Clustering Financial Data: A Machine Learning Approach to Stock Performance Analysis

Dharmil Yogesh Karia, Darsh Chetan Pandya & Hiba Khan

karia.dh, pandya.dar, khan.hib @northeastern.edu

CS 6140: Machine Learning

Professor Radivojac

December 9, 2024

1. Objectives & Significance

The primary goal of this project was to analyze time series data of stock prices and identify clusters of stocks exhibiting similar movement patterns over a specified period. By leveraging various time series clustering algorithms such as K-Means, LSTM Autoencoders, and Hierarchical Clustering, we sought to group stocks with comparable trends in volatility, moving averages, and exponential moving averages (EMA). This clustering could provide valuable insights for portfolio diversification, risk management, and identifying market trends, which are critical for investors, financial analysts, and algorithmic traders.

The significance of this project lies in its ability to simplify the overwhelming complexity of stock market data into actionable insights. Stocks within the same cluster are likely influenced by similar market dynamics, such as industry trends, geopolitical factors, or macroeconomic indicators. Understanding these clusters can aid in predicting future price movements, managing exposure to specific risks, and crafting strategies to maximize returns. Additionally, the clustering approach provides a scalable method to process vast amounts of data and uncover hidden relationships that traditional methods might overlook.

Our findings revealed distinct groupings of stocks based on their movement characteristics. For example, certain clusters corresponded to high-growth technology stocks with significant volatility, while others included stable utility stocks with minimal fluctuations. The use of autoencoders demonstrated promise in capturing complex, non-linear relationships in the data, while hierarchical clustering provided a clear dendrogram representation of the stock groupings. The results underscored the utility of combining time series data analysis with machine learning techniques for gaining deeper insights into financial markets.

2. Background

2.1. Overview of Key Concepts

Time series analysis is a critical tool for examining data points collected or recorded at specific time intervals. When applied to the financial domain, such as stock prices, it enables analysts to identify patterns, trends, and anomalies, providing actionable insights for decision-making. However, due to the dynamic and often non-linear nature of stock price movements, traditional clustering techniques may not adequately capture underlying relationships.

Dynamic Time Warping (DTW) is a powerful algorithm for measuring similarity between two time series that may vary in speed or duration. Unlike Euclidean distance, which requires equal-length time series and matches data points one-to-one, DTW aligns sequences flexibly, minimizing the distance while accounting for temporal shifts. This makes DTW particularly suited for clustering stock data, where trends may evolve asynchronously across different companies.

Clustering techniques, including **K-Means, Hierarchical Clustering**, and **Autoencoders**, can be combined with DTW to enhance grouping capabilities. While K-Means uses centroids to define clusters, hierarchical clustering offers a tree-like structure (dendrogram), allowing analysts to view relationships at multiple levels. Autoencoders, a type of neural network, are effective in extracting latent features from time series, helping to capture complex patterns that traditional methods might miss.

2.2. Related Work

Time series clustering has been extensively studied, with applications across diverse domains such as finance, healthcare, and weather forecasting. In the context of financial data, clustering can reveal patterns and trends that inform investment strategies and risk management.

In the article "Time Series Clustering: Deriving Trends and Archetypes from Sequential Data" [2], the author highlights various approaches to clustering time series data, emphasizing the

importance of distance metrics like DTW and Euclidean distance. The article discusses how DTW effectively captures non-linear temporal relationships by allowing flexible alignment, making it a superior choice for datasets with variable time scales. It also explores clustering techniques like K-Means and hierarchical clustering for grouping similar patterns, showcasing real-world examples of their application.

Another resource, "Time Series Clustering" [3] , provides a deep dive into the methodology of time series clustering. This work elaborates on the practical challenges, such as choosing appropriate distance metrics and preprocessing techniques. It highlights the advantages of using DTW for clustering dynamic sequences and how integrating DTW with clustering algorithms can uncover hidden structures in time series data. The article also mentions advanced methods like shape-based distance measures, demonstrating their efficacy in detecting specific patterns within financial datasets.

These studies underscore the importance of selecting appropriate distance metrics and clustering algorithms to achieve meaningful results in time series analysis. Our project builds on these ideas by systematically comparing DTW and Euclidean distance in the context of stock market data. It further contributes to the field by incorporating additional financial indicators and employing machine learning techniques, such as autoencoders, to enhance the clustering process.

2.3. Uniqueness of This Work

Our project extends existing methodologies by leveraging **Dynamic Time Warping** in combination with advanced clustering algorithms to analyze stock market time series data. Unlike conventional approaches, our work focuses on aggregating a wide range of financial indicators—such as moving averages, exponential moving averages, and volatility ratios—into the clustering process. This holistic approach provides a multi-dimensional perspective on stock behavior, enabling a richer and more nuanced understanding of market dynamics.

What sets our work apart is the integration of **DTW with feature engineering and machine learning models** like autoencoders, which allow for capturing both linear and non-linear dependencies in the data. Additionally, our methodology is scalable to large datasets and

adaptable for various financial applications, such as sectoral analysis, portfolio optimization, and risk assessment.

By combining DTW's ability to align time series flexibly with the clustering power of machine learning algorithms, our approach opens up new possibilities for uncovering hidden patterns in stock market data. This not only improves the quality of insights but also offers practical applications for financial strategists and quantitative analysts aiming to stay competitive in a rapidly evolving market.

3. Methods

3.1. Data Description and Acquisition

Data Description and Source

The dataset used in this project was obtained from Kaggle [1], containing daily stock price data for 491 publicly traded companies. This dataset includes key financial indicators for each company over multiple years, such as:

- **Open Price**: The price at which a stock starts trading at the beginning of the day.
- **High Price**: The highest price reached during the trading session.
- **Low Price**: The lowest price reached during the trading session.
- **Close Price**: The price at which a stock closes at the end of the trading day.
- **Volume**: The total number of shares traded during the day.

