Segmenting Financial Markets: A Clustering Approach to the S&P 500 Index

Table of Contents

Table of Figures

1.1 Introduction

1.1.1 Background

The analysis of financial markets, such as the S&P 500 index, is essential for understanding market dynamics, investment opportunities and economic trends. Financial datasets are often complex, making it challenging for the average person to uncover meaningful patterns without advanced analytical methods and expert domain knowledge. Clustering, an unsupervised learning technique can be used to discover similar characteristics, leading to the segmentation of markets into well-defined patterns. This segmentation can lead to identifying trends, informing investment strategies and mitigating risks.

This study utilizes two clustering algorithms, K-Means and Agglomerative Clustering, to analyse the S&P 500 dataset. These two clustering algorithms utilize different methods in segmenting data points. K-Means partitions data into distinct non-overlapping clusters, while agglomerative clustering creates and hierarchy of nested clusters in the form of a dendrogram. These methods will be used in identifying different market conditions which can serve as a foundation for strategic financial decision-making.

1.1.2 Problem Statement

This study aims to address the challenge of extracting meaningful patterns from financial datasets like the S&P500 dataset by applying clustering algorithms.

1.1.3 Research Question

How can clustering techniques like K-Means and Agglomerative clustering uncover meaningful patterns in financial datasets, and how can these patterns be applied in making decisions?

1.1.4 Relevant works

Numerous studies have employed clustering techniques to identify patterns and discover market patterns in financial data. Lucio and Caiado (2022) used both hierarchical and non-hierarchical clustering algorithms to analyse industry groupings in the S&P500. They aimed to examine the impact of COVID-19 on market volatility providing insights on how different industries were affected. Bin (2020) introduced a portfolio-building algorithm that combines K-Means clustering and the Sharpe ratio to optimize stock selection within the S&P 500.

Additionally**,** Rusu and Boloș (2024) reviewed the application of clustering techniques in financial markets, focusing on how machine learning algorithms like K-Means and Agglomerative Clustering are used to group stocks based on performance and sectoral trends. Their work emphasized how clustering simplifies complex financial data, offering valuable insights for decision-making by identifying correlations and patterns within market behaviour.

Building on these works, this study applies two clustering algorithms to the S&P 500 index dataset to uncover patterns that reflect market conditions aiming to provide actionable insights into market segmentation, portfolio management and risk mitigation.

1.1.5 Methodology

This study adopts the CRISP-DM (Cross Industry Standard Process for Data Mining) methodology, which provides a structured and systematic approach to handling data mining tasks. This process is divided into five phases: Business Understanding, Data Understanding, Data Preparation, Modelling and Evaluation (Larose & Larose, 2015).

1.2 Dataset

The S&P500 index dataset (Shiller's data) used in this report was obtained from Datahub, a public data repository. It contains S&P500 index data collected monthly from 1870 to 2023. The dataset contains 10 features and 1,833 observations.

*Table **Error! No text of specified style in document.**.1 Description of features in the S&P500 dataset*

FEATURE	DESCRIPTION
DATE	Date
SP500	The combined market cap of the 500 largest publicly traded companies
DIVIDEND	Dividend per share for companies in the index
EARNINGS	Aggregate profit
CONSUMER PRICE INDEX	Weighted average of prices of consumer goods and services
LONG INTEREST RATE	Long-term interest rates
REAL PRICE	Inflation-adjusted S&P500 value
REAL DIVIDEND	Inflation-adjusted dividend per share
REAL EARNINGS	Inflation-adjusted earnings
PE10	Cyclically adjusted price-to-earnings ratio (10 years)

1.2.1 Ethical / Social /Legal Issues

The S&P500 dataset has been licenced under the Open Data Commons Public Domain Dedication and License making it a public dataset that ensures no ethical, social or legal issues associated with its use.

1.3 Exploratory Data Analysis and Preprocessing

Exploring and analysing the features in a dataset is essential for gaining a proper understanding and developing a foundation for the modelling stage. This process will include data cleaning and data visualizations that provide valuable insights.

1.3.1 Data Cleaning

Duplicate Handling: No duplicated observations were found in the data.

```
#checking for duplicates
sp500.duplicated().sum()
print(f' Duplicates: {sp500.duplicated().sum()}')
```

```
Duplicates: 0
```

*Figure **Error! No text of specified style in document.**.1 Handling duplicates*

Handling Missing Values: Missing values can raise significant issues during the modelling process if not handled properly. The data provider handled the missing value by imputing with 0, this might significantly affect the modelling result. Therefore 0 values were replaced with null values to be handled appropriately.

```
# replace all the values imputed as 0 with NaN
sp500 = sp500.replace(0,np.nan)
```

*Figure **Error! No text of specified style in document.**.2 Code converting the 0 values to null values*

```
# checking for missing values again
sp500.isna().sum()/len(sp500) * 100
```

```
Date                    0.000000
SP500                   0.000000
Dividend                0.163666
Earnings                0.163666
Consumer Price Index    0.000000
Long Interest Rate      0.000000
Real Price              0.000000
Real Dividend           0.163666
Real Earnings           0.163666
PE10                    6.546645
dtype: float64
```

Figure ***Error! No text of specified style in document.****.3 Code showing missing values in the S&P500 dataset*

Earnings and dividends only have 0.16% of their data missing, so it is appropriate to drop them. While for PE10, 6.5% which corresponds to the data from the first 10 years in the data frame, by the nature of PE10 and the data, simply imputing with mean is highly inappropriate. Therefore, a simple linear interpolation method will suffice.

```
# Handling missing values in PE10 by linear interpolation
sp500['PE10'] = sp500['PE10'].interpolate(method='linear')
```

Figure ***Error! No text of specified style in document.****.4 Handling missing values in PE10 by linear interpolation*

```
# handling missing values in the earnings and dividend column
sp500 = sp500.dropna(subset='Dividend')
sp500.isna().sum()
```

Figure ***Error! No text of specified style in document.****.5 Handling missing values in Dividend and Earnings by dropping them*

