuracy and effectiveness. Well-prepared data, in turn, gives rise to more meaningful clustering results that can be interpreted or acted on in a way that allows more of the true problem under study to be understood.

Parameters considered in the Descriptive analysis are:

1.) **Mean returns :** This analysis showed a difference in performance across the tickers. I found, labels of 54 tickers have returns above average and for the rest, they are below mean line. This is an important detail that emphasizes a solid separation in the stock performances to be implemented in clustering analysis. Using Return Patterns to Cluster Similar Stocks — Clustering stocks with their return patterns helps in categorizing high and low performers of a same cluster will have similar return magnitudes. For investors, the clusters in this division for general segmentation and categorization of various performance factor trends based on history.

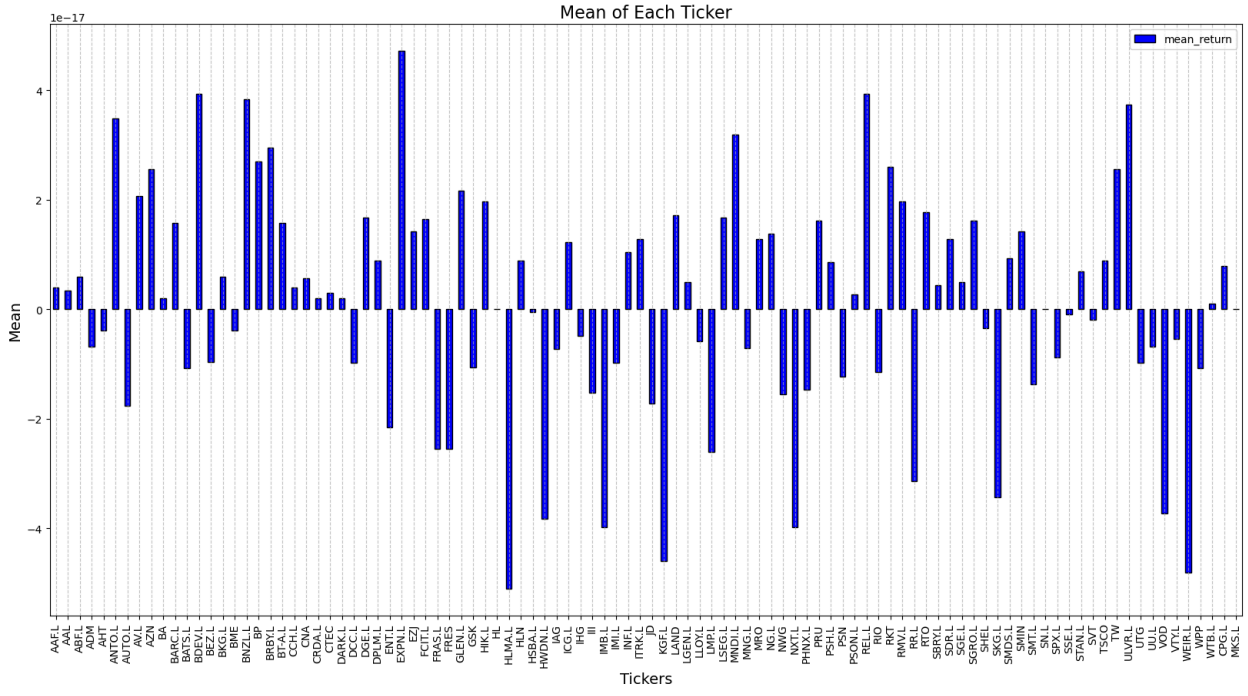


Fig 2: Mean_returns of each stock

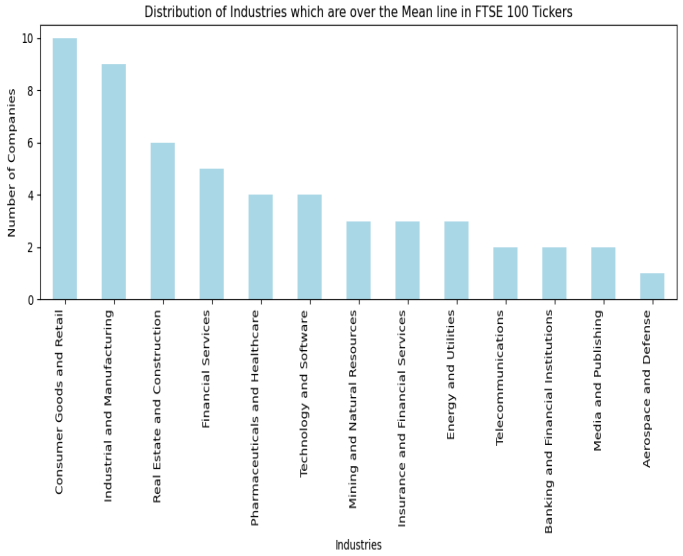


Fig 3: List of industries of tickers over the meanline

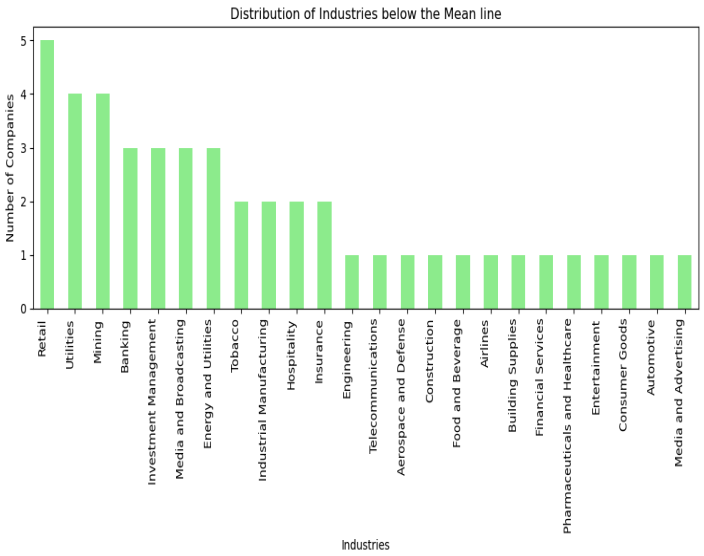


Fig 4: List of industries of tickers below the meanline

The plots indicate that tickers from sectors like consumer goods, industrials and manufacturing, and banking and financials are above the mean, suggesting strong performance and potential profitability for investors. This could also mean these stocks present a favorable risk-reward ratio, attracting more investor interest. Meanwhile the sectors with tickers under that average line indicate some concern for profitability and possibly a riskier view of business or lack of faith in sustainability.

2.) **Standard Deviation(Volatility):** This is an important measure of how returns differ within a group of stocks. It indicates the risk level of each stock and logs, whether the investment is decided or volatile. In general, a high standard deviation provides a greater range of returns with higher risk and reward; whereas a low standard deviation offers less uncertainty (and lower risk). This helps build a compelling investment thesis and an allocation model that can be used for your portfolio management.

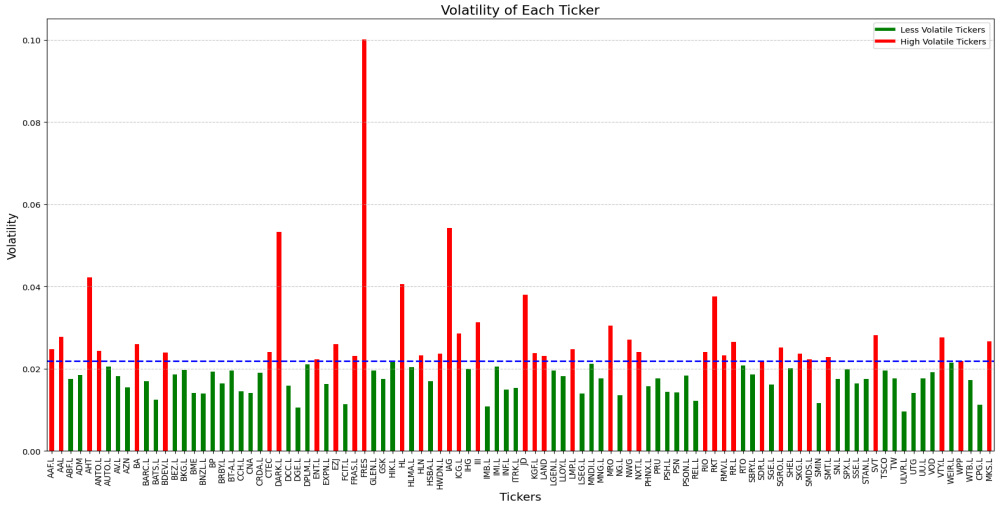


Fig 5: Volatility of the each stock with the threshold

```
Less Volatile Tickers:
['ABF.L', 'ADM', 'AUTO.L', 'AV.L', 'AZN', 'BARC.L', 'BATS.L', 'BEZ.L', 'BKG.L', 'BME', 'BNZL.L', 'BP', 'BRBY.L', 'BT-A.L', 'CC
H.L', 'CNA', 'CRDA.L', 'DCC.L', 'DGE.L', 'DPLM.L', 'EXPN.L', 'FCIT.L', 'GLEN.L', 'GSK', 'HIK.L', 'HLMA.L', 'HSBA.L', 'IHG', 'IM
B.L', 'IMI.L', 'INF.L', 'ITRK.L', 'LGEN.L', 'LLOY.L', 'LSEG.L', 'MNDI.L', 'MNG.L', 'NG.L', 'PHNX.L', 'PRU', 'PSH.L', 'PSN', 'PS
ON.L', 'REL.L', 'RTO', 'SBRY.L', 'SGE.L', 'SHEL', 'SMIN', 'SN.L', 'SPX.L', 'SSE.L', 'STAN.L', 'TSCO', 'TW', 'ULVR.L', 'UTG', 'U
U.L', 'VOD', 'WEIR.L', 'WTB.L', 'CPG.L']
high Volatile Tickers:
['AAF.L', 'AAL', 'AHT', 'ANTO.L', 'BA', 'BDEV.L', 'CTEC', 'DARK.L', 'ENT.L', 'EZJ', 'FRAS.L', 'FRES', 'HL', 'HLN', 'HWDN.L', 'I
AG', 'ICG.L', 'III', 'JD', 'KGF.L', 'LAND', 'LMP.L', 'MRO', 'NWG', 'NXT.L', 'RIO', 'RKT', 'RMV.L', 'RR.L', 'SDR.L', 'SGRO.L',
'SKG.L', 'SMDS.L', 'SMT.L', 'SVT', 'VTY.L', 'WPP', 'MKS.L']
```

From the above Mining, Retail, Construction and Real estate as well financial services are mostly High Volatile tickers. Stable volatile tickers are from industries as Consumer goods, oil & gas, pharmaceuticals & healthcare and Insurance.

3.) **Max drawdown:** It is a prominent risk metric from finance that measures the biggest loss an investment portfolio suffered at least once in time. It measures the largest percentage loss from a peak to a trough preceding a new high (100% amplitude), or vice versa — bear market before a new bull market. It is important for investors to be aware of their portfolio's performance on a risk-adjusted basis — one way of doing that is by looking at the max drawdown, with a lower drawdown being an indicator that risk management and stability are more or less in place.

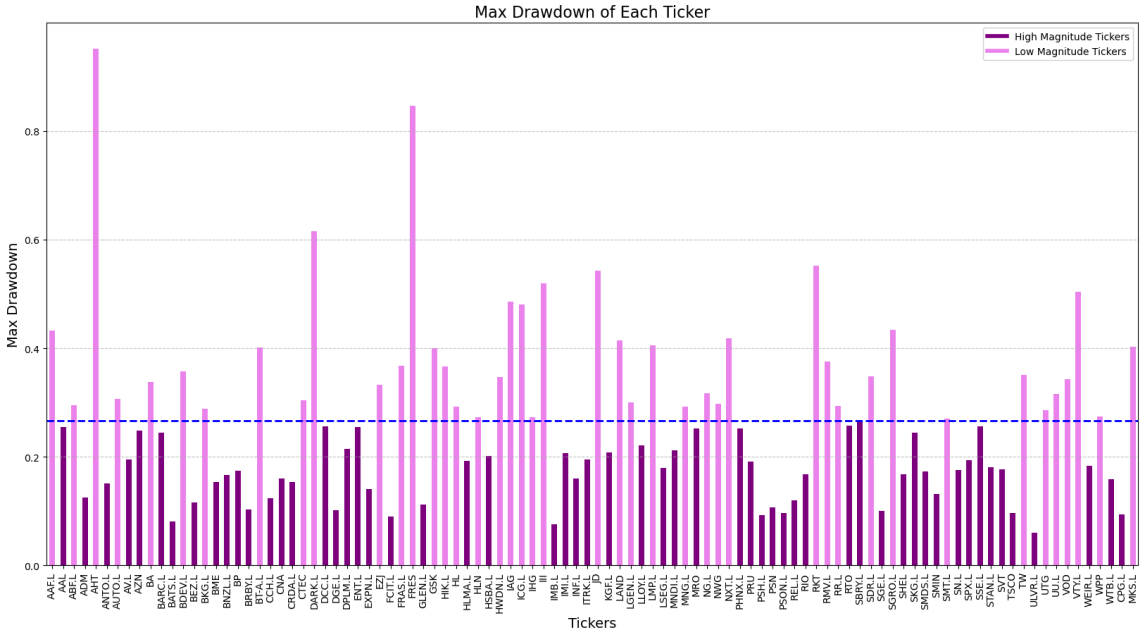


Fig 6: Max Drawdown of each stock with the threshold

```
High Magnitude Drawdown Tickers: ['ABF.L', 'AHT', 'AUTO.L', 'AV.L', 'AZN', 'BA', 'BARC.L', 'BDEV.L', 'BKG.L', 'BME', 'CTEC', 'E
NT.L', 'EZJ', 'FRAS.L', 'GSK', 'HIK.L', 'HLN', 'HSBA.L', 'HWDN.L', 'IAG', 'ICG.L', 'IHG', 'INF.L', 'JD', 'LAND', 'LGEN.L', 'LLO
Y.L', 'LMP.L', 'MNG.L', 'NG.L', 'NWG', 'NXT.L', 'PHNX.L', 'PRU', 'RKT', 'RMV.L', 'RR.L', 'RTO', 'SBRY.L', 'SDR.L', 'SGRO.L', 'S
KG.L', 'SMIN', 'SSE.L', 'TW', 'UTG', 'UU.L', 'VTY.L', 'WPP', 'MKS.L']
Low Magnitude Drawdown Tickers: ['AAF.L', 'AAL', 'ADM', 'ANTO.L', 'BATS.L', 'BEZ.L', 'BNZL.L', 'BP', 'BRBY.L', 'BT-A.L', 'CCH.
L', 'CNA', 'CRDA.L', 'DARK.L', 'DCC.L', 'DGE.L', 'DPLM.L', 'EXPN.L', 'FCIT.L', 'FRES', 'GLEN.L', 'HL', 'HLMA.L', 'III', 'IMB.
L', 'IMI.L', 'ITRK.L', 'KGF.L', 'LSEG.L', 'MNDI.L', 'MRO', 'PSH.L', 'PSN', 'PSON.L', 'REL.L', 'RIO', 'SGE.L', 'SHEL', 'SMDS.L',
'SMT.L', 'SN.L', 'SPX.L', 'STAN.L', 'SVT', 'TSCO', 'ULVR.L', 'VOD', 'WEIR.L', 'WTB.L', 'CPG.L']
```

➔ From the analysis of Max_drawdown, technology and utilities have the greatest amount of high magnitude tickers. Retail, banking, food, and travel are also well-represented. There are fewer tickers in the hotel, airline, and healthcare industries.

