COS30018
Intelligent Systems
Option B: Stock Prediction

Task B.3 – Data Processing

Name: Duc Thuan Tran
Student ID: 104330455
Tutor: Dr. Ru Jia
Tutorial: Friday 2:30 – 4:30

Table of Contents

Introduction

Continuing the enhancement of the v0.1 stock prediction project, Task B.3 introduces advanced data visualizations, including candlestick and boxplot charts, to provide deeper insights into stock market trends. These visual tools complement the LSTM model's predictions by capturing key price movements and distribution patterns.

This report outlines the improvements made in v0.1, focusing on data preprocessing, model development, and the integration of these visualizations to enrich the analysis and understanding of the model's performance.

Overview of Previous Data Processing

In the previous version of the project (v0.1), the focus was on building the foundation for a stock prediction model using four key components: main.py, data_processing.py, model_operations.py, and predictor.py.

- **main.py**: This script orchestrated the overall workflow of the project, including data loading, preprocessing, model training, and prediction. It served as the central hub that coordinated the tasks carried out by the other modules.
- **data_processing.py**: This module handled the essential data preprocessing tasks, such as loading stock market data, handling missing values, scaling features, and splitting the data into training and testing sets. This preparation ensured that the data was in a suitable format for model training.
- **model_operations.py**: This file defined the architecture and operations of the LSTM (Long Short-Term Memory) model. It included functions to build the model, train it on historical stock data, and test its performance on unseen data.
- **predictor.py**: The predictor module was responsible for generating stock price predictions using the trained model. It processed the most recent data to forecast the stock price for the next trading day.

These components worked together to create a functioning stock prediction model, laying the groundwork for further enhancements. The focus in v0.1 was on establishing a reliable prediction pipeline, ensuring that the model could accurately forecast stock prices based on historical data.

In the current task (v0.1), these core components are extended with advanced visualization techniques to provide deeper insights into the data and the model's predictions.

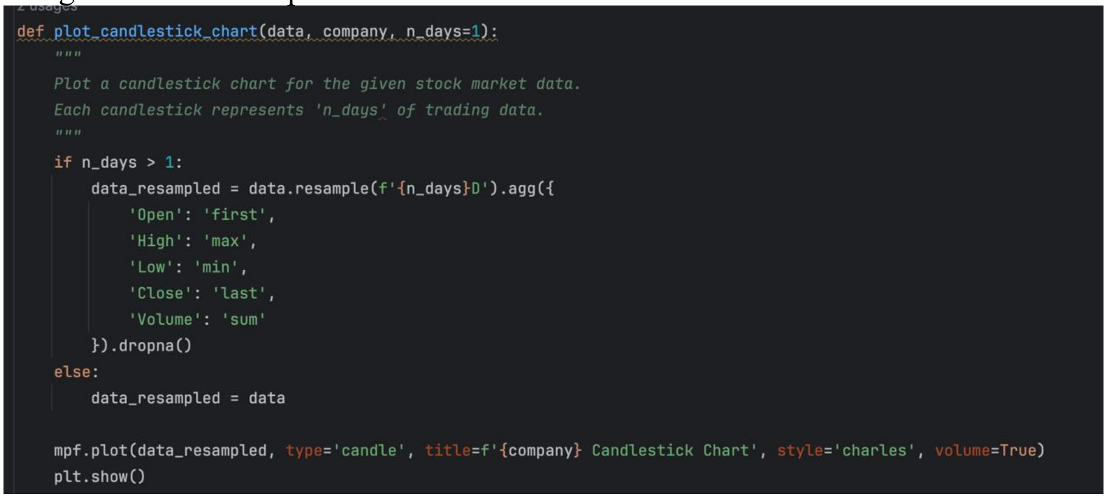
Enhancements in the Current Task

In this task, significant improvements were made to the stock prediction project by introducing advanced data visualization techniques and reorganizing the code to better integrate these new features. The focus was on enhancing the interpretability of the model's predictions through visual tools and making the overall workflow more streamlined and effective.

Visualization Techniques

- **Candlestick Chart Implementation**:
  - A new function was introduced in the newly created visualization.py file to generate candlestick charts. This chart provides a detailed view of the stock's price movements, showing the opening, closing, high, and low prices over a specified period. The candlestick chart is particularly useful for identifying market trends and patterns, making it a valuable addition to the project.