Formatting Datatypes: Each feature must be in the appropriate datatype (e.g. date formatted as date-time format and numbers formatted as floats).

```
#converting the date column to datetime
sp500['Date'] = pd.to_datetime(sp500['Date'])
```

```
# numerical columns
numerical_cols = sp500.drop('Date',axis=1).columns
numerical_cols
```

```
# confirming datatypes
sp500.dtypes
```

```
Date                    datetime64[ns]
SP500                          float64
Dividend                       float64
Earnings                       float64
Consumer Price Index           float64
Long Interest Rate             float64
Real Price                     float64
Real Dividend                  float64
Real Earnings                  float64
PE10                           float64
dtype: object
```

Figure ***Error! No text of specified style in document.****.6 code for formatting and confirming the data types.*

1.3.2 Visualization

1.3.2.1 Univariate Analysis

Distribution of Numerical Features: Each feature was visualized to understand its distribution. All the distributions appear right-skewed, implying that they are not normally distributed.

```
# plotting distribution of CPI, LIR, and Realprice
fig, axes = plt.subplots(1, 3, figsize=(10, 3))
for i, col in enumerate(numerical_cols[3:6]):
    sns.histplot(sp500[col], kde=True, ax=axes[i],edgecolor='black',color='#66C2A5')
    axes[i].set_title(f'Distribution of {col}')
plt.tight_layout(w_pad=3, pad=2)
plt.show()
```

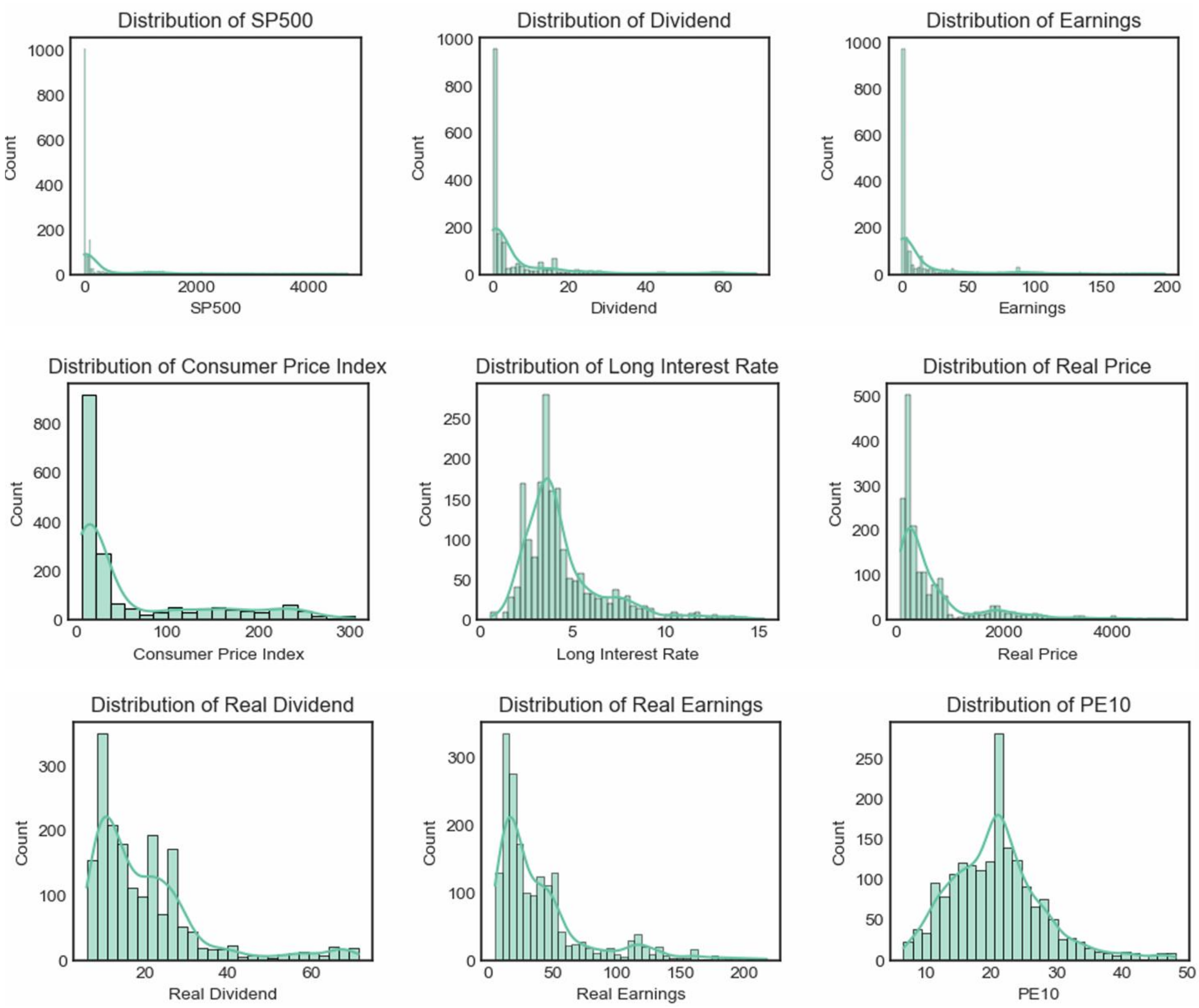

Figure ***Error! No text of specified style in document.****.7 Distribution of all numerical features in the S&P500 dataset*

1.3.2.2 Bivariate Analysis

Times Series Plots: Time series plots of each of the features were plotted to observe the trend over time. All the plots have an upward trend with little to no volatility except for the long interest rate and PE10.

```
# Time series plot
fig, axes = plt.subplots(1, 3, figsize=(12,4))
for i, col in enumerate(numerical_cols[:3]):
    sns.lineplot(x=sp500['Date'], y=sp500[col], ax=axes[i], color='#FC8D62')
    axes[i].set_title(f'{col} Over Time')
plt.tight_layout(w_pad=3, pad=2)
plt.show()
```