➔ And the sectors with the lowest magnitudes are mining, consumer goods, and financials. There are also sizable representations from other businesses, such as media, retail, and electricity.

4.) Skewness: This parameter evaluates a probability distribution's asymmetry. In contrast to a negative skew, which shows the opposite, a positive skew implies that the right side's tail is longer or more substantial than the left. A distribution that is symmetrical is suggested by a zero skewness.

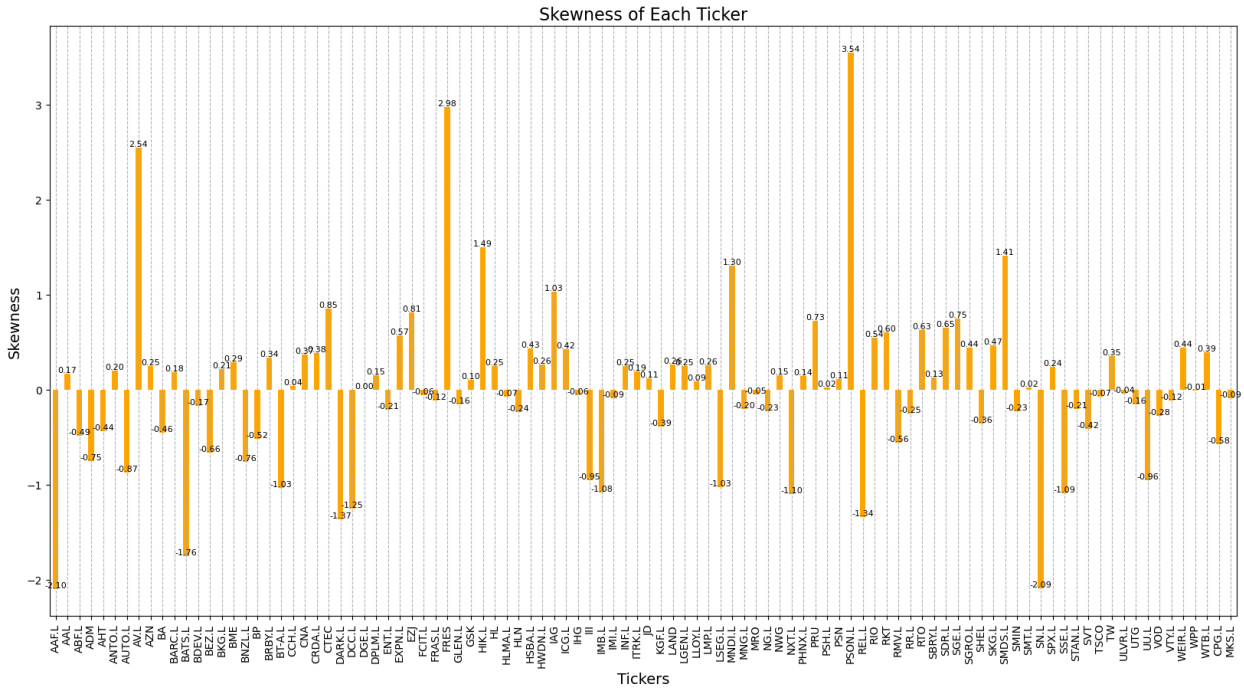


Fig 7: Skewness of all the tickers

```
Positive Skewed Tickers: ['AAL', 'ANTO.L', 'AV.L', 'AZN', 'BARC.L', 'BKG.L', 'BME', 'BRBY.L', 'CCH.L', 'CNA', 'CRDA.L', 'CTEC',
'DGE.L', 'DPLM.L', 'EXPN.L', 'EZJ', 'FRES', 'GSK', 'HIK.L', 'HL', 'HSBA.L', 'HWDN.L', 'IAG', 'ICG.L', 'INF.L', 'ITRK.L', 'JD',
'LAND', 'LGEN.L', 'LLOY.L', 'LMP.L', 'MNDI.L', 'NWG', 'PHNX.L', 'PRU', 'PSH.L', 'PSN', 'PSON.L', 'RIO', 'RKT', 'RTO', 'SBRY.L',
'SDR.L', 'SGE.L', 'SGRO.L', 'SKG.L', 'SMDS.L', 'SMT.L', 'SPX.L', 'TW', 'WEIR.L', 'WTB.L']
Negative Skewed Tickers: ['AAF.L', 'ABF.L', 'ADM', 'AHT', 'AUTO.L', 'BA', 'BATS.L', 'BDEV.L', 'BEZ.L', 'BNZL.L', 'BP', 'BT-A.
L', 'DARK.L', 'DCC.L', 'ENT.L', 'FCIT.L', 'FRAS.L', 'GLEN.L', 'HLMA.L', 'HLN', 'IHG', 'III', 'IMB.L', 'IMI.L', 'KGF.L', 'LSEG.
L', 'MNG.L', 'MRO', 'NG.L', 'NXT.L', 'REL.L', 'RMV.L', 'RR.L', 'SHEL', 'SMIN', 'SN.L', 'SSE.L', 'STAN.L', 'SVT', 'TSCO', 'ULVR.
L', 'UTG', 'UU.L', 'VOD', 'VTY.L', 'WPP', 'CPG.L', 'MKS.L']
```

➔ The skew of the positive returns is pretty mixed by industry, with a nice balance in broad sectors like Finance, Retail, and Healthcare. This means they may be good for a risk-adjusted return profile.

➔ The negative skewed tickers are extensively present in the Food & Beverage, Energy, and Hospitality sectors, indicating potential risk areas or underperformance concerns within these industries.

5.) Kurtosis: The last parameter kurtosis assesses a distribution's "tailedness" to determine whether outliers are present. Whereas low kurtosis (platykurtic) denotes fewer outliers and thinner tails, high kurtosis (leptokurtic) indicates more extreme values and fatter tails. A normal distribution is indicated by a kurtosis of three.

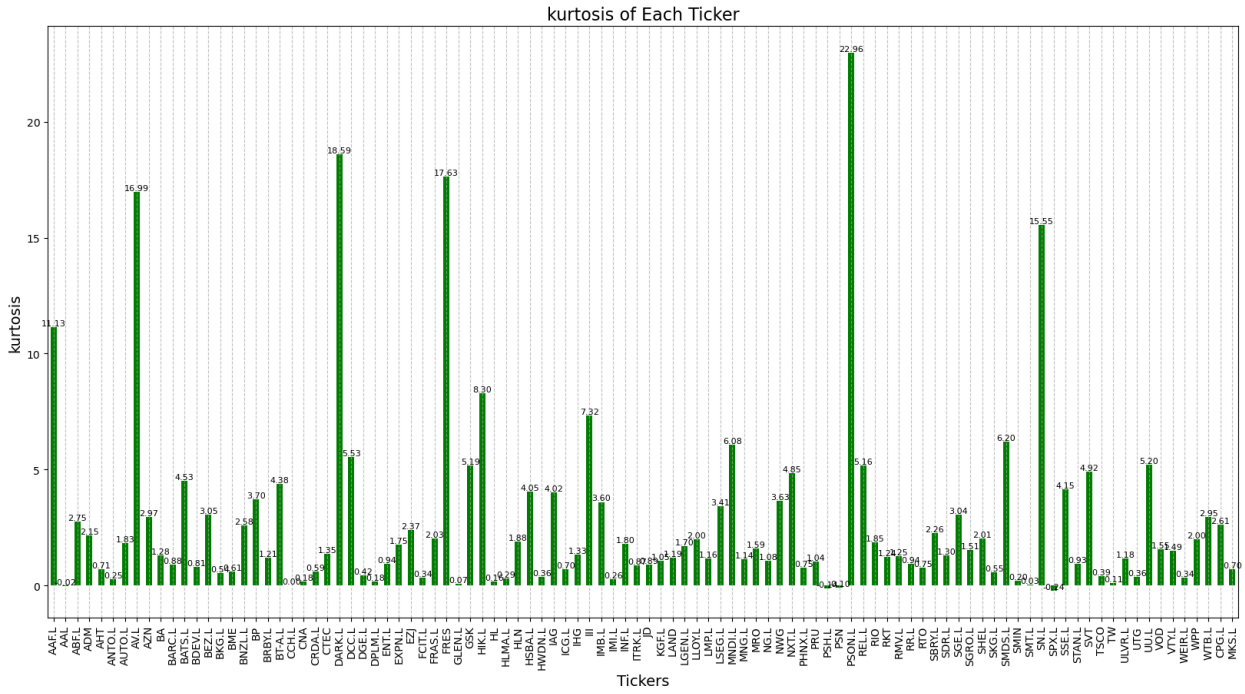


Fig 8: Kurtosis of each ticker

```
High Kurtosis Tickers: ['AAF.L', 'AV.L', 'BATS.L', 'BEZ.L', 'BP', 'BT-A.L', 'DARK.L', 'DCC.L', 'FRES', 'GSK', 'HIK.L', 'HSBA.
L', 'IAG', 'III', 'IMB.L', 'LSEG.L', 'MNDI.L', 'NWG', 'NXT.L', 'PSON.L', 'REL.L', 'SGE.L', 'SMDS.L', 'SN.L', 'SSE.L', 'SVT', 'U
U.L']
Low Kurtosis Tickers: ['AAL', 'ABF.L', 'ADM', 'AHT', 'ANTO.L', 'AUTO.L', 'AZN', 'BA', 'BARC.L', 'BDEV.L', 'BKG.L', 'BME', 'BNZ
L.L', 'BRBY.L', 'CCH.L', 'CNA', 'CRDA.L', 'CTEC', 'DGE.L', 'DPLM.L', 'ENT.L', 'EXPN.L', 'EZJ', 'FCIT.L', 'FRAS.L', 'GLEN.L', 'H
L', 'HLMA.L', 'HLN', 'HWDN.L', 'ICG.L', 'IHG', 'IMI.L', 'INF.L', 'ITRK.L', 'JD', 'KGF.L', 'LAND', 'LGEN.L', 'LLOY.L', 'LMP.L',
'MNG.L', 'MRO', 'NG.L', 'PHNX.L', 'PRU', 'PSH.L', 'PSN', 'RIO', 'RKT', 'RMV.L', 'RR.L', 'RTO', 'SBRY.L', 'SDR.L', 'SGRO.L', 'SH
EL', 'SKG.L', 'SMIN', 'SMT.L', 'SPX.L', 'STAN.L', 'TSCO', 'TW', 'ULVR.L', 'UTG', 'VOD', 'VTY.L', 'WEIR.L', 'WPP', 'WTB.L', 'CP
G.L', 'MKS.L']
```

- ➔ Financial services, mining, utilities, and pharmaceuticals are among the industries with the highest concentration of high Kurtosis tickers. These sectors typically exhibit more extreme values in their returns, which point to higher volatility or atypical price swings.
- ➔ Airlines, food and beverage, retail, and pharmaceuticals are among the industries with the lowest Kurtosis Tickers. A flatter distribution of price changes is the outcome of these sectors' more reliable and stable returns.

Summary from the Descriptive analysis:

The analysis of different industries points toward probable clusters of stocks according to performance metrics, volatility, drawdown risks, mean return, skewness, and kurtosis.

- ➔ Risk-averse investors seeking stable returns might find industries with strong performance and low volatility, like consumer goods, industrials, and financials, forming a protective cluster.
- ➔ Conversely, sectors such as mining, retail, and construction, with their high volatility and significant drawdowns, may attract risk-tolerant investors.
- ➔ Technology and travel stocks, prone to significant price fluctuations, suggest risk factors that might appeal to speculative investors.
- ➔ Negative skewness in food & beverage and hospitality indicates potential underperformance concerns, urging caution.
- ➔ High kurtosis in financial services and mining reflects stocks that may experience sudden price changes, appealing to those pursuing high-risk, high-reward opportunities.

By analyzing these patterns, we can infer future clusters, enabling investors to build portfolios tailored to risk appetite and market scenarios.

3.2.4 Dimensionality Reduction

For clustering and visualization purposes, the dimensionality of data was greatly reduced using Principal Component Analysis (PCA) as an important preprocessing step. PCA is a method for reducing the dimensionality of large datasets while preserving as much important information as possible. The most important patterns are encoded in a new set of uncorrelated variables, created by transforming the initial features into new ones via PCA. PCA was used to transform the original high-dimensional stock price data from many different tickers into a much simpler two-dimensional space. This enabled better visualization of relationships and patterns among the clusters. Ultimately, mapping this high-dimensional data to 2 dimensions allows easier identification and interpretation of behaviors in different stocks.

3.2.5 Clustering Algorithm Application

- ➔ Algorithm Selection: Implemented K-Means, DBSCAN, and Hierarchical Clustering for clustering analysis.
- ➔ Several Iterations: The model ran through several iterations with different initialization and parameters to enhance robustness.
- ➔ Distance Metrics: Checked if the clusters were good using different distance metrics like Euclidean, Manhattan, and Cosine distances.

3.2.6 Cluster Validation

- ➔ Validation Metrics: Used Silhouette Score, Davies-Bouldin Index, and Calinski-Harabasz Index to validate the clusters.
- ➔ Performance Comparison: Compared validation results among algorithms to determine the best-performing clustering method.

3.2.7 Tools and Software

The analysis was conducted using Python, employing libraries such as Pandas for data manipulation, NumPy for numerical computations, Scikit-learn for implementing clustering algorithms, and Matplotlib, plotly and Seaborn for data visualization. Jupyter Notebook was used as the development environment to facilitate interactive analysis.

This study assumes that stock price movements follow patterns that can be effectively captured through clustering. Limitations include potential biases in the dataset, such as market anomalies or external economic factors influencing stock prices, which may affect the clustering outcomes.

4.0 Analysis and Results

4.1 K-Means Clustering

The K-Means clustering algorithm was applied to the preprocessed dataset to identify distinct clusters among the stock price tickers in the FTSE 100 index. The optimal number of clusters, k, was determined using the Elbow Method, which indicated that $k = 3$ was suitable for capturing the underlying structure of the data.

Before applying K-Means, Principal Component Analysis (PCA) was utilized for dimensionality reduction. PCA effectively transformed the high-dimensional stock price data into two principal components, explaining 98.66% of the variance with the first component and an additional 0.62% with the second component. This dimensionality reduction enabled a clearer visualization of the clusters.

