Stock Market Price Prediction

Presented By :-

16010122030 - Shubham Chaurasia
16010122044 - Ved Dhake
16010122051 - Mayur Dumbre
16010122055 - Hriday Gandhi
16010122065 - Farzan Irani

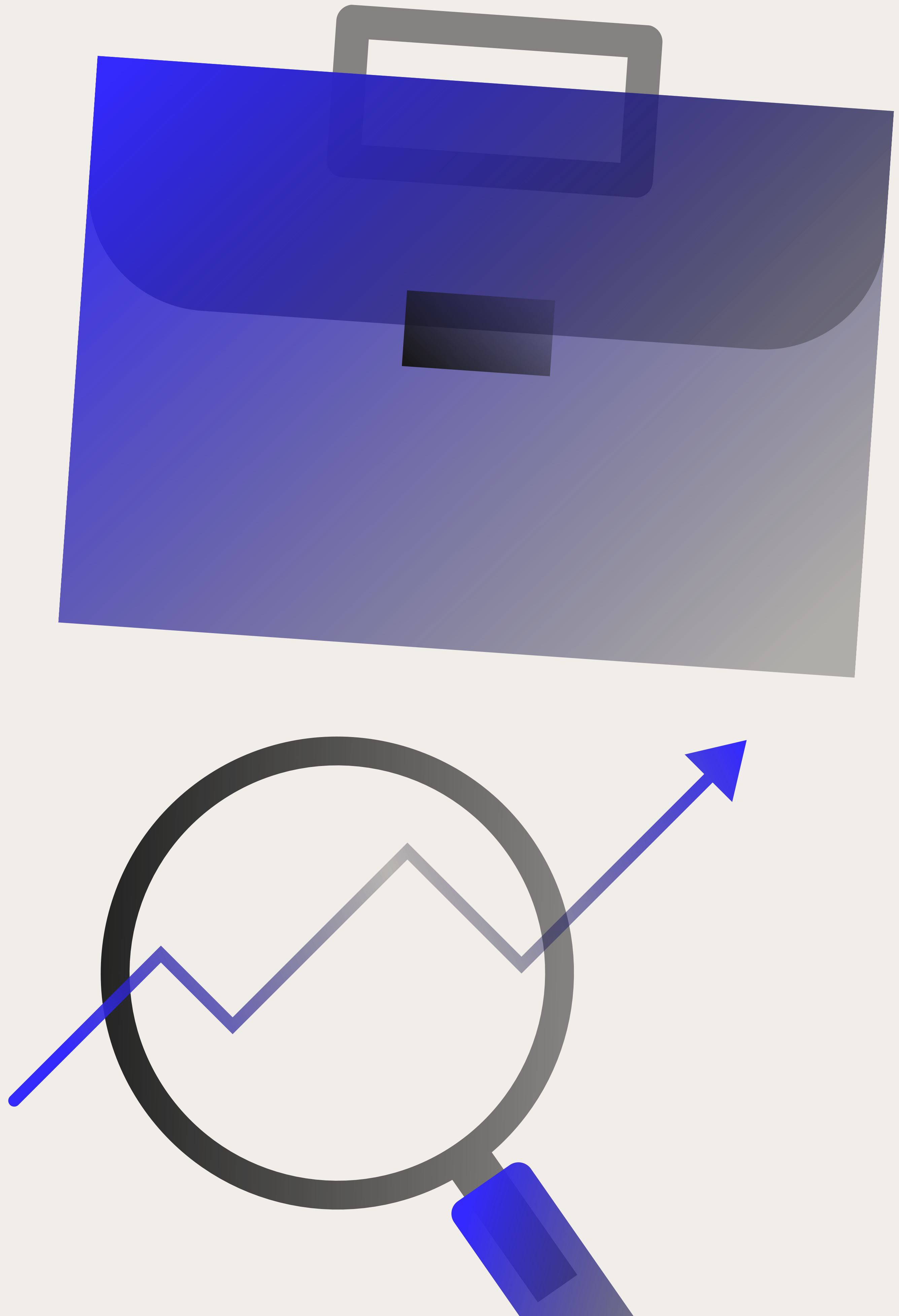
Institution : K.J. Somaiya School Of Engineering
Date : 17-04-2025

Problem Definition

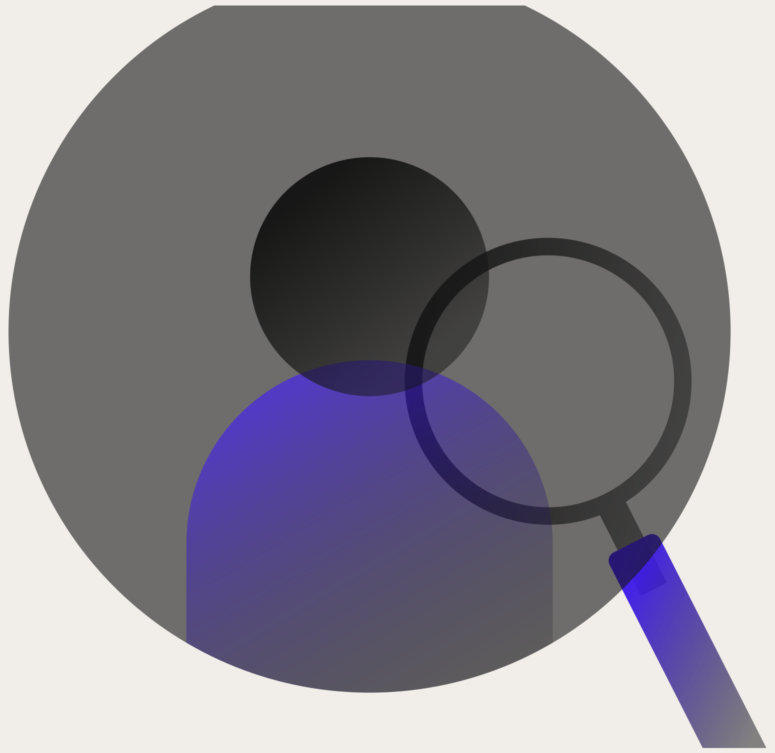
Predicting stock prices is a challenging time series forecasting problem due to the market's non-linear, noisy, and highly dynamic nature. Traditional statistical models often fail to capture long-term dependencies and complex patterns in historical price data.

Machine learning, particularly Long Short-Term Memory (LSTM) networks, offers a more effective approach by learning temporal relationships and trends from past stock prices. LSTMs are designed to handle sequential data and can retain information over long time periods, making them suitable for modeling and forecasting stock prices.

