

Large-scale financial data analysis & trend detection

By: Yohanse

Project Overview

This project is intended to create a system that helps make sense of large financial datasets, like stock prices, by spotting key trends, finding the best and worst times for gains or losses, and flagging unusual patterns. The goal is to give people better tools for making smart decisions in the financial market, whether it's understanding stock performance, catching unexpected price swings, or uncovering new investment opportunities.

The financial markets produce a massive amount of data every day, and it's not easy for analysts to sit and sift through all that information to spot important patterns, trends, or unusual activity. The project tries to take on that challenge by automating the process in two key ways:

- **Spotting Trends and Best or Worst times:** The system will track when stocks saw the biggest gains or losses, helping investors pinpoint ideal times to buy or sell, or be cautious about potential risks.
- **Catching Anomalies:** It will also flag any odd movements in stock prices, which could hint at things like market manipulation, breaking news, or unusual trading behavior that might need a closer look which might lead the investor or analyst to react to it.

The goals of the analysis are to efficiently process large financial datasets by implementing sorting techniques specifically using merge sort, detect key trends by finding periods where stock prices showed the most significant gains or losses, and identify anomalies through unusual spikes or dips in prices that could signal irregular market behavior. Additionally, the project tries to generate visual reports that summarize trends, highlight significant periods, and present detected anomalies, making the results easy to interpret for users.

I have tried to detect trends and periods of maximum gain by identifying the time periods when stocks exhibited the highest gains. This system can help investors recognize favorable investment windows. The other approach was to detect anomalies, this identifies unusual patterns in stock prices, which could signal market manipulation, significant news events, or irregular trading behavior that might require further investigation to benefit from it.

Type Specific Considerations

I used a dataset of 5 stocks covering diverse sectors to provide broader market insights. Daily closing prices were chosen as they are standard for analyzing stock performance. The stock tickers used are:

- META
- TSLA
- MSTR
- NVDA
- SPY

Algorithms Used for Processing and Analysis

1. Data Sorting (Merge Sort):

A custom merge sort organizes the data by date, preparing it for subsequent analyses. In our case the dataset was already sorted so we didn't need to sort the data but I ran it anyways and it stayed the same. For future use merge sort is provided in the project.

2. Trend Detection (2D Kadane's Algorithm):

A 2D extension of Kadane's algorithm identifies the period of maximum gain across two stocks. It detects the subarray with the highest combined gain, making it suitable for trend analysis.

3. Anomaly Detection (Closest Pair of Points):

The closest pair of points algorithm detects anomalies by finding unusually close price points that are far apart in time, indicating potential irregularities such as sharp spikes or dips.

1. Data Preprocessing

- **Functionality:** Loads the dataset, handles missing values, and sorts the data using a custom merge sort algorithm.
- **Purpose:** Ensures the time-series data is clean and sorted for accurate analysis.

2. Trend Analysis (2D Kadane's Algorithm)

- **Functionality:** Detects the period of maximum gain across selected stocks using a 2D extension of Kadane's algorithm.
- **Purpose:** Identifies time periods where stocks show the highest gains, providing insights into significant market trends.

3. Anomaly Detection (Closest Pair of Points Algorithm)

- **Functionality:** Uses the closest pair of points algorithms to detect anomalies in the stock prices, such as sharp spikes or dips.
- **Purpose:** Identifies unusual price movements that may indicate irregularities or significant market events.

4. Visualization

- **Functionality:** Plots the results of the trend analysis, individual stock trends, and detected anomalies.
- **Purpose:** Provides visual representation of the analysis to facilitate easy interpretation of the results.

Structure of the code

The code is organized into distinct functional blocks, each handling a specific aspect of the project:

1. Data Preprocessing:

- **Purpose:** Cleans the dataset, fills missing values, and sorts the data by date.
- **Functions:**
 - `merge_sort(data)`: Custom merge sort function to sort dates in the dataset.
- **Output:** A cleaned and sorted DataFrame that is ready for analysis.