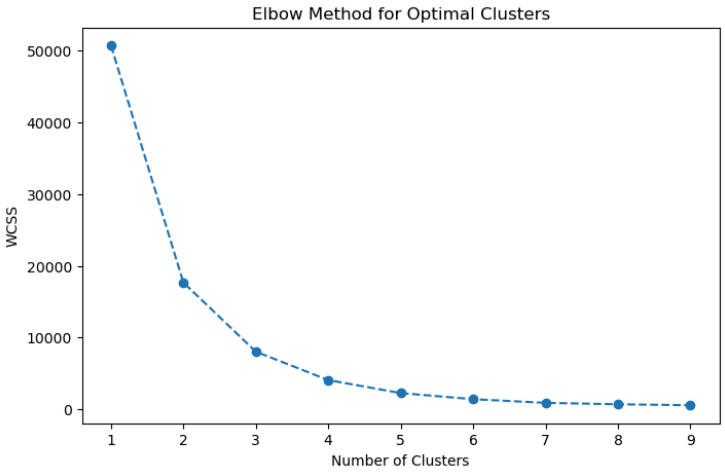


Fig 9 : Elbow method plot for Optimal number of clusters

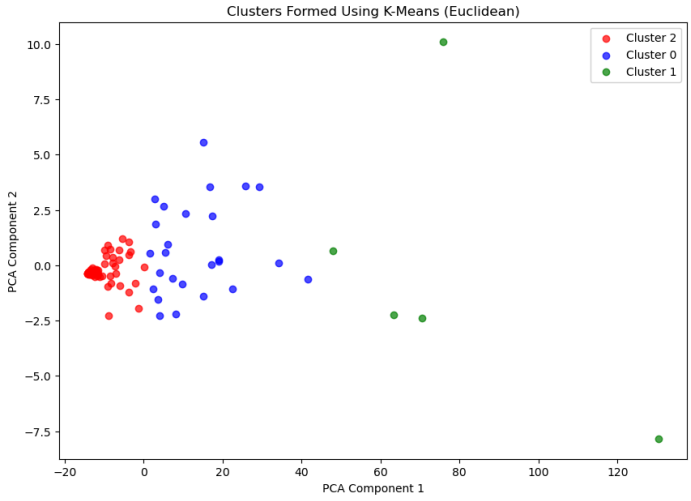


Fig 10 : Clusters formed using the KMeans algorithm with Euclidean Distance

The above plots shows 3 clusters with the tickers in each cluster as follows:

➔ Tickers in Cluster 0(blue): ['ABF.L', 'ANTO.L', 'BATS.L', 'BKG.L', 'BNZL.L', 'BRBY.L', 'CCH.L', 'DGE.L', 'DPLM.L', 'ENT.L', 'EXPN.L', 'HIK.L', 'HLMA.L', 'ICG.L', 'IMB.L', 'IMI.L', 'ITRK.L', 'MNDI.L', 'PSH.L', 'REL.L', 'SKG.L', 'SSE.L', 'ULVR.L', 'WEIR.L', 'WTB.L', 'CPG.L']

The stocks in cluster 0 exhibit a blend of stability and moderate volatility, primarily stemming from their association with industries such as consumer goods, pharmaceuticals, and retail. This clustering suggests that these tickers tend to have similar performance patterns, characterized by lower volatility and a positive skewness in returns, making them attractive for risk-averse investors.

However, the presence of high kurtosis among some tickers indicates potential for extreme price movements, which investors should consider when assessing risk. The mixed skewness also implies a balance of opportunities for growth alongside some inherent risks. Overall, this unified cluster provides a nuanced perspective on how these stocks behave collectively, highlighting their potential for stable returns while also signaling caution regarding their susceptibility to market fluctuations.

➔ Tickers in Cluster 1(green): ['CRDA.L', 'DCC.L', 'LSEG.L', 'NXT.L', 'SPX.L']

The stocks in cluster 1 is characterized by companies that tend to be very stable, with sound, perpetual or fecund profitability streams and driven by sectors such as utilities, financials, and industrials. The volatility in these stocks tends to be lower, which often attracts conservative investors looking for enough in return. Positive skewness implies that we see likely returns, and the relatively low kurtosis means these price changes are generally more predictable over time. Taken together, the lineup has a good investment footprint — slow but steady growth and portfolio resilience in different market conditions make these funds ideal for those seeking stability and an outperforming profile.

➔ Tickers in Cluster 2(red): ['AAF.L', 'AAL', 'ADM', 'AHT', 'AUTO.L', 'AV.L', 'AZN', 'BA', 'BARC.L', 'BDEV.L', 'BEZ.L', 'BME', 'BP', 'BT-A.L', 'CNA', 'CTEC', 'DARK.L', 'EZJ', 'FCIT.L', 'FRAS.L', 'FRES', 'GLEN.L', 'GSK', 'HL', 'HSBA.L', 'HWDN.L', 'IAG', 'IHG', 'III', 'INF.L', 'JD', 'KGF.L', 'LAND', 'LGEN.L', 'LLOY.L', 'LMP.L', 'MNG.L', 'MRO', 'NG.L', 'NWG', 'PHNX.L', 'PRU', 'PSN', 'PSON.L', 'RIO', 'RKT', 'RMV.L', 'RR.L', 'RTO', 'SBRY.L', 'SDR.L', 'SGE.L', 'SGRO.L',

'SHEL', 'SMDS.L', 'SMIN', 'SMT.L', 'SN.L', 'STAN.L', 'SVT', 'TSCO', 'TW', 'UTG', 'UU.L', 'VOD', 'VTY.L', 'WPP', 'MKS.L']

The tickers in cluster 2 appear to be influenced by several market factors, including the ongoing effects of the COVID-19 pandemic, supply chain issues, and changes in consumer habits, which likely led to their similar price movements. Many of these stocks are from sectors heavily impacted by the economy, like consumer goods, energy, and financial services, where price fluctuations were significant. During this time, investor feelings were shaped by changes in interest rates, inflation worries, and government aid, causing their stock prices to move together. Additionally, important financial events, like earnings reports and company actions, might have led to synchronized changes in their prices. Finally, the use of similar trading signals among these stocks likely attracted automated trading strategies, reinforcing their price correlations. Altogether, these factors caused these stocks to behave similarly, leading to their clustering during the chosen time frame.

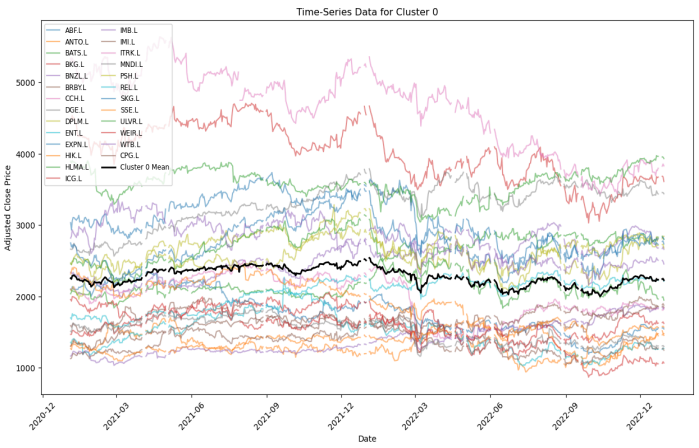


Fig 11: Time series of tickers in Cluster 0

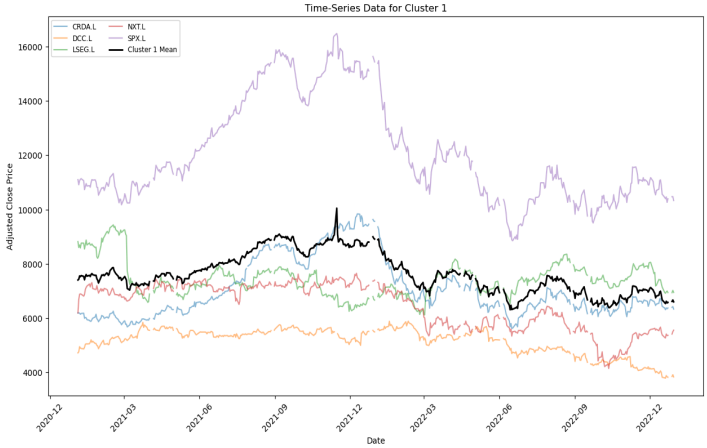


Fig 12: Time series of tickers in Cluster 1

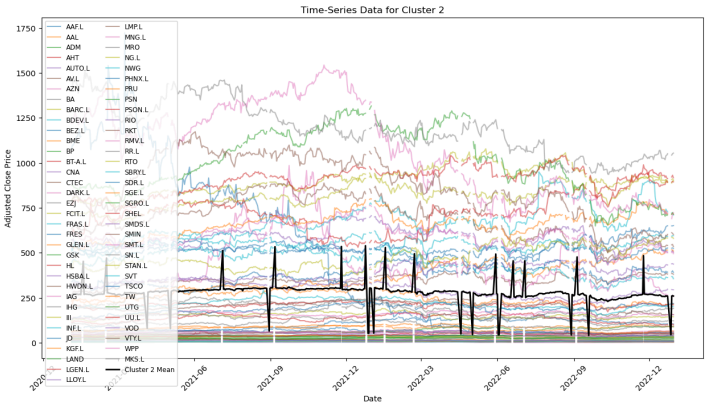


Fig 13: Time series of tickers in ticker 3

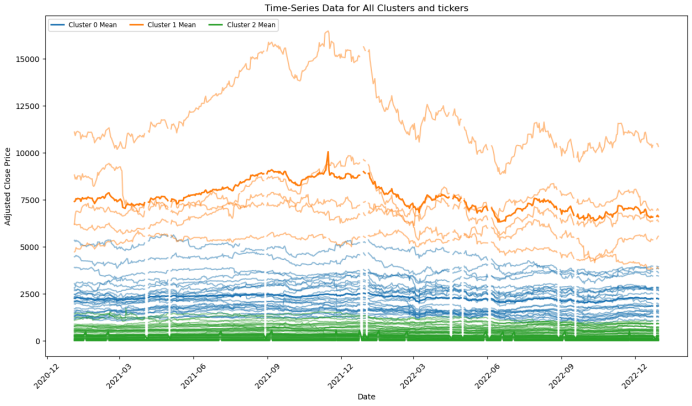


Fig 14: Times series of all tickers along with the 3 cluster means

In the plots above, the time series data for the tickers is displayed along with their cluster mean, and we can see that the curves for the tickers and the mean follow a similar pattern.

The table presents validation metrics and distance measures used to evaluate clustering performance. The results indicate that the Euclidean distance outperforms the other measures, achieving a Silhouette Score of 0.6897, a Calinski-Harabasz Index of 255.0721, and a Davies-Bouldin Index of 0.5043. In contrast, the Manhattan distance shows lower scores across these metrics, while Cosine Similarity has the least favorable results. Based on these findings, the algorithm was tested using different distance measures, with Euclidean distance demonstrating superior clustering performance, which is the basis for the analysis conducted above.

Validation Metric/Distance Measure	**Euclidean**	**Manhattan**	**Cosine Similarity**
Silhouette Score	0.6897	0.5733	0.2416
Calinski-Harabasz Index	255.0721	97.9691	46.4700
Davies-Bouldin Index	0.5043	0.6906	0.74398

Table 1: Scores of the validation metrics for different distances in K-means clustering

In conclusion, the KMeans clustering analysis demonstrates a clear ability to group similar tickers effectively based on the defined parameters.The results show, in general using the distance metric of Euclidean, KMeans reveal great validation results which represent strong intra-cluster cohesion and inter-cluster separation. This implies that the stocks are behaving similarly to each other within a cluster, giving valuable insights to investors who wish to know about shapes or trends in stock performance.

4.2 Hierarchical Clustering

Hierarchical clustering provides a unique way to understand the relationships between stocks by creating a dendrogram that shows how tickers are grouped based on their similarities. Unlike KMeans, which divides stocks into a predetermined number of clusters, hierarchical clustering offers a more flexible analysis, allowing us to see how stocks are linked at different levels of similarity.

From our analyses, we expect that hierarchical clustering will shed more light on the connections among tickers. This method will help us explore subtle differences in performance patterns, especially considering factors like market dynamics, sector-specific influences, and investor sentiment. By studying the resulting dendrogram, we can uncover deeper insights into which stocks exhibit similar behaviors and how they react to external influences such as economic conditions, interest rate changes, and financial announcements.

This approach is especially useful given the variety of sectors in our dataset, including consumer goods, energy, and financial services, which have been significantly affected by recent market shifts. By utilizing hierarchical clustering, we aim to deepen our understanding of these relationships, offering investors valuable insights into groups of stocks that may show similar performance trends in the future.

The algorithm was evaluated using various distance measures and linkages to identify the best combination, as shown in the table below. The analysis indicates that Euclidean distance with Ward linkage offers the strongest clustering performance, demonstrated by high Silhouette and Calinski-Harabasz scores, and a low Davies-Bouldin Index. This combination creates well-defined and compact clusters, making it the most suitable option for grouping FTSE 100 stocks. While Manhattan distance with Complete linkage yields similar results, it slightly falls short in cluster compactness. On the other hand, Cosine distance did not produce meaningful clusters, making it unsuitable for this dataset. Thus, Euclidean distance with Ward linkage is the preferred method for hierarchical clustering.

Distance Metric	**Linkage**	**Silhouette Score**	**Calinski-Harabasz Score**	**Davies Bouldin Index**
Euclidean	Ward	0.6835	451.4424	0.4166
Euclidean	Complete	0.6780	456.9642	0.4283
Euclidean	Average	0.6589	226.2130	0.3462
Euclidean	Single	0.6791	69.0177	0.2313
Manhattan	Complete	0.6835	451.4424	0.4166
Manhattan	Average	0.6589	226.2130	0.3462
Manhattan	Single	0.6791	69.0177	0.2313
Cosine	Complete	0.3061	34.1374	1.0176
Cosine	Average	0.3061	34.1374	1.0176
Cosine	Single	0.2887	28.7653	0.8362

Table 2: Values of the metrics for different distance and linkage methods in Hierarchical Agglomerative clustering

From the analysis above, I created the following dendrograms using the agglomerative method, where each ticker is initially considered its own cluster and then progressively merged with other clusters based on similarity or sector interdependence.

In the plot below, the x-axis illustrates how individual clusters are progressively merged with others until all clusters combine into a single, large cluster. Due to the varying heights at which the clusters merge, some groupings may appear less distinct. However, one of the key benefits of hierarchical clustering is the ability to observe clusters at different levels of granularity by examining the dendrogram at various heights.

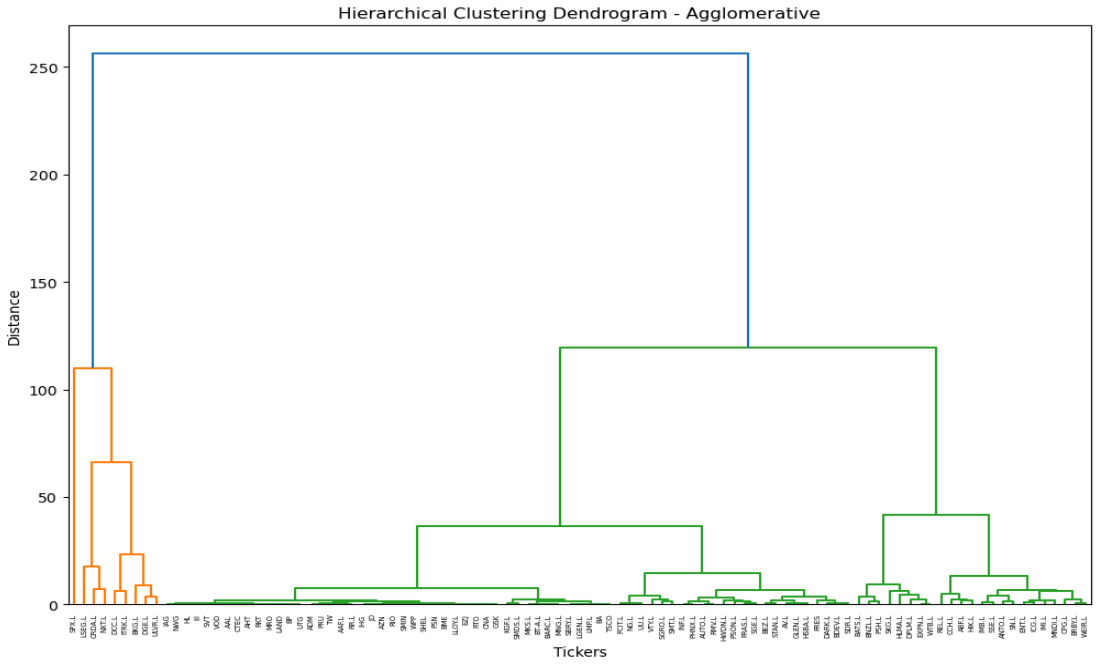


Fig 15: Dendrogram showing 99 tickers using the agglomerative approach

The plot below highlights the number of clusters formed at three different heights—2, 5, and 20. As the height increases, the number of distinct clusters decreases, revealing the broader relationships between the stocks at higher levels while maintaining more granular details at lower heights. This flexibility allows for a deeper exploration of clustering patterns, providing insights at multiple scales.