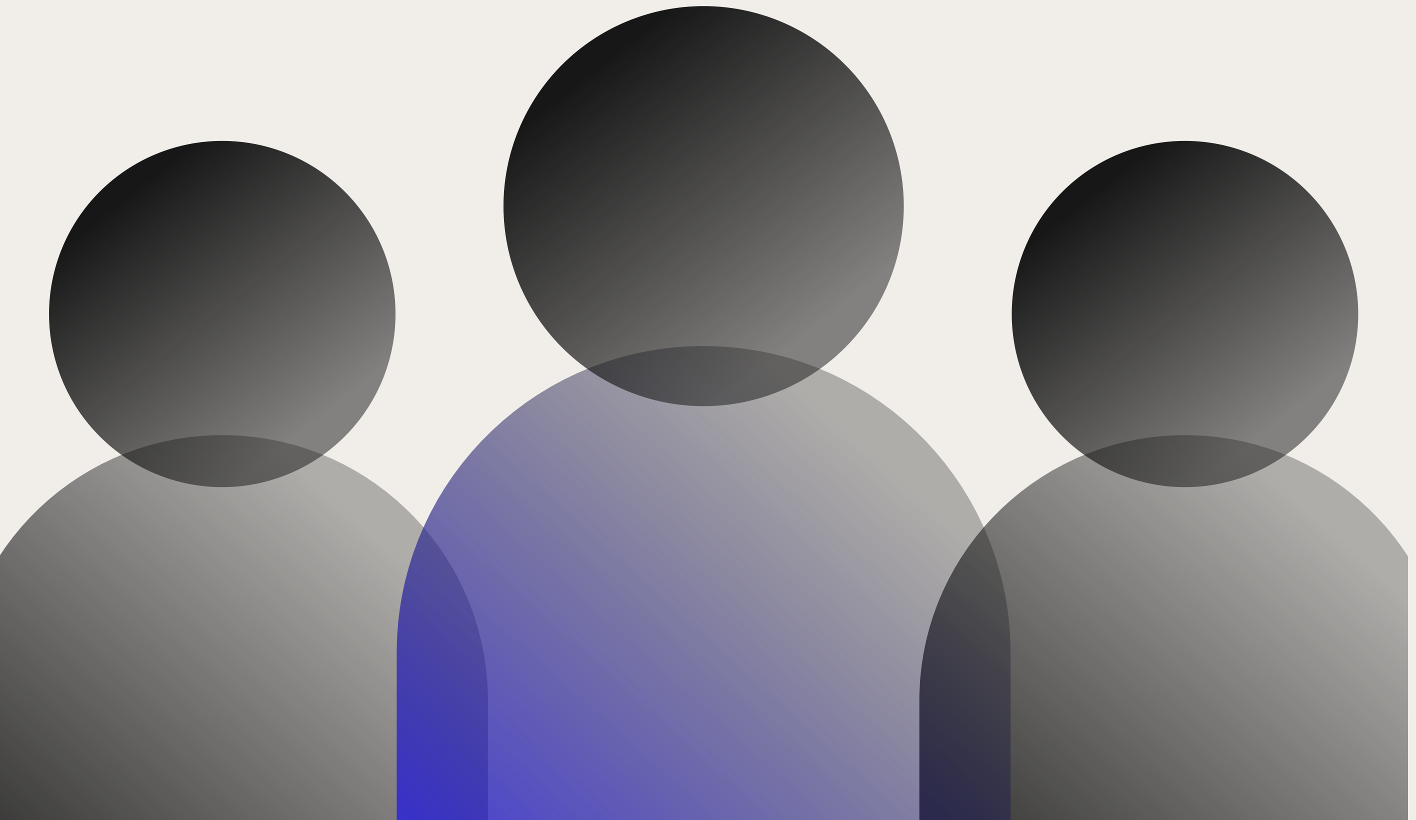
This project focuses on building an LSTM-based predictive model that can learn from historical closing prices and provide accurate short-term future forecasts.



Objectives of Work



- To design and implement a supervised learning pipeline for stock price prediction using Long Short-Term Memory (LSTM) neural networks.
- To preprocess and transform historical stock data into suitable input sequences for time-series modeling.
- To train the LSTM model on closing price trends, optimizing it for low error and high predictive accuracy.
- To evaluate the model's performance using standard regression metrics: RMSE, MAE, and R^2 score.
- To use the trained model to generate future price predictions by iteratively forecasting multiple time steps ahead.



Datasets Used

- **<u>Source:</u>**

  Stock data is retrieved using the yfinance API, which pulls historical data directly from Yahoo Finance.
- **<u>Time Range & Stock Symbols:</u>**

  Data spans from January 1, 2020, to the present. Users can input any valid stock ticker (e.g., GOOG, AAPL) through the app's text field, allowing flexibility in stock selection.
- **<u>Features:</u>**

  The dataset includes Open, High, Low, Close, Adjusted Close, and Volume. However, only the 'Close' price is used for training and prediction to keep the model focused and reduce complexity.
- **<u>Granularity:</u>**

  Daily-level data is used to capture regular market trends and price fluctuations.
- **<u>Data Split:</u>**

  The dataset is split into training (80%) and testing (20%) segments. The last 20% helps validate model accuracy on unseen data.

Tools, Libraries, APIs Used

Python:
The core programming language used for implementation.

yfinance:
To fetch historical stock data directly from Yahoo Finance.

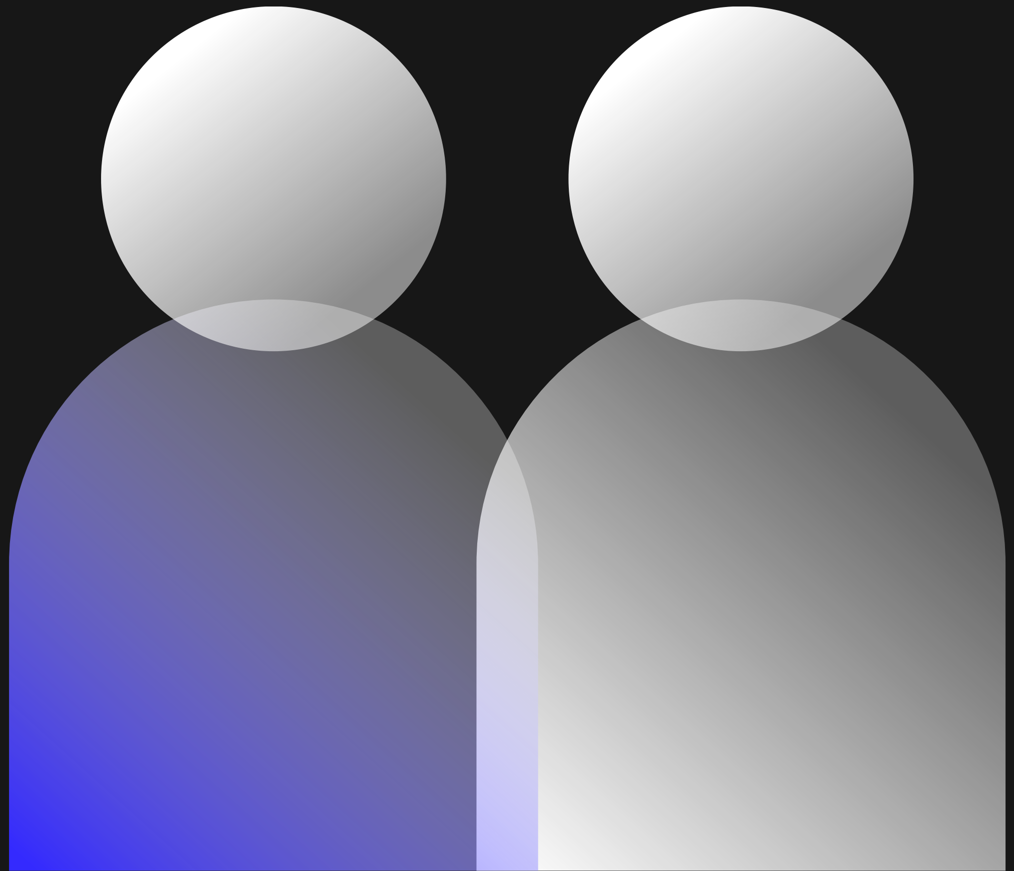
NumPy & Pandas:
For numerical operations and data manipulation.