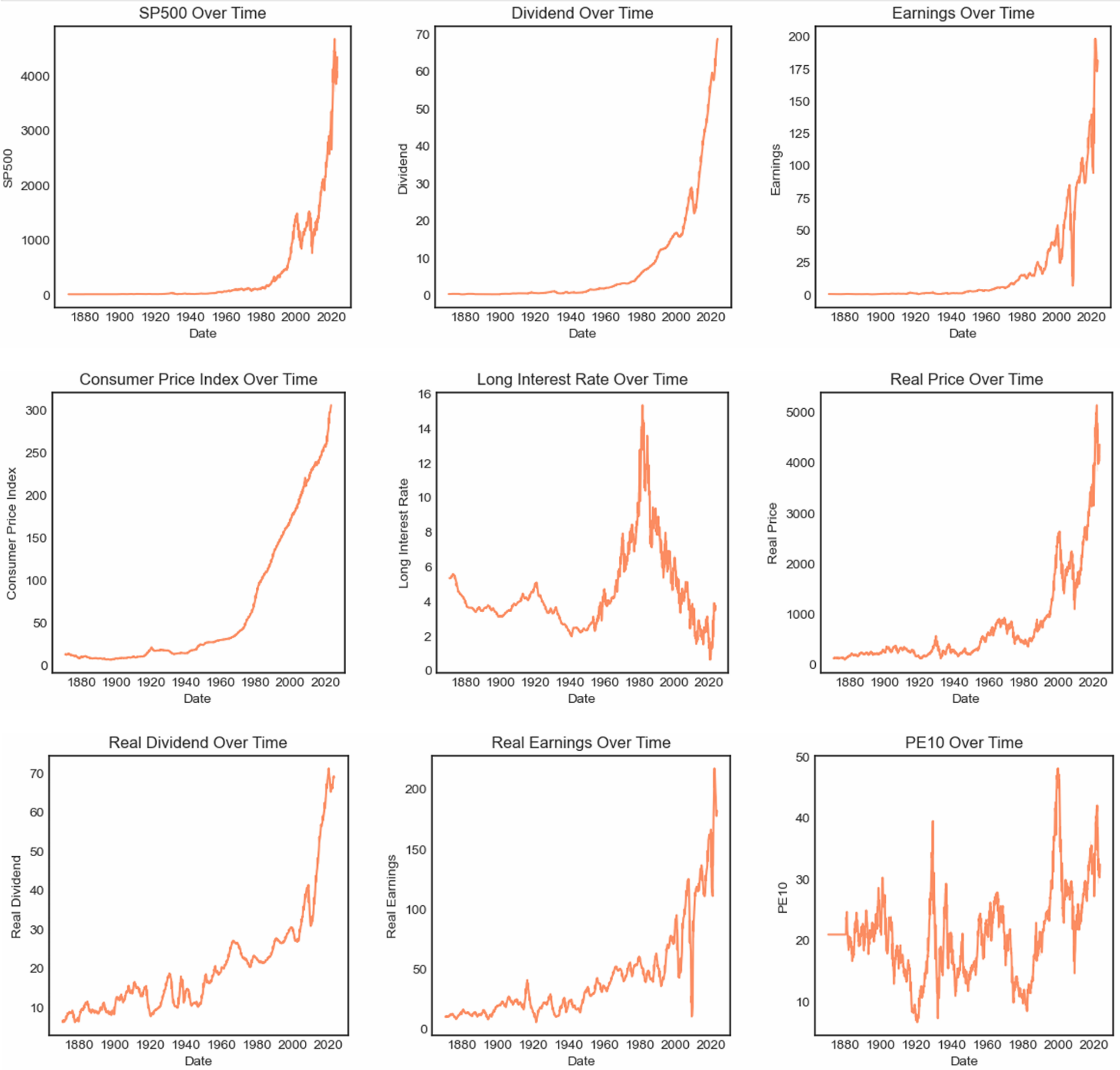

*Figure **Error! No text of specified style in document.**.8 Time series plot of all the numerical features from 1870 to 2023*

Real Earnings vs SP500: as the SP500 index level increases so does the real earnings.

```
# Relationship between SP500 and Real Earnings
plt.figure(figsize=(6, 3))
sns.scatterplot(x='SP500', y='Real Earnings',data=sp500, color='#66C2A5')
plt.title('Relationship between SP500 and Real Earnings')
plt.show()
```

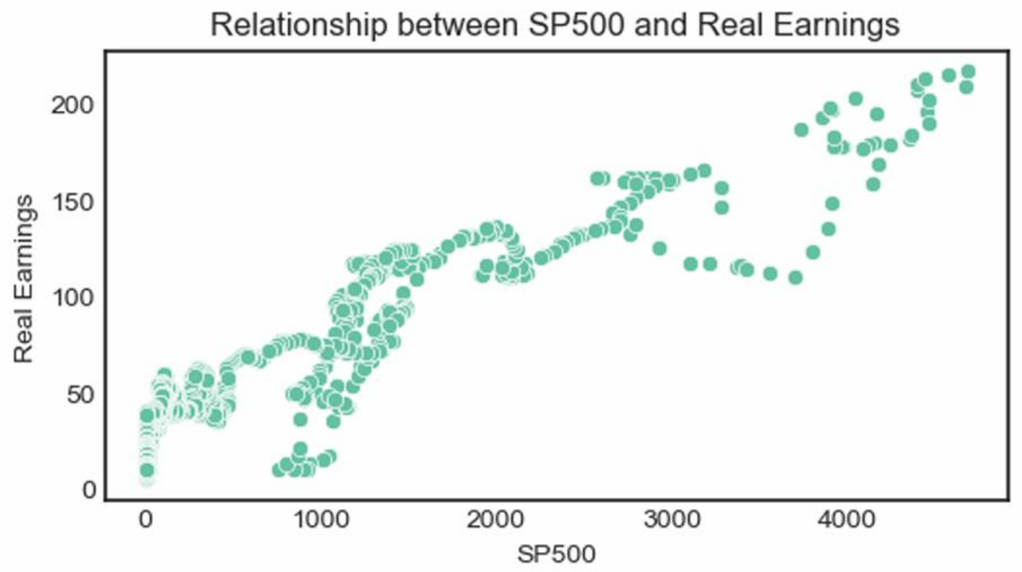

*Figure **Error! No text of specified style in document.**.9 Scatter plot showing the relationship between SP500 and real earnings*

1.3.2.3 Multivariate Analysis

Real Earnings vs Earnings: a comparison of earnings and real earnings over time shows that real earnings are almost always greater than earnings.