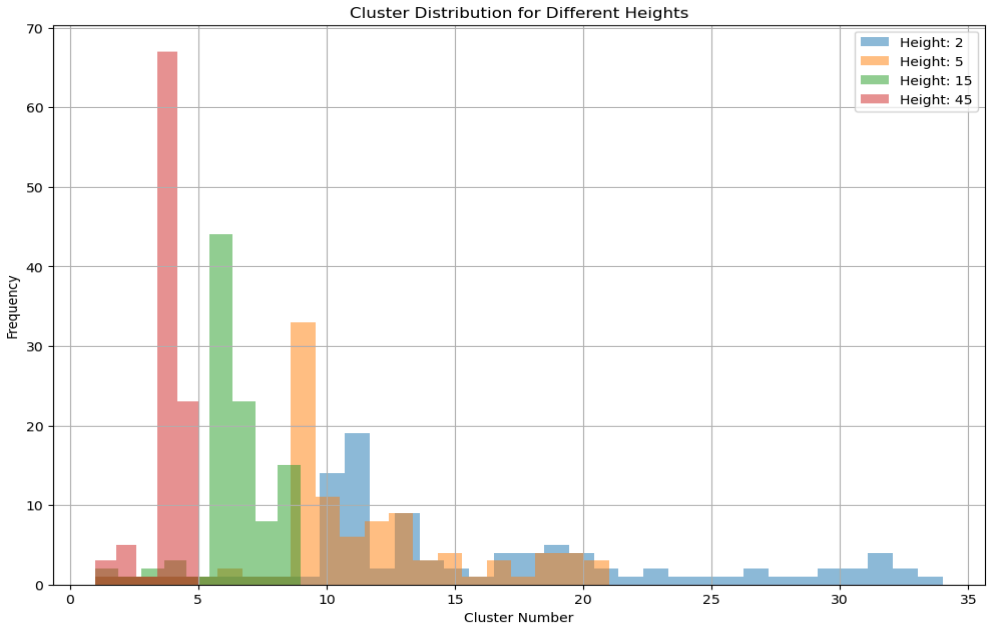


Fig 16: Distribution of clusters at various heights

Height	Number of Clusters
2	34
5	21
15	9
45	5

Table 3: Number of clusters at different height

Figures 17 and 18 provide a more focused and refined view of the clusters previously displayed in Figure 16. These figures offer a truncated and targeted perspective, allowing for a clearer understanding of how individual tickers are merged within the clusters. By eliminating overlap and zooming in on specific sections of the dendrogram, we can more easily identify which tickers have been grouped together based on their similarities, offering better insight into the cluster formation process.

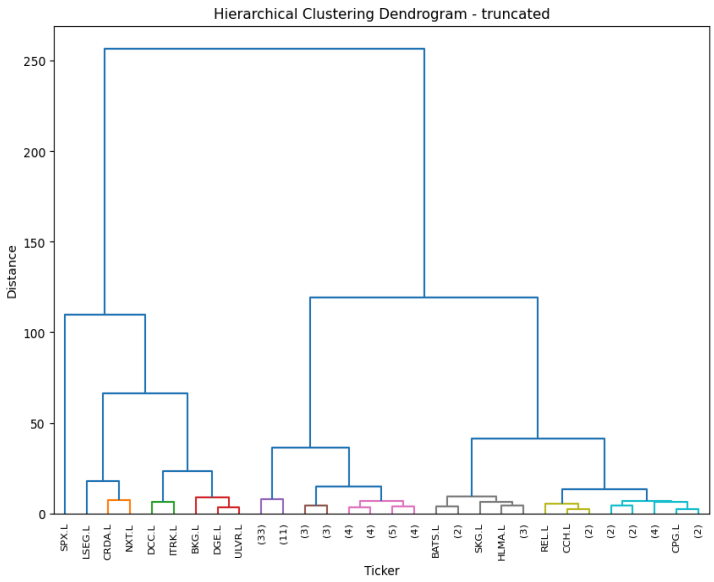


Fig 17: Truncated dendrogram of 99 tickers

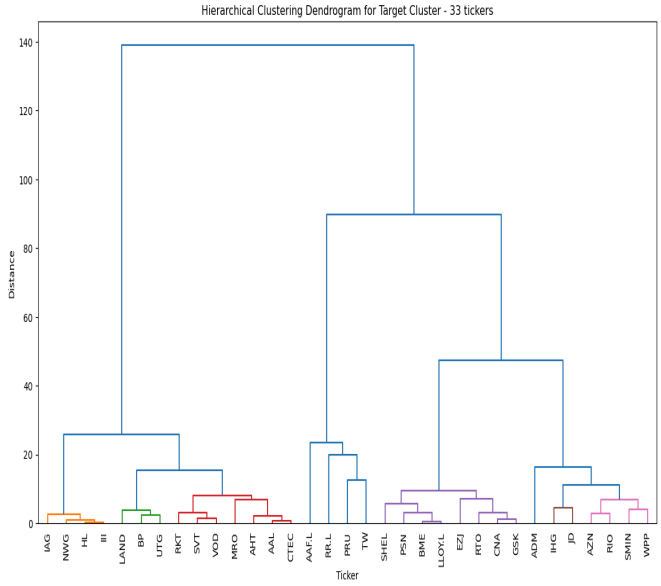


Fig 18: Targeted dendrogram of 33 tickers

For example, let us consider the last grouping of tickers ‘WPP’ and ‘SMIN’ were likely clustered together due to their global exposure, sensitivity to macroeconomic trends, and correlated stock movements influenced by market volatility. Additionally, WPP's advertising services for industrial sectors like SMIN's creates indirect sector interdependencies, further aligning their business cycles.

Example 2: ‘AZN’ and ‘RIO’ have been clustered together not due to direct business connections, but likely because of shared external market influences. Both companies, as large multinationals, are affected by global economic conditions,

such as fluctuations in interest rates, inflation, and geopolitical events. These factors can drive correlated stock price movements across diverse industries, leading to similarities in their time series data. Despite operating in distinct sectors—pharmaceuticals and mining—these broader market dynamics likely contributed to their clustering.

Below is the time series plot displaying the price movements of four tickers. The curves representing WPP and SMIN are shown in two different shades of blue, while RIO and AZN are represented in varying shades of pink. From the plot, we can observe that all four tickers exhibit a similar price pattern over time, which could be a contributing factor to their clustering. This similarity in their movement may indicate that, despite being from different sectors, they are influenced by common market dynamics. Investors can leverage this information to build a diversified portfolio across multiple industries while maintaining a level of cohesion in stock behavior.

Fig 19 : Time series of 4 tickers for the examples above

The clusters at a distance of 45 forming 5 clusters is presented below along with the PCA visualization

```
Max Distance: 45 - Number of Clusters: 5
Clusters at height 45:
Cluster 1: ['CRDA.L', 'LSEG.L', 'NXT.L']
Cluster 2: ['BKG.L', 'DCC.L', 'DGE.L', 'ITRK.L', 'ULVR.L']
Cluster 3: ['SPX.L']
Cluster 4: ['AAF.L', 'AAL', 'ADM', 'AHT', 'AUTO.L', 'AV.L', 'AZN', 'BA', 'BARC.L', 'BDEV.L', 'BEZ.L', 'BME', 'BP', 'BT-A.L', 'C
NA', 'CTEC', 'DARK.L', 'EZJ', 'FCIT.L', 'FRAS.L', 'FRES', 'GLEN.L', 'GSK', 'HL', 'HSBA.L', 'HWDN.L', 'IAG', 'IHG', 'III', 'INF.
L', 'JD', 'KGF.L', 'LAND', 'LGEN.L', 'LLOY.L', 'LMP.L', 'MNG.L', 'MRO', 'NG.L', 'NWG', 'PHNX.L', 'PRU', 'PSN', 'PSON.L', 'RIO',
'RKT', 'RMV.L', 'RR.L', 'RTO', 'SBRY.L', 'SDR.L', 'SGE.L', 'SGRO.L', 'SHEL', 'SMDS.L', 'SMIN', 'SMT.L', 'STAN.L', 'SVT', 'TSC
O', 'TW', 'UTG', 'UU.L', 'VOD', 'VTY.L', 'WPP', 'MKS.L']
Cluster 5: ['ABF.L', 'ANTO.L', 'BATS.L', 'BNZL.L', 'BRBY.L', 'CCH.L', 'DPLM.L', 'ENT.L', 'EXPN.L', 'HIK.L', 'HLMA.L', 'ICG.L',
'IMB.L', 'IMI.L', 'MNDI.L', 'PSH.L', 'REL.L', 'SKG.L', 'SN.L', 'SSE.L', 'WEIR.L', 'WTB.L', 'CPG.L']
```

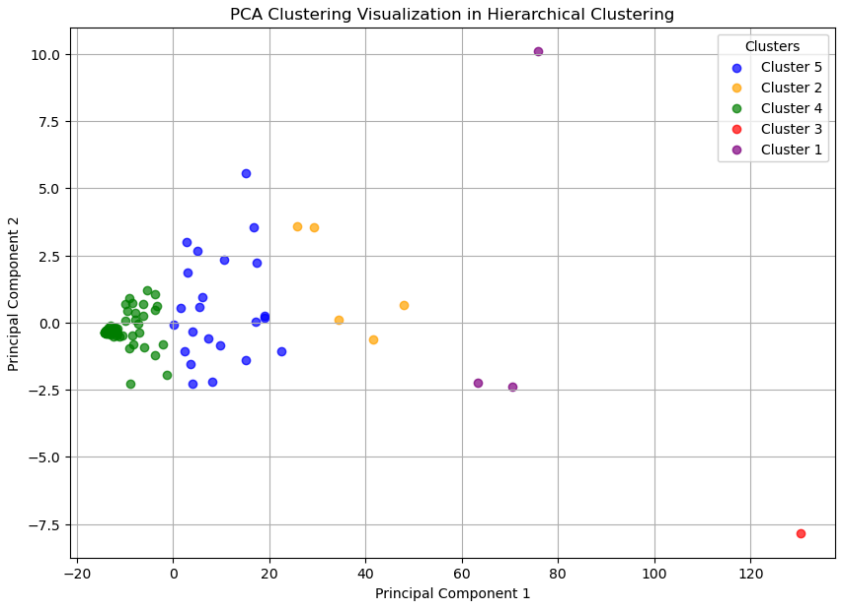


Fig 20: PCA visualization of the Hierarchical clustering

Following reasons can be inferred for the clustering of the tickers at Max Distance 45, forming 5 distinct clusters:

- ➔ Tickers in Cluster 1: ['CRDA.L', 'LSEG.L', 'NXT.L'], These tickers have been clustered together due to their relative stability and strong financial performance during the analysis period. The sectors they belong to, including financial services (LSEG) and retail (NXT), showed lower volatility and consistent growth patterns, making them distinct from more volatile sectors. Their price movements and sectoral resilience suggest why they are grouped together.
- ➔ Tickers in Cluster 2: This cluster includes tickers from sectors like consumer goods (ULVR, DGE) and testing services (ITRK), where relatively stable and defensive stock behavior was observed. These companies have robust fundamentals and steady cash flow, which likely led to less extreme price swings and overall lower volatility, distinguishing them from more cyclical or volatile sectors.
- ➔ Tickers in Cluster 3: SPX.L has been clustered independently, likely due to its unique price behavior or sectoral characteristics, possibly driven by specific factors like market capitalization, sector-specific developments, or atypical volatility patterns that set it apart from other tickers.
- ➔ Tickers in cluster 4: This large cluster represents a wide range of sectors, including pharmaceuticals (AZN), banking (HSBA, LLOY), energy (BP, SHEL), and retail (JD, TSCO). These stocks are linked by their higher volatility and exposure to global macroeconomic conditions such as inflation, interest rate fluctuations, and supply chain disruptions. The diversity within this cluster reflects the shared influence of market-wide factors driving correlated movements across these stocks, despite their sectoral differences.
- ➔ Tickers in Cluster 5: This cluster includes companies from sectors such as consumer goods (ABF.L), manufacturing (IMI.L), and industrials (WEIR.L). The common characteristic here appears to be moderate volatility with consistent price movements, possibly driven by sector resilience or investor confidence in their long-term fundamentals. Their clustering may reflect their shared response to market conditions such as industrial demand or consumption patterns.

The validation metrics calculated for the five clusters yielded the following results:

Validation Metric	**Value**
Silhouette Score	0.6842
Davies-Bouldin Index	0.4135
Calinski-Harabasz Index	454.3982

Table 4: Metrics for the clusters formed using the Hierarchical clustering approach

The Silhouette Score of 0.6842 indicates a good level of separation between the clusters, suggesting that the tickers within each cluster are more similar to each other than to those in other clusters. A score above 0.5 generally reflects well-defined clusters. The Davies-Bouldin Index of 0.4135 further supports this, as lower values indicate better clustering performance, implying that the clusters are compact and well-separated. Finally, the Calinski-Harabasz Index of 454.3982 indicates a high ratio of between-cluster dispersion to within-cluster dispersion, further confirming that the clusters are distinct and meaningful.

In conclusion, the hierarchical clustering algorithm has effectively grouped the FTSE 100 tickers into five distinct clusters based on their similarities, as evidenced by the favorable validation metrics. This approach allows investors to identify patterns and relationships among stocks, enabling more informed decision-making and portfolio diversification. The combination of the validation metrics highlights the robustness of the clustering results, making hierarchical clustering a valuable tool for analyzing stock market data.

4.3 DBSCAN(Density-Based Spatial Clustering of Applications with Noise)

Moving on to the last algorithm of our analysis, the DBSCAN. This algorithm is used to form clusters based on the local density of data points in a space. Unlike classic clustering methods like K-Means or hierarchical clustering that need to pre-assign the number of clusters or assume a spherical cluster shape, DBSCAN is capable of finding clusters with arbitrary shapes and identifying noise in the dataset. In the context of our analysis of the FTSE 100 tickers, DBSCAN provides a non-overlapping view on clustering focusing on the local data point density. This property enables the algorithm to pertain to significantly populated stock sets, further helps in segregating core points (points which are inside clusters) from border points (those on the periphery of a cluster), and noise points(which do not belong to any cluster). This capability is particularly useful in financial markets where outliers and noise are prevalent, providing a more versatile approach to understanding stock behaviors.

We will investigate how DBSCAN enhances existing clustering techniques by providing insights into the underlying structure of the data when we apply it to our dataset of FTSE 100 companies. We hope to improve our comprehension of the connections between the tickers by utilising the density-based technique of DBSCAN, especially in light of the market and economic factors that affect stock price fluctuations. In the end, this research will help investors looking to optimise their portfolios by providing a more thorough understanding of the clustering results.