Scikit-learn:
Used for preprocessing with MinMaxScaler and for evaluation metrics like RMSE, MAE, and R^2.

TensorFlow & Keras:
To build, train, and save the LSTM model using high-level APIs.

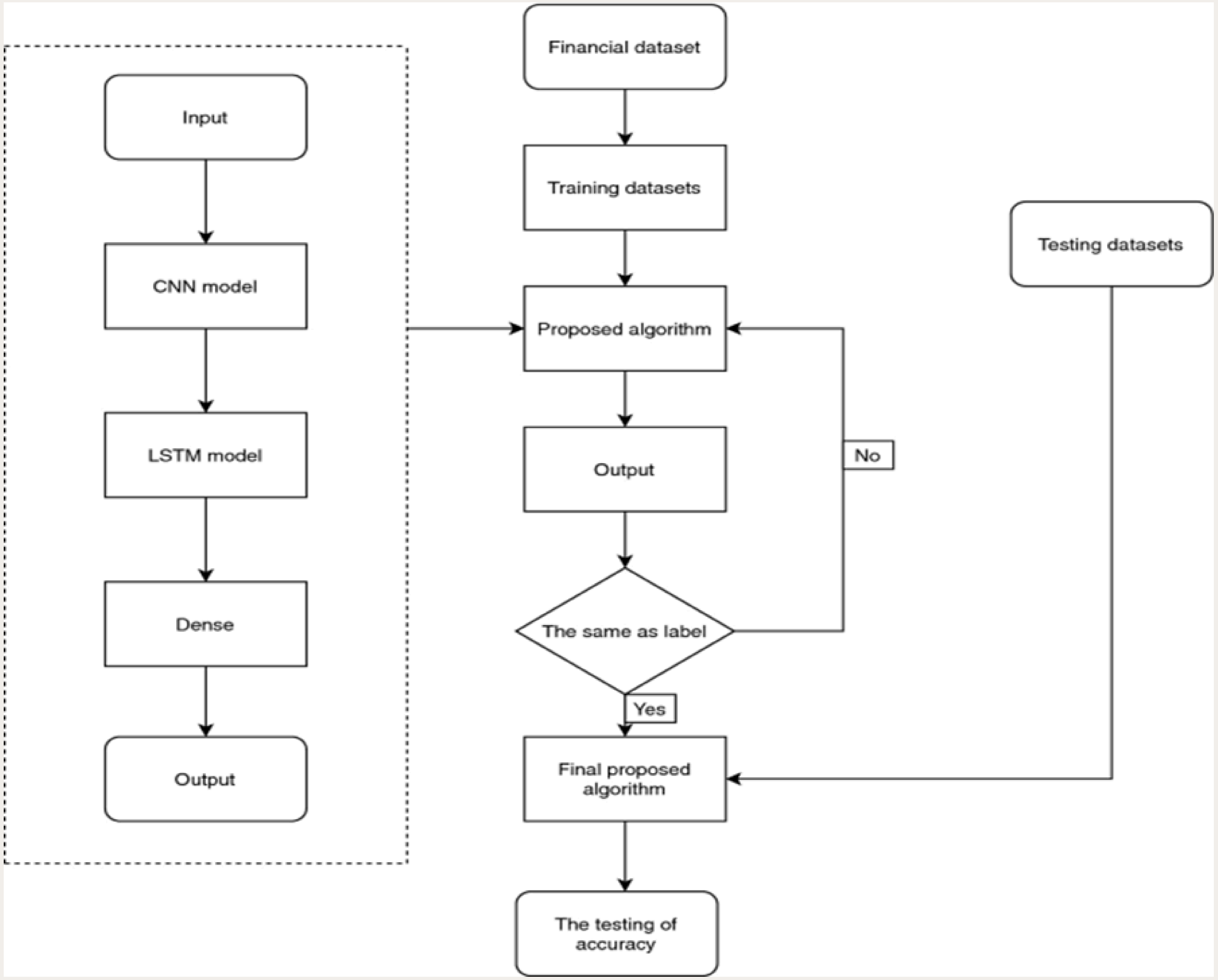
Matplotlib:
For plotting moving averages and prediction results.

Streamlit:
A Python framework used to create the interactive web application for user input, visualization, and downloading predictions.



Implementation Details:

Architecture:



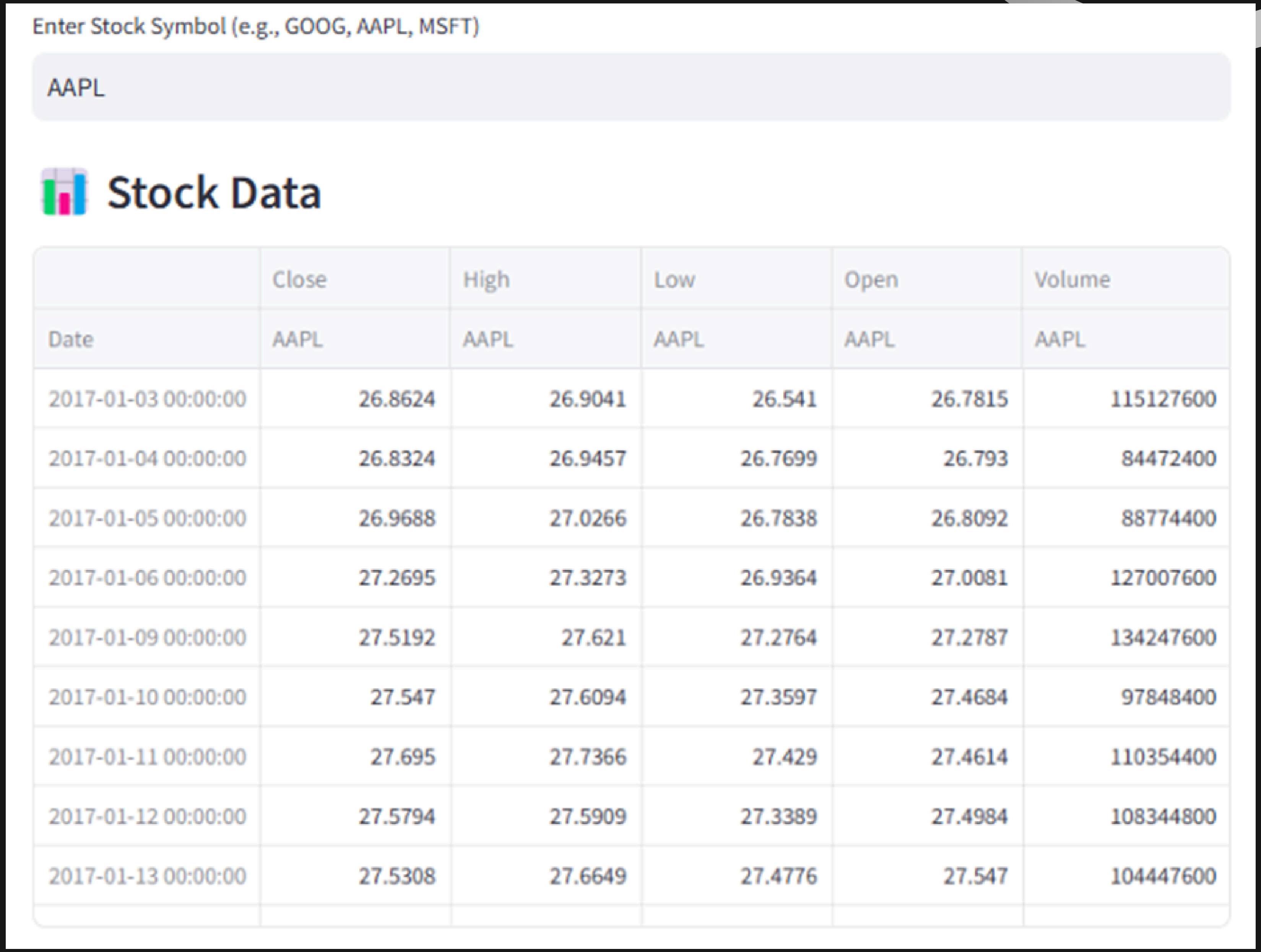
Implementation Details

a. **User Input**

- Stock Symbol: Ticker entered by the user (e.g., AAPL, TSLA).
- Forecast Horizon: Number of future days to predict.
- Training Params: User selects epochs and batch size (affects learning
  behavior).

b. **Data Collection**

- Historical data is fetched using yfinance.
- Data ranges from Jan 1, 2020, to the current date.
- Only the 'Close' price is used for modeling.



Enter Stock Symbol (e.g., GOOG, AAPL, MSFT)

AAPL

Stock Data

	Close	High	Low	Open	Volume
Date	AAPL	AAPL	AAPL	AAPL	AAPL
2017-01-03 00:00:00	26.8624	26.9041	26.541	26.7815	115127600
2017-01-04 00:00:00	26.8324	26.9457	26.7699	26.793	84472400
2017-01-05 00:00:00	26.9688	27.0266	26.7838	26.8092	88774400
2017-01-06 00:00:00	27.2695	27.3273	26.9364	27.0081	127007600
2017-01-09 00:00:00	27.5192	27.621	27.2764	27.2787	134247600
2017-01-10 00:00:00	27.547	27.6094	27.3597	27.4684	97848400
2017-01-11 00:00:00	27.695	27.7366	27.429	27.4614	110354400
2017-01-12 00:00:00	27.5794	27.5909	27.3389	27.4984	108344800
2017-01-13 00:00:00	27.5308	27.6649	27.4776	27.547	104447600