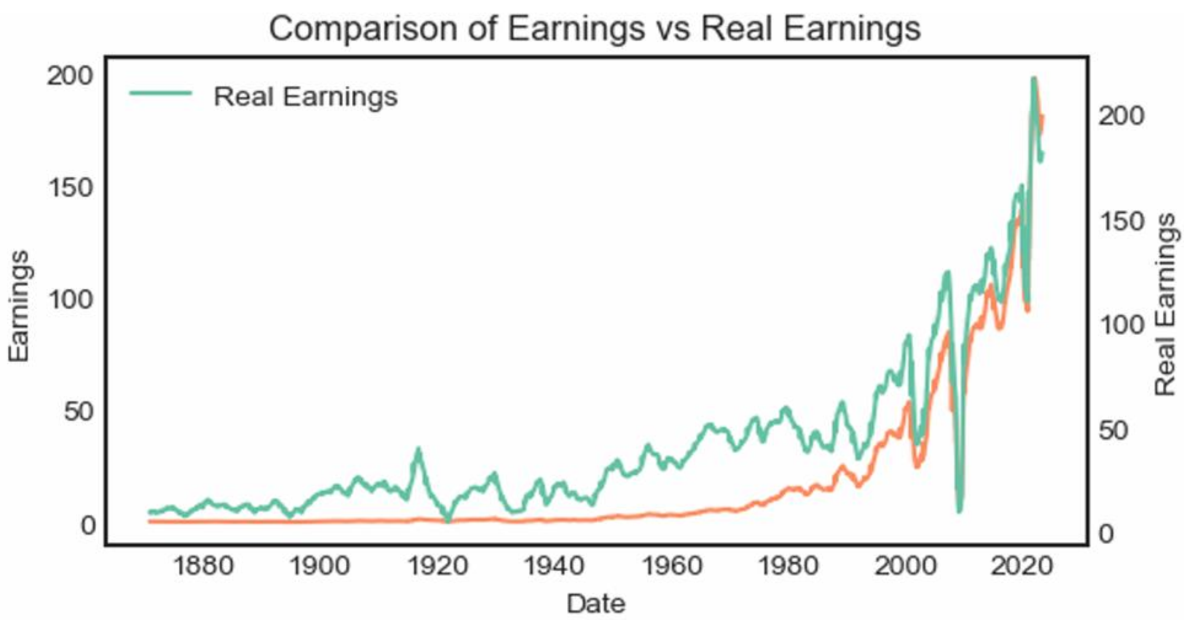

*Figure **Error! No text of specified style in document.**.10 Time series plot of real earnings vs earnings*

Correlation: The SP500 index level appears to be positively correlated with dividends, earnings and CPI, but negatively correlated with long interest rates

```
# Heatmap of correlation
plt.figure(figsize=(12,10))
sns.heatmap(sp500.corr(),annot=True,cmap='Blues')
plt.savefig('corr.png')
plt.show()
```

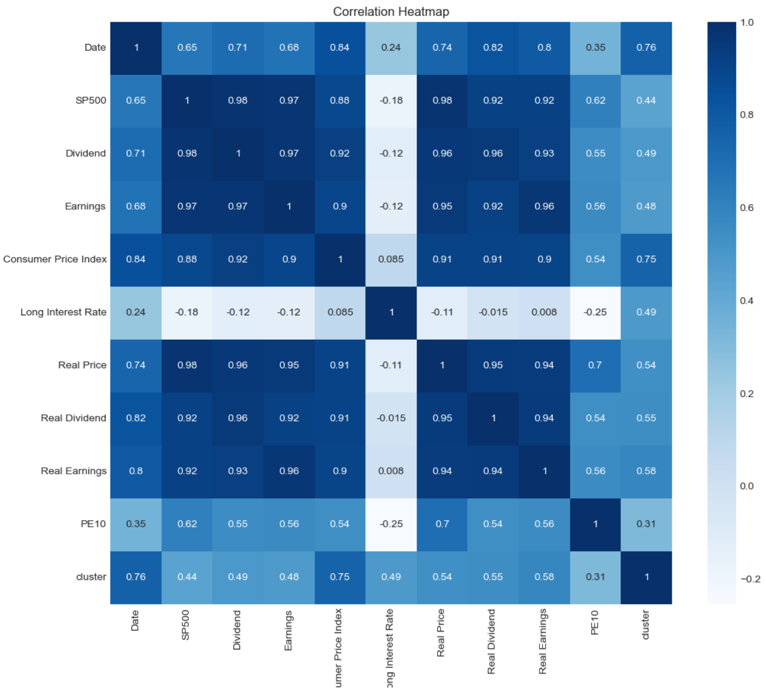

*Figure **Error! No text of specified style in document.**.11 Heatmap showing the correlation between all the numerical features*

1.3.3 Preprocessing

The S&P500 datasets contain all numerical features and dates, therefore only two steps are needed in preparation for the modelling stage.

Selecting Relevant Columns: the date column was dropped due to its irrelevance to the clustering.

```
# numerical columns
numerical_cols = sp500.drop('Date',axis=1).columns
numerical_cols
```

```
# creating a data frame that will be used for modelling
X = sp500[numerical_cols]
```

*Figure **Error! No text of specified style in document.**.12 Code showing the selected columns assigned to X*

