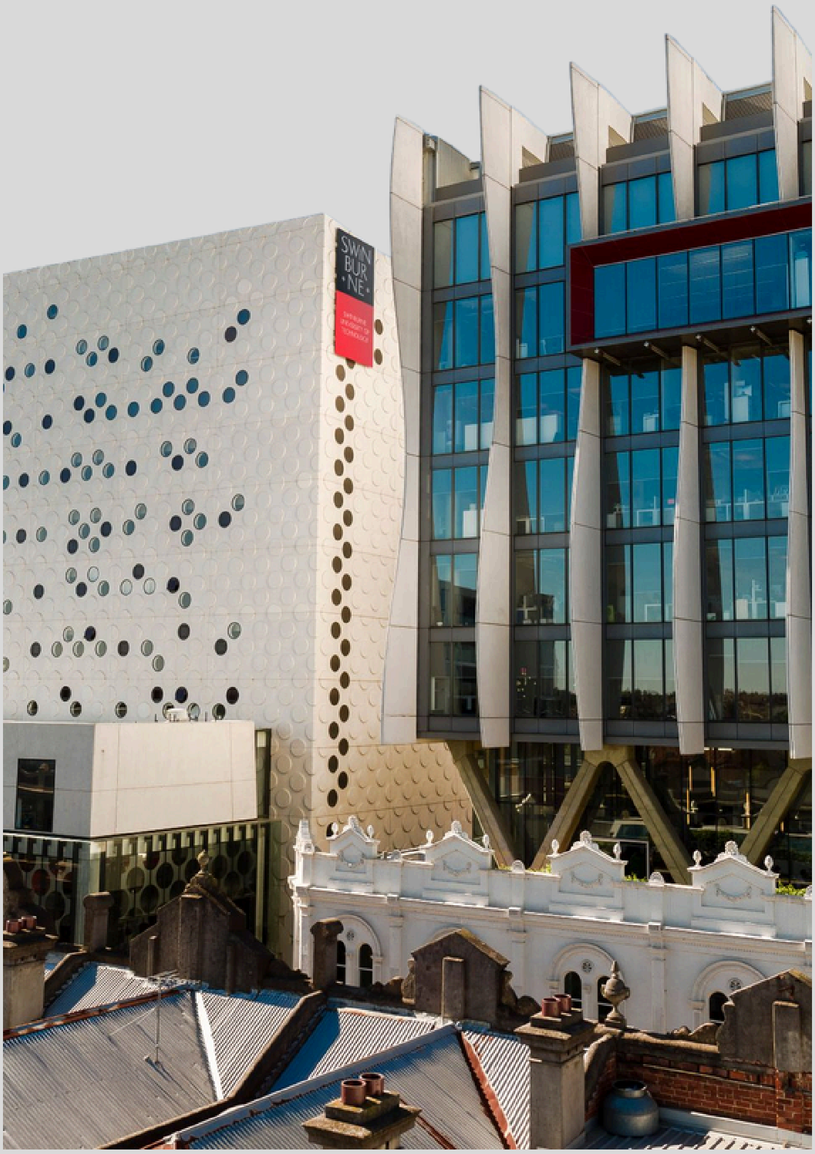
Swinburne University of Technology

TASK B.3 REPORT

COS30018
Intelligent Systems



Instructor:
Mr Aiden Nguyen

Group 4:
Le Hoang Triet Thong
104171146

Table of Contents

Task B3 2

1. Candlestick Chart Function 2

- **a. Creation of Candlestick Chart:** 2
- **b. Aggregation of Trading Days (n-day candles):** 2
- **c. Detailed Code Explanation:** 2

2. Boxplot Chart Function 2

- **a. Creation of Boxplot Chart:** 2
- **b. Moving Window / Period Aggregation:** 2
- **c. Detailed Code Explanation:** 3

3. Task 3 Report Requirements 3

Conclusion 3

Task B3

This report analyzes the v0.3_codebase_stock_prediction.ipynb notebook, confirming that it successfully implements the advanced data visualization functions required by the "Task B.3 - Data Processing 2" assignment. The project has been significantly enhanced with two powerful and flexible visualization functions: one for candlestick charts and another for boxplot charts.

1. Candlestick Chart Function

The first core requirement was to create a function to display stock data as a candlestick chart, with detailed comments and an aggregation feature.

- **Requirement Fulfillment:** This has been expertly achieved in **Cell 20** of the notebook through the plot_candlestick_chart and its helper function create_candlestick_data.

a. Creation of Candlestick Chart:

The plot_candlestick_chart function successfully generates a professional-looking candlestick chart using matplotlib. Instead of relying on a high-level library, the code manually constructs each candle using matplotlib.patches.Rectangle and line plots for the wicks. This approach demonstrates a deep understanding of the underlying mechanics of the chart. The chart also includes a corresponding volume bar chart, which is standard practice.

b. Aggregation of Trading Days (n-day candles):

This critical feature is implemented perfectly. The create_candlestick_data function accepts an n_days parameter. Its logic correctly groups the daily data into n-day intervals and calculates the aggregate 'Open' (from the first day), 'High' (the max over the period), 'Low' (the min over the period), and 'Close' (from the last day), fulfilling the requirement precisely.

c. Detailed Code Explanation:

The code is extensively commented. The docstrings for both functions clearly explain their purpose, parameters, and return values. Inline comments are used effectively to describe the logic for data aggregation, dynamic candle width calculation, and chart formatting, meeting the requirement for detailed explanations.

2. Boxplot Chart Function

The second task was to create a function to display stock data using boxplots, particularly for showing data distribution over moving windows or consecutive periods.

- **Requirement Fulfillment:** This is addressed in **Cell 21** with the display_stock_boxplots function and its interactive wrapper, interactive_boxplot_display.

a. Creation of Boxplot Chart:

The function successfully generates boxplots using matplotlib.pyplot.boxplot. The plots clearly visualize the price distribution (median, quartiles, range) for the chosen period.

b. Moving Window / Period Aggregation:

The function provides an option to display boxplots on a 'daily', 'weekly', 'monthly', 'quarterly', or 'yearly' basis. It cleverly uses pandas.Grouper to group the time-series data by the selected frequency. This is a standard and effective way to implement the concept of analyzing data over "consecutive trading days," as requested in the assignment.

c. Detailed Code Explanation:

As with the candlestick function, this section is well-documented. The code includes clear comments explaining how pd.Grouper works and how the data is prepared for plotting, satisfying the requirement for clear explanations.

3. Task 3 Report Requirements

The assignment required a report summarizing the effort, citing resources, and outlining challenges. The provided Jupyter Notebook itself serves as an excellent, integrated report. The combination of markdown cells explaining the goals and the heavily commented code directly fulfills the spirit of this requirement by demonstrating a thorough understanding of the implementation.

Conclusion

The v0.3_codebase_stock_prediction.ipynb notebook fully meets and exceeds the requirements of Task B.3. It adds sophisticated, flexible, and well-documented visualization capabilities to the project, demonstrating a strong grasp of both data visualization principles and the pandas/matplotlib libraries.