Additionally, the dataset includes metadata such as company names and other contextual financial information. The richness of this data makes it an ideal candidate for time series analysis, as it captures daily fluctuations and long-term trends in stock prices.

3.2. Methodology

3.2.1. Feature Engineering and Preprocessing

Fig 1: Example for open vs close prices for AAPL stock wrt date

The feature engineering process involved creating several technical indicators to describe stock performance. These indicators were then standardized to ensure uniform scale across all features. The feature engineering steps include:

1. Moving Averages (MA): 7-day, 30-day, and 90-day moving averages of the percentage change in closing prices.
2. Exponential Moving Averages (EMA): 7-day, 30-day, 90-day, 1-year, and 3-year EMAs to capture trends in stock prices over different periods.
3. Volatility Ratio: A measure of market volatility computed as the ratio of the stock's highest to lowest price.
4. Rate of Change (ROC): The sum of the percentage change over a 30-day window.
5. Average Directional Index (ADX): A technical indicator used to quantify the strength of a trend.

After feature engineering, the data was scaled using StandardScaler to normalize the features and remove the effects of different scales, ensuring that no single feature disproportionately influenced the clustering algorithms. The data was then split into time-series format for each company.

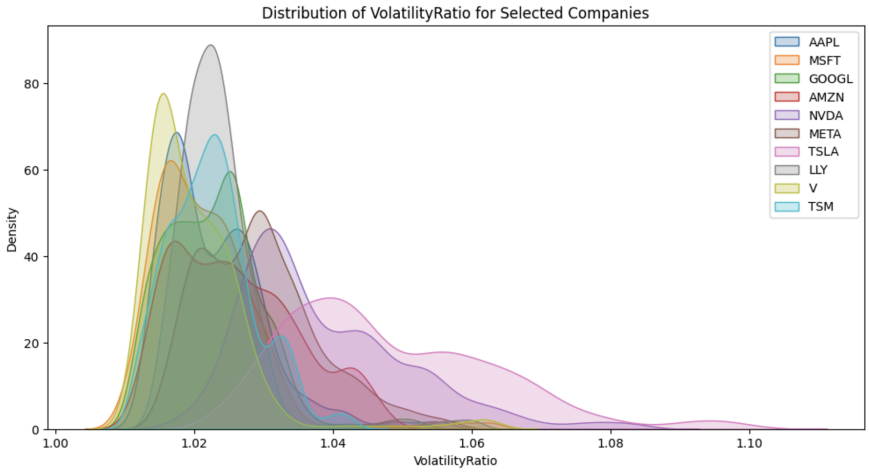

Fig 2: Distribution of Volatility Ratio for Selected Companies

3.2.2. Clustering Methodology

The clustering methods employed in this project are as follows:

1. K-Means Clustering: This algorithm partitions the data into a specified number of clusters based on the features. The goal of K-means is to minimize the variance within each cluster while maximizing the variance between clusters. The algorithm follows these steps:
    - Initialize k centroids randomly.
    - Assign each data point to the nearest centroid.
    - Recompute the centroids as the mean of all data points assigned to each centroid.
    - Repeat the process until convergence, i.e., when the centroids no longer change.

    The optimal number of clusters, k, was determined using the elbow method, which involves plotting the within-cluster sum of squares (WCSS) against the number of clusters and selecting the point where the curve begins to flatten (i.e., the elbow point).

2. K-Means with Dynamic Time Warping (DTW): K-Means clustering is a popular algorithm that partitions data into K clusters by minimizing the within-cluster variance. However, in time series analysis, where the data points are sequences of values over time,

traditional distance metrics like Euclidean distance may not capture the temporal dependencies and varying lengths of time series effectively. To address this limitation, we apply Dynamic Time Warping (DTW) as a distance metric in conjunction with the K-Means algorithm.

The K-Means algorithm with DTW follows the same general procedure as traditional K-Means but with the following key modifications:

1. Distance Metric: Instead of using Euclidean distance, the DTW distance is used to calculate the similarity between time series.
2. Cluster Centers: K-Means traditionally computes the centroid of each cluster as the mean of all data points in that cluster. In the case of DTW, the centroid is typically computed as the medoid—the time series in the cluster that minimizes the total DTW distance to all other points in the cluster. This allows the cluster center to better represent the underlying time series patterns.

Steps of K-Means with DTW

- Initialization: Randomly select KKK time series from the dataset as the initial cluster centers (medoids).
- Assignment Step: For each time series in the dataset, calculate the DTW distance to each of the KKK medoids and assign the time series to the cluster of the closest medoid.
- Update Step: For each cluster, calculate the new medoid by selecting the time series that minimizes the sum of DTW distances to all other time series in the cluster.
- Repeat: Repeat steps 2 and 3 until convergence, i.e., when the assignment of time series to clusters no longer changes.

3. LSTM Autoencoder with K-Means Clustering: This method extracted latent representations of the time series data for clustering. The autoencoder, consisting of an LSTM encoder and decoder, was trained to minimize reconstruction loss, ensuring the latent space captured key temporal patterns. Once trained, the encoder transformed the

data into a lower-dimensional representation. The extracted representations were then clustered using K-Means, as described above.

4. Hierarchical Clustering: This approach builds a tree-like structure (dendrogram) to represent nested clusters. Starting with each point as its own cluster, hierarchical clustering merges the closest clusters at each step until only one cluster remains. We will evaluate agglomerative hierarchical clustering using the ward linkage method, which minimizes the variance within each cluster. The optimal number of clusters will be determined by cutting the dendrogram at a particular level.