  - The candlestick chart was then integrated into the main.py script, ensuring that it is displayed as part of the analysis workflow. This allows users to visually inspect the historical price data alongside the model's predictions.



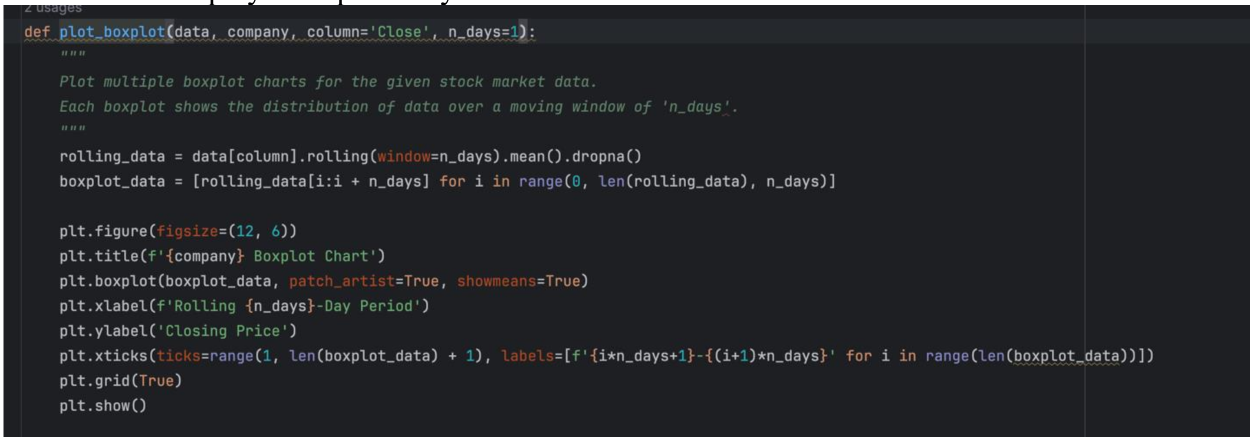
```python
def plot_candlestick_chart(data, company, n_days=1):
    """
    Plot a candlestick chart for the given stock market data.
    Each candlestick represents 'n_days' of trading data.
    """
    if n_days > 1:
        data_resampled = data.resample(f'{n_days}D').agg({
            'Open': 'first',
            'High': 'max',
            'Low': 'min',
            'Close': 'last',
            'Volume': 'sum'
        }).dropna()
    else:
        data_resampled = data

    mpf.plot(data_resampled, type='candle', title=f'{company} Candlestick Chart', style='charles', volume=True)
    plt.show()
```

Figure 1. plot_canlestick_chart function

- **Boxplot Chart Implementation**:
  - Another function was added in visualization.py to generate boxplot charts, which visualize the distribution of stock prices over a moving window of trading days. This chart helps in identifying the spread and central tendency of the stock prices, providing insights into price volatility and outliers.

  - Like the candlestick chart, the boxplot chart was also incorporated into main.py, allowing it to be displayed sequentially after the candlestick chart.



```python
def plot_boxplot(data, company, column='Close', n_days=1):
    """
    Plot multiple boxplot charts for the given stock market data.
    Each boxplot shows the distribution of data over a moving window of 'n_days'.
    """
    rolling_data = data[column].rolling(window=n_days).mean().dropna()
    boxplot_data = [rolling_data[i:i + n_days] for i in range(0, len(rolling_data), n_days)]

    plt.figure(figsize=(12, 6))
    plt.title(f'{company} Boxplot Chart')
    plt.boxplot(boxplot_data, patch_artist=True, showmeans=True)
    plt.xlabel(f'Rolling {n_days}-Day Period')
    plt.ylabel('Closing Price')
    plt.xticks(ticks=range(1, len(boxplot_data) + 1), labels=[f'{i*n_days+1}-{(i+1)*n_days}' for i in range(len(boxplot_data))])
    plt.grid(True)
    plt.show()
```

