UNIVERSITY OF LIVERPOOL

Big Data Analytics: Assignment 2

COMP 336: Big Data Analytics

Prepared by
Matthew Paver
201582813

Table of Contents

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:

```
Task 1: Data loaded successfully into a DataFrame
         date    open   high    low  close   volume Name
0  2019-07-02  73.009  75.61  76.18  75.25  1877000    A
1  2019-07-03  74.158  76.80  76.93  75.70  1153100    A
2  2019-07-05  73.144  75.75  76.47  75.30  1493100    A
3  2019-07-08  71.300  73.84  75.99  73.50  2217400    A
4  2019-07-09  70.923  73.45  74.08  73.11  2092200    A
```

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}")
```

Output:

```
Task 2a: ['A', 'AAPL', 'ABBV', 'ABNB', 'ABT', 'ACGL', 'ACN', 'ADBE', 'ADI', 'ADM', 'ADP', 'ADSK', 'AEE', 'AEP', 'AES',
D', 'AME', 'AMGN', 'AMP', 'AMT', 'AMZN', 'ANET', 'ANSS', 'AON', 'AOS', 'APA', 'APD', 'APH', 'APTV', 'ARE', 'ATO', 'AVB'
Y', 'BDX', 'BEN', 'BG', 'BIIB', 'BK', 'BKNG', 'BKR', 'BLDR', 'BLK', 'BMY', 'BR', 'BRO', 'BSX', 'BWA', 'BX', 'BXP', 'C',
, 'CEG', 'CF', 'CFG', 'CHD', 'CHRW', 'CHTR', 'CI', 'CINF', 'CL', 'CLX', 'CMCSA', 'CME', 'CMG', 'CMI', 'CMS', 'CNC', 'CN
CRWD', 'CSCO', 'CSGP', 'CSX', 'CTAS', 'CTLT', 'CTRA', 'CTSH', 'CTVA', 'CVS', 'CVX', 'CZR', 'D', 'DAL', 'DAY', 'DD', 'DE
V', 'DOW', 'DPZ', 'DRI', 'DTE', 'DUK', 'DVA', 'DVN', 'DXCM', 'EA', 'EBAY', 'ECL', 'ED', 'EFX', 'EG', 'EIX', 'EL', 'ELV'
N', 'ETR', 'EVRG', 'EW', 'EXC', 'EXPD', 'EXPE', 'EXR', 'F', 'FANG', 'FAST', 'FCX', 'FDS', 'FDX', 'FE', 'FFIV', 'FI', 'F
, 'GE', 'GEHC', 'GEN', 'GEV', 'GILD', 'GIS', 'GL', 'GLW', 'GM', 'GNRC', 'GOOG', 'GOOGL', 'GPC', 'GPN', 'GRMN', 'GS', 'G
PE', 'HPQ', 'HRL', 'HSIC', 'HST', 'HSY', 'HUBB', 'HUM', 'HWM', 'IBM', 'ICE', 'IDXX', 'IEX', 'IFF', 'INCY', 'INTC', 'INT
 'JBL', 'JCI', 'JKHY', 'JNJ', 'JNPR', 'JPM', 'K', 'KDP', 'KEY', 'KEYS', 'KHC', 'KIM', 'KKR', 'KLAC', 'KMB', 'KMI', 'KMX
LNT', 'LOW', 'LRCX', 'LULU', 'LUV', 'LVS', 'LW', 'LYB', 'LYV', 'MA', 'MAA', 'MAR', 'MAS', 'MCD', 'MCHP', 'MCK', 'MCO',
', 'MO', 'MOH', 'MOS', 'MPC', 'MPWR', 'MRK', 'MRNA', 'MRO', 'MS', 'MSCI', 'MSFT', 'MSI', 'MTB', 'MTCH', 'MTD', 'MU', 'N
'NTAP', 'NTRS', 'NUE', 'NVDA', 'NVR', 'NWS', 'NWSA', 'NXPI', 'O', 'ODFL', 'OKE', 'OMC', 'ON', 'ORCL', 'ORLY', 'OTIS', '
, 'PGR', 'PH', 'PHM', 'PKG', 'PLD', 'PLTR', 'PM', 'PNC', 'PNR', 'PNW', 'PODD', 'POOL', 'PPG', 'PPL', 'PRU', 'PSA', 'PSX
D', 'ROK', 'ROL', 'ROP', 'ROST', 'RSG', 'RTX', 'RVTY', 'SBAC', 'SBUX', 'SCHW', 'SHW', 'SJM', 'SLB', 'SMCI', 'SNA', 'SNP
WKS', 'SYF', 'SYK', 'SYY', 'T', 'TAP', 'TDG', 'TDY', 'TECH', 'TEL', 'TER', 'TFC', 'TFX', 'TGT', 'TJX', 'TMO', 'TMUS', '
', 'TYL', 'UAL', 'UBER', 'UDR', 'UHS', 'ULTA', 'UNH', 'UNP', 'UPS', 'URI', 'USB', 'V', 'VICI', 'VLO', 'VLTO', 'VMC', 'V
'WEC', 'WELL', 'WFC', 'WM', 'WMB', 'WMT', 'WRB', 'WST', 'WTW', 'WY', 'WYNN', 'XEL', 'XOM', 'XYL', 'YUM', 'ZBH', 'ZBRA',
Task 2b: Total number of names: 500
Task 2c: First 5 names: ['A', 'AAPL', 'ABBV', 'ABNB', 'ABT'], Last 5 names: ['XYL', 'YUM', 'ZBH', 'ZBRA', 'ZTS']
```

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 3b: Removed names: {'CEG', 'KVUE', 'GEV', 'VLTO', 'SOLV', 'CARR', 'ABNB', 'OTIS', 'PLTR', 'GEHC'}
Task 3c: Remaining valid names: 490
```