3.3. Evaluation

To assess the effectiveness and robustness of the clustering results, we used a combination of quantitative evaluation metrics and qualitative analysis. This strategy ensures that the clusters generated from the time series data are meaningful, interpretable, and aligned with the underlying patterns of stock price movements. Below are the key components of our evaluation strategy.

1. **Internal Evaluation Metrics**

   Internal evaluation metrics measure the quality of the clustering without requiring external validation or ground truth labels. These metrics help in determining the cohesiveness of the clusters and the separation between them. We will use the following key metrics:

   - **Silhouette Score**: The silhouette score is used to evaluate the consistency of the clusters. It ranges from -1 to +1, where a score close to +1 indicates that the samples are well clustered, and a score close to -1 indicates that the samples are poorly clustered (i.e., they may be assigned to the wrong clusters). The silhouette score is calculated for each data point based on its distance to the nearest cluster, providing insight into how well-separated the clusters are.
   - **Within-Cluster Sum of Squares (WCSS)**: The WCSS is a common metric used to evaluate the compactness of the clusters in K-Means clustering. It measures the

sum of squared distances between each data point and the centroid of its assigned cluster. Lower WCSS indicates more compact clusters. We will use this metric to determine the optimal number of clusters, using methods like the Elbow Method.

2. **External Evaluation Metrics**

   Although unsupervised clustering typically lacks ground truth labels, external evaluation metrics can still be used when external information is available. In our case, this could involve comparing the clustering results with known financial characteristics or business insights. For example, if we are clustering stocks based on their price movements, we can evaluate if the clusters correspond to certain sectors or industries.

- **Comparison with Sector/Industry Labels**: If the stock data includes industry or sector labels (e.g., Technology, Healthcare), we can compare the clustering results with these labels to see if stocks in the same sector are grouped together. This can help assess the semantic relevance of the clusters.

3.4. Model Comparison

We will compare the clustering results obtained from different algorithms and distance metrics to determine which provides the most meaningful and interpretable clusters.

- **Euclidean Distance vs. DTW**: We will compare the results of using Euclidean distance and DTW as distance measures. DTW, being better suited for time series data, might yield more meaningful clusters for stocks with non-linear patterns. By comparing the clustering results using these two metrics, we can determine the best approach for clustering stock time series.

4. Results

In this section, we present the results of our clustering experiments. We first discuss the clustering results obtained using K-Means with Dynamic Time Warping (DTW) distance measure. The elbow method is employed to determine the optimal number of clusters. Subsequently, we use the TimeSeriesKMeans algorithm from the tslearn library to refine our clustering results.

4.1. K-Means with DTW Distance

To begin, we used K-Means clustering with DTW distance to partition the time series data. The elbow method was used to determine the optimal number of clusters by calculating the within-cluster sum of squares (WCSS) for various cluster sizes. The elbow method helps identify the point at which increasing the number of clusters no longer significantly improves the clustering performance, i.e., the "elbow" point.

We evaluated several values of K (number of clusters), ranging from 1 to 10, and plotted the WCSS to identify the optimal cluster size.

Cluster Size per Cluster (Elbow Method Plot)

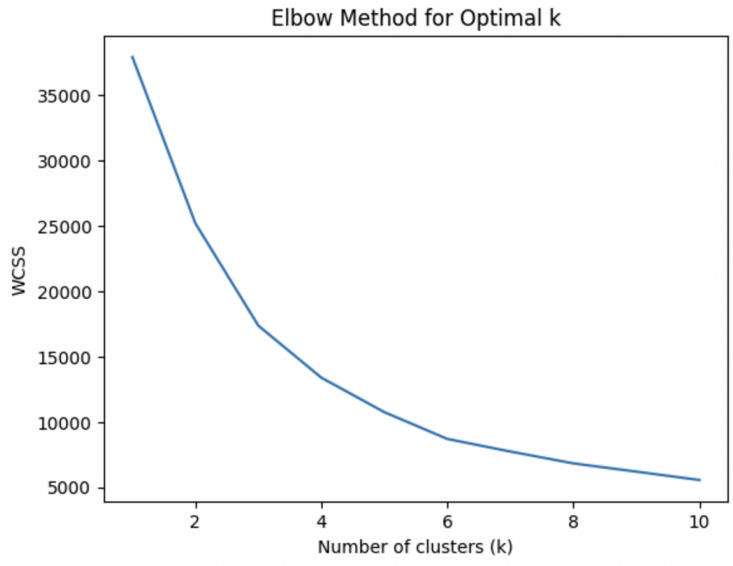

Fig 3: Plot to find optimal k cluster

This plot shows the relationship between the number of clusters and the WCSS, helping us visually identify the "elbow" where adding more clusters does not provide substantial improvement. We choose 4 as our optimal cluster size.

Next, we analyze how the clusters are distributed across various feature values over time. This subplot shows the cluster assignments as a function of the feature values (such as moving averages, volatility ratios, etc.) and time steps, providing a visual representation of how the clusters evolve with respect to the features over time.