In this analysis, we have tested the algorithm with multiple eps and minimum sample values

eps	min_samples	clusters	noise	silhouette_score
0.5	2	4	53	0.124758
0.5	3	2	57	0.166564
0.5	5	2	57	0.166564
0.5	7	1	66	0.148835
1	2	4	41	0.240371
1	3	2	45	0.267314
1	5	2	50	0.229905
1	7	1	55	0.331852
4	2	2	13	0.5207
4	3	2	13	0.5207
4	5	1	19	0.688027
4	7	2	21	0.48971
5	2	2	10	0.580573
5	3	2	10	0.580573
5	5	2	10	0.580573
5	7	2	10	0.580573
5.5	2	2	9	0.581026
5.5	3	2	9	0.581026
5.5	5	2	9	0.581026
5.5	7	2	10	0.580573

Table 5: Different iterations of eps and min samples to determine the best values

The table above presents the results of testing various combinations of the epsilon (eps) and min_samples parameters for the DBSCAN clustering algorithm. Among the different parameter sets evaluated, the combination of eps = 5.5 and min_samples = 2 emerged as the most effective. With these parameters, the algorithm identified 2 clusters while classifying 9 data points as noise. The corresponding silhouette score of 0.581026 indicates a moderate level of separation between the clusters, suggesting a reasonably effective clustering outcome.

Below are the list of Clusters along with the tickers and the noise formed for the selected parameters:

```
Tickers in each cluster for the best parameters:
Cluster 0: AAF.L, AAL, ABF.L, ADM, AHT, ANTO.L, AUTO.L, AV.L, AZN, BA, BARC.L, BDEV.L, BEZ.L, BME, B
P, BRBY.L, BT-A.L, CCH.L, CNA, CTEC, DARK.L, ENT.L, EZJ, FCIT.L, FRAS.L, FRES, GLEN.L, GSK, HIK.L, H
L, HSBA.L, HWDN.L, IAG, ICG.L, IHG, III, IMB.L, IMI.L, INF.L, JD, KGF.L, LAND, LGEN.L, LLOY.L, LMP.L,
MNDI.L, MNG.L, MRO, NG.L, NWG, PHNX.L, PRU, PSN, PSON.L, REL.L, RIO, RKT, RMV.L, RR.L, RTO, SBRY.L, S
DR.L, SGE.L, SGRO.L, SHEL, SMDS.L, SMIN, SMT.L, SN.L, SSE.L, STAN.L, SVT, TSCO, TW, UTG, UU.L, VOD, V
TY.L, WEIR.L, WPP, CPG.L, MKS.L
Cluster 1: BATS.L, BNZL.L, DPLM.L, EXPN.L, HLMA.L, PSH.L, SKG.L, WTB.L
Cluster -1: BKG.L, CRDA.L, DCC.L, DGE.L, ITRK.L, LSEG.L, NXT.L, SPX.L, ULVR.L
```

- ➔ Cluster 0 includes a diverse set of companies such as AAF.L, AZN, BP, and WPP. These tickers are likely clustered together due to their similar price movements and market behaviors, which could stem from shared economic factors, sector performance, or investor sentiment. Many of these companies belong to sectors like consumer goods, energy, and pharmaceuticals, which tend to react similarly to broader market changes, thus leading to correlated price movements.

- ➔ Cluster 1 consists of tickers such as BATS.L and EXPN.L. The clustering of these stocks may indicate that they share similar characteristics or market dynamics, perhaps related to specific industry trends or financial performance, distinguishing them from the larger group in Cluster 0.
- ➔ The tickers in Cluster -1 (noise), including BKG.L, CRDA.L, and DCC.L, did not fit into the defined clusters. This could be attributed to their distinct price behaviors, lack of correlation with other stocks, or insufficient data points to establish a meaningful cluster. Being classified as noise suggests that these companies either operate in different market conditions or have price patterns that diverge from the majority, highlighting their unique positions in the market.

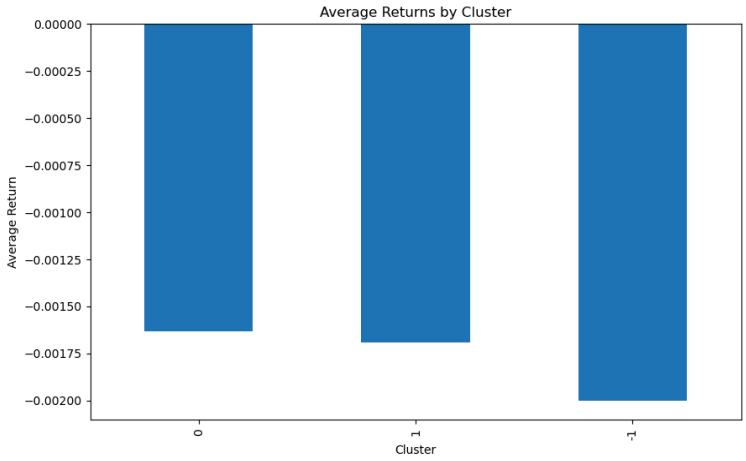


Fig 21: Average returns by each cluster and the noise

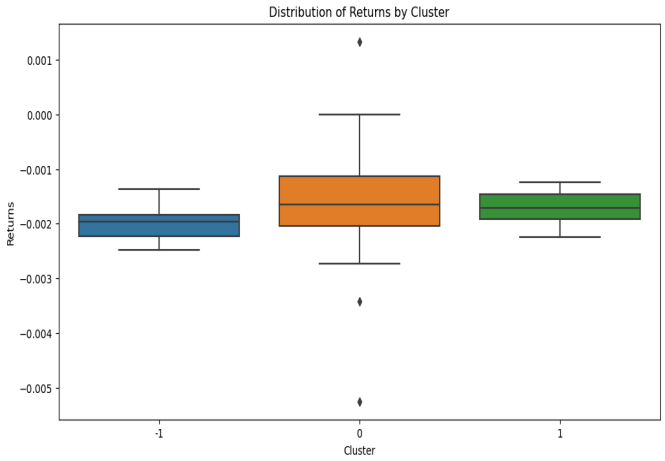


Fig 22: Distribution of returns by each cluster

The average returns by cluster reveal that all clusters experienced negative returns during the observed period. Cluster 0 had an average return of -0.001635, slightly outperforming Cluster 1, which had an average return of -0.001694. The stocks classified as noise in Cluster -1 performed the worst, with an average return of -0.002002. This suggests that while all groups faced declines, the stocks in Cluster 0 fared marginally better, highlighting performance variations among the clustered stocks. Overall, these negative returns reflect broader market challenges affecting all tickers.

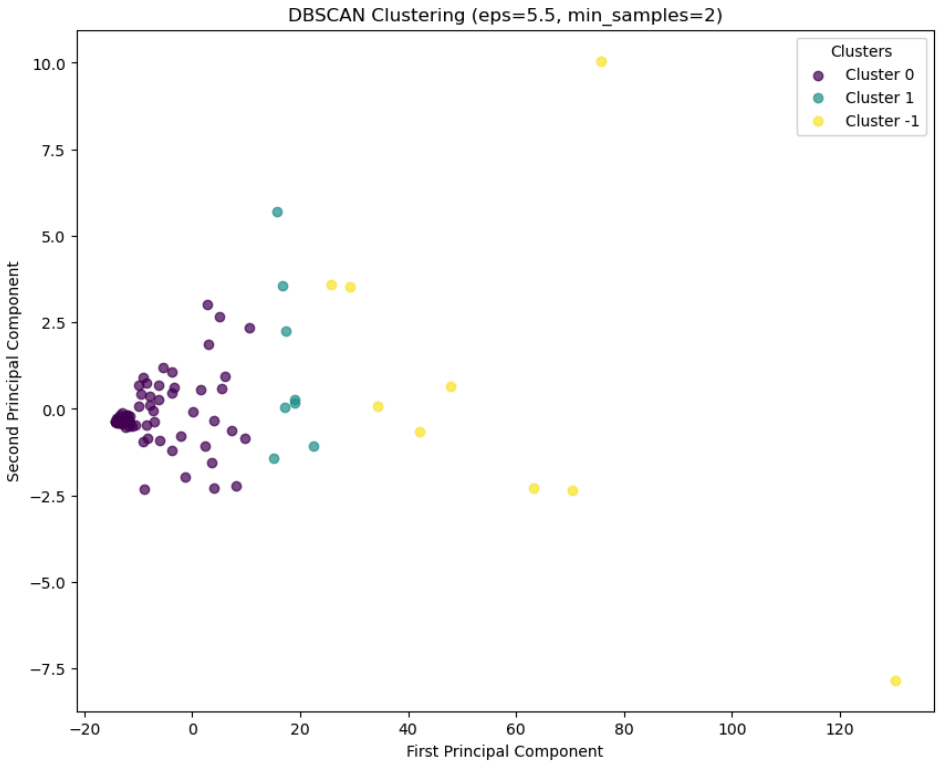


Fig 23: PCA visualization of clusters formed by DBSCAN algorithm

The above visualization demonstrates the results of applying PCA to represent the two distinct clusters, along with the noise points identified by the DBSCAN algorithm. In this plot, points that are situated in less densely populated areas are categorized as noise, while the more densely concentrated regions have been divided into two separate clusters. This separation indicates that the algorithm effectively distinguishes between areas of high and low data density, allowing for a clearer understanding of the underlying structure within the dataset.

Validation Metric	**Values**
silhouette_score	0.5810
davies_bouldin	0.6443
calinski_harabasz	16.4660

Table 6: Validation metrics for the clusters in DBSCAN

The validation metrics obtained from the DBSCAN clustering algorithm give a useful insight about the different values of performance we can ever get on clustering our dataset using this algorithm. The Silhouette Score (0.5810) shows very good separation but not ideal, it means the clusters are well apart from each other and there is still some overlapping areas that can make them unique. Davies-Bouldin Index: With a value of 0.6443 it depicts the good distinctness between the clusters; higher value corresponds to better separation. The Calinski-Harabasz Index, indicating 16.4660, shows a good trade-off between within-cluster compactness and between-cluster separation, which implies that the clusters are

more well-defined to each other. In summary, the DBSCAN algorithm demonstrates a moderate clustering performance with reasonable separation and cohesion among the identified clusters.

5.0 Validation of Clusters

For our study, we have used different clustering algorithms and analyzed their results with three specific validation metrics: the Silhouette Score, the Davies-Bouldin Index, and the Calinski-Harabasz Index. These metrics are really useful since each takes very different kinds of information on board, and we can learn from all three about distinguishing groups well (how your clustering algorithm separates similar points) vs. how much errors particles sharing clusters would cause, etc. By examining these metrics on the output of each clustering algorithm, it will help us rank them in terms of how well and reliable they are, which assists us in selecting the most fitted algorithm needed for our analysis.

Validation metric/ Algorithm	K- Means	Hierarchical Agglomerative	DBSCAN
Silhouette Score	0.6897	0.6842	0.5810
Davies-Bouldin Index	0.5043	0.4135	0.6443
Calinski-Harabasz Index	255.0721	454.3982	16.4660

Table 7 : Comparison of metrics for the three clustering algorithms

Using a variety of validation metrics, the table presents an overview of the performance of three clustering algorithms: K-Means, Hierarchical Agglomerative Clustering, and DBSCAN. Aspects of clustering quality including cohesiveness, separation, and the general structure of clusters are evaluated differently by each metric. In comparison to DBSCAN, the results show that K-Means and Hierarchical Agglomerative Clustering display better-defined clusters that suggest higher cohesion and separation. This analysis helps determine which clustering technique is best in this situation by illuminating how successful each algorithm is. Overall, the analysis indicates that when compared to DBSCAN, K-Means and Hierarchical Agglomerative grouping are more successful in producing meaningful grouping.

6.0 Conclusion:

Ultimately, this project has successfully analyzed the clustering behavior of FTSE 100 tickers using various algorithms. Among the methods examined, Hierarchical Clustering emerges as the most effective for several reasons.

Firstly, the Hierarchical Clustering algorithm provides a deeper insight into the structure of the data through the generation of a dendrogram, which visually represents all clusters. This approach allows for a more detailed understanding of the data, which is not achievable with K-Means due to its fixed pre-defined cluster counts. The varying heights of the clusters in the dendrogram illustrate the similarities among tickers, aiding investors in making informed decisions about portfolio diversification and risk management.

Secondly, validation metrics indicate that Hierarchical Clustering consistently scores high in terms of cluster cohesion and separation. The Silhouette Score reflects this optimization, showing that the clusters are well-defined, meaning the algorithm effectively groups similar tickers while maintaining clear boundaries between different groups. Additionally, the Davies-Bouldin and Calinski-Harabasz indices further validate its superiority in identifying meaningful patterns in the data.

Moreover, the hierarchical nature of this algorithm is particularly suited for financial stocks, as they often exhibit complex interrelationships. The flexibility of Hierarchical Clustering allows it to handle higher-dimensional cases, such as time series data, without the need to impose a strict number of clusters, making it especially applicable for financial analyses.

In conclusion, this project highlights Hierarchical Clustering as the optimal method for clustering FTSE 100 tickers, owing to its adaptability, superior validation metrics, and capability to uncover intricate structures within the data. This eventually provides better opportunities for investors to enhance their portfolio returns.

7.0 Challenges and Future Work:

The large dimensionality of financial time series data made the clustering process more difficult, and we had to deal with concerns related to data quality and the difficulty of choosing the best clustering parameters. Further complicating the development of successful investment strategies was figuring out the economic value of the emerging clusters.

To enhance clustering performance, technical indicators and macroeconomic aspects could be incorporated into expanded feature engineering in future work. It might be possible to get more timely insights by using dynamic clustering approaches that adjust to changes in the market. Furthermore, incorporating risk assessment frameworks and doing comparative research with alternative clustering techniques could enhance our comprehension of ticker groups and the financial implications they entail. Finally, creating a real-time clustering tool would enable investors to decide wisely in response to fresh information. The analysis will be strengthened and improved investment results will be obtained by addressing these issues and going in these areas.

8.0 References

Warren Liao, T. (2005). Clustering of time series data—a survey. Pattern Recognition, [online] 38(11), pp.1857–1874. doi:https://doi.org/10.1016/j.patcog.2005.01.025.

Wikipedia Contributors (2019). FTSE 100 Index. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/FTSE_100_Index.

Jolliffe, I.T. (2013). Principal Component Analysis. Springer Science & Business Media.

Brockwell, P.J. and Davis, R.A. (2013). Introduction to Time Series and Forecasting. Springer Science & Business Media.

Suganthi, R. and Kamalakannan, P. (2015). Analyzing Stock Market Data Using Clustering Algorithm. International Journal of Future Computer and Communication, 4(2), pp.108–111. doi:https://doi.org/10.7763/ijfcc.2015.v4.366.

Journal of Corporate Finance | ScienceDirect.com by Elsevier. [online] Available at: https://www.sciencedirect.com/journal/journal-of-corporate-finance.

Arbelaitz, O., Gurrutxaga, I., Muguerza, J., Pérez, J.M. and Perona, I. (2013). An extensive comparative study of cluster validity indices. Pattern Recognition, [online] 46(1), pp.243–256. doi:https://doi.org/10.1016/j.patcog.2012.07.021.

Vendramin, L., Campello, R.J.G.B. and Hruschka, E.R. (2010). Relative clustering validity criteria: A comparative overview. Statistical Analysis and Data Mining, p.n/a-n/a. doi:https://doi.org/10.1002/sam.10080.

Müllner, D. (2011). Modern hierarchical, agglomerative clustering algorithms. arXiv:1109.2378 [cs, stat]. [online] Available at: https://arxiv.org/abs/1109.2378.

Neil Hardy Spencer (2014). Essentials of multivariate data analysis. Boca Raton: Crc Press/Taylor & Francis Group.

Eryk Lewinson (2020). PYTHON FOR FINANCE COOKBOOK : over 50 recipes for applying modern python libraries to ... quantitative finance to analyze data..

