STOCK PRICE PREDICTION REPORT

1. Introduction

This project aims to develop an LSTM-based machine learning model for predicting stock price movements using historical financial data. The report outlines the steps involved in data preprocessing, feature engineering, model development, and evaluation.

2. Data Preprocessing

2.1. Data

The dataset was obtained using the **Yahoo Finance API** and includes daily stock price movements of **Tesla Inc. (TSLA)** from **January 1, 2020, to January 1, 2025**. The key features used in the analysis: Open Price, High Price, Low Price, Close Price and Volume

2.2. Data Processing Workflow and Descriptions

To ensure the model functions correctly, data is processed in the following sequence:

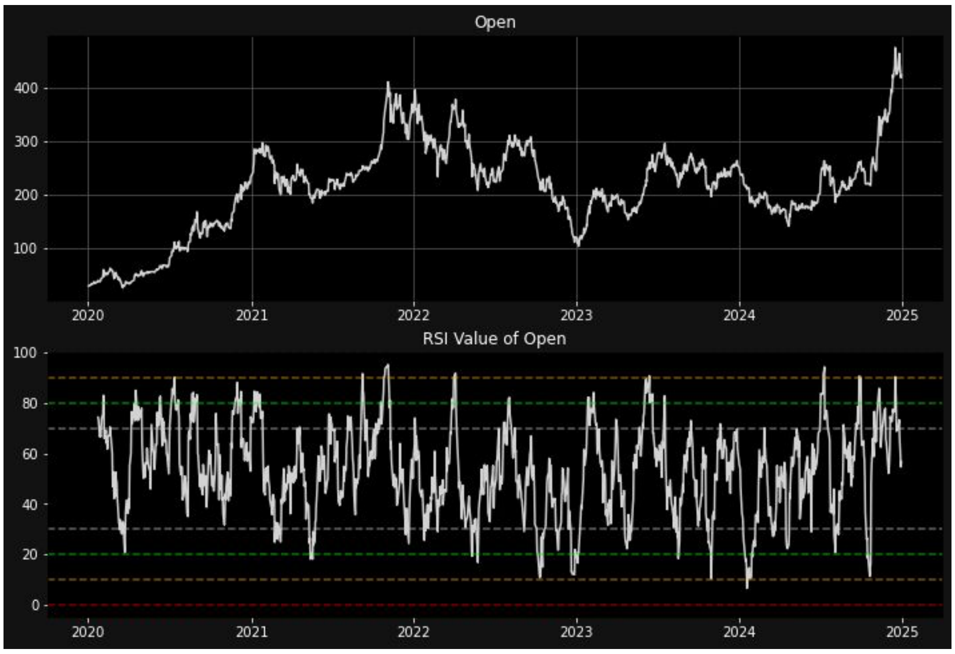
Note that, for the functions have underscore (_) in front of them signifies that they are coded specifically within the file itself in order to make sure that code is robust and thrustable. (Helper functions.)

1. **data_stock.py**: Retrieves stock data from Yahoo Finance API. Fills missing values and ensures correct formatting.
   - **get_stock_data():** Fetches data for a given company symbol, date range, and selected features.
   - **_check_date():** Ensures the date format is YYYY-MM-DD.
   - **_check_company_names():** Validates stock symbols.
   - **_check_interval():** Checks valid time intervals.
   - **_check_features():** Verifies selected financial features.
2. **prepare_data.py**: After having the stock price data, data preparation process shall start. This, splits data into training and testing sets. Normalizes the data for better model performance. Converts time series into input-output sequences.
   - **prepare_train_test_data():** Splits and normalizes the dataset.
   - **_get_time_sequences():** Converts data into 90-day (can be changed, explain later below) time steps.
   - **get_input_output_dims():** Defines input and output dimensions for the LSTM model. (This code is not a mandatory, however, intended to make it simpler when a need emerges for the input & output dimensions.)
3. **visualize_data.py**: To display or plot the stock price data. This also, by inspection, enables one to detect anomalies within the data.
   - **visualize_data():** Generates price changes over time along with volume charts.

