Project 1 Report

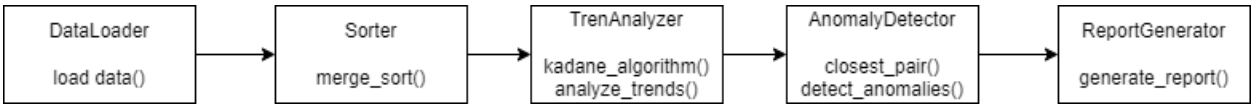
Description of the Project

The project aims to develop a system for analyzing large-scale financial datasets, such as stock prices, using Python. The system applies various divide-and-conquer algorithms to sort, analyze trends, and detect anomalies efficiently. It focuses on making financial analysis more manageable by identifying periods of maximum gain, loss, and potential market anomalies. This approach can be valuable for decision-making in finance, such as identifying optimal buy/sell periods or flagging unusual trading activity.

Structure of the code with diagram

The project is designed in a modular manner, with each key task handled by a separate class:

- **DataLoader:** Handles the retrieval of stock price data using the yfinance library. Takes stock ticker, start date, and end date as input and returns a DataFrame of stock prices.
- **Sorter:** Implements merge sort to ensure that the stock data is sorted by date before any analysis is performed. This helps in efficient analysis.
- **TrendAnalyzer:** Uses Kadane's algorithm to detect periods of maximum gain or loss in stock prices by analyzing daily changes.
- **AnomalyDetector:** Detects anomalies in stock price changes using a closest pair approach, flagging dates where significant deviations occur.
- **ReportGenerator:** Produces visual output, including trend lines and markers for detected anomalies, to summarize the analysis results.



Instructions on how to use the system

Step 1: Load Data

Replace the default stock ticker with your preferred financial data source by modifying the ticker parameter in the DataLoader class.

Step 2: Run the Analysis

Run the script to sort the data, identify the period of maximum gain or loss, and detect anomalies.

Step 3: Generate Report

The script automatically generates visual reports, including trend graphs and anomaly highlights. Customize the generate_report function in the ReportGenerator class to adjust report details or save the plots to a specific directory.

Step 4: Customize Parameters

Adjust analysis parameters such as date range or anomaly detection thresholds to refine the results.

Verification of code functionality

- **Merge Sort.** The Merge Sort algorithm was tested on the AAPL (Apple) stock price data retrieved using yfinance for the period between 2022-01-01 and 2023-01-01. The output is verified by checking that dates are sorted chronologically, ensuring that subsequent analysis operates on correctly ordered data.

- **Kadane's Algorithm.** Identifying the period of maximum gain for AAPL stock prices using TrendAnalyzer. Kadane's algorithm is applied to daily price changes to find the sub-period with the highest cumulative gain. The algorithm correctly identifies the sub-period where the stock price increase is the highest from 2022-06-16 to 2022-08-17, demonstrating its ability to pinpoint profitable periods in stock trading.

- **Closest Pair of Points.** Detecting anomalies in AAPL stock prices using AnomalyDetector. The closest pair algorithm is applied to pairs of dates and closing prices to find anomalies, defined as significant deviations between consecutive trading days. The system correctly identifies potential anomaly, such as the same price at different timestamps ('2022-06-24 00:00:00'), price: 141.66, and ('2022-06-27 00:00:00'), price: 141.66 with distance 0.0, indicating possible irregularities or market activity.

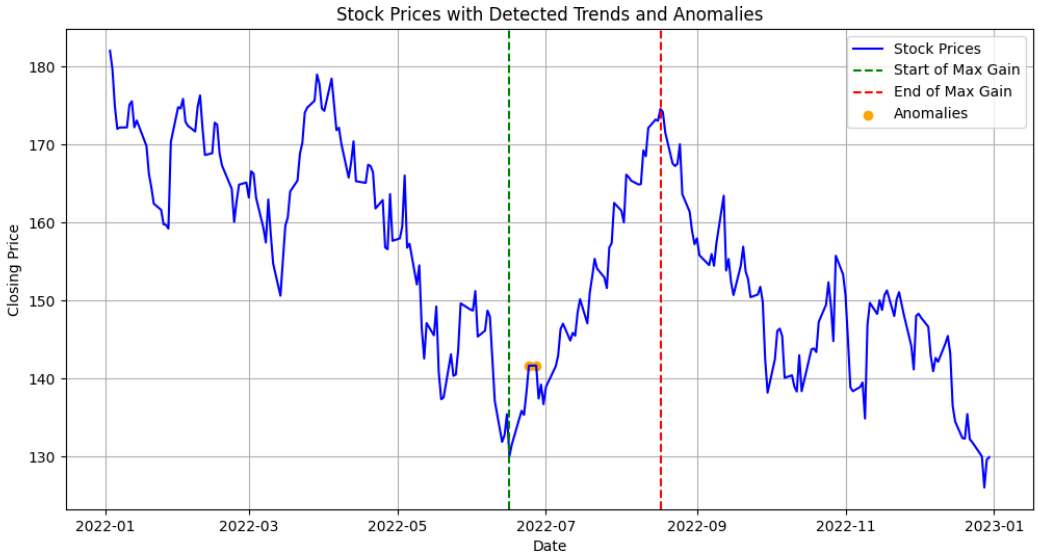
Sample Scenario:

Analyzing AAPL stock prices from 2022-01-01 to 2023-01-01.

Generating a visual report using the ReportGenerator class, displaying trends and anomalies.

The generate_report function generates line plots for stock prices and marks detected anomalies.

The graph is visually inspected to ensure that trends and anomalies are correctly highlighted, providing clear insights into the stock's performance over time.



Discussion of findings

- The project successfully detected periods of maximum growth and decline in stock prices, offering potential insights for investment decisions.
- The closest pair of points algorithm was effective in highlighting price deviations, which could indicate market anomalies or errors in data.
- The divide-and-conquer approach enabled efficient analysis of large datasets, making it suitable for high-frequency trading data.

Limitations

- The algorithm's performance may decrease with extremely large datasets due to memory constraints.
- While the divide-and-conquer techniques handled large datasets efficiently, processing times could still be improved, especially for very large datasets spanning multiple years.

Future Improvement

- Building a simple graphical user interface (GUI) for users to input stock tickers and view results directly could make the system more user-friendly.
- Integrating machine learning models for anomaly detection could help filter out insignificant anomalies, improving accuracy.