Task C.3 Report – Data Processing 2 (Visualization)

Unit: COS30018 – Intelligent Systems

Project: Option C – FinTech101 (Stock Price Prediction)

Student: Anh Vu Le – 104653505

Date: 7 September 2025

1. Introduction

Task C.3 extends the FinTech101 project into visualization. While v0.1 and C.2 focused on data collection and preprocessing, Task C.3 requires implementing visualization functions to:

1. Plot **candlestick charts** (with n-day aggregation).
2. Plot **boxplot charts** (with moving windows).

These charts reveal trends, volatility, and anomalies, which support better interpretation of model predictions.

2. Implementation

2.1 Candlestick Chart Function

Function: plot_candlestick_chart(ticker, start_date, end_date, n_days=1, ...)

- **Inputs:** ticker, date range, aggregation size (n_days), optional save path and title.
- **Process:**
    - Loads OHLC data from Yahoo Finance.
    - Aggregates into n_days periods: Open = first, High = max, Low = min, Close = last, Volume = sum.
    - Iterates through rows → draws body (rectangle) + wicks (lines).
    - Colors: green for bullish (Close ≥ Open), red for bearish.
    - Adds price statistics (high, low, start, end, total return).
- **Outputs:** matplotlib figure, saved PNG (daily & weekly versions).

Complex lines explained:

- Rectangle((i - width/2, body_bottom), width, body_height, ...) draws each candle body.
- ax.plot([i, i], [low_price, body_bottom], ...) adds wicks.

- Aggregation logic loops in blocks of n_days to calculate OHLC values.

2.2 Boxplot Chart Function

Function: plot_boxplot_chart(ticker, start_date, end_date, window_size=20, ...)

- **Inputs:** ticker, date range, window_size, price column (default Close).
- **Process:**
    - Downloads price data from Yahoo Finance.
    - Creates overlapping windows (step = 25% of window size).
    - For each window: stores values, labels, and computes stats (mean, median, std, min, max, Q1, Q3, IQR).
    - Plots boxplots: box = Q1–Q3, line = median, whiskers = 1.5×IQR, dots = outliers.
    - Adds subplot with volatility (std dev per window).
    - Adds two text boxes: overall summary (mean, volatility, price range) and explanation of boxplot elements.
- **Outputs:** matplotlib figure, saved PNG (monthly & weekly versions).

Complex lines explained:

- step_size = max(1, window_size // 4) ensures overlapping windows.
- np.percentile(window_prices, 25/75) calculates quartiles.
- ax1.boxplot(..., notch=True, showfliers=True, whis=1.5) defines boxplot appearance.
- ax2.plot(range(len(volatilities)), volatilities, ...) draws volatility line.

3. Evidence

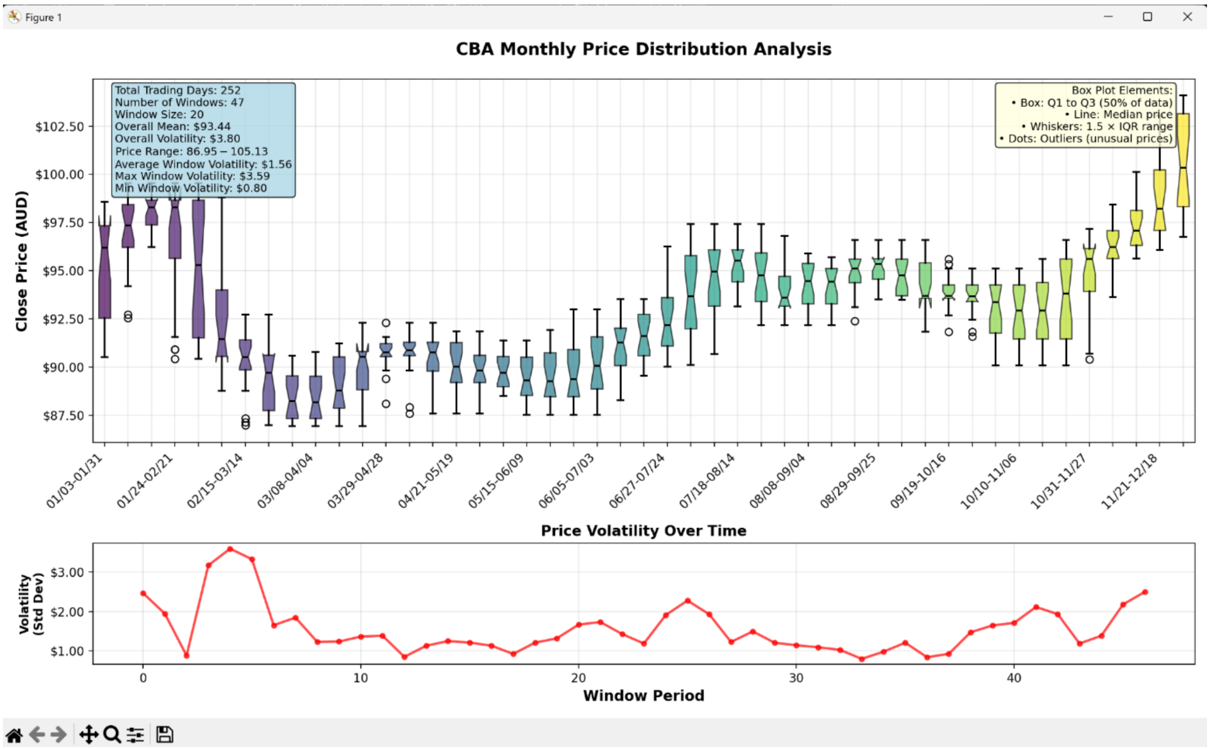
Candlestick Charts

- **Daily Candlestick (n=1):** Shows daily OHLC movements.