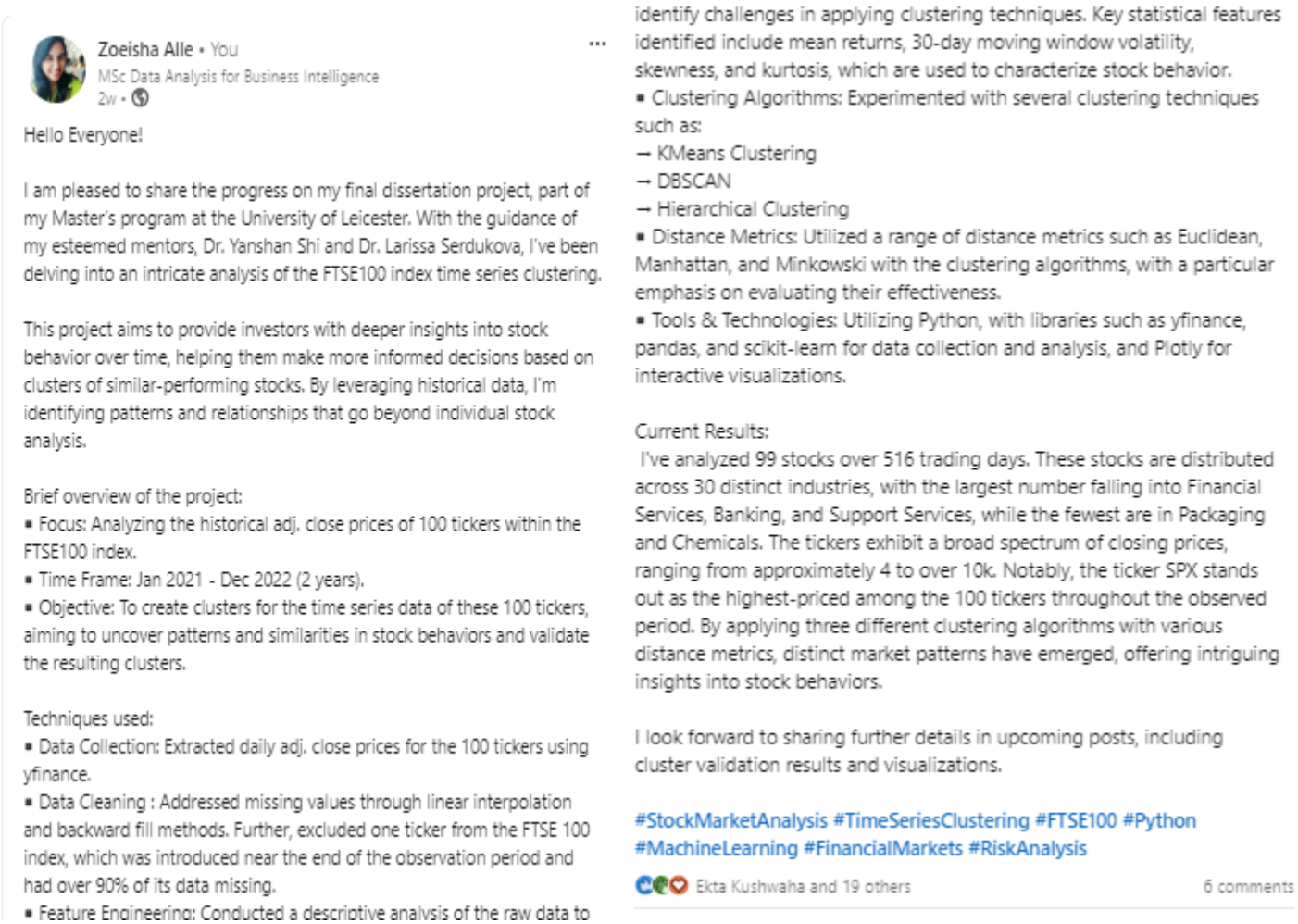
Aggarwal, C.C. and Reddy, C.K. (2018). Data Clustering. CRC Press.

Atsalakis, G.S. and Valavanis, K.P. (2009). Surveying stock market forecasting techniques – Part II: Soft computing methods. Expert Systems with Applications, 36(3), pp.5932–5941. doi:https://doi.org/10.1016/j.eswa.2008.07.006.

Maharaj, E.A. and D'Urso, P. (2011). Fuzzy clustering of time series in the frequency domain. Information Sciences, 181(7), pp.1187–1211. doi:https://doi.org/10.1016/j.ins.2010.11.031.

9.0 LinkedIn Posts

Post 1:



Zoeisha Alle • You
MSc Data Analysis for Business Intelligence
2w •

Hello Everyone!

I am pleased to share the progress on my final dissertation project, part of my Master's program at the University of Leicester. With the guidance of my esteemed mentors, Dr. Yanshan Shi and Dr. Larissa Serdukova, I've been delving into an intricate analysis of the FTSE100 index time series clustering.

This project aims to provide investors with deeper insights into stock behavior over time, helping them make more informed decisions based on clusters of similar-performing stocks. By leveraging historical data, I'm identifying patterns and relationships that go beyond individual stock analysis.

Brief overview of the project:
▪ Focus: Analyzing the historical adj. close prices of 100 tickers within the FTSE100 index.
▪ Time Frame: Jan 2021 - Dec 2022 (2 years).
▪ Objective: To create clusters for the time series data of these 100 tickers, aiming to uncover patterns and similarities in stock behaviors and validate the resulting clusters.

Techniques used:
▪ Data Collection: Extracted daily adj. close prices for the 100 tickers using yfinance.
▪ Data Cleaning : Addressed missing values through linear interpolation and backward fill methods. Further, excluded one ticker from the FTSE 100 index, which was introduced near the end of the observation period and had over 90% of its data missing.
▪ Feature Engineering: Conducted a descriptive analysis of the raw data to identify challenges in applying clustering techniques. Key statistical features identified include mean returns, 30-day moving window volatility, skewness, and kurtosis, which are used to characterize stock behavior.
▪ Clustering Algorithms: Experimented with several clustering techniques such as:
→ KMeans Clustering
→ DBSCAN
→ Hierarchical Clustering
▪ Distance Metrics: Utilized a range of distance metrics such as Euclidean, Manhattan, and Minkowski with the clustering algorithms, with a particular emphasis on evaluating their effectiveness.
▪ Tools & Technologies: Utilizing Python, with libraries such as yfinance, pandas, and scikit-learn for data collection and analysis, and Plotly for interactive visualizations.

Current Results:
I've analyzed 99 stocks over 516 trading days. These stocks are distributed across 30 distinct industries, with the largest number falling into Financial Services, Banking, and Support Services, while the fewest are in Packaging and Chemicals. The tickers exhibit a broad spectrum of closing prices, ranging from approximately 4 to over 10k. Notably, the ticker SPX stands out as the highest-priced among the 100 tickers throughout the observed period. By applying three different clustering algorithms with various distance metrics, distinct market patterns have emerged, offering intriguing insights into stock behaviors.

I look forward to sharing further details in upcoming posts, including cluster validation results and visualizations.

#StockMarketAnalysis #TimeSeriesClustering #FTSE100 #Python #MachineLearning #FinancialMarkets #RiskAnalysis

Ekta Kushwaha and 19 others 6 comments

Post 2:

Zoeisha Alle • You
MSc Data Analysis for Business Intelligence
1w •

Hello Everyone!

Following up on my previous post, I'm pleased to share more progress on my final dissertation project, which focuses on time series clustering of the FTSE100 index. Throughout this process, I've been uncovering significant patterns in stock performance over time.

Refining Stock Behavior Patterns with PCA and Data Scaling:
→ PCA: Before the application of the algorithms to manage the complexity of stock time series data, I employed Principal Component Analysis (PCA) for dimensionality reduction. This approach allowed me to concentrate on the key aspects of stock behavior, simplifying the dataset and reducing computational demands. It also enhanced the effectiveness and efficiency of the clustering models.
→ Data Scaling: Properly scaling the data ensured that the clustering models detected patterns based on relative movements rather than the raw stock price values.

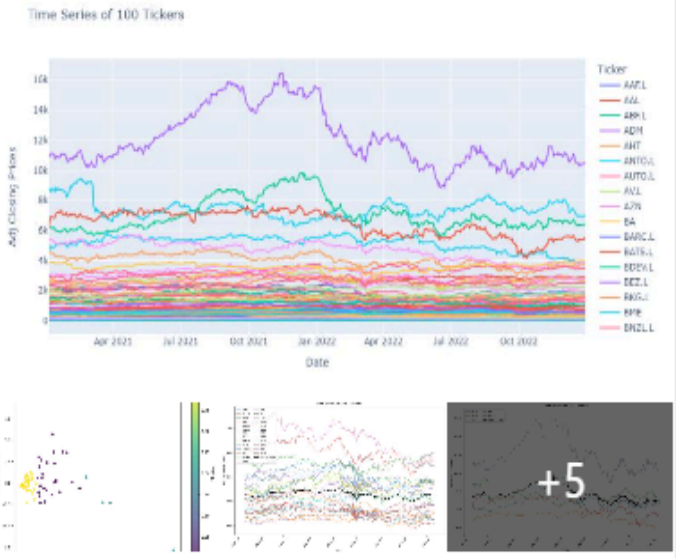
Clustering Algorithms & Key Insights:
→ KMeans Clustering: This algorithm grouped stocks based on similar volatility patterns, identifying tickers with comparable price rises and drops. To determine the optimal number of clusters, I applied the elbow method, which helped identify the point where adding more clusters yields diminishing returns. This technique revealed strong sector-based groupings, particularly within Consumer Staples and Discretion and Finance.

→ DBSCAN: This algorithm effectively identified outliers, highlighting stocks that deviated from the majority. In my analysis of the 99 tickers, I discovered only two main clusters, with several noise points. Many of these noise points belong to the Technology, Finance and Consumer sectors, which are known for their volatility. Their classification as noise suggests these stocks exhibit behaviors that are distinct from the broader trends in the market, indicating they may require a different analytical approach for investors looking to understand their unique dynamics.

→ Hierarchical Clustering: Among the three algorithms, Hierarchical Clustering provided the most detailed insights. It highlighted the nested relationships between stocks, offering a clear hierarchical view of how stocks within sectors, such as Mining, Advertising and Hospitality were related. By adjusting the dendrogram height, I was able to view clusters at various levels of granularity, from broader sector groupings at higher heights to more specific stock groupings at lower heights. This flexibility in visualizing clusters made it easier to capture industry-specific patterns that were less visible with other algorithms.

I'm excited to share some visualizations of the results from the algorithm analysis. In future posts, I'll explore the validation of these clusters and dive deeper into the interpretation of the findings.

#StockMarketAnalysis #TimeSeriesClustering #FTSE100 #PCA #MachineLearning #DataScience #FinancialMarkets



Udayeni Kumarkalva and 18 others 2 comments

Post 3:

Zoeisha Alle • You
MSc Data Analysis for Business Intelligence
1d •

Hello Everyone!

As I reach the final stage of my dissertation project, I'm excited to share the results of my cluster validation analysis. Through this project, I've gained valuable insights into time series clustering and stock behavior patterns.

I've utilized three key metrics to validate the quality of the clusters
→ Silhouette Score: Evaluated how well stocks fit within their clusters and how distinct they are from others.
→ Davies-Bouldin Index: Focused on the separation between clusters by measuring their average similarity.
→ Calinski-Harabasz Index: Examined the balance between intra-cluster cohesion and inter-cluster dispersion, ensuring well-formed clusters.

Among the clustering methods applied, Hierarchical Clustering emerged as the top performer across all validation metrics. It consistently produced higher Silhouette Scores, lower Davies-Bouldin values, and demonstrated superior cluster dispersion based on the Calinski-Harabasz Index.

Hierarchical Clustering also provided greater flexibility compared to KMeans, allowing me to explore clusters at various levels of granularity. By adjusting the dendrogram height, I was able to reveal both broad sector groupings and more specific stock relationships that other algorithms, such as KMeans and DBSCAN, couldn't capture as clearly. This algorithm also formed well-defined boundaries between clusters, offering a deeper understanding of stock behavior patterns and their market implications.

Throughout this project, I've developed a strong understanding of time series clustering and its application in financial markets. Working with large datasets, exploring different clustering techniques, and validating results have been incredibly enriching experiences. These skills will undoubtedly help me in future endeavors in data analytics.

Finally, I'd like to express my gratitude to the University of Leicester and my mentors, Dr. Yanshan Shi and Dr. Larissa Serdukova, for their unwavering support and guidance throughout this project. Their expertise and encouragement have been instrumental in my success.

Thank you for following along with my journey!