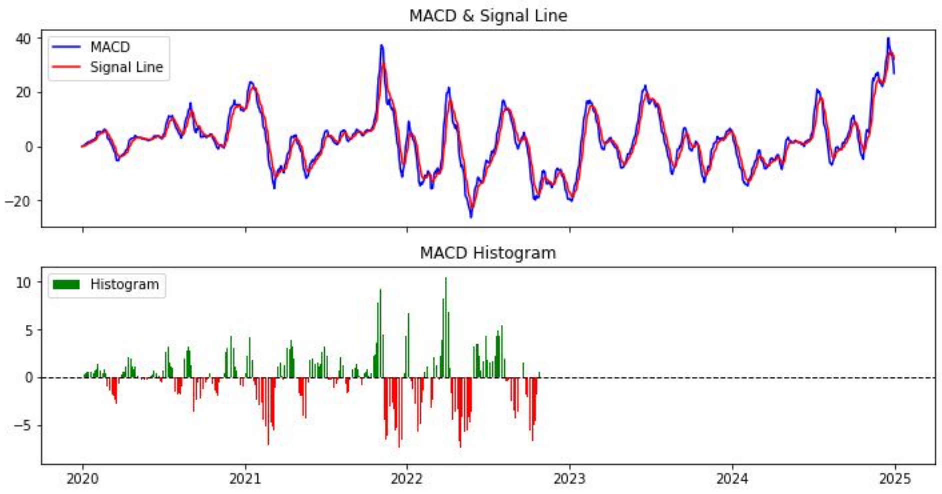
4. **model.py**: A Bidirectional LSTM (Long Short-Term Memory) Network was implemented to capture complex temporal dependencies.
    - **build_lstm_model():** Constructs and compiles the LSTM model.
        - Four LSTM layers with 128, 64, 32, and 16 nodes. Fours layers are employed to make the model more aware about further complex and intriguing findings easily. This is an ad-hoc process where one needs to try one and change the structure if needed until it fulfills his/her needs. The reason why "bidirectional" LSTM was utilized in the first layers is -despite of sequential layer- stock prices do not follow a certain trend instead, they are usually altered fast subject to changes around the globe. Furthermore, the dimensions of the layers are chosen as above however, they are free to change and can be investigated in the training section down below.
        - Dropout layers to prevent overfitting.
        - A last layer for final price prediction. Last layer's dimension is changed in accordance with number of companies along with their corresponding features willing to be train. Basically, the number of companies x number of features will yield the total "prediction values" needed. Failing to update the last layer will subject to error in training, therefore, this dimension determination is made autonomously.
        - L2 Regularization for better generalization. L2 is the simplest yet effective rule to employ, others are not even tried.
        - Adam optimizer for efficient training. Adam gave the best convergence among other optimizers.(RMSprop and SGD not documented)

5. **visualize_lstm_output.py**: Compares predicted vs. actual prices. Converts normalized values back to original scale.
    - **visualize_lstm_output():** Compares predicted vs. actual prices.
        - Plots predicted data on validation set and their groundtruths as well as losses (both training and validation) curves to analyze training stability.

6. **Feature Engineering**: To improve model accuracy, additional financial indicators were included:
    - **rsi.py**: Measures price momentum and overbought/oversold conditions.
        - **get_rsi():** Calculates RSI values for selected time periods.
        - **visualize_rsi():** Plots RSI trends, highlighting overbought (>70) and oversold (<30) levels.

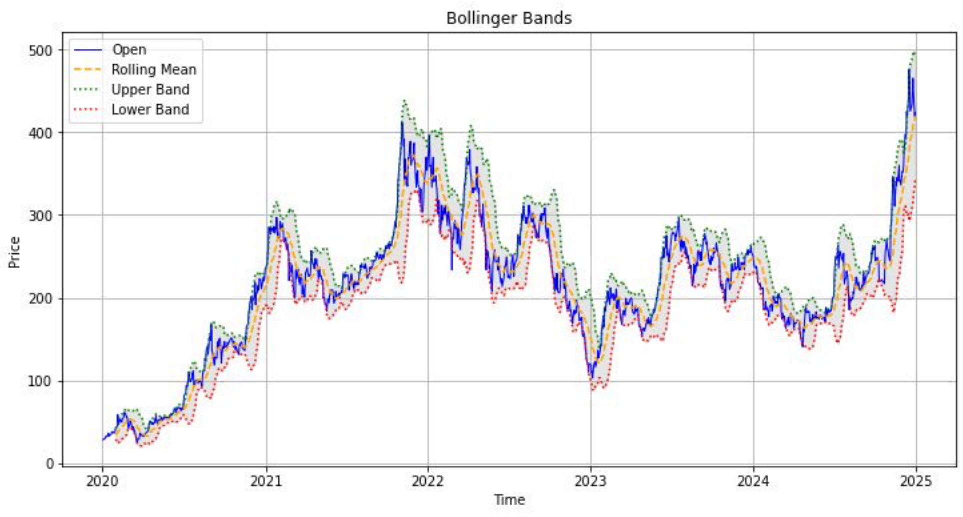


- **macd.py**: Tracks price momentum and trend direction.
  - **get_macd():** Computes MACD line, signal line, and histogram.
  - **visualize_macd():** Plots MACD indicators.
  -



- **bollinger_bands.py**: Evaluates price volatility and identifies price extremes.