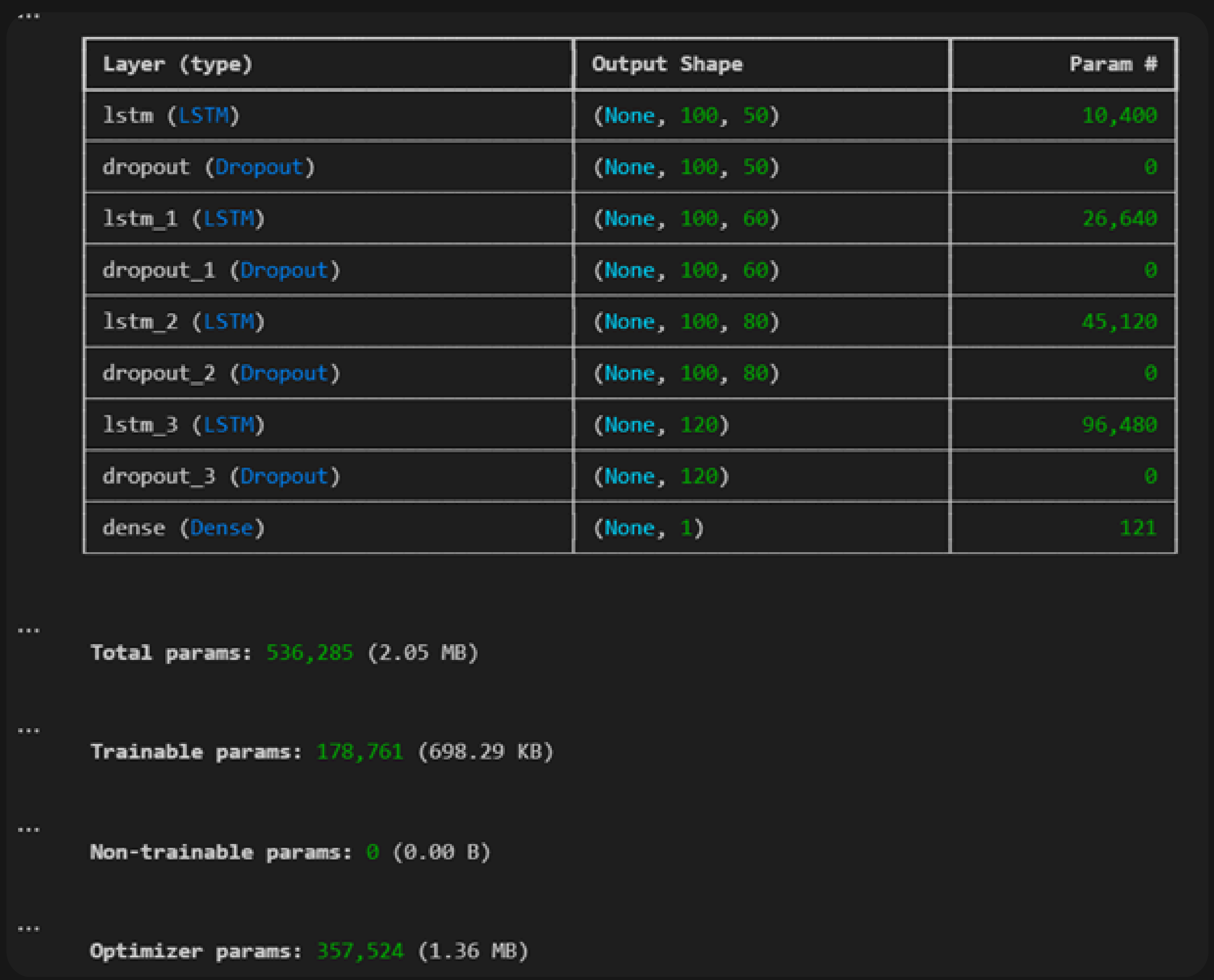
Implementation Details

c. Data Preprocessing

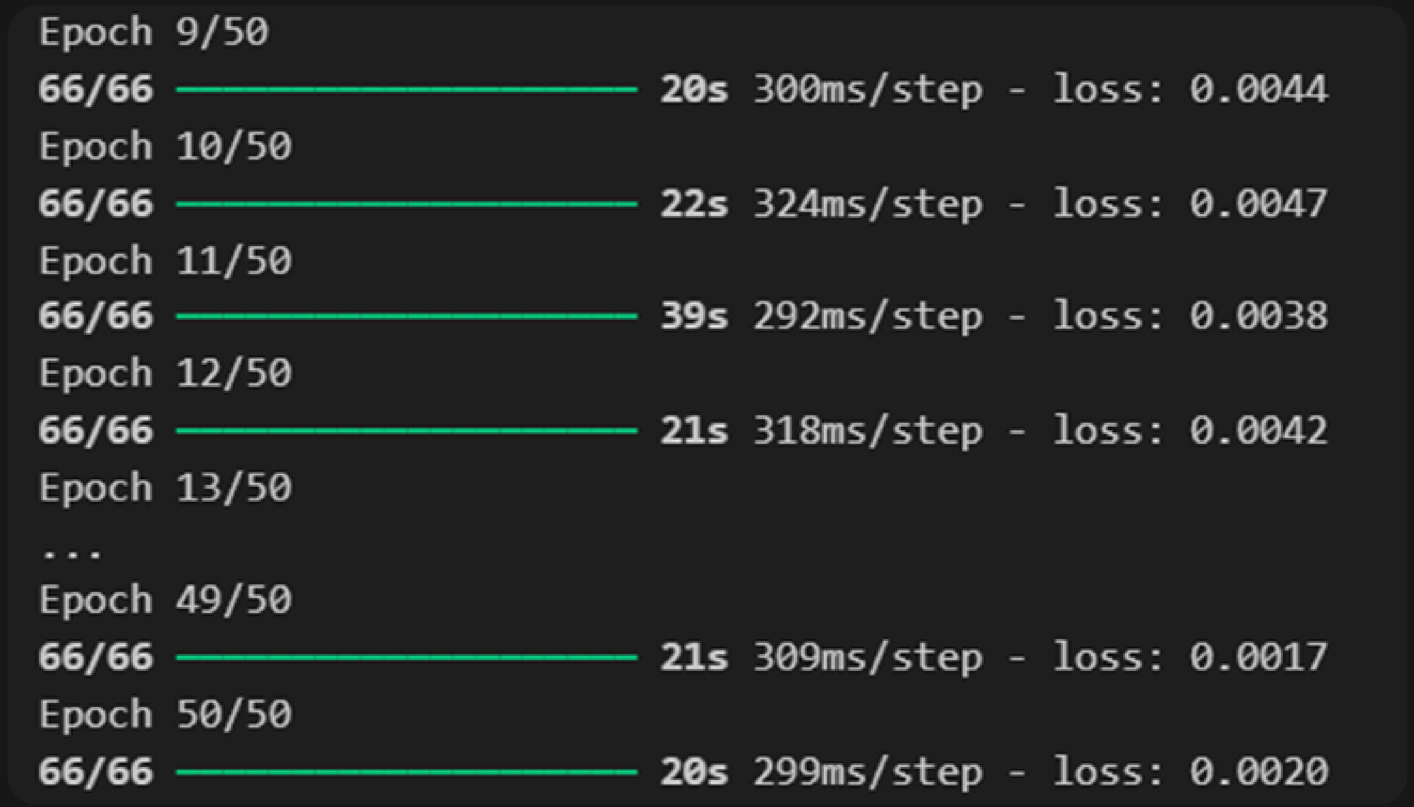
- **Normalization:** Close prices scaled to [0,1] using MinMaxScaler.
- **Windowing:** 100-day sliding window to form input sequences (X) and next-day target (Y).
- **Split:** 80% training, 20% testing.

d. Model Training

- If a model exists, it's loaded; else:
  - LSTM model with 3 stacked layers (50 units), dropout layers (0.2), and a Dense output layer is built.
  - Compiled with Adam optimizer and MSE loss.
  - Model is trained and saved.

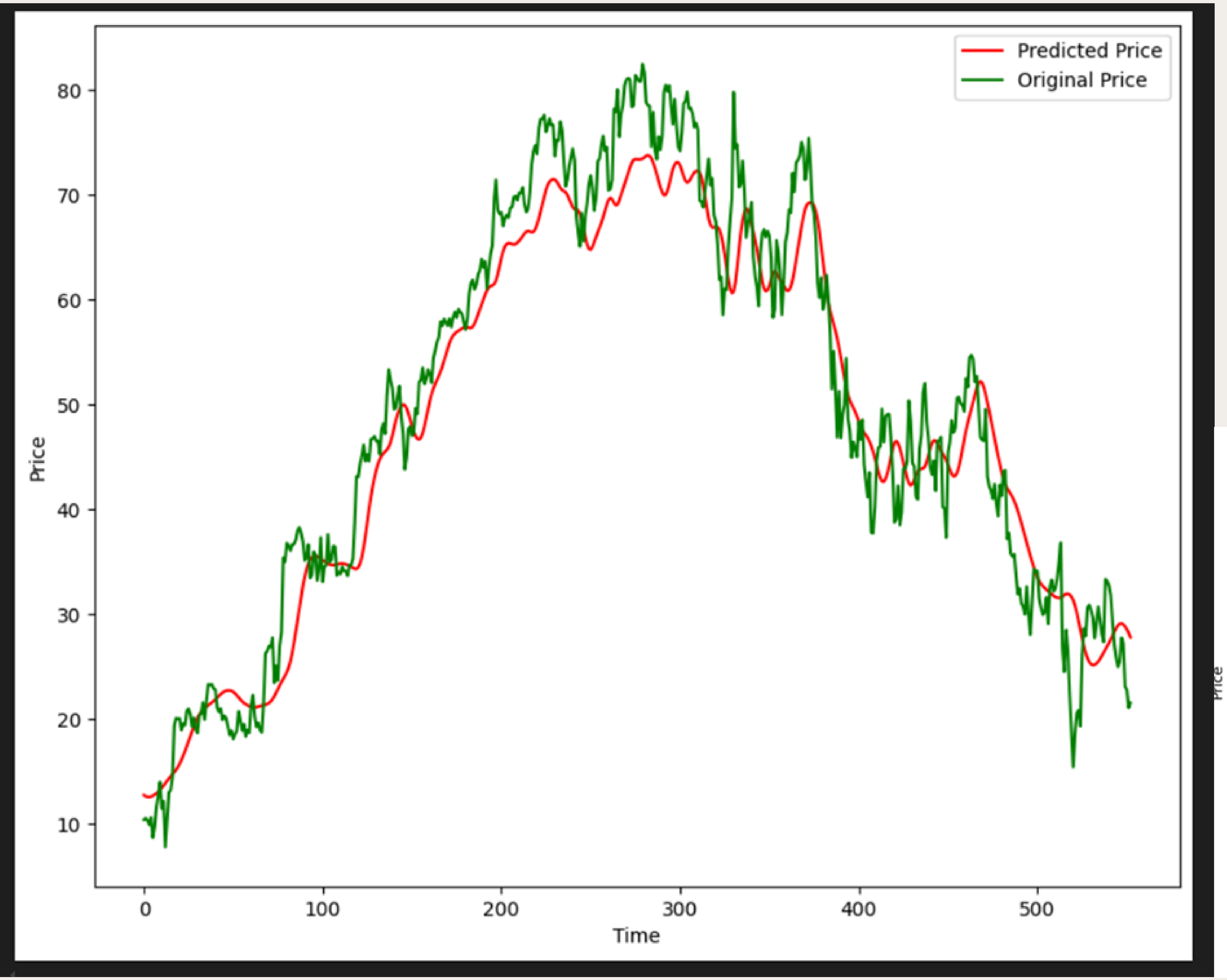


```
...
┌──────────────────────────────┬───────────────────────┬──────────┐
│ Layer (type)                 │ Output Shape          │  Param # │
├──────────────────────────────┼───────────────────────┼──────────┤
│ lstm (LSTM)                  │ (None, 100, 50)       │   10,400 │
├──────────────────────────────┼───────────────────────┼──────────┤
│ dropout (Dropout)            │ (None, 100, 50)       │        0 │
├──────────────────────────────┼───────────────────────┼──────────┤
│ lstm_1 (LSTM)                │ (None, 100, 60)       │   26,640 │
├──────────────────────────────┼───────────────────────┼──────────┤
│ dropout_1 (Dropout)          │ (None, 100, 60)       │        0 │
├──────────────────────────────┼───────────────────────┼──────────┤
│ lstm_2 (LSTM)                │ (None, 100, 80)       │   45,120 │
├──────────────────────────────┼───────────────────────┼──────────┤
│ dropout_2 (Dropout)          │ (None, 100, 80)       │        0 │
├──────────────────────────────┼───────────────────────┼──────────┤
│ lstm_3 (LSTM)                │ (None, 120)           │   96,480 │
├──────────────────────────────┼───────────────────────┼──────────┤
│ dropout_3 (Dropout)          │ (None, 120)           │        0 │
├──────────────────────────────┼───────────────────────┼──────────┤
│ dense (Dense)                │ (None, 1)             │      121 │
└──────────────────────────────┴───────────────────────┴──────────┘
...
 Total params: 536,285 (2.05 MB)
...
 Trainable params: 178,761 (698.29 KB)
...
 Non-trainable params: 0 (0.00 B)
...
 Optimizer params: 357,524 (1.36 MB)
```