2. Trend Detection:

- **Purpose:** Identifies the period of maximum gain across two selected stocks using the 2D Kadane's algorithm.
- **Functions:**
 - `kadane_2d_algorithm(matrix)`: Finds the maximum gain period in a 2D array of daily price changes.
 - `kadane_algorithm(array)`: Standard 1D Kadane's algorithm used within the 2D implementation.
- **Output:** The time range and indices where the maximum gain occurred.

3. Anomaly Detection:

- **Purpose:** Detects unusual price movements using a closest pair algorithm.
- **Functions:**
 - `distance(p1, p2)`: Calculates the Euclidean distance between two points.
 - `closest_pair(points)`: Identifies the closest pair of points in the dataset to flag potential anomalies.
- **Output:** The detected anomaly points and their distance, indicating potential irregularities in the stock prices.

4. Visualization:

- **Purpose:** Creates plots for the detected trends and anomalies, as well as individual stock trends.
- **Functions:**
 - `plt.plot()`, `plt.axvspan()`, `plt.scatter()`: Used to create various plots that visually represent the analysis.
- **Output:** Saves the generated plots as image files, like `max_gain_2d_analysis_final.png`, `max_gain_anomaly_analysis.png`, and individual stock trend plots (e.g., `TSLA_trend.png`).

Flow Chart outline

☐ Start

- **Load Dataset → Handle Missing Values → Sort Data (Merge Sort)**

☐ Trend Analysis

- **Calculate Daily Changes → Apply 2D Kadane's Algorithm → Detect Maximum Gain Period**

☐ Anomaly Detection

- **Prepare Points (Index, Price) → Find Closest Pair**

☐ Visualization

- **Plot Stock Trends → Highlight Maximum Gain Period → Mark Detected Anomalies**

☐ End

Instructions on How to Use the System

To use the system effectively follow these steps I am going to provide.

Python Environment is important to use, recommend PyCharm.

Required libraries are **Pandas and matplotlib**.

Download the dataset (5 stocks) provided into your computer. It is named multi_stock_data.csv. make sure you provide the location of the path where you saved the csv file when running the program.

```
12 file_path = "C:/CMPSC 463/multi_stock_data.csv"
```

In this example I have used C:\CMPSC 463 folder in my computer

Understanding the outputs

In the code

1. max_gain_2d_analysis_final.png: is a plot showing the period of maximum gain detected across two selected stocks

```
163 plt.savefig('C:/CMPSC 463/max_gain_2d_analysis_final.png')
```

2. max_gain_anomaly_analysis.png: is a plot showing the detected maximum gain period along with marked anomalies.

```
250 plt.savefig('C:/CMPSC 463/max_gain_anomaly_analysis.png')
```

3. {stock}_trend.png: is a plot displaying individual stock trend. Each of the tickers are inserted one by one so that each stock trend is displayed and in our case we have 5 stocks.

