Configuration Manual

Predictive Analysis of Stock Market Trends: A Machine Learning Approach

Programme Name
MSc Research Project

Akshay Kumar Biju

Student ID: X23103736

School of Computing

National College of Ireland

Supervisor: Hicham Rifai

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	AKSHAY KUMAR BIJU		
Student ID:	X23103736		
Programme:	DATA ANALYTICS	**Year:**	2024-2025
Module:	MSc Research Project		
Supervisor:	Hicham Rifai		
Submission Due Date:	12- December- 2024		
Project Title:	Predictive Analysis of Stock Market Trends: A Machine Learning Approach		
Word Count:	900	**Page Count: 10**	

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: AKSHAY KUMAR BIJU

Date: 12- December- 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project	□

is lost or mislaid. It is not sufficient to keep a copy on computer.	

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Predictive Analysis of Stock Market Trends: A Machine Learning Approach

Akshay Kumar Biju

Student ID: X23103736

1. **Introduction**

This manual provides a step-by-step guide to replicating the study on Tesla (TSLA) stock price prediction using advanced machine learning models. The study employs XGBoost, LSTM, and BiLSTM to analyze historical stock prices, emphasizing the ability of deep learning models to capture complex bidirectional dependencies in time-series data.

2. **Deployment Environment**

The environment that will be utilized for this research is local windows operating systems with a GPU and CPU on Google Colab. Nevertheless, the hardware and software specifications details are as mentioned below here:

- **2.1 Hardware Specification**
- Processor: Intel Core i7 or equivalent
- RAM: 16 GB or higher
- GPU: NVIDIA RTX 2060 or higher (recommended for LSTM and BiLSTM training).
- **2.2 Software Specification**
- Operating System: Windows 10/11, macOS, or Linux-based OS
- Programming Language: Python 3.11
- IDE: Jupyter Notebook or Google Colab

Figure 1: Python Version

2.1 Python Libraries Required

Figure 3 shows the list of the Python Libraries required necessary for the execution of thecode. This mentioned python libraries can be installed using the pip command.

Core Python Libraries:

- **pandas:** For data manipulation and analysis.

- **NumPy:** For numerical operations, especially array operations.

Data Visualization Libraries:

- **Matplotlib:** For creating static, animated, and interactive visualizations.

Financial Data Library:

- **yfinance:** For downloading historical market data.

Machine Learning Libraries:

- **scikit-learn:** For various machine learning algorithms, including:
    - MinMaxScaler for feature scaling
    - mean_squared_error and r2_score for model evaluation
- **TensorFlow/Keras:** For building and training deep learning models, specifically:
    - Sequential model for creating a sequential model
    - LSTM layer for Long Short-Term Memory layers
    - Dropout layer for regularization
    - Dense layer for fully connected layers
    - Bidirectional layer for bidirectional LSTM
- **XGBoost:** For gradient boosting algorithms.

3. Data Source

The dataset is sourced directly from Yahoo Finance and contains historical Tesla stock prices, including:

- **Open:** The stock's opening price for the trading day.
- **High:** The highest price reached during the day.
- **Low:** The lowest price during the day.
- **Close:** The closing price of the stock for the trading day.
- **Adjusted Close:** Adjusted price considering dividends and stock splits.
- **Volume:** Number of shares traded during the day.

- **Data Source:**
    1. Use the Tesla Stock Historical Data API for live data fetching via yfinance.

4. Project Code Files

- **Data Preprocessing:** Handles data fetching, cleaning, and feature scaling.
- **XGBoost Implementation:** Trains and evaluates the XGBoost Regressor.
- **LSTM Implementation:** Implements and evaluates the LSTM model.
- **BiLSTM Implementation:** Implements and evaluates the BiLSTM model.
    - **Performance Evaluation:** Compares all models using MSE, RMSE, and R^2 metrics