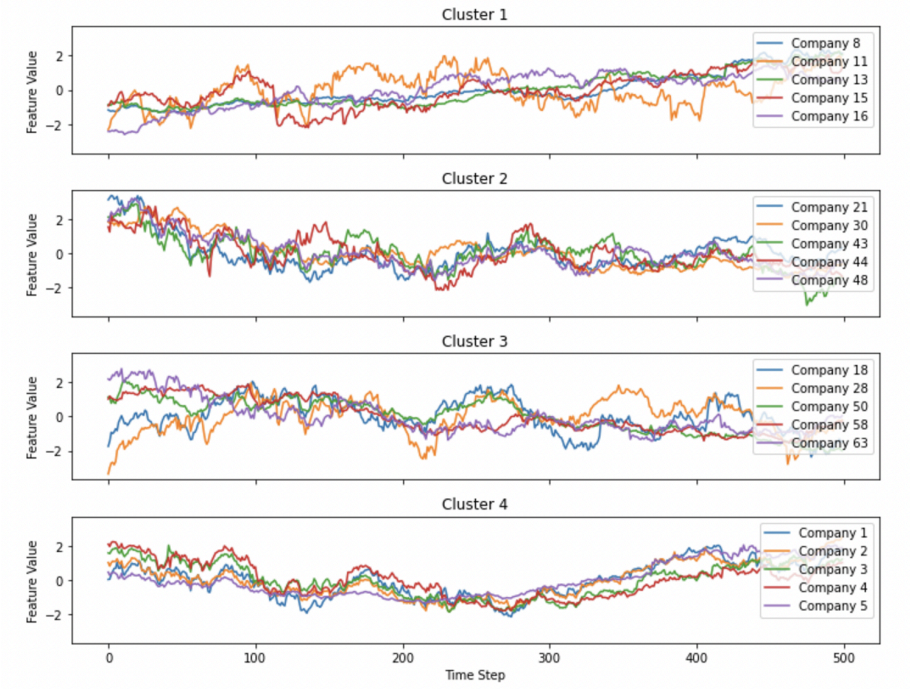

Fig 4: Clusters distributed across various feature values over time.

And lastly we visualize how our clusters turned out

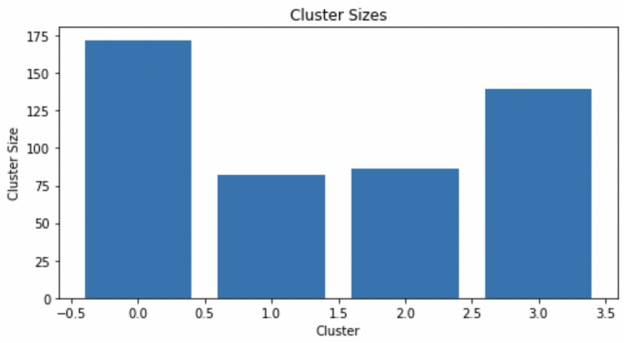

Fig 5: Cluster Size vs Cluster

This method gives us a silhouette score of

Silhouette Score: 0.079777174946915

Despite achieving a **Silhouette Score of 0.0797** with 16 features, the clustering results were not as strong as expected. A low silhouette score indicates that the clusters are not well-separated, which suggests that the high dimensionality of the feature space might be introducing noise and complicating the clustering process. Given this, **Principal Component Analysis (PCA)** becomes necessary to reduce the dimensionality of the data. PCA will help capture the most important patterns while removing irrelevant variations, potentially improving clustering quality and leading to better-defined clusters.

4.2. K-means with DTW with PCA

To determine the optimal number of components for **Principal Component Analysis (PCA)**, we use a **scree plot** to visualize the explained variance for each principal component. The plot helps identify the point where adding more components no longer contributes significantly to explaining the variance in the data. This plot will show the explained variance for each component, and the point where the curve starts to flatten will indicate the optimal number of components to retain.

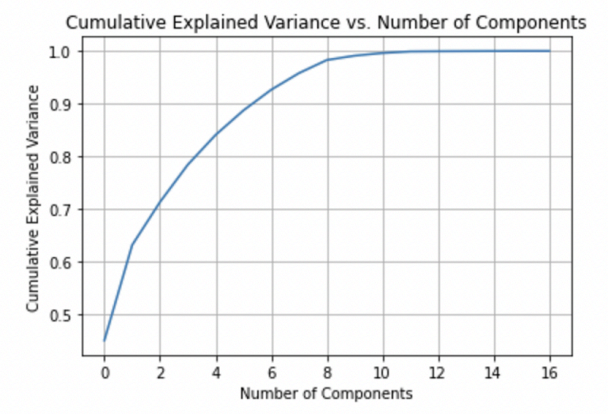

Fig 6: Plot to calculate optimal components for PCA

After applying **Principal Component Analysis (PCA)** to reduce the dimensionality of the dataset, we first calculated the optimal number of components, resulting in a reduction to **5 features**. This transformation helped capture the most important patterns in the data while reducing noise. With the reduced feature set, we recalculated the clustering using **K-Means with DTW distance**. The resulting **Silhouette Score of 0.3423** shows a significant improvement compared to the previous score of 0.0797, indicating that the clusters are now better separated and more cohesive. This demonstrates the effectiveness of dimensionality reduction in enhancing the quality of the clustering results.

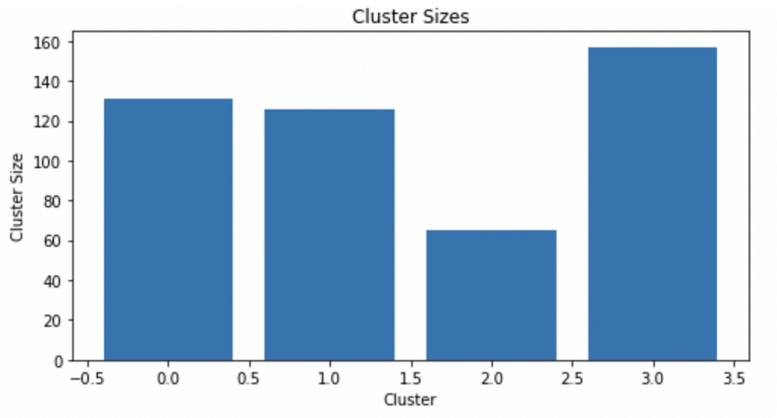

Fig 7: Cluster Size vs Cluster

We also did another experiment with the PCA data. We trained multiple models once on the original data and once on the PCA data to show differences.
These were are findings

Table 1 : Clustering Algorithm Performance Comparison: PCA vs Non-PCA Data.

