A blue and white wavy lines

Description automatically generated

Big Data Analytics:

Assignment 2

COMP 336: Big Data Analytics

Prepared by

Matthew Paver

201582813

Table of Contents

[Introduction 2](#_Toc183218461)

[Dataset Description 2](#_Toc183218462)

[Methodology 2](#_Toc183218463)

[Decomposition of Tasks 3](#_Toc183218464)

[Task 1: Data Loading 3](#_Toc183218465)

[Task 2: Stock Names Identification 3](#_Toc183218466)

[Task 3: Filtering Stock Names 4](#_Toc183218467)

[Task 4: Weeks with More Than 100 Data Points 4](#_Toc183218468)

[Task 5: Close Value DataFrame 5](#_Toc183218469)

[Task 6: Returns DataFrame 6](#_Toc183218470)

[Task 7: PCA 6](#_Toc183218471)

[Task 8: Explained Variance Ratios 7](#_Toc183218472)

[Task 9: Cumulative Variance 8](#_Toc183218473)

[Task 10: Normalised PCA 9](#_Toc183218474)

[Interpretation of PCA Results 11](#_Toc183218475)

[Summary 11](#_Toc183218476)

Introduction

This report examines past stock market data of the S&P 500 index through Principal Component Analysis (PCA). The aim is to simplify the dataset while preserving its variations and revealing market patterns and individual stock behaviours.

Principal Component Analysis (PCA) is useful in determining the factors that impact stock performance, which is an aspect in activities such as risk assessment and optimising investment portfolios.

Dataset Description

The dataset contains historical daily market data for S&P 500 stocks, with seven attributes:

* **Date:** The date of observation.
* **Open, High, Low, Close:** Stock prices at different points in the trading day.
* **Volume:** Total traded volume.
* **Name:** Stock ticker symbol.

This analysis focuses on the Date, Close, and Name columns for PCA.

Methodology

The project was implemented in Python 3.7 using pandas for data manipulation and scikit-learn for Principle Component Analysis (PCA). The dataset was pre-processed by filtering stocks and dates to ensure consistency. PCA was performed to reduce dimensionality and identify principal components. Normalisation was applied to assess its impact on PCA.

When executing the code, please ensure *stockdata.csv* is in the same directory for the code to run successfully.

Decomposition of Tasks

## Task 1: Data Loading

The dataset was loaded into a pandas DataFrame, as shown below:

**Code:**

# Task 1: Load the data

# Change the current working directory to the directory where the script is located

script\_dir = os.path.dirname(os.path.abspath(\_\_file\_\_))

file\_path = os.path.join(script\_dir, "stockdata.csv")

data = pd.read\_csv(file\_path)

print("Task 1: Data loaded successfully into a DataFrame")

print(data.head()) # Display the first few rows of the DataFrame

**Output:**

The dataset was successfully loaded and previewed:

*A close-up of numbers

Description automatically generated*

*--------------------------------------------------------------------------------------------------------------*

## Task 2: Stock Names Identification

* **2a:** All stock names were identified and sorted alphabetically.
* **2b:** Total unique stock names: **500**.
* **2c:**
  + First 5 names: ['A', 'AAPL', 'ABBV', 'ABNB', 'ABT'].
  + Last 5: ['XYL', 'YUM', 'ZBH', 'ZBRA', 'ZTS'].

**Code:**

# Task 2: Identify and sort all stock names

all\_names = sorted(data['Name'].unique())

num\_names = len(all\_names)

first\_5\_names = all\_names[:5]

last\_5\_names = all\_names[-5:]

print(f"Task 2a: {all\_names}")

print(f"Task 2b: Total number of names: {num\_names}")

print(f"Task 2c: First 5 names: {first\_5\_names}, Last 5 names: {last\_5\_names}")

A close-up of a computer code

Description automatically generated**Output:**

*--------------------------------------------------------------------------------------------------------------*

## Task 3: Filtering Stock Names

* (3a) Removed stocks with insufficient date coverage (before 2019-11-01 or after 2022-10-31).
* (3b) Removed 10 stocks ({'OTIS', 'GEV', 'KVUE', 'SOLV', 'VLTO', 'GEHC', 'ABNB', 'CARR', 'CEG', 'PLTR'}).
* (3c) Retained 490 valid stock names for further analysis.