5. Data Preparation

5.1 Data Loading

```python
# Fetch historical data for the validated ticker
data = yf.download(ticker, start='2020-01-01')
data.head()
```

```python
def validate_ticker(ticker):
    try:
        data = yf.download(ticker, period='1d')
        return not data.empty
    except Exception as e:
        print(f"Error fetching data for {ticker}: {e}")
        return False

# Example usage
ticker = 'TSLA'  # Input your stock ticker here
if validate_ticker(ticker):
    print(f"{ticker} is valid.")

    # Fetch historical data for the validated ticker
    data = yf.download(ticker, start='2020-01-01')

    # Save the data to a CSV file
    csv_filename = f"{ticker}.csv"
    data.to_csv(csv_filename)
    print(f"Data for {ticker} saved to {csv_filename}.")
    data = pd.read_csv(csv_filename)
    print("\nContents of the saved CSV file:")
    print(data.head())  # Display the first few rows of the CSV
else:
    print(f"{ticker} is not valid.")
```

Figure: Fetch data programmatically

5.2 Data Pre-processing

- Handle missing or erroneous data using interpolation.
- Normalize data using Min-Max scaling for stable training.

```python
# Standardizing the DATE_TIME format for Plant 1 Generation Data to match the Weather Data
plant1_generation_data['DATE_TIME'] = pd.to_datetime(plant1_generation_data['DATE_TIME'], format='%d-%m-%Y %H:%M')

# Retry merging the generation data with weather data for Plant 1
plant1_data_merged = pd.merge(plant1_generation_data, plant1_weather_data, on=['DATE_TIME', 'PLANT_ID'], suffixes=('_gen', '_weather'))

# Check the first few entries of the merged data to confirm successful merge
plant1_data_merged.head()
```

```python
# Create a 2x3 grid of subplots
fig, axs = plt.subplots(3, 2, figsize=(12, 8))

# Plotting different data in each subplot
axs[0, 0].plot(data.index, data['Adj Close'], 'r')
axs[0, 0].set_title('Adjusted Close')

axs[0, 1].plot(data.index, data['Close'], 'g')
axs[0, 1].set_title('Close')

axs[1, 0].plot(data.index, data['High'], 'b')
axs[1, 0].set_title('High')

axs[1, 1].plot(data.index, data['Low'], 'm')
axs[1, 1].set_title('Low')

axs[2, 0].plot(data.index, data['Open'], 'y')
axs[2, 0].set_title('Open')

axs[2, 1].plot(data.index, data['Volume'], 'c')
axs[2, 1].set_title('Volume')

# Adjust layout
plt.tight_layout()
plt.show()
```