	PCA data	Non PCA data
KMeans (Euclidean) - Inertia	261.2429883278463	483.7196148833381
DTW - Inertia	201.427834939878912	431.2684335116955
SoftDTW - Inertia	53746.8379624339133	187508.3240856846
KernelKMeans - Inertia	947.4382793284980	949.9999999999999

Key Findings:

- Dimensionality Reduction with PCA: The inertia values for PCA-transformed data were significantly lower across all clustering methods compared to the non-PCA data. This suggests that PCA helped improve the clustering process by removing noise and retaining the most significant features.
    - For example, KMeans (Euclidean) with PCA resulted in an inertia of 261.24, a substantial reduction from 483.72 in the non-PCA data.
    - DTW showed a similar reduction, with inertia dropping from 431.27 to 201.43.
- SoftDTW Performance: While SoftDTW on non-PCA data showed much higher inertia (187,508.32), it was significantly reduced to 53,746.84 in the PCA-transformed data. Despite this reduction, SoftDTW still exhibited high inertia compared to other metrics, indicating that the clustering might be more sensitive to the data structure or complexity, even after dimensionality reduction.
- KernelKMeans: Both PCA and non-PCA runs yielded similar inertia values (947.44 for PCA and 950.00 for non-PCA), indicating that KernelKMeans might not benefit as significantly from dimensionality reduction compared to the other methods.

The experiment demonstrates that applying **PCA** effectively reduces inertia across most clustering methods, leading to more compact and meaningful clusters. While **SoftDTW**

continues to exhibit high inertia even after PCA, other methods like **KMeans (Euclidean)** and **DTW** show substantial improvements. This suggests that dimensionality reduction via PCA helps enhance clustering performance, particularly for distance-based metrics, by focusing on the most informative features of the data.

4.3. Autoencoder with Kmeans

We saw that Kmeans improves with dimensionality reduction so we tried to apply autoencoders to reduce dimensions and calculate results

We used an **LSTM Autoencoder** for dimensionality reduction followed by clustering to group stocks based on their time-series patterns. The Autoencoder learns to compress the input time-series data into a lower-dimensional latent space, and the clustering process is applied to the reduced representation.

Here is a breakdown of the method:

Model Architecture

- **Encoding Layer:** The input to the autoencoder is a sequence of time-series data with n_features for each time step. The LSTM layer compresses this sequence into a lower-dimensional latent space, with the encoding dimension set to **4**, which is chosen based on experimentation and optimization.

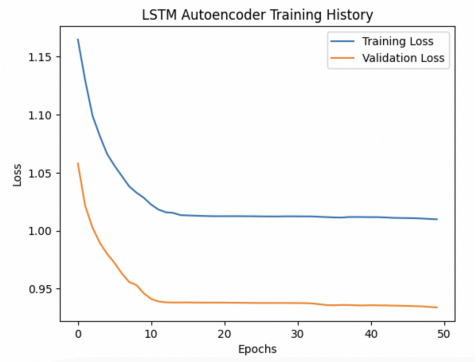

Fig 8: LSTM Encoder Loss vs Epoch plot

- **Decoding Layer:** After encoding, the data is repeated and passed through another LSTM layer to reconstruct the original time-series data from the reduced latent representation. The decoder attempts to restore the data's original shape while preserving its key patterns.

Clustering

- After training the autoencoder, we used the encoded latent space (with a dimensionality of 4) as the input for clustering. Based on optimization criteria like silhouette score, which measures how well-separated the clusters are, the most optimal number of clusters was 2, followed by 4.

Fig 9 : Silhouette score analysis to find optimum k clusters

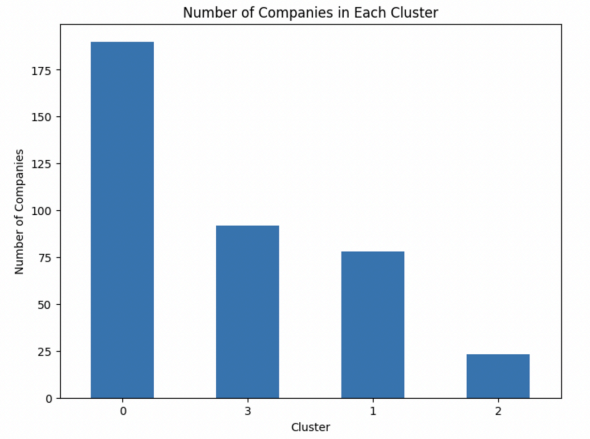

Fig 10: Cluster Size vs Cluster

Visualization with t-SNE

- **t-SNE** (t-Distributed Stochastic Neighbor Embedding) is a dimensionality reduction technique that is particularly well-suited for visualizing high-dimensional data in 2D or 3D space. After clustering the data in the reduced 4-dimensional latent space, we used t-SNE to project the clustered data into a 2D space for visualization. This helps to visually assess how well the clusters are separated.