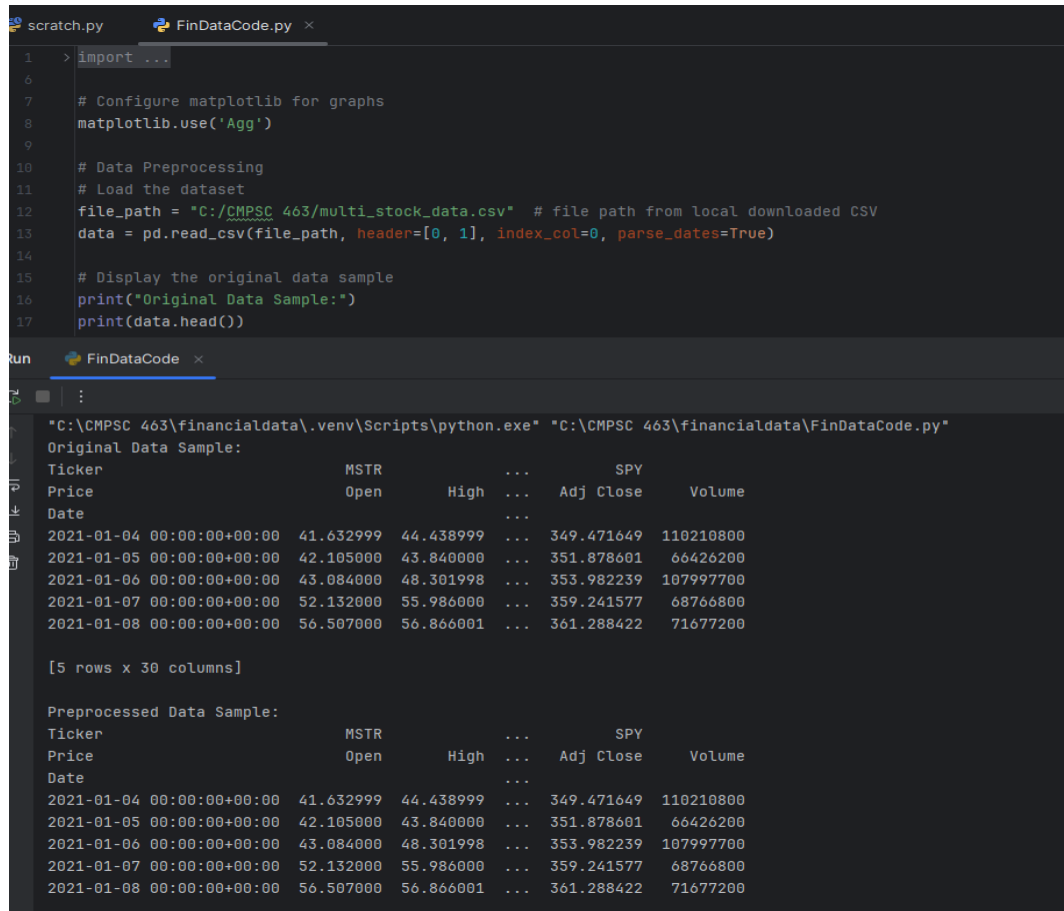
```
218 plt.savefig(f'C:/CMPSC 463/{stock}_trend.png')
```

The .png files shows the max gain, trend and anomlies of the stocks.

Verification of code functionality

Here are verification of code functionality

First view when run



The screenshot shows a Jupyter Notebook interface with two tabs: 'scratch.py' and 'FinDataCode.py'. The 'FinDataCode.py' tab is active, displaying the following Python code:

```
1 > import ...
6
7 # Configure matplotlib for graphs
8 matplotlib.use('Agg')
9
10 # Data Preprocessing
11 # Load the dataset
12 file_path = "C:/CMPSC 463/multi_stock_data.csv" # file path from local downloaded CSV
13 data = pd.read_csv(file_path, header=[0, 1], index_col=0, parse_dates=True)
14
15 # Display the original data sample
16 print("Original Data Sample:")
17 print(data.head())
```

Below the code, the 'Run' button is visible, and the output of the code execution is shown in the 'FinDataCode' tab. The output displays the 'Original Data Sample' as a table with 5 rows and 30 columns. The first 6 columns are visible, showing data for 'MSTR' and 'SPY' tickers. The data is in chronological order.

Ticker	MSTR	...	SPY
Price	Open	High	Adj Close
Date			
2021-01-04 00:00:00+00:00	41.632999	44.438999	349.471649
2021-01-05 00:00:00+00:00	42.105000	43.840000	351.878601
2021-01-06 00:00:00+00:00	43.084000	48.301998	353.982239
2021-01-07 00:00:00+00:00	52.132000	55.986000	359.241577
2021-01-08 00:00:00+00:00	56.507000	56.866001	361.288422

[5 rows x 30 columns]

The output also shows the 'Preprocessed Data Sample' with the same structure and data.

The outputs are in chronological order.