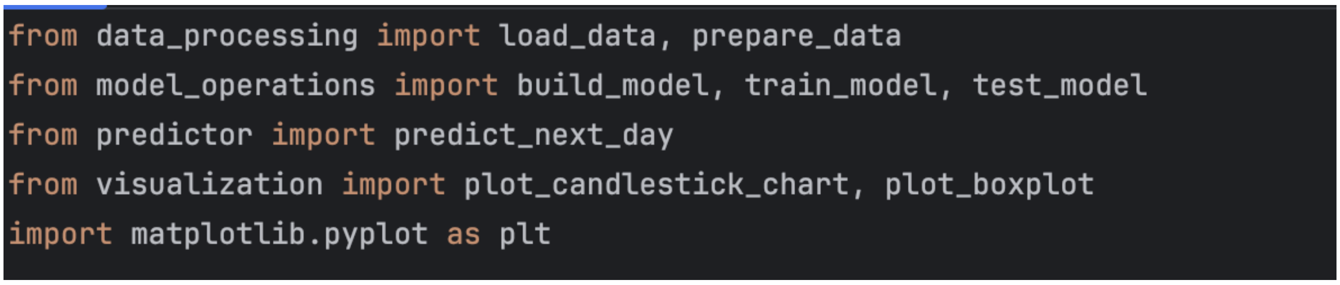
Figure 2. plot_boxplots function

Key Changes in main.py

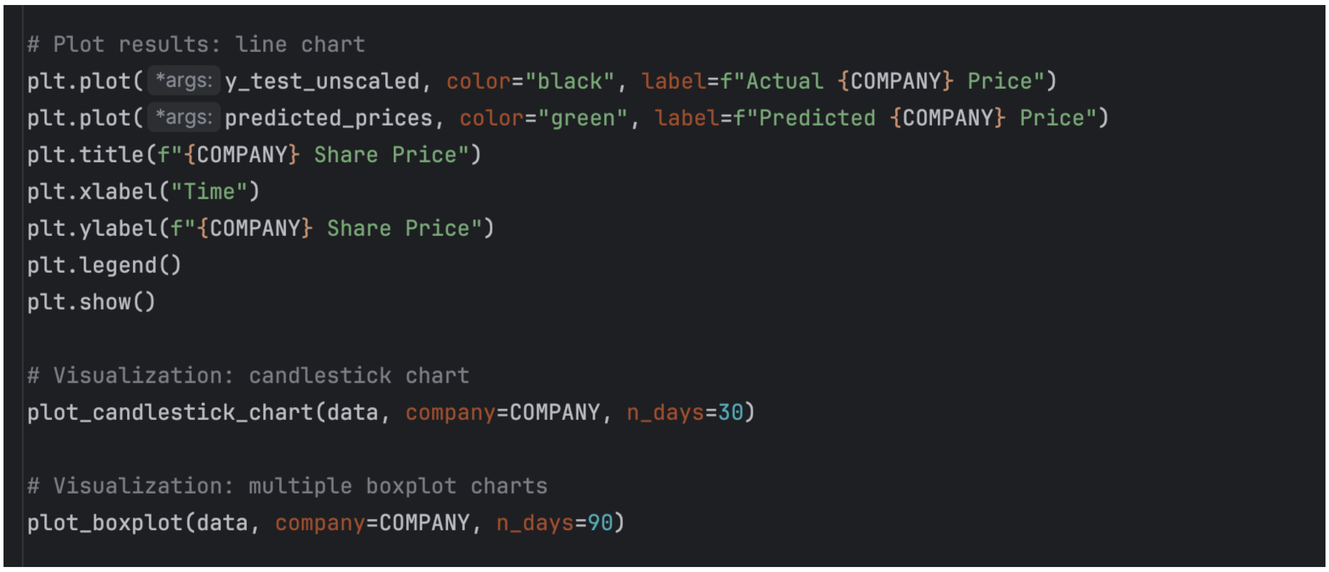
- **Reordering of Plotting Functions**:
  - The main.py code was modified to incorporate the new visualization functions. Specifically, the candlestick and boxplot visualizations were positioned before the plotting of actual vs. predicted prices. This change ensures that the visual analysis of historical data precedes the evaluation of the model's predictions, offering a logical flow to the analysis.