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 4c: Remaining dates: 755
Task 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']
```

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:

```
Task 5b: Close value DataFrame:
Name            A     AAPL     ABBV     ABT     ACGL     ACN     ADBE     ADI     ADM     ADP     ADSK     AEE  ...     WRB
date                                                                                                           ...
2019-11-01   76.32   62.290   79.13   82.61   41.35   186.11   277.27   107.56   42.01   162.13   147.61   77.09  ...  30.813
2019-11-04   76.73   63.845   82.14   82.22   40.85   187.39   276.05   110.35   42.50   161.71   149.55   75.72  ...  30.067
2019-11-05   75.47   64.080   81.47   80.90   40.23   185.48   283.60   111.34   42.91   156.89   149.08   74.61  ...  29.964
2019-11-06   74.94   63.842   80.35   82.38   40.63   186.91   285.61   110.65   42.83   157.35   150.48   75.00  ...  30.049
2019-11-07   75.68   64.527   81.08   82.91   40.42   188.67   287.00   111.53   43.07   159.88   151.00   73.74  ...  29.987
...            ...      ...      ...     ...     ...      ...      ...      ...     ...      ...      ...     ...  ...     ...
2022-10-25  132.30  149.360  147.29   97.66   47.90   275.49   316.74   144.41   88.56   235.32   208.92   77.84  ...  45.093
2022-10-26  134.00  148.040  150.70   97.88   48.60   278.00   316.54   140.54   91.89   233.88   211.01   79.21  ...  45.807
2022-10-27  135.33  144.130  152.06   96.77   49.41   278.45   316.87   140.04   93.79   233.71   209.58   79.89  ...  46.907
2022-10-28  135.02  147.820  142.34   97.20   55.11   279.06   318.67   141.15   93.44   235.57   208.59   80.60  ...  47.533
2022-10-31  137.35  151.920  144.08   98.15   56.17   283.27   317.42   140.60   95.21   239.46   214.00   81.09  ...  48.547

[755 rows x 490 columns]
```

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:

```
Task 6b: Returns DataFrame:
Name              A      AAPL      ABBV       ABT      ACGL       ACN      ADBE       ADI       ADM  ...        WY
date                                                                                                  ...
2019-11-04  0.005372  0.024964  0.038039 -0.004721 -0.012092  0.006878 -0.004400  0.025939  0.011664  ...  0.024632
2019-11-05 -0.016421  0.003681 -0.008157 -0.016054 -0.015177 -0.010193  0.027350  0.008971  0.009647  ... -0.011686
2019-11-06 -0.007023 -0.003714 -0.013747  0.018294  0.009943  0.007710  0.007087 -0.006197 -0.001864  ... -0.004054
2019-11-07  0.009875  0.010730  0.009085  0.006434 -0.005169  0.009416  0.004867  0.007953  0.005604  ... -0.006106
2019-11-08 -0.000396 -0.004866  0.012457  0.003257 -0.002721 -0.002120  0.005157 -0.003228 -0.003947  ... -0.002389
...              ...       ...       ...       ...       ...       ...       ...       ...       ...  ...       ...
2022-10-25  0.019182  0.023014 -0.009482  0.018352 -0.010944  0.019880  0.034422  0.014258 -0.006172  ...  0.004343
2022-10-26  0.012850 -0.008838  0.023152  0.002253  0.014614  0.009111 -0.000631 -0.026799  0.037602  ...  0.018629
2022-10-27  0.009925 -0.026412  0.009025 -0.011340  0.016667  0.001619  0.001043 -0.003558  0.020677  ...  0.011757
2022-10-28 -0.002291  0.025602 -0.063922  0.004444  0.115361  0.002191  0.005681  0.007926 -0.003732  ... -0.031633
2022-10-31  0.017257  0.027736  0.012224  0.009774  0.019234  0.015086 -0.003923 -0.003897  0.018943  ...  0.008333