  - **bollinger_bands():** Computes upper and lower bands based on a 20-day moving average.
  - **visualize_bollinger_bands():** Plots Bollinger Bands to visualize volatility.



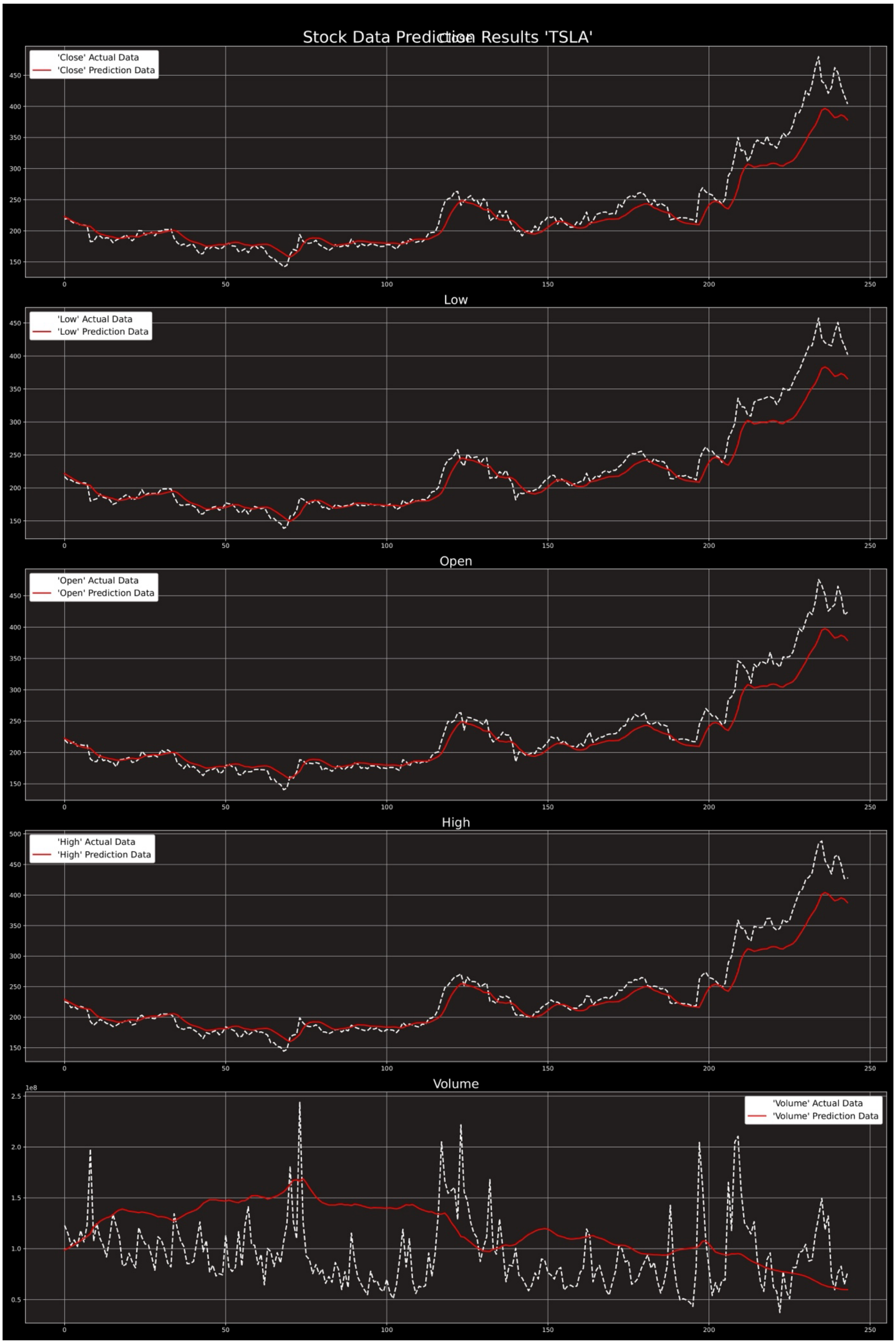
7. **train.py**: This script is responsible for training the LSTM model to predict stock prices. (You can actually give it a shot! Go ahead and run the train.py file line by line (one may use shift+enter) to see what the training process takes place.) This script withinself, calls all the other external methods residing in other modules/files that are detailed above.
    - Loading Required Modules: **EarlyStopping:** Prevents overfitting by stopping training if performance does not improve after a certain number of epochs. (Typically this is chosen to be epoch //2.) **MinMaxScaler:** Normalizes data between 0 and 1 to enhance LSTM's learning efficiency. In other words, scaling provides to achieve the data which has been freed from its prior possible biase and mitigates such problems (exploding and vanishing gradients) thereby leveraging training quality of the modal furher.
    - Importing Custom Modules:
        - **build_lstm_model:** Defines and returns the LSTM model.
        - **get_stock_data:** Fetches historical stock data from Yahoo Finance API.
        - **prepare_train_test_data:** Splits and normalizes the dataset for training and testing. In other words, makes sure pure stock data is ready for training.
        - **get_input_output_dims:** Determines the input-output dimensions for the LSTM network.
        - **visualize_data:** Generates graphical analysis of the stock data.
        - **visualize_lstm_output:** Compares the model's predictions with actual stock prices in a plot.
        - **get_macd, get_rsi, bollinger_bands:** Computes technical indicators for trend analysis.
        - **visualize_macd, visualize_rsi, visualize_bollinger_bands:** Plots financial indicators.
    - Data Preparation: The user specifies the company, start and end dates, and features for analysis. Example: Tesla Inc. (TSLA) stock data from January 1, 2020, to January 1, 2025. Features used: Open, Close, High, Low, Volume. Data