- **Integration of New Functions**:
  - The functions plot_candlestick_chart and plot_boxplot from visualization.py were integrated into main.py. These functions are called directly within the main workflow, allowing for the sequential display of the candlestick chart, boxplot, and stock prediction plot.

- **Enhancing Workflow Structure**:
  - The adjustments to main.py not only included adding new lines of code for visualization but also involved reorganizing the existing code to ensure a smooth and cohesive workflow. The plotting of the stock prediction results was moved to follow the visualizations, creating a logical progression from historical data analysis to predictive modeling.



```
from data_processing import load_data, prepare_data
from model_operations import build_model, train_model, test_model
from predictor import predict_next_day
from visualization import plot_candlestick_chart, plot_boxplot
import matplotlib.pyplot as plt
```



```
# Plot results: line chart
plt.plot( *args: y_test_unscaled, color="black", label=f"Actual {COMPANY} Price")
plt.plot( *args: predicted_prices, color="green", label=f"Predicted {COMPANY} Price")
plt.title(f"{COMPANY} Share Price")
plt.xlabel("Time")
plt.ylabel(f"{COMPANY} Share Price")
plt.legend()
plt.show()

# Visualization: candlestick chart
plot_candlestick_chart(data, company=COMPANY, n_days=30)

# Visualization: multiple boxplot charts
plot_boxplot(data, company=COMPANY, n_days=90)
```

Figure 3. Change in main.py

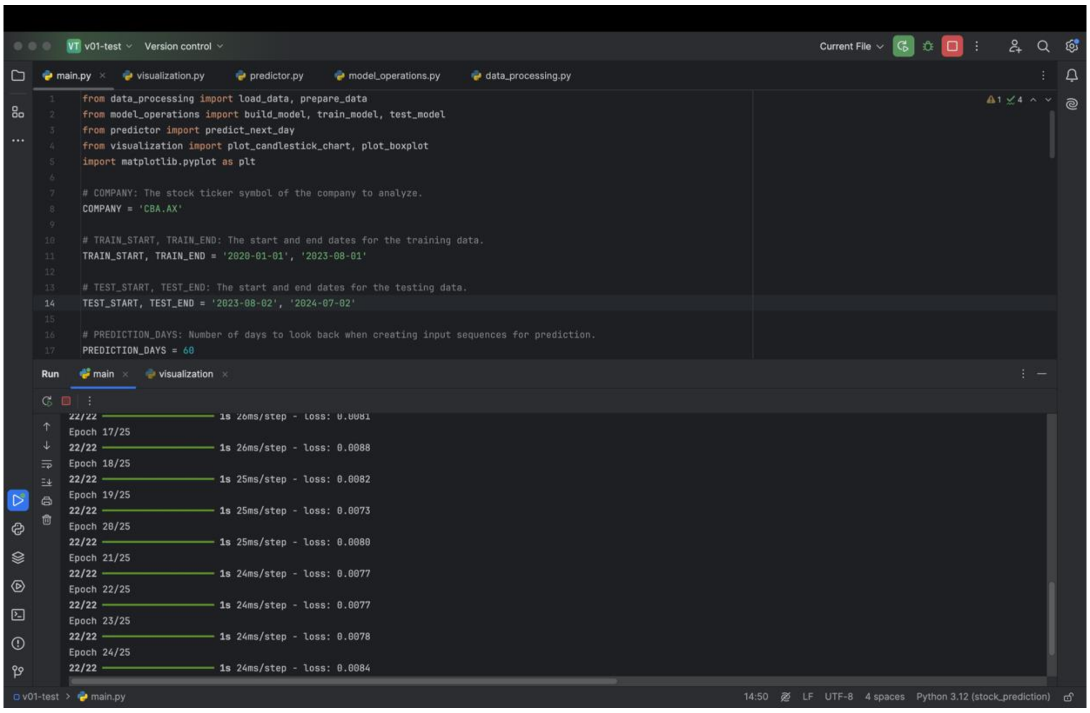
Model Execution and Outputs

This section outlines the execution of the enhanced stock prediction model and presents the resulting outputs. The goal is to provide a clear understanding of how the model performs and how the newly integrated visualization techniques aid in interpreting the stock price predictions.