**Code:**

# Task 3: Filter data for names meeting the date criteria

data['date'] = pd.to\_datetime(data['date'])

min\_date = pd.Timestamp('2019-11-01')

max\_date = pd.Timestamp('2022-10-31')

grouped = data.groupby('Name')

valid\_names = [

name for name, group in grouped

if group['date'].min() <= min\_date and group['date'].max() >= max\_date

]

removed\_names = set(all\_names) - set(valid\_names)

print(f"Task 3b: Removed names: {removed\_names}")

print(f"Task 3c: Remaining valid names: {len(valid\_names)}")

**Output:**



*--------------------------------------------------------------------------------------------------------------*

## Task 4: Weeks with More Than 100 Data Points

* (4a) Found dates common to all 490 valid stocks.
* (4b) Filtered these dates to the range 2019-11-01 to 2022-10-31.
* (4c) Remaining dates: **755**.
* (4d)
  + First 5 dates: ['2019-11-01', '2019-11-04', '2019-11-05', '2019-11-06', '2019-11-07'].
  + Last 5 dates: ['2022-10-25', '2022-10-26', '2022-10-27', '2022-10-28', '2022-10-31'].

To maintain consistency in the analysis process for all stock data sets, keep those dates shared by all stocks within the time frame of November 1st 2019, to October 31st 2022, inclusive. Initially, in the datasets, timestamps were overly precise with details such as T00 hours and multiple zeros, which made them appear cluttered and hard to read at a glance; converting them into the more standard YYYY-MM-DD format improved readability significantly without changing any of the actual data. Furthermore, an error-handling mechanism was put in place to ensure that if no dates were found to be shared among all stocks after filtering, precautions were taken beforehand to prevent any disruptions in the analysis process.

**Code:**

# Task 4: Identify common dates and filter

filtered\_data = data[data['Name'].isin(valid\_names)]

# Identify common dates where all valid stocks have data

common\_dates = (

filtered\_data.groupby('date')['Name']

.nunique()

.reset\_index()

)

# Ensure 'common\_dates' is in datetime format

common\_dates['date'] = pd.to\_datetime(common\_dates['date'])

# Filter dates where all valid stocks have data

valid\_date\_count = len(valid\_names)

common\_dates = common\_dates[common\_dates['Name'] == valid\_date\_count]['date']

# Filter dates based on the required range

filtered\_dates = common\_dates[

(common\_dates >= min\_date) & (common\_dates <= max\_date)

]

# Check if filtered\_dates is empty

if filtered\_dates.empty:

print("Warning: No common dates found after filtering!")

# Convert dates to a more readable format

filtered\_dates = filtered\_dates.dt.strftime('%Y-%m-%d')

# Calculate remaining dates and extract first/last 5 dates

num\_remaining\_dates = len(filtered\_dates)

first\_5\_dates = filtered\_dates[:5]

last\_5\_dates = filtered\_dates[-5:]

# Print results

print(f"Task 4c: Remaining dates: {num\_remaining\_dates}")

print(f"Task 4d: First 5 dates: {list(first\_5\_dates)}, Last 5 dates: {list(last\_5\_dates)}")

**Output:**

****

*--------------------------------------------------------------------------------------------------------------*

## Task 5: Close Value DataFrame

Constructed a DataFrame containing the close prices for each of the 490 stocks across 755 valid dates.

**Code:**

# Task 5: Create a DataFrame of 'close' values

pivot\_data = filtered\_data.pivot(index='date', columns='Name', values='close')

pivot\_data = pivot\_data.loc[filtered\_dates]

print("Task 5b: Close value DataFrame:")

print(pivot\_data)

**Output:**

**A table of numbers with numbers

Description automatically generated**

*--------------------------------------------------------------------------------------------------------------*

## Task 6: Returns DataFrame

The returns DataFrame calculates daily percentage changes in stock prices, excluding the first day. This is due to the NaN being the first row

**Code:**

# Task 6: Calculate returns

returns = pivot\_data.pct\_change().dropna()

print("Task 6b: Returns DataFrame:")

print(returns)

**Output:**

A number of numbers on a white background

Description automatically generated

*--------------------------------------------------------------------------------------------------------------*

## Task 7: PCA

* Calculated the principal components of the returns DataFrame.
* Extracted the top 5 principal components (ranked by eigenvalues).

**Code:**

# Task 7: Perform PCA

pca = PCA()

pca.fit(returns.fillna(0))

top\_5\_components = pca.components\_[:5]

print("Task 7b: Top 5 principal components by eigenvalue:")

print(top\_5\_components)

**Output:**

A number of numbers on a white background

Description automatically generated

*--------------------------------------------------------------------------------------------------------------*

## Task 8: Explained Variance Ratios

* (8a) Extracted explained variance ratios from PCA.
* (8b) First principal component explains 43.65% of variance.
* (8c) Plotted the first 20 explained variance ratios.
* (8d) Identified the elbow point (~5 components).