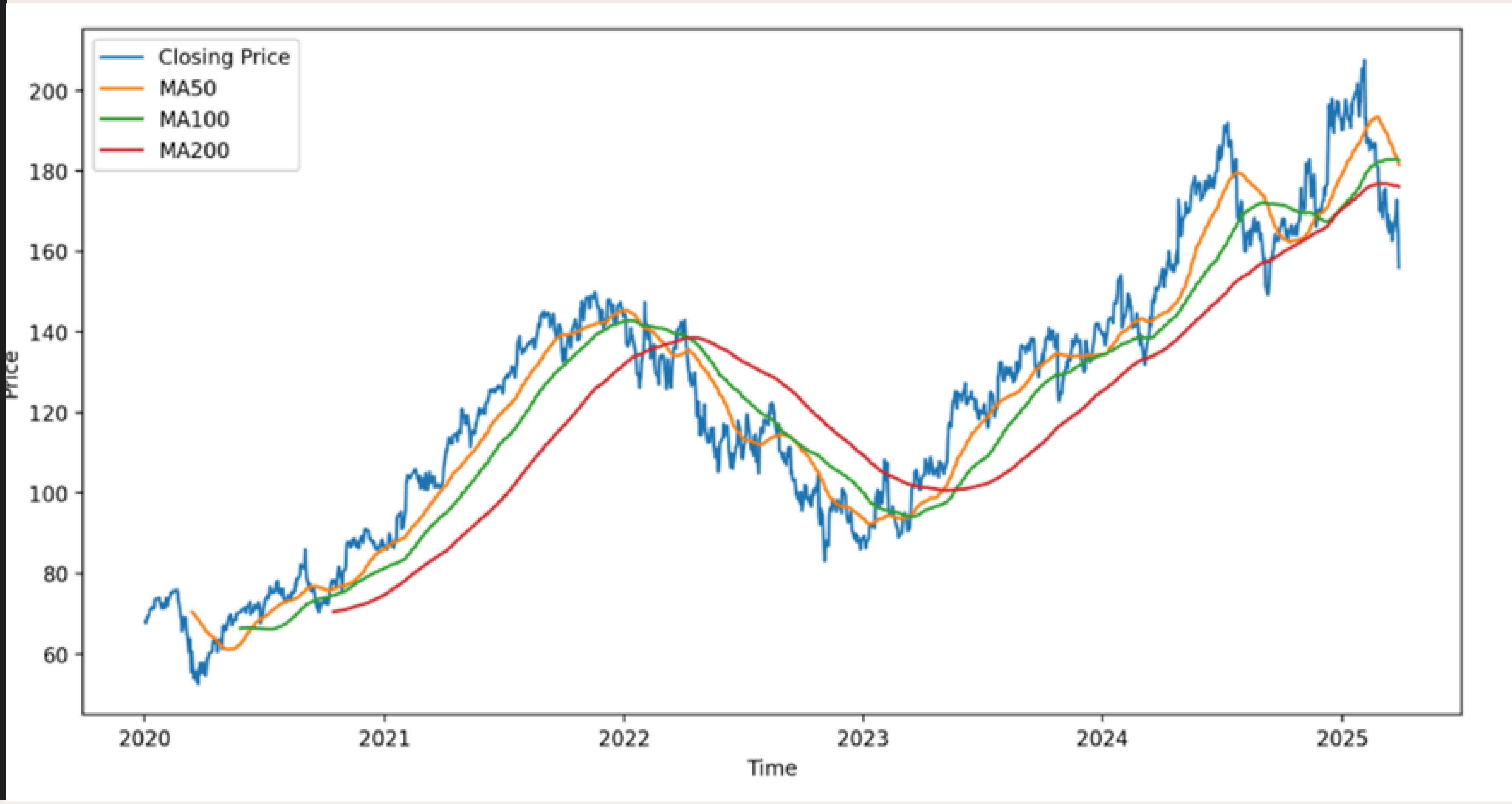
```
Epoch 9/50
66/66 ━━━━━━━━━━━━━━━━━━━━ 20s 300ms/step - loss: 0.0044
Epoch 10/50
66/66 ━━━━━━━━━━━━━━━━━━━━ 22s 324ms/step - loss: 0.0047
Epoch 11/50
66/66 ━━━━━━━━━━━━━━━━━━━━ 39s 292ms/step - loss: 0.0038
Epoch 12/50
66/66 ━━━━━━━━━━━━━━━━━━━━ 21s 318ms/step - loss: 0.0042
Epoch 13/50
...
Epoch 49/50
66/66 ━━━━━━━━━━━━━━━━━━━━ 21s 309ms/step - loss: 0.0017
Epoch 50/50
66/66 ━━━━━━━━━━━━━━━━━━━━ 20s 299ms/step - loss: 0.0020
```

Implementation



e. Prediction on Test Set:

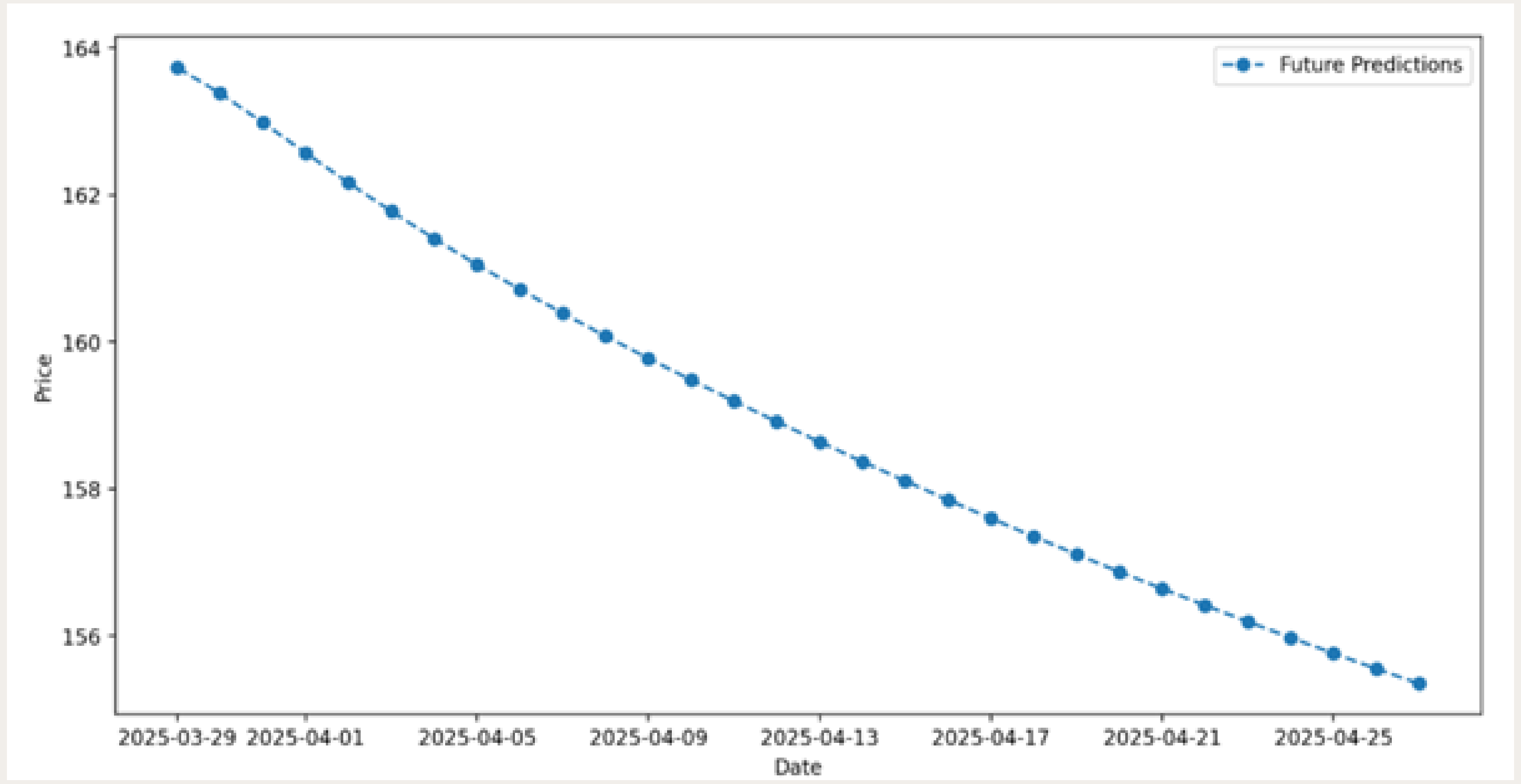
- Combines last 100 train values with test data.
- Predicts test targets, reverse normalization, and plots predicted vs actual

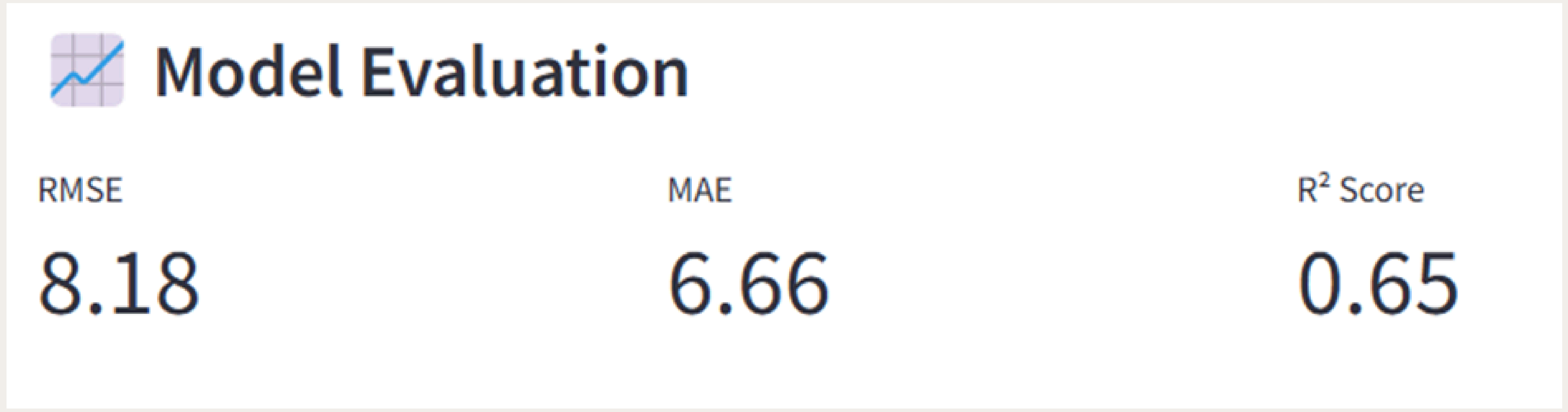


Implementation

f. Future Forecasting:

- Uses last 100 known values to predict 1 day ahead repeatedly.
- New predictions are fed into input window recursively.
- Final output is inverse-scaled for readability.