Execution Process

The model was executed using CBA.AX stock data, with a training period from January 2020 to August 2023 and a testing period from August 2023 to July 2024. The model was trained over 25 epochs, during which the loss consistently decreased, indicating that the model effectively learned from the data.

Key enhancements in this task include the integration of advanced visualization techniques that provide additional insights into the model's performance and the underlying stock data.



```python
from data_processing import load_data, prepare_data
from model_operations import build_model, train_model, test_model
from predictor import predict_next_day
from visualization import plot_candlestick_chart, plot_boxplot
import matplotlib.pyplot as plt

# COMPANY: The stock ticker symbol of the company to analyze.
COMPANY = 'CBA.AX'

# TRAIN_START, TRAIN_END: The start and end dates for the training data.
TRAIN_START, TRAIN_END = '2020-01-01', '2023-08-01'

# TEST_START, TEST_END: The start and end dates for the testing data.
TEST_START, TEST_END = '2023-08-02', '2024-07-02'

# PREDICTION_DAYS: Number of days to look back when creating input sequences for prediction.
PREDICTION_DAYS = 60
```

```
22/22 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0081
Epoch 17/25
22/22 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0088
Epoch 18/25
22/22 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0082
Epoch 19/25
22/22 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0073
Epoch 20/25
22/22 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0080
Epoch 21/25
22/22 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0077
Epoch 22/25
22/22 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0077
Epoch 23/25
22/22 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0078
Epoch 24/25
22/22 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0084
```

Figure 4. Training process

Visual Output

The following visualizations were generated to provide a comprehensive analysis of the stock prediction model:

- **Candlestick Chart**:
  - This chart visualizes the historical price movements of CBA.AX, showing the opening, closing, high, and low prices for each trading day. The candlestick chart helps identify market trends and price patterns, offering context to the model's predictions.

- **Boxplot Chart**:
  - The boxplot illustrates the distribution of stock prices over a moving window, highlighting the spread, central tendency, and potential outliers in the data. This visualization is crucial for understanding price volatility during the testing period.