The explained variance plot shows diminishing returns in variance explained:

**Code:**

# Task 8: Explained variance ratio and plot

explained\_variance\_ratio = pca.explained\_variance\_ratio\_

first\_component\_variance = explained\_variance\_ratio[0] \* 100

print(f"Task 8b: Variance explained by the first component: {first\_component\_variance:.2f}%")

plt.figure()

plt.bar(range(1, 21), explained\_variance\_ratio[:20])

plt.xlabel('Principal Component')

plt.ylabel('Explained Variance Ratio')

plt.title('Top 20 Explained Variance Ratios')

plt.axvline(x=5, color='r', linestyle='--', label='Elbow')

plt.legend()

plt.show()

**Output:**

**A graph of a graph

Description automatically generated**

*--------------------------------------------------------------------------------------------------------------*

## Task 9: Cumulative Variance

* (9a) Calculated cumulative variance ratios using numpy.cumsum.
* (9b) Plotted cumulative variance for all components.
* (9c) Identified that 195 components are required to explain **95%** of cumulative variance.

**Code:**

# Task 9: Cumulative variance plot

cumulative\_variance = np.cumsum(explained\_variance\_ratio)

plt.figure()

plt.plot(range(1, len(cumulative\_variance) + 1), cumulative\_variance, marker='o')

plt.xlabel('Principal Component')

plt.ylabel('Cumulative Variance Ratio')

plt.title('Cumulative Variance')

threshold\_index = np.argmax(cumulative\_variance >= 0.95) + 1

plt.axvline(x=threshold\_index, color='r', linestyle='--', label='95% Threshold')

plt.legend()

plt.show()

**Output:**

A graph with a curved line

Description automatically generated

*--------------------------------------------------------------------------------------------------------------*

## Task 10: Normalised PCA

The returns dataset was normalised to have zero mean and unit variance to prevent stocks with large price ranges (e.g., Amazon, Apple) from dominating PCA results. Normalisation ensures that each stock is given equal weight for each stock, in the PCA process, leading to an analysis that captures broader trends.

The amount of elements needed to account for 95% of the variance increased from 195 (without normalisation) to 216. This trend is predictable as normalisation distributes the variance uniformly among elements and lessens the influence of the few components.

**Code:**

# Task 10: Normalise data and repeat PCA

normalised\_returns = (returns - returns.mean()) / returns.std()

pca\_normalised = PCA()

pca\_normalised.fit(normalised\_returns.fillna(0))

explained\_variance\_normalised = pca\_normalised.explained\_variance\_ratio\_

cumulative\_variance\_normalised = np.cumsum(explained\_variance\_normalised)

threshold\_index\_normalised = np.argmax(cumulative\_variance\_normalised >= 0.95) + 1

# Repeat plots for normalised data

plt.figure()

plt.bar(range(1, 21), explained\_variance\_normalised[:20])

plt.xlabel('Principal Component')

plt.ylabel('Explained Variance Ratio')

plt.title('Top 20 Explained Variance Ratios (normalised Data)')

plt.axvline(x=5, color='r', linestyle='--', label='Elbow')

plt.legend()

plt.show()

plt.figure()

plt.plot(range(1, len(cumulative\_variance\_normalised) + 1), cumulative\_variance\_normalised, marker='o')

plt.xlabel('Principal Component')

plt.ylabel('Cumulative Variance Ratio')

plt.title('Cumulative Variance (normalised Data)')

plt.axvline(x=threshold\_index\_normalised, color='r', linestyle='--', label='95% Threshold')

plt.legend()

plt.show()

# Display results for cumulative variance

print(f"Task 9c: Principal component for 95% variance: {threshold\_index}")

print(f"Task 10d: Principal component for 95% variance (normalised): {threshold\_index\_normalised}")

**Output:**

A graph with a red line

Description automatically generated

A blue curve with a red line

Description automatically generated

* The PCA analysis of normalised data highlights that individual stocks contribute more evenly to the overall variance.
* The normalised results benefit financial models that rely on relative importance rather than absolute scale.

*--------------------------------------------------------------------------------------------------------------*

Interpretation of PCA Results

The PCA results highlight the following:

* The first principal component explains 43.65% of the variance, which reflects that there are broad market-wide factors influencing most stocks. This highlights the impact of market factors that affect various stocks, such as investor sentiment towards specific stocks.
* Identification of the Elbow Point: The elbow point, typically around five components, marks the point of diminishing returns. Any additional components contribute minimally to explaining variance. This ultimately simplifies complexity while preserving information.
* The effects of normalisation include an increase in the required components for achieving 95% variance from 195 to 216 due to a balanced distribution of variance among components that prevents any individual stock from overly influencing the analysis.

The findings showcase the ability of PCA to uncover overarching trends and patterns in the market that are useful for analysis.

Summary

From this S&P 500 data, the key findings consist of

* The first component explains 43.65% of the variance.
* 195 components suffice for 95% cumulative variance, increasing to 216 after normalisation.