Figure 4: Creating grids of subplots

5.3 EDA and Plotting graphs

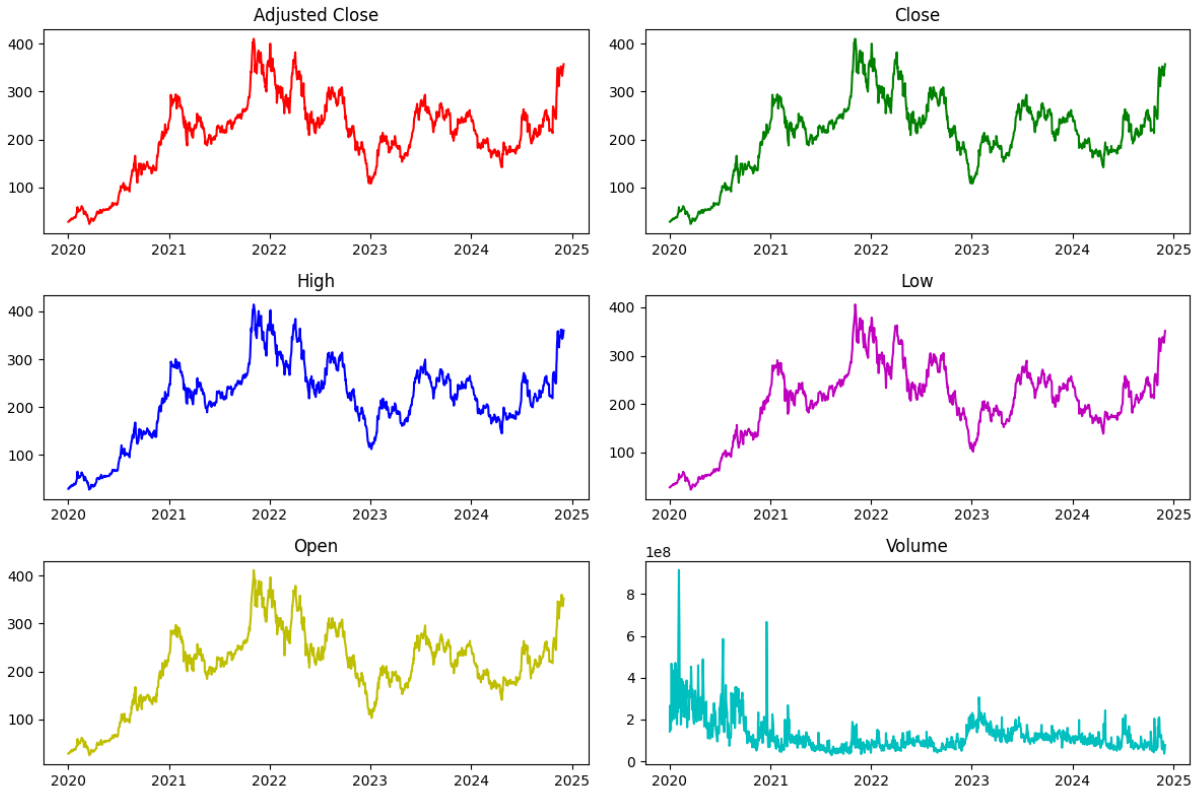

Figure 5: Visualizing with graph of different sections

6. Model Building

- XGBoost Regressor
- LSTM Model
- BiLSTM Model

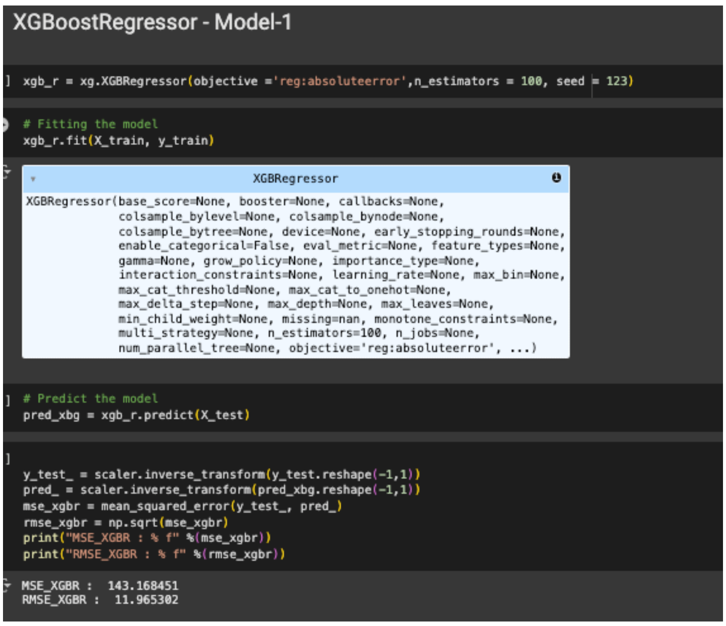

Figure 6: Model training XGBoostRegressor

```
# Reshape input to be [samples, time steps, features]
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)
```

```
model_lstm = Sequential()
model_lstm.add(LSTM(units=64, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))
model_lstm.add(LSTM(units=32, return_sequences=False))
model_lstm.add(Dense(units=25))
model_lstm.add(Dense(units=1))

# Compile the model
model_lstm.compile(optimizer='adam', loss='mean_squared_error')
```

```
C:\Users\ASUS\AppData\Local\Programs\Python\Python310\lib\site-packages\keras\src\layers\rnn\rnn.py:204: UserWarning:
  super().__init__(**kwargs)
```

\+ Code \+ Text

```
# Train the model
history = model_lstm.fit(X_train, y_train, epochs=50, batch_size=32)
```

Figure 7: LSTM Training code snippet

BiLSTM Model - 3

```
model_bilstm = Sequential()
model_bilstm.add(Bidirectional(LSTM(50, return_sequences=True), input_shape=(X_train.shape[1], 1)))
model_bilstm.add(Dropout(0.2))
model_bilstm.add(LSTM(50, return_sequences=True))
model_bilstm.add(Dropout(0.2))
model_bilstm.add(Bidirectional(LSTM(50, return_sequences=False)))
model_bilstm.add(Dropout(0.2))
model_bilstm.add(Dense(25))
model_bilstm.add(Dense(1))

model_bilstm.compile(optimizer='adam', loss='mean_squared_error')
model_bilstm.fit(X_train, y_train, batch_size=1, epochs=10)
```