is split into 80% training and 20% testing, then normalized. The model uses a 40-day time window to predict the next day’s stock price.

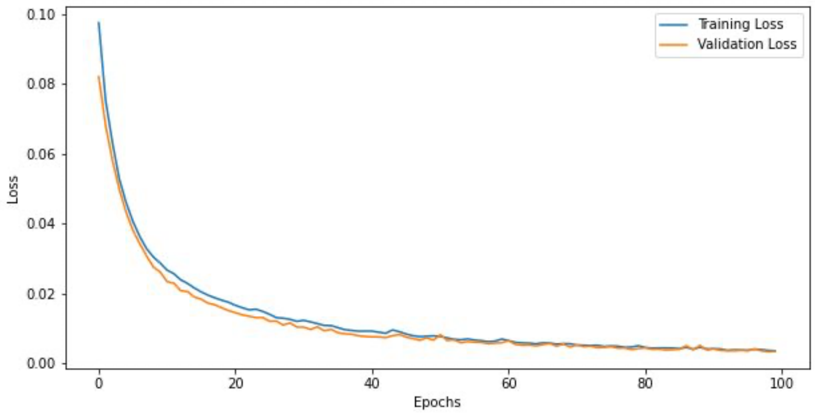
- Training Process: After having obtained the prior modules, training process now can take place. Before start, make sure that hidden dimension, dropout factors, learning rate, number of epochs, patiance, and lastly regularization coefficient should be fine tuned or adjusted carefully.
- Model Predictions Visualization: After trining finishes, the predicted stock prices are visualized and compared with actual values to assess the model’s performance.
- Technical Indicators Calculation: Key financial indicators (RSI, MACD, and Bollinger Bands) are computed to analyze market trends and price movements.
- Performance Evaluation: Model performance was evaluated using evaluation metrics and results can be seen below (RMSE, MAPE and R^2).

3. Performance Metrics and Results

After the model got trained, the results can be obtained and they are detailed below.


Stock Data Prediction Results 'TSLA'
Close
'Close' Actual Data
'Close' Prediction Data
Low
'Low' Actual Data
'Low' Prediction Data
Open
'Open' Actual Data
'Open' Prediction Data
High
'High' Actual Data
'High' Prediction Data
Volume
1e8
'Volume' Actual Data
'Volume' Prediction Data
0
50
100
150
200
250

The image shows the predicted vs. actual stock prices and volume for TSLA over an unseen 20% validation dataset, covering approximately 250 days. White dashed lines: Represent actual stock prices. Red solid lines: Represent LSTM model predictions. The model successfully captures the overall upward trend, but lags behind in highly volatile movements (e.g., sharp increases or drops). For all price features (Close, Low, Open, High), the model follows actual price movements but smooths out fluctuations. The model accurately predicts overall trends in stock prices. The predicted values align well with actual movements over a long period (250 days).



The graph shows the training loss (blue line) and validation loss (orange line) over 100 epochs. Initially, the loss starts around 0.10 and drops below 0.01, meaning it has decreased by a factor of 10. This indicates that the model is learning effectively and reducing error over time.

Parameters Training Took On	**RMSE**	**MAPE**	**R²**

epoch = 100
hidden_dim1 = 50
hidden_dim2 =30
hidden_dim3 = 20
dense_dim = 10
lr = 0.0004
	0.0399	16.97%	0.70

The LSTM model has demonstrated strong performance by minimizing error and enhancing generalization capability. This study successfully developed a stock price prediction model using financial data and technical indicators. Future research can incorporate additional macroeconomic factors and explore alternative deep learning architectures to further improve prediction accuracy.