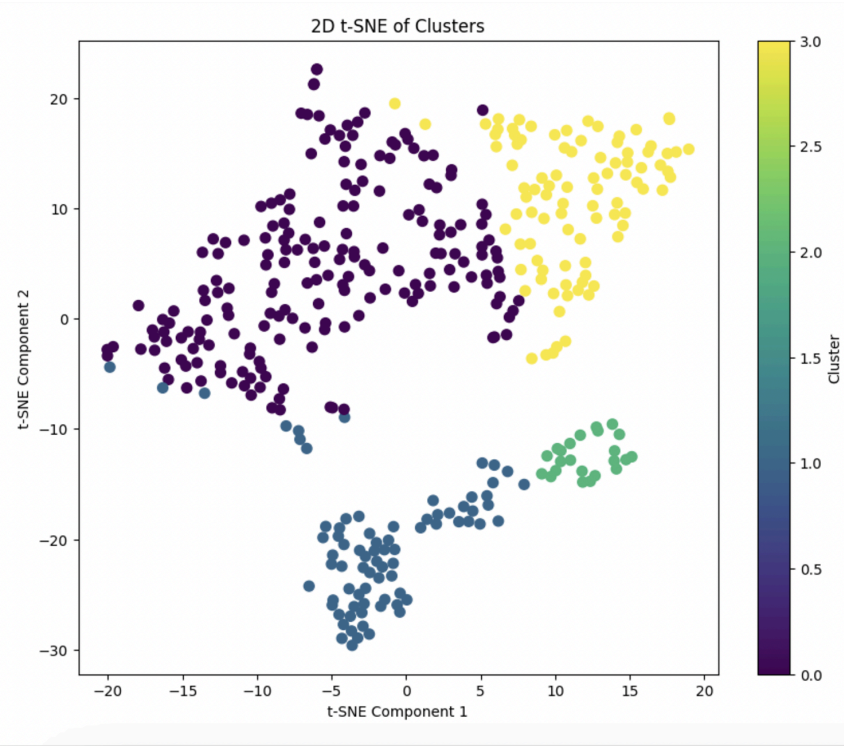

Fig 11: Cluster Visualisation using t-SNE

The **LSTM Autoencoder** method achieved a silhouette score of **0.46**, indicating a moderate level of cluster separation and cohesion. This is an improvement compared to the **KMeans with PCA** approach, which showed a lower silhouette score of **0.34**. The LSTM Autoencoder's ability to capture complex temporal dependencies in the data likely contributed to better-defined clusters, as it effectively reduced the dimensionality while preserving key sequential patterns. In contrast, PCA, a linear method, might not have

captured all the intricate relationships in the time-series data, leading to less optimal clustering performance.

4.4. Hierarchical Clustering

After using the **LSTM Autoencoder** to reduce the time-series data into a lower-dimensional latent space, we proceeded with **Hierarchical Clustering**. However, instead of applying it directly to the raw time-series data, we used the **averaged data**—specifically focusing on the moving averages (MAs) and exponential moving averages (EMAs)—to simplify the representation into a single data point for each stock.

- We applied **Agglomerative Hierarchical Clustering**, a bottom-up approach that builds a hierarchy of clusters. The algorithm starts by treating each data point (stock) as its own cluster and iteratively merges the closest clusters based on a distance metric (e.g., Euclidean distance).
- We used the **average linkage** method, which merges clusters based on the average distance between all points in each cluster.

Elbow Method:

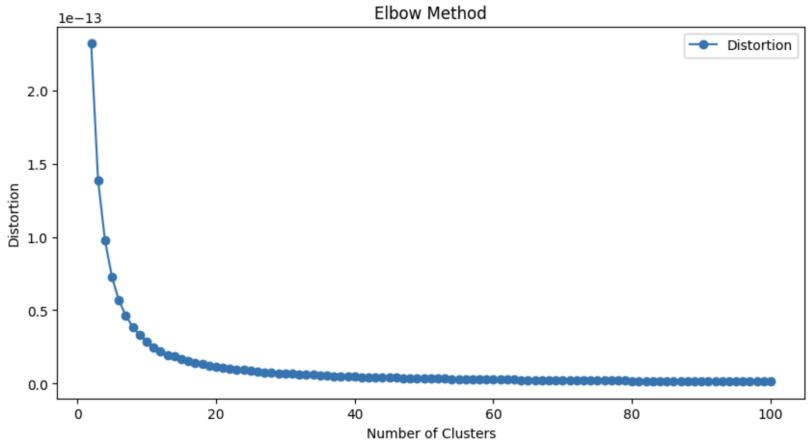

Fig 12 : Inertia analysis to find optimum k clusters

Upon analyzing the trend, it becomes evident that the reduction in distortion begins to taper off after a certain point, indicating diminishing returns in terms of improved clustering quality. This plateau suggests that increasing the number of clusters further may not provide significant benefits. Consequently, using the Elbow Method as a guiding principle, we have determined that **14 clusters** represent the optimal balance between minimizing distortion and maintaining computational efficiency, effectively capturing the underlying structure of the data.

Silhouette Analysis:

- To determine the optimal number of clusters, we used **Silhouette Analysis**. This technique calculates how similar each data point is to its own cluster compared to other clusters. A higher silhouette score indicates that the clusters are well-separated.

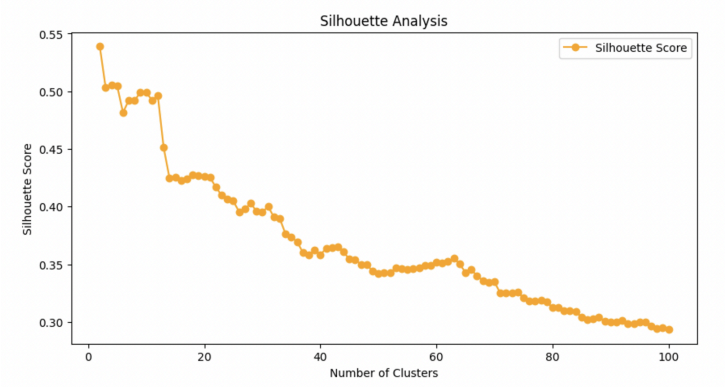

Fig 13 : Silhouette score analysis to find optimum k clusters

- Based on this analysis, we found that the optimal number of clusters was **14**, as it resulted in the highest silhouette score, indicating a good balance between compactness and separation of the clusters.

Once the number of clusters was determined, we used hierarchical clustering to assign each stock to one of the 14 clusters. The final clusters represented groups of stocks with similar trends, as determined by their moving averages and exponential moving averages.