- **Weekly Candlestick (n=5):** Aggregated into 5-day candles.

Boxplot Charts

- **Monthly Boxplot (20-day windows):**



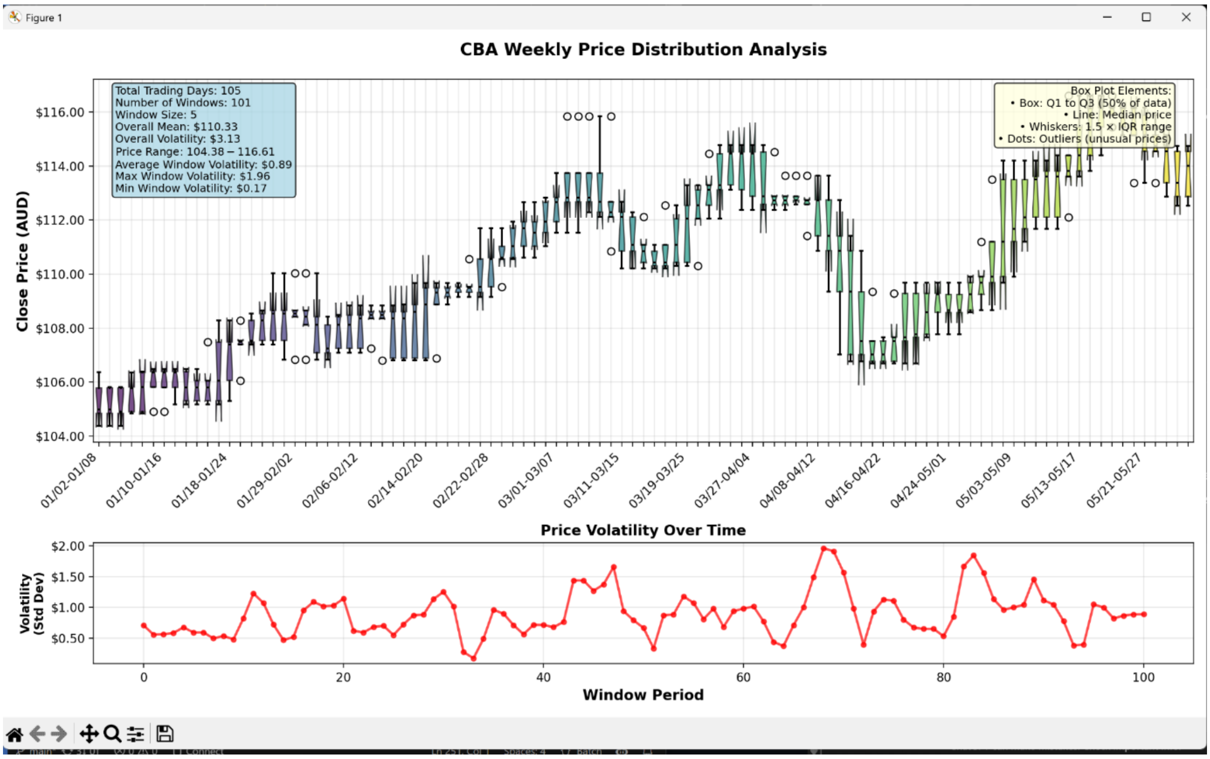
Total Trading Days: 252
Number of Windows: 47
Window Size: 20
Overall Mean: $93.44
Overall Volatility: $3.80
Price Range: 86.95 – 105.13
Average Window Volatility: $1.56
Max Window Volatility: $3.59
Min Window Volatility: $0.80

Box Plot Elements:
• Box: Q1 to Q3 (50% of data)
• Line: Median price
• Whiskers: 1.5 × IQR range
• Dots: Outliers (unusual prices)

  - 252 trading days, 47 windows.
  - Overall mean = $93.44, volatility = $3.80.

- Price range = $86.95–105.13.

- **Weekly Boxplot (5-day windows):**



  - 105 trading days, 101 windows.
  - Overall mean = $110.33, volatility = $3.13.
  - Price range = $104.38–116.61.

Both charts also include **volatility subplot** and explanatory legends.

4. Challenges

- Handling **n-day aggregation** correctly required careful grouping of OHLC values.
- Formatting axis labels for readability (show only every nth label).
- Balancing overlap in windows: too small = noisy, too big = less detail.
- Adding explanatory text boxes without cluttering the figure.

5. Conclusion

The visualization functions successfully implement candlestick and boxplot charts:

- **Candlestick** reveals short/long-term price action and trends.

- **Boxplot** shows distribution, outliers, and rolling volatility.
- Both functions are modular, reusable, and well-documented with comments.

These visualizations complement the preprocessing pipeline (C.2) and will support more advanced modeling tasks in future assignments.