Standardization: clustering algorithms typically rely on distance metrics to measure similarity but features in the S&P500 dataset have various scales and units, which can affect distance calculations. To solve this, StandardScaler was used to scale the features in the dataset to have a mean and standard deviation of approximately 0 and 1 respectively.

```
#Instantiating the scaler
scaler = StandardScaler()
```

```
# Scaling numerical features
X[numerical_cols]=scaler.fit_transform(X[numerical_cols])
```

```
# summary statistics
X.describe().T
```

	count	mean	std	min	25%	50%	75%	max
SP500	1830.0	3.106198e-17	1.000273	-0.464396	-0.457824	-0.445055	-0.236211	5.439613
Dividend	1830.0	-3.106198e-17	1.000273	-0.535226	-0.517634	-0.480250	0.016363	4.488083
Earnings	1830.0	9.318593e-17	1.000273	-0.503982	-0.492050	-0.464022	-0.077668	5.216044
Consumer Price Index	1830.0	0.000000e+00	1.000273	-0.736990	-0.686730	-0.540724	0.531075	3.017825
Long Interest Rate	1830.0	-6.212396e-17	1.000273	-1.682651	-0.580992	-0.299324	0.266187	4.711974
Real Price	1830.0	0.000000e+00	1.000273	-0.724858	-0.583853	-0.459283	0.081961	4.874278
Real Dividend	1830.0	1.242479e-16	1.000273	-1.046311	-0.713223	-0.295753	0.406877	3.773100
Real Earnings	1830.0	6.212396e-17	1.000273	-0.957486	-0.672310	-0.375313	0.225256	4.677754
PE10	1830.0	-4.348677e-16	1.000273	-1.997175	-0.709917	0.016687	0.496295	3.831098

*Figure **Error! No text of specified style in document.**.13 Result after standardizing the data*

1.4 Implementation

K-Means clustering and Agglomerative clustering were chosen to analyse because they handle clustering with different approaches. K-means works by partitioning data objects into non-overlapping clusters that minimize within-cluster variance, while agglomerative clustering uses a hierarchical approach to progressively merge data points into a tree-like structure (dendrogram) based on similarity.

1.4.1 K-Means Clustering

K-Means clustering is one of the top ten clustering algorithms used in data mining, it combines simplicity, and flexibility with its low computational complexity to solve clustering problems effectively (Ikotun et al., 2023). K-means aims to minimize the within-cluster sum of squares (WCSS) by assigning data points to clusters based on distance from the cluster centroids.

1.4.1.1 Implementation

Assigning random state value: The value of 123 was assigned to an object “random”, which will be used when setting random states for hyperparameters.

```
#setting random state
random = 123
```

*Figure **Error! No text of specified style in document.**.14 Assignin random state, to maintain consistency*

Determining the optimal number of clusters: The K-Means algorithm was applied by varying the number of clusters (k) from 2 to 10. For each value of k, the within-cluster sum of square and silhouette scores were plotted on a line graph. The optimal number of clusters was identified as 4 by using the elbow method and validated with the highest silhouette score.

```python
# finding the optimal number of clusters
WCSS = []
s_score = []
k = 10
for i in range(2,k+1):
    kmeans = KMeans(n_clusters=i,init='k-means++',random_state=random)
    kmeans.fit(X)
    WCSS.append(kmeans.inertia_)
    s_score.append(silhouette_score(X, kmeans.labels_))
```

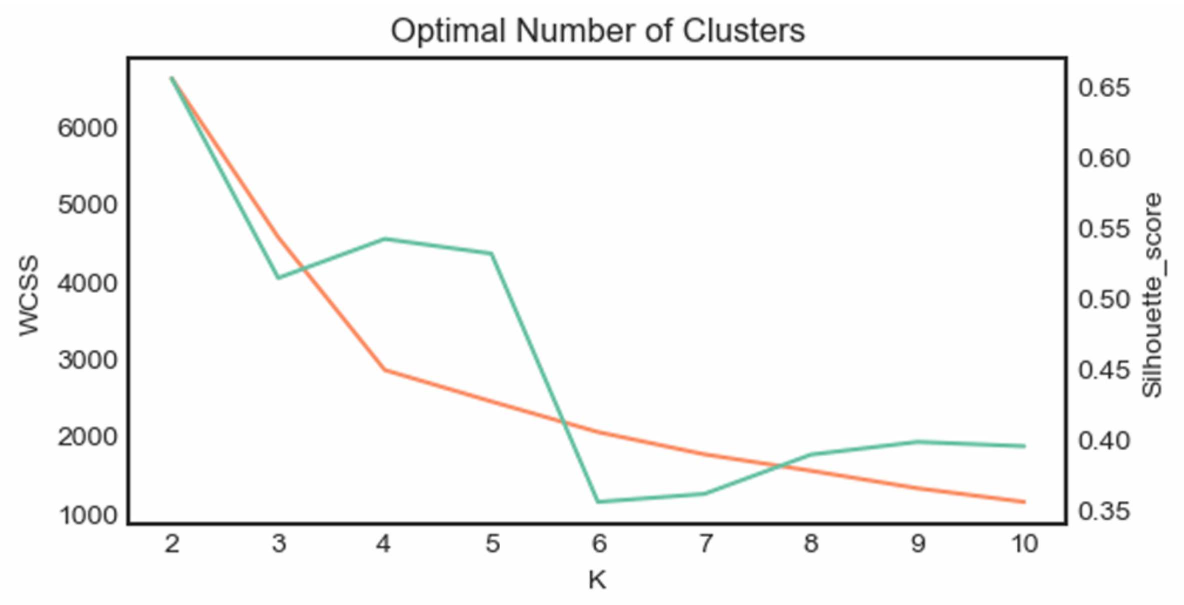

*Figure **Error! No text of specified style in document.**.15 Line graph showing the optimal number of clusters*

Hyperparameter tuning:

- **n_clusters:** this parameter sets the number of clusters, 4 was chosen as the number of clusters based on the result from the elbow method.
- **init:** this parameter sets how the initial k-centroids are initialized, “k-means++” ensures the best possible location to start from instead of just random.
- **random_state:** random state was set to 123, to ensure reproducibility.

```python
# Clustering based on optimal k
kmeans = KMeans(n_clusters=4, init='k-means++', random_state=random)
kmeans.fit(X)
```

```
KMeans
KMeans(n_clusters=4, random_state=123)
```

*Figure **Error! No text of specified style in document.**.16 Code used in fitting the K-Means model*

1.4.1.2 PCA

Principal component analysis (PCA) is a dimensionality reduction technique used in this analysis to simplify the visualization of the clusters formed by the K-means clustering. PCA was applied to reduce the dataset to 2 dimensions while retaining 91% of the variance to visualize the clusters. Here are the steps taken:

1. Instantiating PCA: the number of components was set to two and a random state of 123 was assigned to ensure reproducibility.
2. Transforming the dataset: The scaled dataset was fitted and transformed using the PCA model, resulting in two components (PCA1 and PCA2).
3. Explained variance: The two components captured 91% of the total variance in the dataset, retaining the majority of the information.
4. Visualizing clusters: the clusters were plotted in the 2D PCA space.

```
# Applying PCA to reduce the dataset to 2 dimensions for visualization
pca = PCA(n_components=2, random_state=random)
pca_result = pca.fit_transform(X)
```

```
# Adding the PCA components to the SP500 dataset
pca_df = sp500[['cluster']]
pca_df[['PCA1','PCA2']] = pca_result
```

```
# Explained Variance ratio
pca.explained_variance_ratio_.sum()
```

```
# Visualize the clusters in the 2D PCA space
plt.figure(figsize=(10, 8))
sns.scatterplot(
    data=pca_df, x='PCA1', y='PCA2',
    hue='cluster', palette='Set2', alpha=0.7
)
plt.title('Cluster Visualization in PCA Space')
plt.show()
```

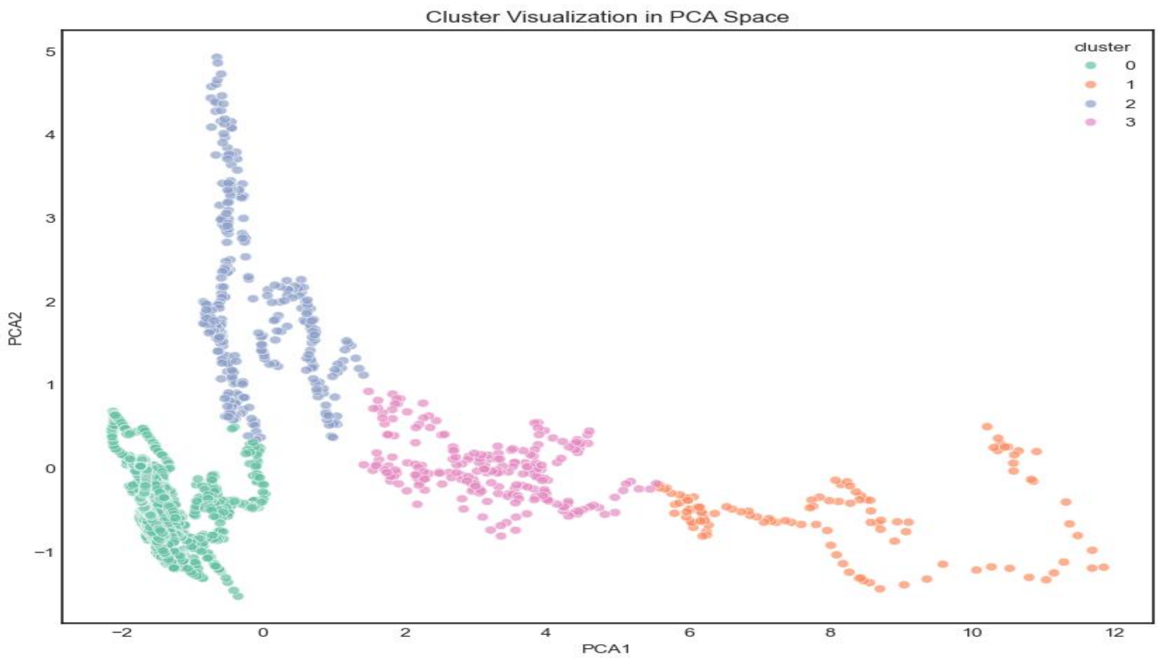

*Figure **Error! No text of specified style in document.**.17 2D visualization of clusters*

1.4.2 Agglomerative Clustering

This is also known as a bottom-up approach, it doesn't require a pre-specified number of clusters. It treats each data point as an individual cluster at the start, then progressively agglomerate pairs until all data points have been merged into a single cluster.

Determining the optimal number of clusters: A dendrogram was constructed to visualize the hierarchical clustering process showing how individual data points are linked. Wards linkage which uses Euclidean distance to measure dissimilarity between clusters was used to minimize the within-cluster variance.

```
# creating dendrogram to find optimal number of cluster
plt.figure(figsize=(15,6))
dendrogram(linkage(X,method='ward'))
plt.title('Dendrogram')
plt.xlabel('Period')
plt.ylabel('Euclidean distances')
```

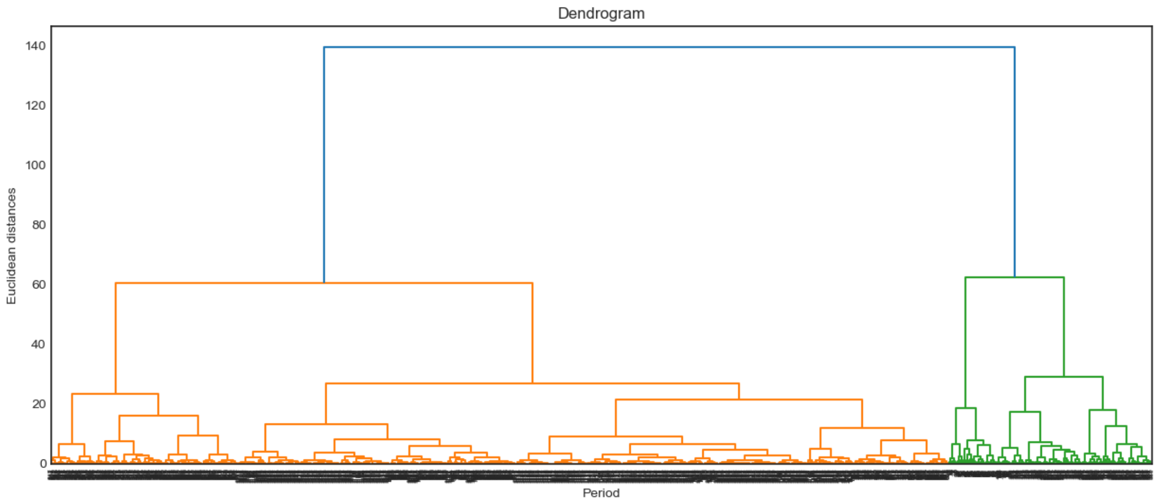

*Figure **Error! No text of specified style in document.**.18 A dendrogram showing how each datapoint are grouped*