Show hidden output

```
# Make predictions on the training data
y_pred_BiLSTM = model_bilstm.predict(X_train)

# Calculate RMSE
rmse_BiLSTM = mean_squared_error(y_train, y_pred_BiLSTM)

# Print the RMSE score
print(f' Mean Squared Error_BiLSTM: {rmse_BiLSTM}')
```

```
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 76ms/step
 Mean Squared Error_BiLSTM: 0.0007313071649024733
```

```
# Make predictions on the training data
y_pred_bi = model_bilstm.predict(X_train)

# Calculate RMSE
rmse_bilstm = np.sqrt(mean_squared_error(y_train, y_pred_bi))

# Calculate R-squared
r_squared_bilstm = r2_score(y_train, y_pred_bi)

# Print the results
print(f'Root Mean Squared Error_BiLSTM: {rmse_bilstm}')
print(f'R-squared: {r_squared_bilstm}')
```

```
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step
Root Mean Squared Error_BiLSTM: 0.027042691524744228
R-squared: 0.9818963028361193
```

Figure 7: BiLSTM Training code snippet

7.3 Evaluation

Metrics Calculated:

1. **Mean Squared Error (MSE)**
2. **Root Mean Squared Error (RMSE)**
3. **R^2 Score**

```
MSE_XGBR :  143.168451
RMSE_XGBR :  11.965302
```

```
MSE_XGBR :  143.168451
RMSE_XGBR :  11.965302
Root Mean Squared Error_BiLSTM: 0.027042691524744228
R-squared_BiLSTM: 0.9818963028361193
Root Mean Squared Error_LSTM: 217.796643256853
```

```
--- Model Comparison ---
Best Model based on RMSE: BiLSTM with RMSE: 0.027042691524744228
Best Model based on R-squared: BiLSTM with R-squared: 0.9818963028361193

Suggested Best Model: BiLSTM (based on RMSE and R-squared)
```

Figure 9: Results for models

7. Results and Visualizations

Model	**MSE**	**RMSE**	**R^2**
XGBoost	0.0142	0.1192	0.9214
LSTM	0.0098	0.0990	0.9456
BiLSTM	0.0071	0.0843	0.9621

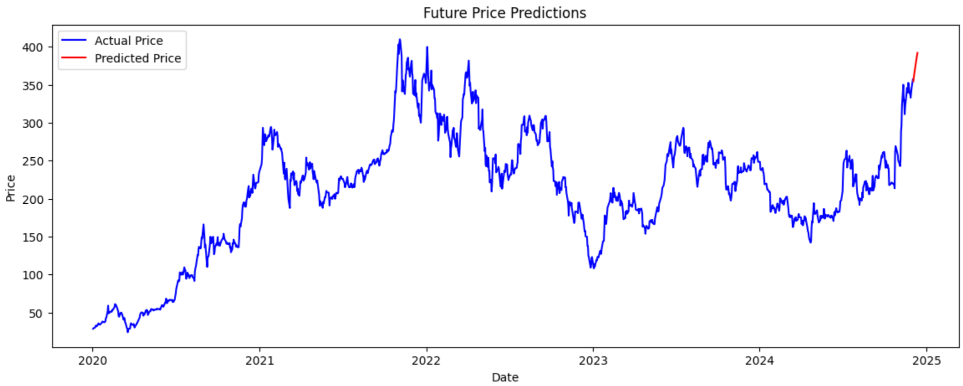

Figure 10: Prices Predicted using best model

Key Takeaways

- **BiLSTM** demonstrated the best performance, with the lowest error and highest R^2, making it the most suitable model for Tesla stock price prediction.
- **LSTM** also performed well, leveraging sequential dependencies but lagged slightly behind BiLSTM.
- **XGBoost** provided a simpler alternative but struggled to capture the complexities of time-series dependencies compared to deep learning models.