[754 rows x 490 columns]
```

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:

```
Task 7b: Top 5 principal components by eigenvalue:
[[-0.03053307 -0.03228756 -0.02491468 ... -0.04591686 -0.04311538
  -0.0321947 ]
 [-0.05702523 -0.06911258 -0.01543889 ... -0.00071617 -0.04360575
  -0.05302061]
 [-0.00428185  0.00599959 -0.02495018 ... -0.00225642 -0.00013562
  -0.00782042]
 [-0.0256462  -0.03204411 -0.00718046 ...  0.02355439 -0.02285197
   0.00351133]
 [ 0.01751229  0.02690375 -0.05168101 ... -0.02504652  0.049497
  -0.04116101]]
```

Task 8: Explained Variance Ratios

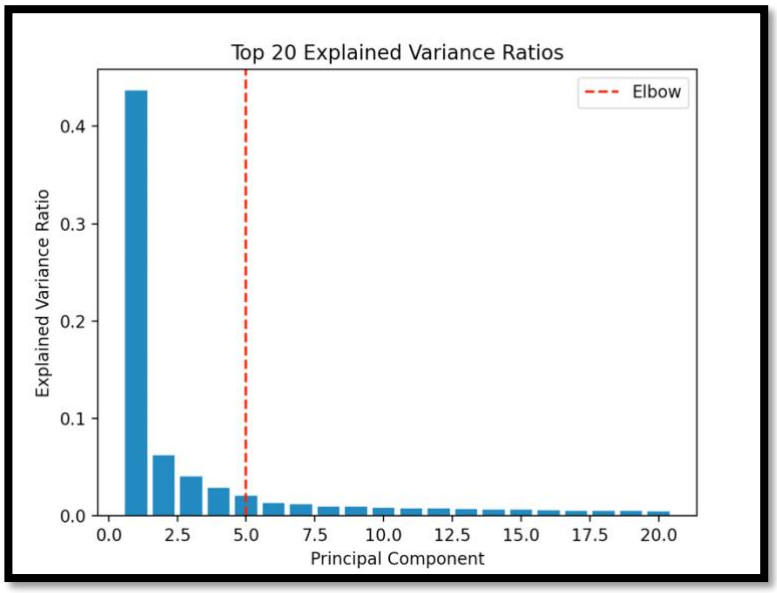
- (8a) Extracted explained variance ratios from PCA.
- (8b) First principal component explains 43.65% of variance.
- (8c) Plotted the first 20 explained variance ratios.
- (8d) Identified the elbow point (~5 components).

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:



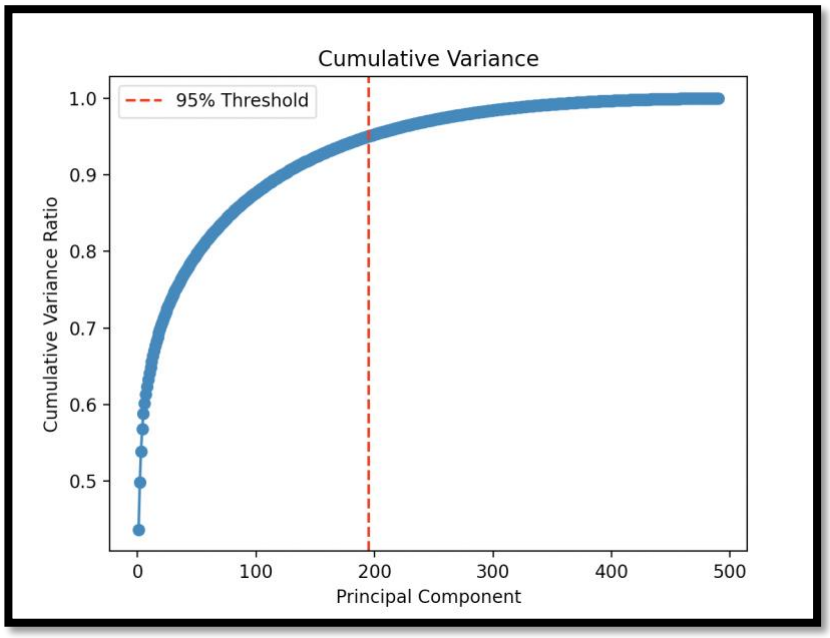