Original Data sample:

```
"C:\CMPSC 463\financialdata\.venv\Scripts\python.exe" "C:\CMPSC 463\financialdata\FinDataCode.py"
Original Data Sample:
Ticker          MSTR      ...      SPY
Price           Open      High    ...    Adj Close    Volume
Date            ...
2021-01-04 00:00:00+00:00  41.632999  44.438999  ...    349.471649  110210800
2021-01-05 00:00:00+00:00  42.105000  43.840000  ...    351.878601  66426200
2021-01-06 00:00:00+00:00  43.084000  48.301998  ...    353.982239  107997700
2021-01-07 00:00:00+00:00  52.132000  55.986000  ...    359.241577  68766800
2021-01-08 00:00:00+00:00  56.507000  56.866001  ...    361.288422  71677200

[5 rows x 30 columns]
```

Preprocessed data sample:

```
Preprocessed Data Sample:
Ticker          MSTR      ...      SPY
Price           Open      High    ...    Adj Close    Volume
Date            ...
2021-01-04 00:00:00+00:00  41.632999  44.438999  ...    349.471649  110210800
2021-01-05 00:00:00+00:00  42.105000  43.840000  ...    351.878601  66426200
2021-01-06 00:00:00+00:00  43.084000  48.301998  ...    353.982239  107997700
2021-01-07 00:00:00+00:00  52.132000  55.986000  ...    359.241577  68766800
2021-01-08 00:00:00+00:00  56.507000  56.866001  ...    361.288422  71677200

[5 rows x 30 columns]
```

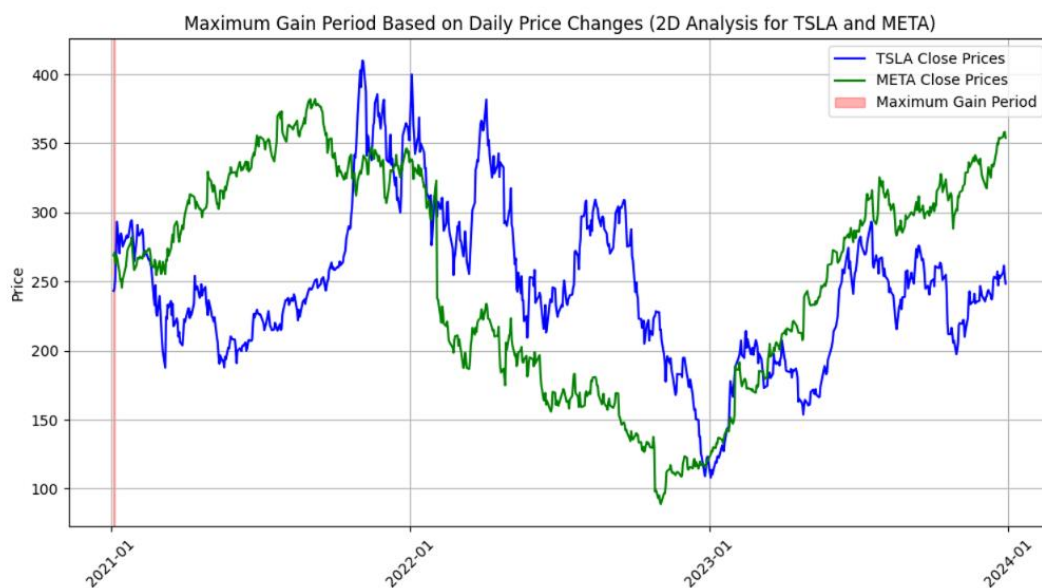
Merge sorted data sample:

```
Sorted Data Sample:
Ticker          MSTR      ...      SPY
Price           Open      High    ...    Adj Close    Volume
Date            ...
2021-01-04 00:00:00+00:00  41.632999  44.438999  ...    349.471649  110210800
2021-01-05 00:00:00+00:00  42.105000  43.840000  ...    351.878601  66426200
2021-01-06 00:00:00+00:00  43.084000  48.301998  ...    353.982239  107997700
2021-01-07 00:00:00+00:00  52.132000  55.986000  ...    359.241577  68766800
2021-01-08 00:00:00+00:00  56.507000  56.866001  ...    361.288422  71677200

[5 rows x 30 columns]
```