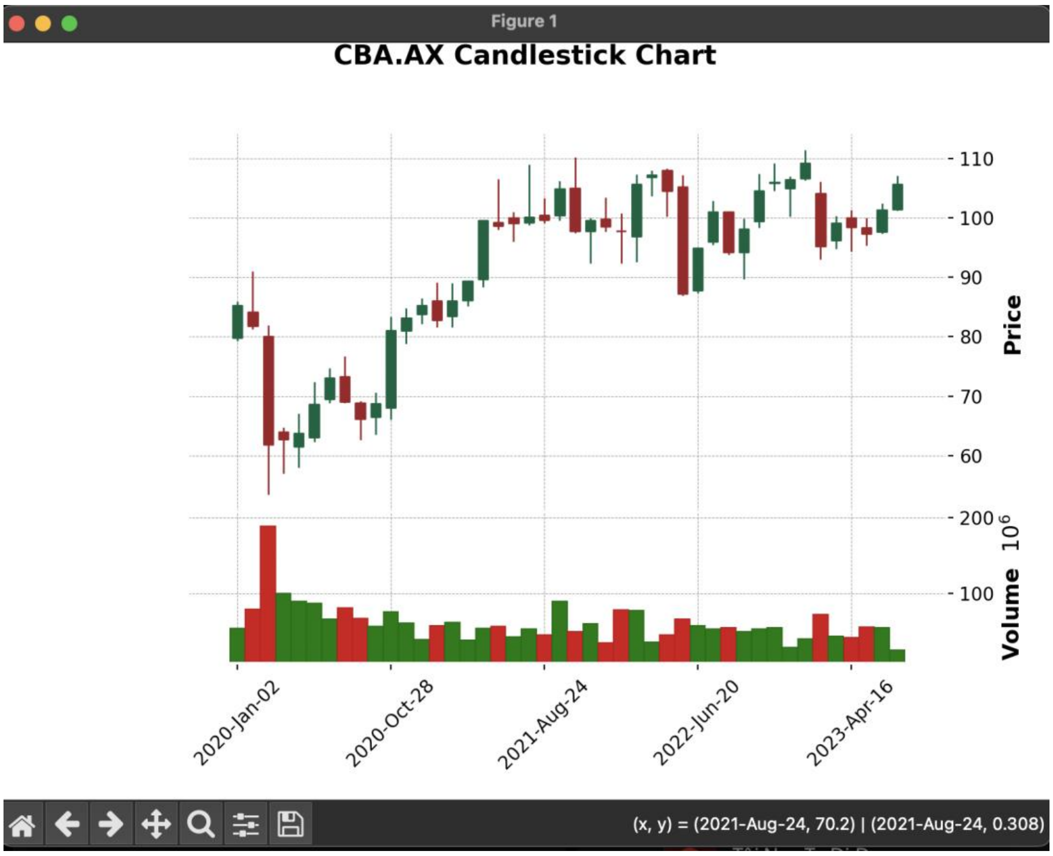


Figure 5. Candlestick chart

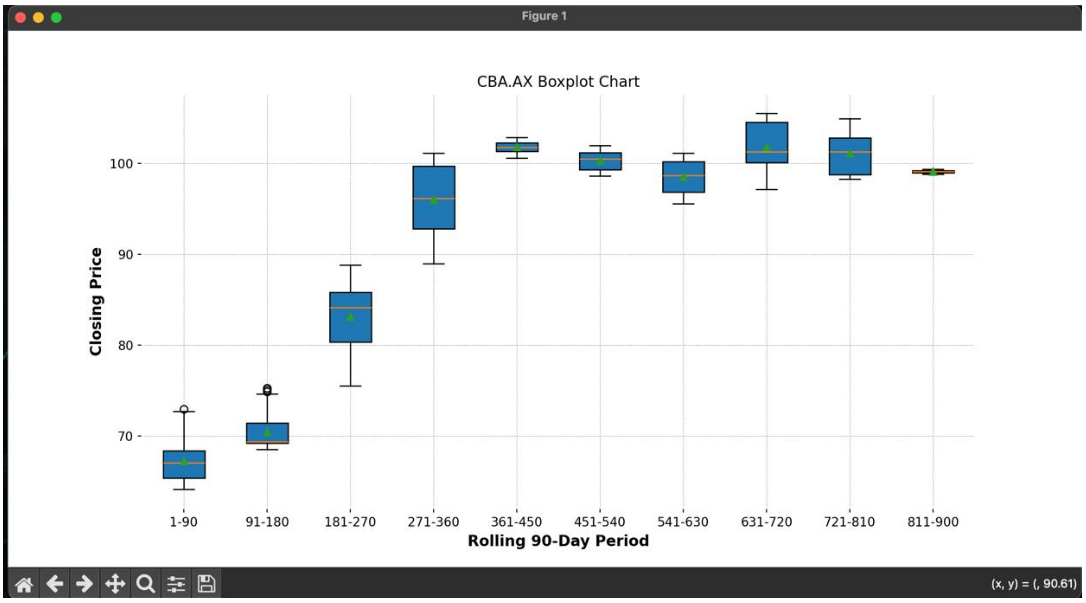


Figure 6. Boxplot chart

This output shows that the model generally captures the trend of the stock prices, though some discrepancies are observed, particularly during periods of high volatility. These differences highlight opportunities for further tuning and refinement of the model to improve prediction accuracy.

GitHub Repository

The project's code base is hosted on GitHub for version control and review. Everyone can access the repository via the following link: GitHub Repository. This repository contains all necessary files for the project and is available for the tutor to review the work.

References

- https://www.youtube.com/watch?v=UuBigNaO_18NeuralNine. (2022, October 1). *Stock Price Prediction using LSTM in Python*. [Video]. YouTube. Retrieved from
- https://www.youtube.com/watch?v=iufeGhOBVNg
- https://www.youtube.com/watch?v=CeTR_-ALdRw&list=PL7yh-TELLS1G9mmnBN3ZSY8hYgJ5kBOg-&index=4
- GeeksforGeeks. (n.d.). *Best Python Libraries for Machine Learning*. Retrieved from https://www.geeksforgeeks.org/best-python-libraries-for-machine-learning/
- https://coderzcolumn.com/tutorials/data-science/candlestick-chart-in-python-mplfinance-plotly-bokeh#1