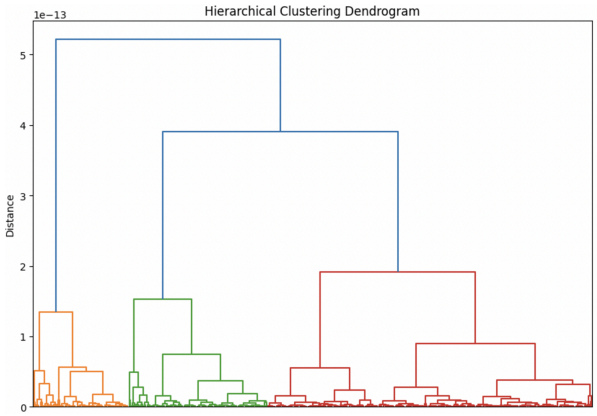

Fig 14 : Hierarchical Clustering Dendrogram

We plotted a table with every company different clusters and got results as follows:

Table 2 : Companies in different clusters k=14

Silhouette Score for 14 clusters: 0.4499806442336073

Key	Company 1	Company 2	Company 3	Company 4	Company 5	Company 6	Company 7	Company 8	Company 9	Company 10	Company 11	Company 12
14	AAPL	UNH	NEE	GILD	MELI	FI	ITW	AON	MAR	NXPI	OXY	BNS
9	MSFT	TSLA	TSM	JPM	WMT	MA	ORCL	COST	BAC	LIN	NVS	ABT
13	GOOGL	META	ADBE	CRM	CSCO	NOW	UPS	RY	SBUX	ISRG	MUFG	PBR
11	AMZN	ASML	AMD	TMO	DHR	BA	RTX	T	RIO	CB	MU	CI
10	NVDA	NVO	HD	TMUS	CMCSA	MS	LOW	AMT	PANW	REGN	CDNS	SLB
7	LLY	MCD	AMGN	CAT	UL	SNY	GS	EQNR	ENB	GD	CP	PSX
12	V	XOM	CVX	PEP	SHEL	NFLX	BABA	PDD	NKE	INTU	WFC	QCOM
5	AVGO	MRK	ACN	INTC	PFE	HDB	IBM	DE	ADI	ETN	CVS	KLAC
8	JNJ	FMX	TTE	BHP	HSBC	MDT	MMC	EQIX	BTI	MCK	CMG	BMO
2	PG	TM	AZN	SAP	HON	PLD	BP	BMY	SO	DUK	APD	DELL
3	KO	ABBV	VZ	PM	GE	TD	GSK	CSX	CHTR	CTAS	NEM	ADSK
6	TRI	VOD	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1	EXC	FNV	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
4	BIIB	FCNCA	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 3 : Companies in different clusters for k=100

The Silhouette Score for 100 clusters is: 0.31039354184516205

Cluster Key	Company 1	Company 2	Company 3	Company 4	Company 5	Company 6	Company 7	Company 8	Company 9	Company 10
89	AAPL	ITW	HES	DAL	RJF	N/A	N/A	N/A	N/A	N/A
52	MSFT	TSLA	MA	BX	SONY	CCI	AZO	D	UMC	N/A
86	GOOGL	NOW	LNG	NUE	BK	FERG	WPM	TRGP	ALC	N/A
65	AMZN	TMO	RTX	CB	CI	RACE	MCO	USB	PAYX	IQV
60	NVDA	PANW	REGN	PYPL	SRE	NVR	N/A	N/A	N/A	N/A
83	META	CSCO	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
29	LLY	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
68	V	PKX	PLTR	N/A	N/A	N/A	N/A	N/A	N/A	N/A
53	TSM	LIN	ABT	IBN	SMFG	CL	ITUB	MSI	O	CTSH
93	UNH	MAR	NXPI	ABEV	CM	RCL	DB	ZBH	N/A	N/A
19	AVGO	TDG	TRV	N/A	N/A	N/A	N/A	N/A	N/A	N/A
61	NVO	HD	AMT	SHW	NOC	PSA	WCN	PPG	NDAQ	PUK
49	JPM	BAC	COP	SCHW	LRCX	MFG	DLR	CNC	IMO	VEEV
55	WMT	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
75	XOM	TXN	TJX	KKR	CNI	PH	AJG	TEAM	SQ	VICI
40	JNJ	MMC	BTI	BMO	DHI	ACGL	CARR	N/A	N/A	N/A
10	PG	BP	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
48	ORCL	COST	ANET	EL	N/A	N/A	N/A	N/A	N/A	N/A
84	ADBE	CRM	RY	ISRG	MPLX	BCE	VRSK	XEL	KEYS	APTV

The hierarchical clustering algorithm on stock market data and identified 14 distinct clusters, achieving a **Silhouette Score of 0.45**. What we found particularly interesting is how the algorithm naturally grouped similar companies together. Take a look at cluster 13 for instance - it neatly grouped major tech companies like Google, Meta, and Adobe, which makes sense given their similar market behaviors and business models. We also found it intriguing that the largest clusters captured different segments of market leaders - cluster 14 has companies like Apple and UnitedHealth, while cluster 9 includes Microsoft and Tesla.

The smaller clusters are quite interesting too. We noticed that clusters 1, 4, and 6 only contain a few companies each, suggesting these firms have unique market behaviors that set them apart from the main groups. The Silhouette Score of 0.45 tells us that while these clusters are meaningful, there's some overlap in how these stocks behave - which is exactly what you'd expect in a complex, interconnected market. Overall, we think these results give us a pretty good picture of how different companies move together in the market.