#DataScience #StockMarketAnalysis #TimeSeriesClustering #FTSE100

10.0 Appendix

#Code for K-Means

```
#importing required libraries
import yfinance as yf
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from sklearn.metrics import silhouette_score, calinski_harabasz_score, davies_bouldin_score
from sklearn.decomposition import PCA
import plotly.graph_objects as go
print("All required libraries imported successfully.")

###Loading the data
tick_98 = ['HSBA.L', 'HLMA.L', 'FRAS.L', 'CCH.L', 'ITRK.L', 'GLEN.L', 'SN.L', 'DPLM.L',
        'BDEV.L', 'LSEG.L', 'BARC.L', 'SMDS.L', 'NG.L', 'PHNX.L', 'BT-A.L', 'MNDI.L',
        'AUTO.L', 'HWDN.L', 'EXPN.L', 'PSON.L', 'BNZL.L', 'SBRY.L', 'INF.L', 'ULVR.L',
        'BRBY.L', 'SDR.L', 'FCIT.L', 'LLOY.L', 'ENT.L', 'IMI.L', 'ANTO.L', 'LGEN.L',
        'UU.L', 'SKG.L', 'KGF.L', 'ABF.L', 'IMB.L', 'STAN.L', 'WTB.L', 'SMT.L', 'AAF.L', 'REL.L',
        'SPX.L', 'WEIR.L', 'BKG.L', 'BEZ.L', 'CRDA.L', 'SSE.L', 'DGE.L', 'MNG.L', 'LMP.L', 'BATS.L',
        'SGRO.L', 'VTY.L', 'HIK.L', 'RMV.L', 'AV.L', 'DCC.L', 'AAL', 'ADM', 'AHT', 'AZN', 'BA', 'BME',
        'BP', 'CNA', 'CTEC', 'EZJ', 'FRES', 'GSK', 'HL', 'IAG', 'ICG.L', 'IHG', 'III', 'JD',
        'LAND', 'MRO', 'NWG', 'NXT.L', 'PRU', 'PSH.L', 'PSN', 'RIO', 'RKT', 'RR.L', 'RTO', 'SGE.L', 'SHEL',
        'SMIN', 'SVT', 'TSCO', 'TW', 'UTG', 'VOD', 'WPP', 'DARK.L', 'HLN']

# Time Period
start_date = '2021-01-01'
end_date = '2022-12-31'
# Fetch the historical data for the FTSE 100 index
data_98= yf.download(tick_98, start=start_date, end=end_date)['Adj Close']

# 2 tickers
ticker = ['CPG.L','MKS.L']
start_date = "2021-01-01"
end_date = "2022-12-31"
try:
   # Download historical data
   data_2 = yf.download(ticker, start=start_date, end=end_date)['Adj Close']
   if data_2.empty:
      print(f"No data found for {ticker}.")
   else:
      print(data_2.head())
except Exception as e:

data = pd.concat([data_98, data_2], axis=1)
data.head()

# Check for missing values
```

```
missing_values = data.isnull().sum()
# Print columns with missing values
print("Columns with missing values:\n", missing_values[missing_values > 20])

# Create a bar plot for all columns
plt.figure(figsize=(30, 10))
ax = missing_values.plot(kind='bar', color='skyblue')
plt.title('Number of Missing Values per Column')
plt.xlabel('Columns')
plt.ylabel('Number of Missing Values')
plt.xticks(rotation=90)

# Annotate bars with the number of missing values
for i in ax.patches:
    ax.text(i.get_x() + i.get_width() / 2, i.get_height() + 5,  # Adjust the vertical position as needed
            str(int(i.get_height())), ha='center', va='bottom')

plt.show()

#function to extract features from a single ticker's returns data
def extract_features(returns):
    features = {
        'mean_return': returns.mean(),
        'volatility': returns.std(),
        'max_drawdown': (returns.cumsum().cummax() - returns.cumsum()).max(),
        'skewness': returns.skew(),
        'kurtosis': returns.kurtosis(),
    }
    return features

# Calculate returns
returns_data = data.pct_change().dropna()
# Apply the feature extraction to all tickers' returns
features_dict = {ticker: extract_features(returns_data[ticker]) for ticker in returns_data.columns}
# Create DataFrame and set the index to be the ticker names
features_df1 = pd.DataFrame.from_dict(features_dict, orient='index')
# Set the index name to 'Ticker'
features_df1.index.name = 'Ticker'
features_df1.head()

#Skewness
# Define lists for positive and negative skewed tickers
positive_skewed_tickers = features_df1[features_df1['skewness'] > 0].index.tolist()
negative_skewed_tickers = features_df1[features_df1['skewness'] < 0].index.tolist()

# Print the lists
print("Positive Skewed Tickers:", positive_skewed_tickers)
print("Negative Skewed Tickers:", negative_skewed_tickers)

# Plot Skewness
plt.figure(figsize=(20, 10))
ax = features_df1['skewness'].plot(kind='bar', color='orange')
```

```
plt.title('Skewness of Each Ticker', fontsize=16)
plt.xlabel('Tickers', fontsize=14)
plt.ylabel('Skewness', fontsize=14)
plt.xticks(rotation=90, fontsize=10)
plt.grid(axis='x', linestyle='--', alpha=0.9)

# Annotate the bars with skewness values
for p in ax.patches:
   ax.annotate(f'{p.get_height():.2f}',
            (p.get_x() + p.get_width() / 2, p.get_height()),
            ha='center', va='bottom', fontsize=8)
plt.show()

#Kurtosis
# Define a threshold for kurtosis
kurtosis_threshold = 3  # This is a common threshold for high kurtosis

# Get high and low kurtosis tickers
high_kurtosis_tickers = features_df1[features_df1['kurtosis'] > kurtosis_threshold].index.tolist()
low_kurtosis_tickers = features_df1[features_df1['kurtosis'] <= kurtosis_threshold].index.tolist()

# Print the lists
print("High Kurtosis Tickers:", high_kurtosis_tickers)
print("Low Kurtosis Tickers:", low_kurtosis_tickers)

# Plot Kurtosis
plt.figure(figsize=(20, 10))
ax = features_df1['kurtosis'].plot(kind='bar', color='green')
plt.title('Kurtosis of Each Ticker', fontsize=16)
plt.xlabel('Tickers', fontsize=14)
plt.ylabel('Kurtosis', fontsize=14)
plt.xticks(rotation=90, fontsize=10)
plt.grid(axis='x', linestyle='--', alpha=0.7)

# Annotate the bars with kurtosis values
for p in ax.patches:
   ax.annotate(f'{p.get_height():.2f}',
            (p.get_x() + p.get_width() / 2, p.get_height()),
            ha='center', va='bottom', fontsize=8)
plt.show()

#Mean return
# Calculate overall mean return across all tickers
overall_mean_return = features_df1['mean_return'].mean()
overall_mean_return

# Calculate the overall mean return
overall_mean_return = features_df1['mean_return'].mean()

# Determine tickers above and below the overall mean return
above_mean = features_df[features_df1['mean_return'] > overall_mean_return]
below_mean = features_df[features_df1['mean_return'] <= overall_mean_return]
```

```
# Count and list tickers
num_above = len(above_mean)
num_below = len(below_mean)

# Print results
print(f"Number of tickers above the overall mean return: {num_above}")
print(f"Number of tickers below or equal to the overall mean return: {num_below}")

# Plot Mean
plt.figure(figsize=(20, 10))
ax = features_df['mean_return'].plot(kind='bar', color='blue', edgecolor='black')
plt.title('Mean of Each Ticker', fontsize=16)
plt.xlabel('Tickers', fontsize=14)
plt.ylabel('Mean', fontsize=14)
plt.xticks(rotation=90, fontsize=10)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.legend()

tickers_above = above_mean.index.tolist()
tickers_below = below_mean.index.tolist()

print("\nList of tickers above the overall mean return:")
print(tickers_above)

print("\nList of tickers below or equal to the overall mean return:")
print(tickers_below)

#Volatility
import matplotlib.pyplot as plt
import numpy as np
volatility_threshold = features_df1['volatility'].mean()
colors = np.where(features_df1['volatility'] < volatility_threshold, 'green', 'red')
# Plot Volatility
plt.figure(figsize=(20, 10))
# Plot all tickers with assigned colors
ax = features_df1['volatility'].plot(kind='bar', color=colors)
plt.axhline(y=volatility_threshold, color='blue', linestyle='--', linewidth=2, label='Volatility Threshold')
plt.title('Volatility of Each Ticker', fontsize=16)
plt.xlabel('Tickers', fontsize=14)
plt.ylabel('Volatility', fontsize=14)
plt.xticks(rotation=90, fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
handles = [plt.Line2D([0], [0], color='green', lw=4),
        plt.Line2D([0], [0], color='red', lw=4)]
plt.legend(handles, ['Less Volatile Tickers', 'High Volatile Tickers'] + ['Volatility Threshold'])
plt.show()

less_volatile_tickers = features_df1[features_df1['volatility'] < volatility_threshold]
less_volatile_ticker_list = less_volatile_tickers.index.tolist()  # Assuming the index contains the ticker names
print("Less Volatile Tickers:")
print(less_volatile_ticker_list)
```

```
High_volatile_tickers = features_df1[features_df1['volatility'] > volatility_threshold]
high_volatile_ticker_list = High_volatile_tickers.index.tolist()  # Assuming the index contains the ticker names
print("high Volatile Tickers:")
print(high_volatile_ticker_list)

#Max_drawdown
# Define a threshold for high and low magnitude drawdown
drawdown_threshold = features_df1['max_drawdown'].mean()  # Adjust this if needed
high_drawdown_tickers = features_df1[features_df1['max_drawdown'] >= drawdown_threshold].index.tolist()
low_drawdown_tickers = features_df1[features_df1['max_drawdown'] < drawdown_threshold].index.tolist()
print("High Magnitude Drawdown Tickers:", high_drawdown_tickers)
print("Low Magnitude Drawdown Tickers:", low_drawdown_tickers)

colors = np.where(features_df1['max_drawdown'] < drawdown_threshold, 'purple', 'violet')
plt.figure(figsize=(20, 10))
ax = features_df1['max_drawdown'].plot(kind='bar', color=colors)
plt.axhline(y=drawdown_threshold, color='blue', linestyle='--', linewidth=2, label='Drawdown Threshold')
plt.title('Max Drawdown of Each Ticker', fontsize=16)
plt.xlabel('Tickers', fontsize=14)
plt.ylabel('Max Drawdown', fontsize=14)
plt.xticks(rotation=90, fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
handles = [plt.Line2D([0], [0], color='purple', lw=4),
        plt.Line2D([0], [0], color='violet', lw=4)]
plt.legend(handles, ['High Magnitude Tickers', 'Low Magnitude Tickers'] + ['drawdown_threshold'])
plt.show()

data
#Handling Missing values
# Calculate the percentage of NaN values in each column
nan_percentage = data.isna().mean() * 100

#Filter out columns where the percentage of NaN values exceeds 20%
filtered_data = data.loc[:, nan_percentage <= 20]
filtered_data.head()

# Fill missing values using linear interpolation first
df_interpolated = filtered_data.interpolate(method='linear')

# Check for remaining missing values after interpolation
remaining_missing_values_after_interp = df_interpolated.isnull().sum().sum()
print('Remaining missing values after linear interpolation:', remaining_missing_values_after_interp)
# Fill remaining missing values using backward fill
df = df_interpolated.fillna(method='bfill')
# Check for remaining missing values after backward fill
remaining_missing_values_after_bfill = df.isnull().sum().sum()
print('Remaining missing values after backward fill:', remaining_missing_values_after_bfill)

df
df.to_csv('df.csv')
df= df.T
```

```
df
# Initialize the scaler
scaler = StandardScaler()
# Apply the scaler row-wise
scaled_data = scaler.fit_transform(df)
n_components=2
# Apply PCA before clustering
pca = PCA(n_components)
data_reduced = pca.fit_transform(scaled_data)

# Explained variance
explained_variance = pca.explained_variance_ratio_
print(f"Explained variance by {n_components} components: {explained_variance}")
# optimal number of clusters using the Elbow Method
wcss = []
for i in range(1, 10):
   kmeans = KMeans(n_clusters=i, random_state=42)
   kmeans.fit(data_reduced)
   wcss.append(kmeans.inertia_)

# Plot the Elbow Method
plt.figure(figsize=(8, 5))
plt.plot(range(1, 10), wcss, marker='o', linestyle='--')
plt.title('Elbow Method for Optimal Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()

# From the plot, choose the optimal number of clusters
optimal_clusters = 3
# Apply KMeans Clustering
kmeans = KMeans(n_clusters=optimal_clusters, random_state=42)
cluster_labels = kmeans.fit_predict(data_reduced)
# Add cluster labels back to the original DataFrame
df['Cluster'] = cluster_labels
df['Ticker'] = df.index
# Display the tickers with their corresponding cluster labels
print(df[['Ticker', 'Cluster']])
for cluster_num in range(optimal_clusters):
   print(f"\nTickers in Cluster {cluster_num}:")
   print(df[df['Cluster'] == cluster_num]['Ticker'].tolist())

# Add the 2D data and cluster labels to the DataFrame
df['PCA1'] = data_reduced[:, 0]
df['PCA2'] = data_reduced[:, 1]

plt.figure(figsize=(10, 7))
unique_clusters = df['Cluster'].unique()
colors = ['red', 'blue', 'green', 'orange', 'purple']  # Define colors for each cluster
for i, cluster in enumerate(unique_clusters):
   cluster_data = df[df['Cluster'] == cluster]
   plt.scatter(cluster_data['PCA1'], cluster_data['PCA2'], color=colors[i % len(colors)], label=f'Cluster {cluster}', alpha=0.7)
```

```
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.title('Clusters Formed Using K-Means (Euclidean)')
plt.legend()
plt.show()

from sklearn.metrics import silhouette_score, davies_bouldin_score, calinski_harabasz_score
silhouette = silhouette_score(data_reduced, cluster_labels)
db_index = davies_bouldin_score(data_reduced, cluster_labels)
calinski_index = calinski_harabasz_score(data_reduced, cluster_labels)

print(f'Silhouette Score: {silhouette}')
print(f'Davies-Bouldin Index: {db_index}')
print(f'Calinski-Harabasz Index: {calinski_index}')

import matplotlib.dates as mdates

# Plot time series for each cluster
for cluster_num in range(optimal_clusters):
    cluster_tickers = df[df['Cluster'] == cluster_num]['Ticker']
    cluster_data = data[cluster_tickers]
    plt.figure(figsize=(15, 8))
    for ticker in cluster_tickers:
        plt.plot(cluster_data.index, cluster_data[ticker], label=ticker)

    plt.title(f'Time-Series Data for Cluster {cluster_num}')
    plt.xlabel('Date')
    plt.ylabel('Adjusted Close Price')
    plt.legend(loc='upper left', fontsize='small', ncol=2)
    plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=3))
    plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
    plt.xticks(rotation=45)
    plt.show()

# Plot time series for each cluster along with cluster mean
for cluster_num in range(optimal_clusters):
    # Get the tickers for the current cluster
    cluster_tickers = df[df['Cluster'] == cluster_num]['Ticker']
    cluster_data = data[cluster_tickers]
     plt.figure(figsize=(15, 8))
    for ticker in cluster_tickers:
        plt.plot(cluster_data.index, cluster_data[ticker], label=ticker, alpha=0.5)
    # Calculate and plot the mean time series for the cluster
    cluster_mean = cluster_data.mean(axis=1)
    plt.plot(cluster_data.index, cluster_mean, label=f'Cluster {cluster_num} Mean', color='black', linewidth=2)
    plt.title(f'Time-Series Data for Cluster {cluster_num}')
    plt.xlabel('Date')
    plt.ylabel('Adjusted Close Price')
    plt.legend(loc='upper left', fontsize='small', ncol=2)
    plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=3))
    plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m')
```

```
    plt.xticks(rotation=45)
  plt.show()

# Plot time series for all clusters on the same plot with specific colors
for cluster_num in range(optimal_clusters):
    cluster_tickers = df[df['Cluster'] == cluster_num]['Ticker']
    cluster_data = data[cluster_tickers]
    cluster_color = colors[cluster_num]

    # Plot each ticker's time series with the cluster's color and transparency
    for ticker in cluster_tickers:
        plt.plot(cluster_data.index, cluster_data[ticker], color=cluster_color, alpha=0.5)

    # Calculate and plot the mean time series for the cluster with higher opacity
    cluster_mean = cluster_data.mean(axis=1)
    plt.plot(cluster_data.index, cluster_mean, label=f'Cluster {cluster_num} Mean', color=cluster_color, linewidth=2)

plt.title('Time-Series Data for All Clusters and tickers')
plt.xlabel('Date')
plt.ylabel('Adjusted Close Price')
plt.legend(loc='upper left', fontsize='small', ncol=12)
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=3))
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
plt.xticks(rotation=45)
plt.show()

df

from sklearn_extra.cluster import KMedoids
from scipy.spatial.distance import cdist
df_reduced = df.iloc[:, :-4]
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df_reduced)
pca = PCA(n_components=2)
pca_data = pca.fit_transform(scaled_data)
df_pca = pd.DataFrame(pca_data, columns=['PCA1', 'PCA2'])

# Function to visualize clusters
def visualize_clusters(df_pca, cluster_labels, title):
    plt.figure(figsize=(10, 7))
    sns.scatterplot(x='PCA1', y='PCA2', hue=cluster_labels, palette='viridis', data=df_pca)
    plt.title(title)
    plt.xlabel('PCA Component 1')
    plt.ylabel('PCA Component 2')
    plt.legend(loc='upper right')
    plt.show()

# Validation function
def validate_clusters(data, labels):
    silhouette = silhouette_score(data, labels)
    ch_score = calinski_harabasz_score(data, labels)
    db_score = davies_bouldin_score(data, labels)
```

```
    return silhouette, ch_score, db_score

# KMeans with Euclidean Distance
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans_labels = kmeans.fit_predict(scaled_data)

# Visualization and Validation
visualize_clusters(df_pca, kmeans_labels, 'KMeans (Euclidean)')
euclidean_silhouette, euclidean_ch, euclidean_db = validate_clusters(scaled_data, kmeans_labels)
print(f"Euclidean KMeans - Silhouette: {euclidean_silhouette}, CH: {euclidean_ch}, DB: {euclidean_db}")

# KMedoids with Manhattan (Cityblock) Distance
kmedoids_manhattan = KMedoids(n_clusters=3, metric='manhattan', random_state=42)
manhattan_labels = kmedoids_manhattan.fit_predict(scaled_data)

# Visualization and Validation
visualize_clusters(df_pca, manhattan_labels, 'KMedoids (Manhattan)')
manhattan_silhouette, manhattan_ch, manhattan_db = validate_clusters(scaled_data, manhattan_labels)
print(f"Manhattan KMedoids - Silhouette: {manhattan_silhouette}, CH: {manhattan_ch}, DB: {manhattan_db}")

# KMedoids with Cosine Distance
kmedoids_cosine = KMedoids(n_clusters=3, metric='cosine', random_state=42)
cosine_labels = kmedoids_cosine.fit_predict(scaled_data)

# Visualization and Validation
visualize_clusters(df_pca, cosine_labels, 'KMedoids (Cosine)')
cosine_silhouette, cosine_ch, cosine_db = validate_clusters(scaled_data, cosine_labels)
print(f"Cosine KMedoids - Silhouette: {cosine_silhouette}, CH: {cosine_ch}, DB: {cosine_db}")
```