Task 9: Cumulative Variance

- (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:



Task 10: Normalised PCA

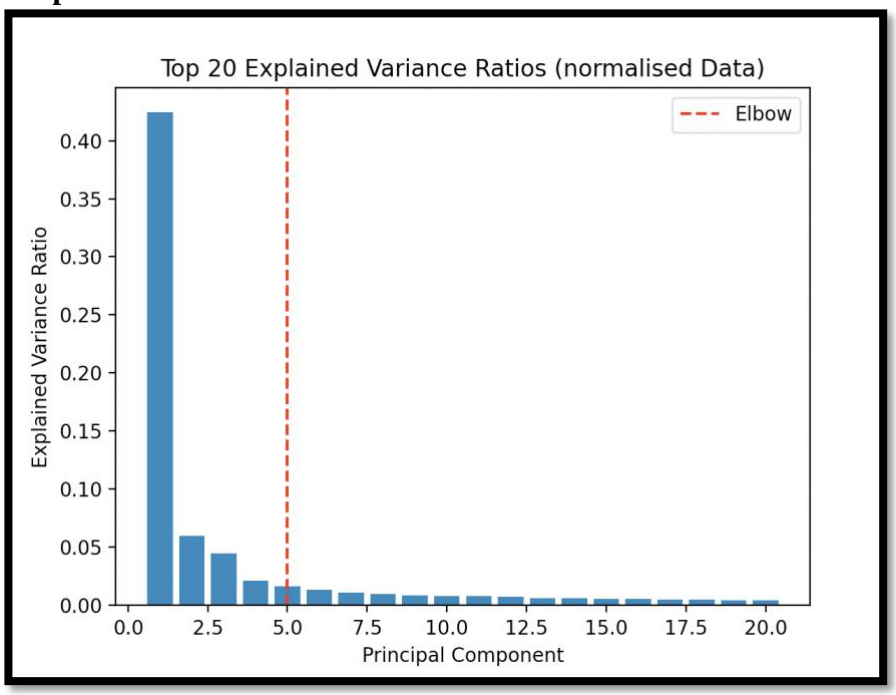
The returns dataset was normalised to have zero mean and unit variance to prevent stocks with large price ranges (e.g., Amazon, Apple) from dominating PCA results. Normalisation ensures that each stock is given equal weight for each stock, in the PCA process, leading to an analysis that captures broader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

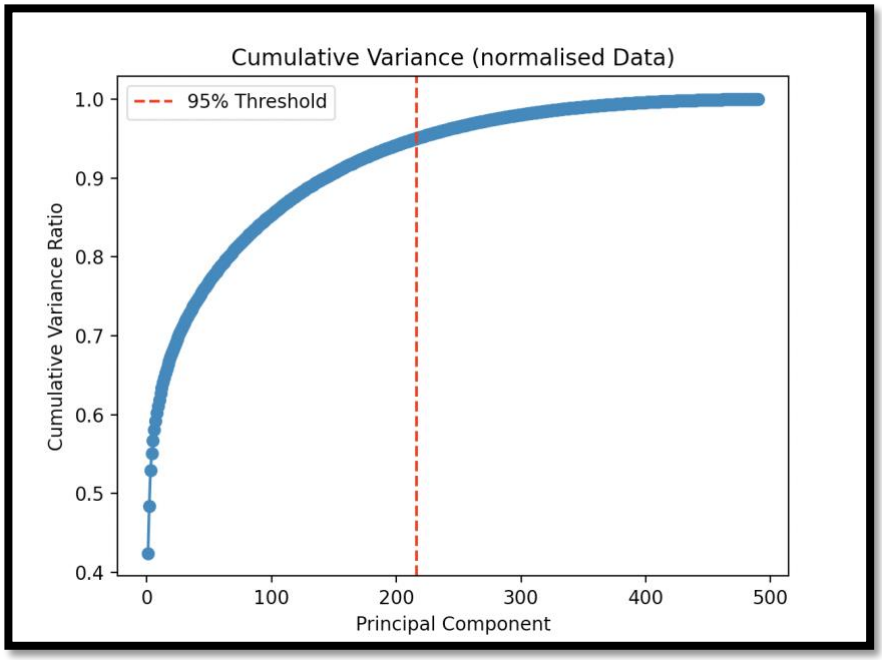
The amount of elements needed to account for 95% of the variance increased from 195 (without normalisation) to 216. This trend is predictable as normalisation distributes the variance uniformly among elements and lessens the influence of the few components.

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:





- 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.