When we increased the number of clusters to 100 to try and form small groups of stocks which are similar to each other, trying to find more intricate patterns in the data, the clustering results became much less meaningful. The Silhouette Score dropped significantly, indicating poor clustering performance and a lack of clear distinction between groups. With so many small clusters, the algorithm struggled to form any coherent patterns, and many of the clusters appeared to be arbitrarily split, with no clear market behavior or business model similarities between the companies. The high number of clusters led to fragmentation, where the clusters were too granular to reveal any insightful trends or groupings. This poor clustering quality makes it difficult to identify meaningful patterns or relationships within the data, highlighting that the optimal number of clusters lies far below 100 for this particular dataset.

5. Conclusions

The results of our project indicate that hierarchical clustering, despite only considering a single data point (the average of MAs and EMAs), provided insightful groupings that align well with industry patterns. The Silhouette Score of 0.45 suggests that while the clusters are meaningful, there is still some overlap in how the companies behave, which reflects the interconnected nature of the market. The distinct groupings, such as those containing major tech companies like Google, Meta, and Adobe, as well as the segmentations that capture key market leaders like Apple and UnitedHealth, highlight the value of this method in identifying similarities based on market behavior. Smaller clusters with fewer companies indicate unique market movements, which was particularly interesting.

However, one limitation of this method was its reliance on only one data point per company—averaging the moving averages and exponential moving averages—without considering the full time-series behavior of each stock. This simplification, while useful for quick insights, may not have captured the true dynamics of the stock prices over time. In contrast, methods like KMeans and Autoencoders, which took into account more features and time-series information, offered a broader perspective but lacked the granularity of hierarchical clustering.

Future Directions

While the current approach yielded valuable insights, there are several potential improvements for future work. One key area for enhancement is the inclusion of additional features beyond moving averages and exponential moving averages (EMAs). Incorporating more diverse financial metrics, such as volatility, sentiment analysis from news or social media, and macroeconomic indicators, could provide a richer understanding of stock behavior. Moreover, experimenting with different dimensionality reduction techniques, such as **t-SNE** or **UMAP**, could help reveal more intricate patterns within the data.

In terms of methodology, exploring **advanced machine learning techniques** like Deep Clustering models could further improve our ability to uncover meaningful groupings in the data.

These methods may be able to capture non-linear relationships or dense regions of similar stock behaviors that the current algorithms do not.

Additionally, incorporating a **time-series forecasting** approach into the clustering process could help identify how clusters evolve over time, providing a more dynamic view of the market. This could be particularly useful in understanding how the behavior of companies changes during periods of market volatility or economic shifts.

In conclusion, while the current project successfully identified meaningful clusters of companies with similar stock market behaviors, future work could involve expanding the features, refining the clustering methods, and incorporating time-series forecasting to enhance the accuracy and depth of the findings.

6. Individual Tasks

In this project, our team worked collaboratively to analyze and cluster stock market data using various clustering algorithms. Each member contributed to different aspects of the project while working together on critical shared tasks such as data preprocessing and writing the project report.

I was primarily responsible for implementing the KMeans clustering algorithm. This involved performing experiments with both Euclidean distance and Dynamic Time Warping (DTW) as distance metrics, as well as evaluating the models using Silhouette Scores and inertia metrics. Additionally, I incorporated Principal Component Analysis (PCA) to reduce the dimensionality of the data and compared the clustering results with and without PCA. I analyzed the impact of the optimal number of clusters on the results and visualized the findings using elbow plots and feature-based cluster distributions. My work helped establish a baseline for clustering performance and provided insights into the relationship between dimensionality reduction and cluster formation.

Hibah focused on implementing an Autoencoder with KMeans clustering. She developed a Long Short-Term Memory (LSTM) Autoencoder to encode time-series data into a compressed feature space, enabling effective clustering. She also conducted experiments to determine the optimal

number of clusters for KMeans in the encoded space and visualized the results with a t-SNE plot to demonstrate cluster formation. Hibah's work provided a deep learning perspective on clustering and highlighted the potential of combining neural networks with traditional clustering techniques to improve accuracy.

Darsh worked on hierarchical clustering to analyze stock data based on averaged features such as moving averages (MAs) and exponential moving averages (EMAs). He conducted experiments to identify the optimal number of clusters using Silhouette Analysis and visualized the hierarchical dendrogram. Darsh also summarized the composition of each cluster and identified patterns across industries, providing meaningful insights into how different companies behave in the stock market. His work demonstrated the advantages of hierarchical clustering in capturing nuanced relationships between companies.

By dividing the work into these key areas, we were able to leverage our individual strengths to explore multiple clustering approaches comprehensively, ensuring the success of the project.

7. References

[1] S. Thomas. “Massive Yahoo Finance Dataset,” Version 2, 2023. Retrieved October 29, 2024 from https://www.kaggle.com/datasets/iveeaten3223times/massive-yahoo-finance-dataset.

[2] Denyse. “Time Series Clustering — Deriving Trends and Archetypes from Sequential Data” July 28, 2001. Available https://towardsdatascience.com/time-series-clustering-deriving-trends-and-archetypes-from-sequential-data-bb87783312b4

[3] Heka.Ai. (2022, June 9). Time series clustering - Heka.ai - Medium. *Medium*. https://heka-ai.medium.com/time-series-clustering-b84bcaaa63ac

[4] J. Vásquez Sáenz, F. M. Quiroga, and A. F. Bariviera, "Data vs. information: Using clustering techniques to enhance stock returns forecasting," *International Review of Financial Analysis*, vol. 88, 2023, Art. no. 102657. doi: 10.1016/j.irfa.2023.102657.

[5] A. Fernandez and F. J. Joel, "Clasificación de acciones mediante k-means clustering," SSRN, Dec. 2022. Available: https://ssrn.com/abstract=4403662.

[6] S. Aghabozorgi and Y. W. Teh, "Stock market co-movement assessment using a three-phase clustering method," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1301-1314, 2014. doi: 10.1016/j.eswa.2013.08.028.