Hyperparameter tuning:

- **n_clusters:** 4 was chosen as the number of clusters based on the dendrogram.
- **linkage:** Ward's method was selected for its ability to minimize within-cluster variance.

```
# Fitting the Hierarchical cluster
hc = AgglomerativeClustering(n_clusters=4,metric='euclidean',linkage='ward')
hc.fit(X)
```

```
AgglomerativeClustering
AgglomerativeClustering(n_clusters=4)
```

*Figure **Error! No text of specified style in document.**.19 Fitting the Agglomerative clustering algorithm with 4 number of clusters*

1.5 Result Analysis and Discussion

1.5.1 Performance Metrics

The silhouette score was used in evaluating the quality of the clustering by measuring how similar each data point cluster is to other clusters. The score ranges from +1 (well-separated clusters) to -1 (misclassified clusters). A value close to 0 suggests clusters are overlapping.

- K-Means Clustering obtained a silhouette score of 0.542, indicating well-separated clusters
- Agglomerative Clustering obtained a slightly lower score of 0.535.

```
# Silhouette score
silhouette_score(X,kmeans.labels_)
```

```
0.5417040177666343
```

*Figure **Error! No text of specified style in document.**.20 Silhouette score for the K-Means model*

```
# Silhouette score
silhouette_score(X,hc.labels_)
```

```
0.5357933976598958
```

*Figure **Error! No text of specified style in document.**.21 Silhouette score for the Agglomerative clustering model*

1.5.2 Model Comparison

Both Models are relatively similar in terms of performance with a negligible difference in their silhouette scores. The main difference is the method by which both models were visualized. Using PCA the K-means clusters were more distinctly separated. The agglomerative clusters results provided a hierarchical view of the data in the form of a dendrogram revealing how each cluster is formed. The K-means algorithm was chosen as the better model only because it had a slightly higher silhouette score.

1.5.3 Results

The K-Means algorithm generated 4 clusters based on the elbow method, with a silhouette score of 0.54, indicating clusters are well split. The cluster distribution shows that cluster 0 represented 64.2% of the observations, while clusters 1, 2, &3 represented 6.2%, 17.3% & 12.2% respectively.

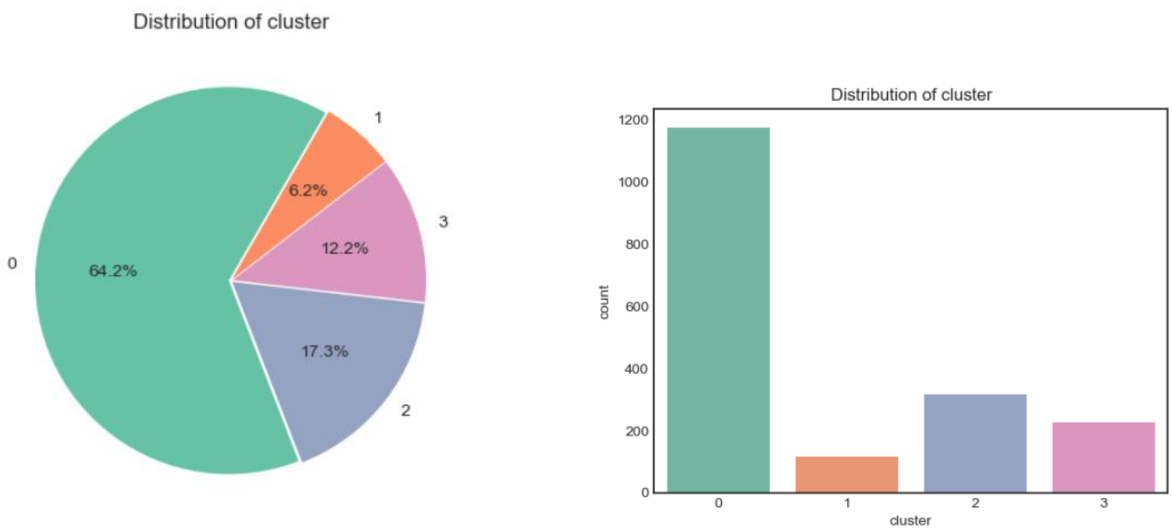

*Figure **Error! No text of specified style in document.**.22 Pie Chart and bar chart showing cluster distribution*

1.5.3.1 Cluster Summary

The cluster characteristics of K-Means can be determined by analysing either the cluster centroids extracted from the K-Means algorithm or by calculating the mean values of the features in each cluster.

```
Kmeans_centroids = pd.DataFrame(kmeans.cluster_centers_, columns=X.columns)
Kmeans_centroids
```

	SP500	Dividend	Earnings	Consumer Price Index	Long Interest Rate	Real Price	Real Dividend	Real Earnings	PE10
0	-0.445398	-0.493665	-0.472250	-0.621699	-0.379780	-0.498298	-0.537724	-0.539292	-0.233394
1	3.233690	3.268575	3.108803	2.410372	-0.992399	2.995180	3.067765	2.817408	1.611208
2	-0.216308	-0.034828	-0.102440	0.306276	1.784861	-0.094950	0.245617	0.188530	-0.588275
3	0.996750	0.975355	1.040019	1.601002	-0.028686	1.223880	0.911786	1.128214	1.236802

*Figure **Error! No text of specified style in document.**.23 Showing cluster centroids for each cluster in the standardized data*