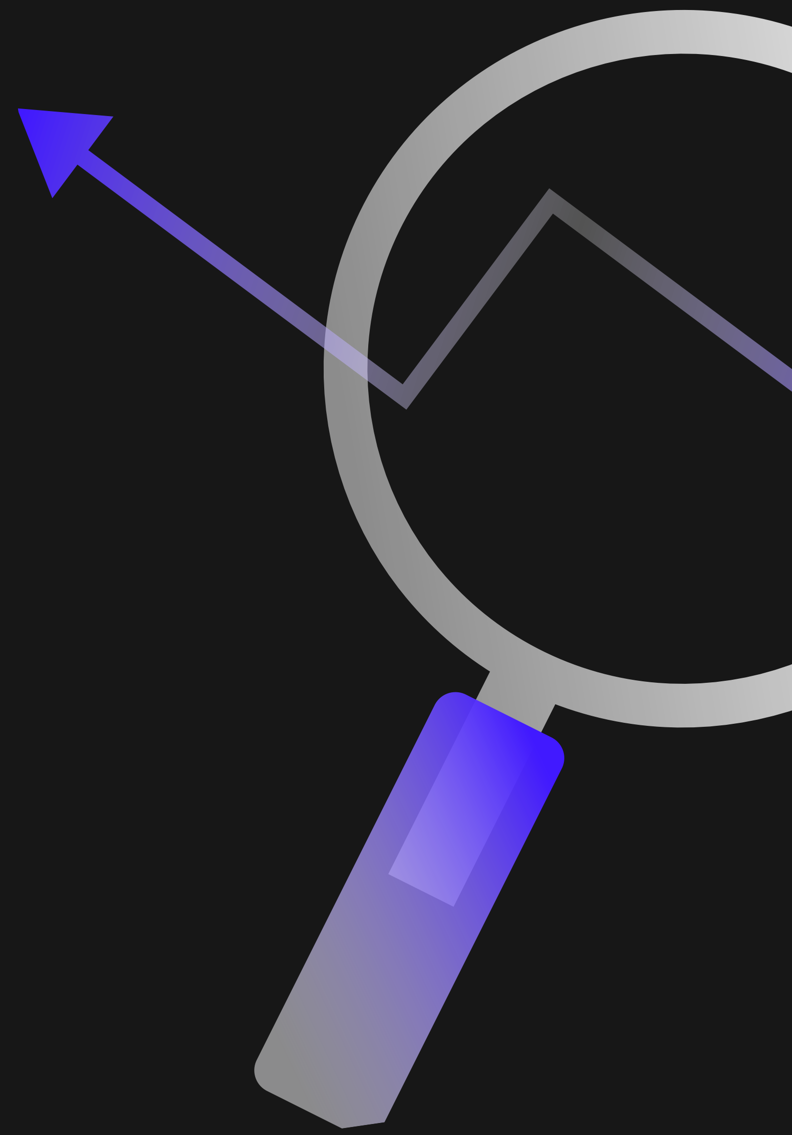
Model Evaluation

RMSE	MAE	R^2 Score
8.18	6.66	0.65

g. Evaluation Metrics:

- **RMSE:** Penalizes large errors.
- **MAE:** Measures average error.
- **R^2 Score:** Indicates model fit (closer to 1 = better).

Result Analysis



1. Prediction Accuracy
The predicted prices (red) align closely with actual prices (green), especially during clear upward and downward trends —showing the model has effectively captured price direction.

2. Evaluation Metrics

- RMSE: 8.18 — moderate average prediction error.
- MAE: 6.66 — low average absolute error.
- R^2 Score: 0.65 — captures 65% of price variance

3. Model Strengths

- Accurately tracks trends in stable phases (e.g., time step 50–300).
- Smooth predictions reduce market noise and provide clearer forecasts.

4. Forecast Visualization
Visual plots clearly illustrate how predictions mirror actual movements, enhancing interpretability for users.

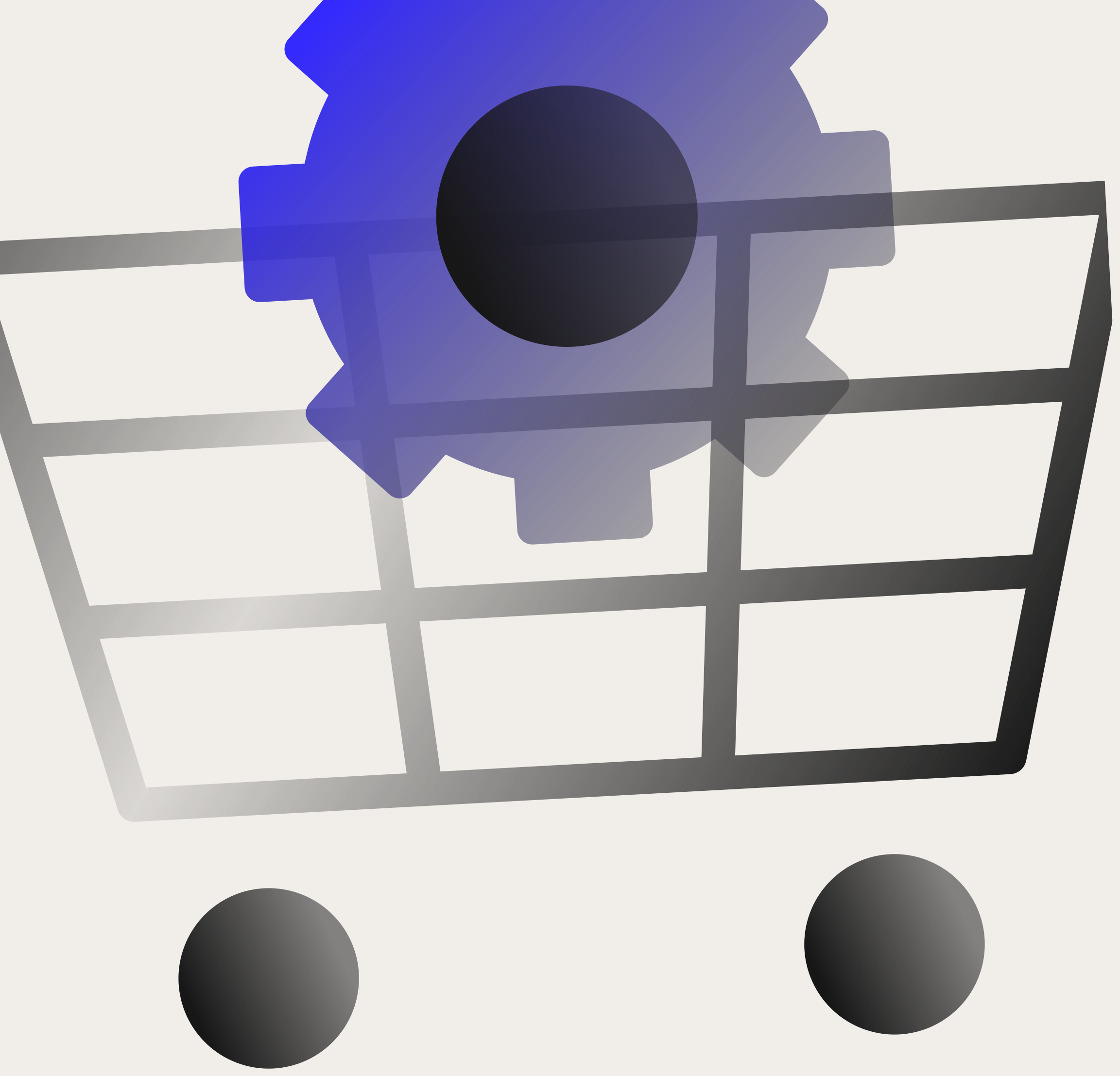
5. Generalization
The model performs consistently across a broad time range, reflecting good generalization on unseen data.

REFERENCES

[1] A. Gupta, M. Patel, Akansha, V. Pratap, and K. Joshi, "Stock Market Prediction using Machine Learning Techniques: A Systematic Review," 2023 International Conference on Power, Instrumentation, Control, and Computing (PICC), IEEE, 2023.

[2] "Stock Price Prediction Model Based on RBF-SVM Algorithm," 2024 International Conference on Computer Engineering and Intelligent Control (ICCEIC), IEEE, 2024.

[3] S. Khapre, P. A. Gunturu, R. Joseph, and E. S. Revant, "Survey of Stock Market Price Prediction Trends using Machine Learning Techniques," 2023 International Conference on Artificial Intelligence and Applications (ICAIA), IEEE, 2023.



Thank You