#Code for Hierarchical Clustering

```
#importing the libraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import AgglomerativeClustering
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score, calinski_harabasz_score, davies_bouldin_score
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.cluster.hierarchy import dendrogram
from tqdm import tqdm
# Load the data
df = pd.read_csv('df.csv', index_col='Date')
df = df.T
tickers = df.index
df_reset = df.reset_index()
print("Data shape:", df.shape)
print("\
First few rows of the data:")
df.head()
```

```
data = df.values
# Define clustering parameters
n_clusters = 5
distance_metrics = ['euclidean', 'cityblock', 'cosine']
linkages = ['ward', 'complete', 'average', 'single']
scalers = {
    'StandardScaler': StandardScaler(),
}

# Define the number of PCA components
pca_components = 2
scaled_and_pca_data = {}
for name, scaler in scalers.items():
    # Scale the data
    if scaler:
        scaled_data = scaler.fit_transform(data)
    else:
        scaled_data = data

    # Apply PCA to reduce dimensions
    pca = PCA(n_components=pca_components)
    pca_data = pca.fit_transform(scaled_data)
    scaled_and_pca_data[name] = pca_data
print("Scaled and PCA data types:")
for name, pca_data in scaled_and_pca_data.items():
    print(f"{name}: {type(pca_data)}, Shape: {pca_data.shape}")

def perform_clustering(data, n_clusters, distance_metric, linkage):
    if linkage == 'ward' and distance_metric != 'euclidean':
        # Skip invalid combinations of Ward with non-Euclidean metrics
        print(f"Skipping: Ward linkage requires Euclidean distance, but {distance_metric} was provided.")
        return None, None, None, pd.Series()

    clustering = AgglomerativeClustering(n_clusters=n_clusters, affinity=distance_metric, linkage=linkage)
    cluster_labels = clustering.fit_predict(data)

    silhouette = silhouette_score(data, cluster_labels)
    calinski = calinski_harabasz_score(data, cluster_labels)
    DB_I = davies_bouldin_score(data, cluster_labels)

    cluster_counts = pd.Series(cluster_labels).value_counts().sort_index()
    return silhouette, calinski, DB_I, cluster_counts  # Ensure this matches the number of expected values

# Perform clustering and calculate scores for each scaled and PCA-reduced dataset
for name, pca_data in scaled_and_pca_data.items():
    print(f"\nResults for {name} with PCA:")
    for distance_metric in distance_metrics:
        for linkage in linkages:
            print(f"\nDistance Metric: {distance_metric}, Linkage: {linkage}")
                silhouette, calinski, DB_I, cluster_counts = perform_clustering(pca_data, n_clusters, distance_metric, linkage)

            if silhouette is not None:
```

```
            print(f"Silhouette Score: {silhouette:.4f}")
            print(f"Calinski-Harabasz Score: {calinski:.4f}")
            print(f"Davies Bouldin Index Score: {DB_I:.4f}")
            print(f"Cluster Counts:\n{cluster_counts}")

# Perform clustering with different parameters on PCA-reduced data
results = []

for distance_metric in distance_metrics:
    for linkage in linkages:
        if distance_metric != 'euclidean' and linkage == 'ward':
            continue
        for scaler_name, pca_data_scaled in scaled_and_pca_data.items():  # Use PCA-transformed data
            print(f"Processing: Distance Metric: {distance_metric}, Linkage: {linkage}, Scaler: {scaler_name}")
            silhouette, calinski, DB_I, cluster_counts = perform_clustering(pca_data_scaled, n_clusters, distance_metric, linkage)  # Ensure four values are returned

            if silhouette is not None:
                results.append({
                    'Distance Metric': distance_metric,
                    'Linkage': linkage,
                    'Scaler': scaler_name,
                    'Silhouette Score': silhouette,
                    'Calinski-Harabasz Score': calinski,
                    'Davies Bouldin Index': DB_I,  # Use the calculated DB_I
                    'Cluster Counts': cluster_counts.to_dict() if cluster_counts is not None else None
                })

# Create a DataFrame with the results
results_df = pd.DataFrame(results)
print("\nClustering Results:")
print(results_df)

best_result = results_df.loc[results_df['Silhouette Score'].idxmax()]
print("\nBest Clustering Result:")
print(best_result)
best_pca_data = scaled_and_pca_data[best_result['Scaler']]
best_clustering = AgglomerativeClustering(
    n_clusters=n_clusters,
    affinity=best_result['Distance Metric'],
    linkage=best_result['Linkage']
)
best_labels = best_clustering.fit_predict(best_pca_data)
df['Cluster'] = best_labels
print("\nFirst few rows of the DataFrame with cluster labels:")
print(df.head())

print("\nNumber of tickers in each cluster:")
print(df['Cluster'].value_counts().sort_index())
import matplotlib.pyplot as plt


import scipy.cluster.hierarchy as sch
```

```
from scipy.cluster.hierarchy import linkage, fcluster

# Step 2: Perform Hierarchical Clustering on PCA-reduced data
Z = linkage(pca_df, method='ward')
labels = pca_df.index
plt.figure(figsize=(12, 8))
sch.dendrogram(Z, labels=labels.tolist(), orientation='top')
plt.title('Hierarchical Clustering Dendrogram - Agglomerative')
plt.xlabel('Tickers')
plt.ylabel('Distance')
plt.show()

# Define multiple height thresholds to cut the dendrogram
height_thresholds = [2, 5, 15, 45]

# Create a dictionary to store clusters for each height threshold
clusters_dict = {}
for max_d in height_thresholds:
    clusters = fcluster(Z, max_d, criterion='distance')
    clusters_dict[max_d] = clusters

plt.figure(figsize=(12, 8))
for max_d, clusters in clusters_dict.items():
    num_clusters = len(set(clusters))
    plt.hist(clusters, bins=num_clusters, alpha=0.5, label=f'Height: {max_d}')
plt.title('Cluster Distribution for Different Heights')
plt.xlabel('Cluster Number')
plt.ylabel('Frequency')
plt.legend(loc='best')
plt.grid(True)
plt.show()

# Number of clusters for each height threshold
for max_d, clusters in clusters_dict.items():
    num_clusters = len(set(clusters))
    print(f'Max Distance: {max_d} - Number of Clusters: {num_clusters}')
    # Show tickers in each cluster
    clusters_df = pd.DataFrame({'Ticker': tickers, 'Cluster': clusters})
    print(f'Clusters at height {max_d}:')
    for cluster_num in sorted(set(clusters)):
        cluster_tickers = clusters_df[clusters_df['Cluster'] == cluster_num]['Ticker']
        print(f'Cluster {cluster_num}: {list(cluster_tickers)}')

#Code for the truncated dendrogram
tickers = df.index.tolist()
distance = 'euclidean'
method = 'ward'
Z = linkage(pca_df, method='ward')
plt.figure(figsize=(10,7))
plt.title('Hierarchical Clustering Dendrogram - truncated')
plt.xlabel('Ticker')
plt.ylabel('Distance')
```

```
# Dendrogram with color threshold to view clusters separately
dendrogram(
   Z,
   labels=tickers,
   leaf_rotation=90.,
   leaf_font_size=8.,
   color_threshold=10,
   truncate_mode='lastp'
)
plt.show()

#Code to view the targeted cluster
tickers = df.index.tolist()  # Extract ticker names for labeling
cluster_labels = fcluster(Z, t=5, criterion='distance')
cluster_df = pd.DataFrame({'Ticker': tickers, 'Cluster': cluster_labels})
target_cluster_size = 40
target_cluster = cluster_df['Cluster'].value_counts().idxmax()

# Filter tickers belonging to the target cluster
tickers_in_cluster = cluster_df[cluster_df['Cluster'] == target_cluster]['Ticker'].tolist()
subset_df = df.loc[tickers_in_cluster]

# Standardize the subset data
subset_scaled_data = scaler.fit_transform(subset_df)
n_components = 2
pca = PCA(n_components=n_components)
subset_pca_data = pca.fit_transform(subset_scaled_data)
explained_variance = pca.explained_variance_ratio_
print(f"Explained variance by {n_components} components: {explained_variance}")
Z_subset_pca = linkage(subset_pca_data, method='ward')
plt.figure(figsize=(15, 8))
plt.title('Hierarchical Clustering Dendrogram for Target Cluster (44  tickers)')
plt.xlabel('Ticker')
plt.ylabel('Distance')
dendrogram(
   Z_subset_pca,
   labels=tickers_in_cluster,
   leaf_rotation=90.,
   leaf_font_size=10.,
   color_threshold=10
)
plt.show()

#PCA Visualization
def plot_pca_clusters(pca_data, clusters):
   plt.figure(figsize=(10, 7))
   cluster_labels_map = {
      0: 'Cluster 5',
      1: 'Cluster 2',
      2: 'Cluster 4',
      3: 'Cluster 3',
```

```
    4: 'Cluster 1'
  }

  # Create a color array based on cluster labels
  colors = ['blue', 'orange', 'green', 'red', 'purple']
  for i, label in enumerate(cluster_labels_map.values()):
    plt.scatter(pca_data[clusters == i, 0], pca_data[clusters == i, 1],
            label=label, color=colors[i], alpha=0.7)
  plt.title('PCA Clustering Visualization in Hierarchical Clustering')
  plt.xlabel('Principal Component 1')
  plt.ylabel('Principal Component 2')
  plt.legend(title='Clusters')
  plt.grid(True)
  plt.show()

# Perform PCA on the scaled data
pca = PCA(n_components=2)
pca_data_2d = pca.fit_transform(scaled_df)
optimal_clusters = 5
clustering = AgglomerativeClustering(n_clusters=optimal_clusters, linkage='ward')
clusters = clustering.fit_predict(pca_data_2d)

# Plot the PCA visualization
plot_pca_clusters(pca_data_2d, clusters)

#Validation of clusters
silhouette = silhouette_score(pca_data_2d, clusters)
db_index = davies_bouldin_score(pca_data_2d, clusters)
calinski_index = calinski_harabasz_score(pca_data_2d, clusters)

print(f'Silhouette Score: {silhouette}')
print(f'Davies-Bouldin Index: {db_index}')
print(f'Calinski-Harabasz Index: {calinski_index}')
```

Code for DBSCAN

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score
import seaborn as sns

# Load the dataset
df = pd.read_csv('df.csv', index_col=0)
# Display basic information about the dataset
print(df.info())
print(df.head())

# Transpose the data
```

```
df_transposed = df.T
df_transposed

# Prepare the data
X = StandardScaler().fit_transform(df_transposed)

# Function to apply DBSCAN and visualize results
def apply_dbscan(eps, min_samples):
    # Apply DBSCAN
    dbscan = DBSCAN(eps=eps, min_samples=min_samples)
    clusters = dbscan.fit_predict(X)
    silhouette_avg = silhouette_score(X, clusters) if len(set(clusters)) > 1 else 0

    # Apply PCA for visualization
    pca = PCA(n_components=2)
    X_pca = pca.fit_transform(X)

    # Count number of clusters and noise points
    n_clusters = len(set(clusters)) - (1 if -1 in clusters else 0)
    n_noise = list(clusters).count(-1)

    # Get tickers for each cluster
    cluster_tickers = {}
    for cluster in set(clusters):
        cluster_tickers[cluster] = df_transposed.index[clusters == cluster].tolist()
        return clusters, silhouette_avg, n_clusters, n_noise, cluster_tickers

# List of eps values to try
eps_values = [0.5, 1.5, 2.5,  5.5]
min_samples_values = [2,3,5,7]
results = []
for eps in eps_values:
    for min_samples in min_samples_values:
        clusters, silhouette_avg, n_clusters, n_noise, cluster_tickers = apply_dbscan(eps, min_samples)
        results.append({
            'eps': eps,
            'min_samples': min_samples,
            'n_clusters': n_clusters,
            'n_noise': n_noise,
            'silhouette_score': silhouette_avg,
            'davies_bouldin': davies_bouldin,
            'calinski_harabasz': calinski_harabasz,
            'cluster_tickers': cluster_tickers
        })

# Create a dataframe of results
results_df = pd.DataFrame(results)
print(results_df[['eps', 'min_samples', 'n_clusters', 'n_noise', 'silhouette_score', 'davies_bouldin', 'calinski_harabasz']]))

# Visualize the effect of eps and min_samples on number of clusters and silhouette score
fig, axes = plt.subplots(2, len(min_samples_values), figsize=(20, 12))
for i, min_samples in enumerate(min_samples_values):
```

```
    subset = results_df[results_df['min_samples'] == min_samples]


    axes[0, i].plot(subset['eps'], subset['n_clusters'], marker='o')
    axes[0, i].set_xlabel('Epsilon')
    axes[0, i].set_ylabel('Number of Clusters')
    axes[0, i].set_title(f'Effect of Epsilon on Number of Clusters (min_samples={min_samples})')

    axes[1, i].plot(subset['eps'], subset['silhouette_score'], marker='o')
    axes[1, i].set_xlabel('Epsilon')
    axes[1, i].set_ylabel('Silhouette Score')
    axes[1, i].set_title(f'Effect of Epsilon on Silhouette Score (min_samples={min_samples})')

plt.tight_layout()
plt.show()

# Select the best parameters based on silhouette score
best_result = results_df.loc[results_df['silhouette_score'].idxmax()]
best_eps = best_result['eps']
best_min_samples = best_result['min_samples']
print(f"\
Best parameters: eps={best_eps}, min_samples={best_min_samples}")

# Apply DBSCAN with the best parameters
best_clusters, _, _, _, _ = apply_dbscan(best_eps, best_min_samples)
df_transposed['Cluster'] = best_clusters
# Calculate average returns for each cluster
df_transposed['Returns'] = df_transposed.pct_change(periods=1, axis=1).mean(axis=1)
cluster_returns = df_transposed.groupby('Cluster')['Returns'].mean().sort_values(ascending=False)
# Create a bar plot of average returns by cluster
plt.figure(figsize=(10, 6))
cluster_returns.plot(kind='bar')
plt.title('Average Returns by Cluster')
plt.xlabel('Cluster')
plt.ylabel('Average Return')
plt.show()
print("\
Average Returns by Cluster:")
print(cluster_returns)

# Create a box plot of returns distribution by cluster
plt.figure(figsize=(12, 6))
sns.boxplot(x='Cluster', y='Returns', data=df_transposed)
plt.title('Distribution of Returns by Cluster')
#plt.savefig('dbscan_cluster_returns_distribution.png')
plt.show()

# Function to print tickers for each cluster
def print_cluster_tickers(eps, min_samples):
    _, _, _, _, cluster_tickers = apply_dbscan(eps, min_samples)
    print(f"\
Tickers in each cluster for eps={eps}, min_samples={min_samples}:")
```

```
    for cluster, tickers in cluster_tickers.items():
        print(f"Cluster {cluster}: {', '.join(tickers)}")

# Print tickers for each cluster for different eps and min_samples values
for eps in eps_values:
    for min_samples in min_samples_values:
        print_cluster_tickers(eps, min_samples)
```