Maximum gain: this shows the average max price for those two stocks

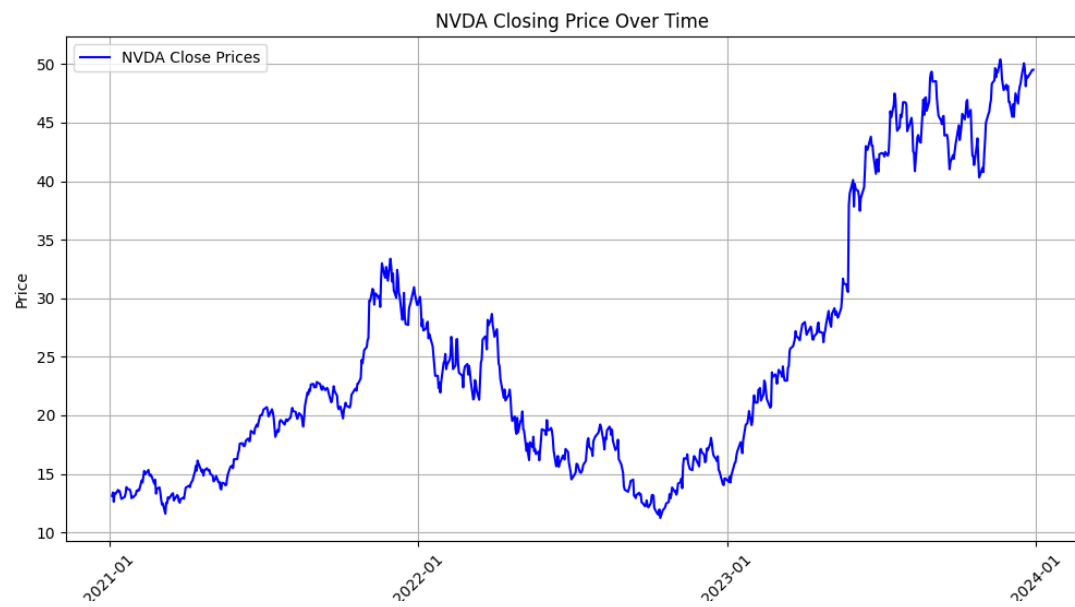
```
Maximum gain for the 2D analysis: 393.2899932861328
Time Period: From index 0 to index 1
Stocks range: From row 500 to row 750
The analysis is complete and the visualization has been saved as 'max_gain_2d_analysis_final.png'.
```



Trend for each stock, as example I will show SPY and NVDA chart separately

```
The trend for MSTR has been saved as 'MSTR_trend.png'.
The trend for META has been saved as 'META_trend.png'.
The trend for NVDA has been saved as 'NVDA_trend.png'.
The trend for TSLA has been saved as 'TSLA_trend.png'.
The trend for SPY has been saved as 'SPY_trend.png'.
```

This shows that .png files are downloaded to the intended file in the computer. When opening the .png file, it shows the trend of closing price overtime.



Discussion of findings

This project provided valuable insights into financial data analysis, especially in using algorithms and visualizations to detect trends from stock price data. One of the most rewarding aspects was learning how to create graphs that illustrate trends using just a dataset. This hands-on experience helped me understand the practical application of algorithms in financial analysis.

Implementing the 2D Kadane's algorithm was a challenging yet educational experience. It highlighted the significance of identifying periods of maximum gains and losses in stocks, which is crucial for making informed investment decisions. I encountered difficulties in extending the algorithm to two dimensions, as it required more complex data processing than traditional 1D scenarios.

One limitation I faced was plotting the maximum profit in the correct location on the graph. Despite multiple attempts, there were instances where the highlighted region did not align precisely with the calculated period of maximum gain. Given more time and effort, I believe this could be improved by refining the algorithm implementation and troubleshooting the plotting process.

The project has room for improvement, such as optimizing the plotting of the maximum gain period and incorporating more sophisticated methods for data processing. With further development, the system could become a more reliable tool for financial trend analysis.