The cluster centroids from the algorithm might be harder to interpret because it is scaled, so a heatmap visualization using the mean values of the features in each cluster might generate more insight.

```
# Visualizing cluster summary
cluster_summary = sp500.drop('Date',axis=1).groupby('cluster').mean()
```

	cluster	SP500	Dividend	Earnings	Consumer Price Index	Long Interest Rate	Real Price	Real Dividend	Real Earnings	PE10
0	0	17.763311	0.746987	1.257025	15.455540	3.615047	287.522145	12.980987	20.872255	19.148017
1	1	2929.150593	52.072982	125.059474	256.765351	2.206754	3443.193509	61.690965	147.230088	32.291930
2	2	199.049905	7.006625	14.041924	89.309148	8.591136	651.867697	23.563880	48.270000	16.619274
3	3	1158.984241	20.787946	53.538497	192.350982	4.422143	1843.172277	32.563795	83.642991	29.624063

*Figure **Error! No text of specified style in document.**.24 Showing Cluster mean for each cluster*

```
# Heatmap of cluster summary
plt.figure(figsize=(12, 8))
sns.heatmap(cluster_summary.T, cmap='Set2', annot=True)
plt.title('Cluster Means')
plt.show()
```

*Figure **Error! No text of specified style in document.**.25 Code to visualize cluster means*

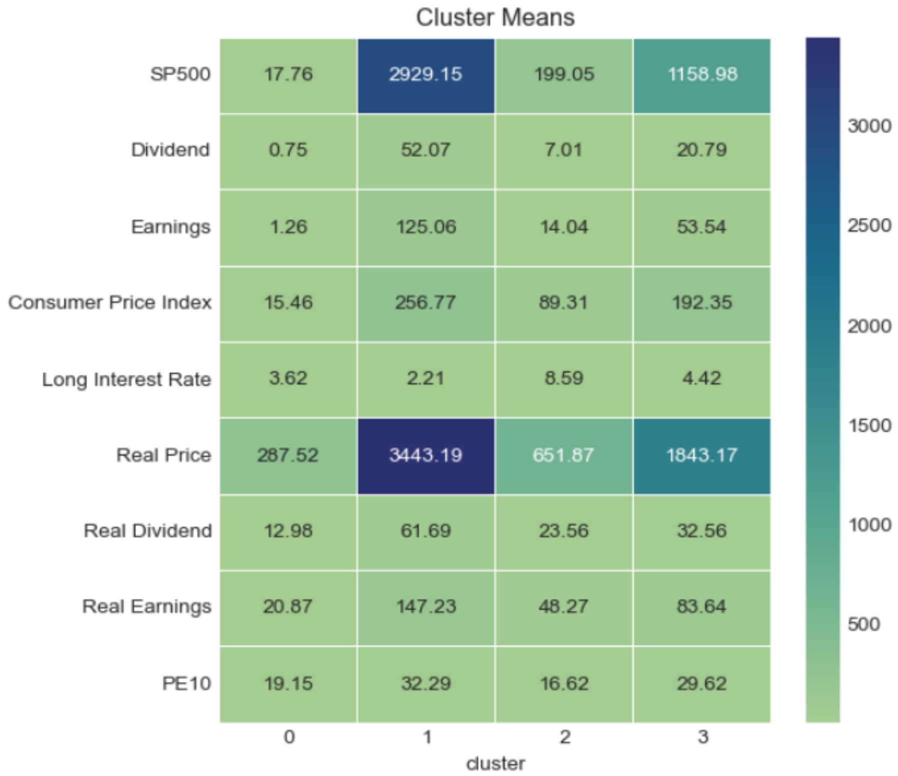

*Figure **Error! No text of specified style in document.**.26 Heatmap showing cluster means*

1.5.3.2 Cluster Interpretations

Cluster 0: Low Market Activity

Characteristics:

- Very low SP500, Dividends, Earnings, Real price and PE10.
- Moderate-low interest rates and CPI suggest stable but unremarkable economic conditions.

Interpretation:

- Likely early periods or economic downturn periods (e.g. the great depression).
- Limited growth opportunities for businesses and investors.

Cluster 1: Economic Boom and Bull Markets

Characteristics:

- Extremely high SP500, Dividends, Earnings, Real Price, and PE10.
- Low interest rate and high CPI.

Interpretation:

- Strong Bull markets.
- Lots of opportunities for businesses and investors.

Cluster 2: Transition Periods

Characteristics:

- Low-moderate SP500, dividends and earnings.

- Very high interest rate and moderate CPI, indicating periods of inflation control.

Interpretation:

- Likely reflects transitional or recovery periods.
- Businesses may experience low growth due to high interest rates.

Cluster 3: Stable Growth

Characteristics:

- Moderate-high SP500, dividends and earnings and real price.
- Moderate interest rate and balanced CPI, reflecting stability.

Interpretation:

- Periods of balanced economic growth.
- Ideal for long-term business planning.

1.5.4 Business Implications

- Market Segmentation: The clusters can be used for market segmentation, enabling investors and businesses to categorize market periods into phases such as growth and stability.
- Risk Management: Insights gained from this clustering can help in mitigating risk by identifying potential periods of market instability.
- Business Planning: Financial institutions and asset managers can use these clusters to optimize investment portfolios based on market conditions.

1.6 Conclusion

The application of **K-Means** and **Agglomerative Clustering** to the S&P 500 dataset has successfully addressed the research question by uncovering meaning patterns and revealing insights into market behaviour. While both algorithms identified similar clusters, K-Means provided clearer separation, making it more suitable for straightforward clustering tasks.

This study was validated by strong performance metrics (Silhouette score of 0.54) and cluster characteristics which align with established market behaviours. Clustering has proven to be effective for segmenting the S&P500 data, offering actionable recommendations for financial planning and investment strategies.

1.6.1 Actionable Recommendations

- Cluster 0: Focus on strategies that hedge against stagnation, such as bond investments or defensive stocks.
- Cluster 1: Prioritize growth stocks, IPOs, and sectors benefiting from bullish trends.
- Cluster 2: Adapt to inflationary environments with commodities or high-yield investments.
- Cluster 3: Focus on balanced portfolios to benefit from stable growth.

In summary, the clustering results provide a solid foundation for financial decision-making, enabling firms to better understand market dynamics, segment periods of market activity, and tailor strategies to specific conditions. By leveraging these insights, organizations can improve performance, optimize investment portfolios, and adapt to changing market environments with greater confidence.