Stock Price Prediction with LSTM Networks

Ajay Kumar Pudi

January 8, 2025

Contents

1 Introduction **2**

2 Installation of Required Modules **2**

3 Data Loading **2**

4 Exploratory Data Analysis (EDA) **3**

5 Data Preprocessing **3**

6 LSTM Model Construction **4**

7 Model Training and Evaluation **4**

8 Model Performance Metrics **5**

9 Conclusion **6**

1 Introduction

Stock price prediction is a complex and challenging task due to the volatility and dynamic nature of financial markets. Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) capable of learning long-term dependencies, making them suitable for time series prediction tasks, such as forecasting stock prices.

2 Installation of Required Modules

Ensure that all necessary Python libraries are installed using the following command:

```
pip install yfinance pandas numpy matplotlib seaborn tensorflow scikit-learn
```

3 Data Loading

We use the Yahoo Finance API to download historical stock price data. The following Python code snippet demonstrates how to fetch the data:

```
import yfinance as yf
import pandas as pd

symbol = 'AAPL'
start_date = '2010-01-01'
end_date = '2023-12-31'
```

```
data = yf.download(symbol, start=start_date, end=end_date)
print(data.head())
```

4 Exploratory Data Analysis (EDA)

Analyzing the stock price data helps to understand its structure and distribution. Visualizing the closing prices over time can reveal trends and patterns.

```
import matplotlib.pyplot as plt

plt.figure(figsize=(14, 7))
plt.plot(data['Close'])
plt.title('Closing Price Over Time')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.show()
```

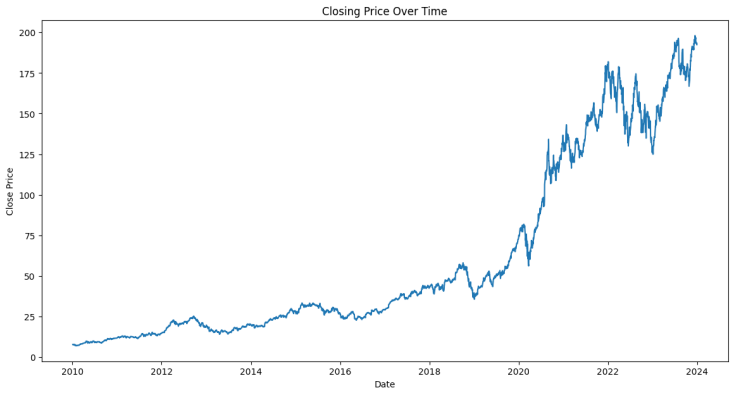

Figure 1: Closing Price Over Time

5 Data Preprocessing

Prepare the data for input into the LSTM model by normalizing and creating sequences. The code snippet below shows how to perform this:

```
from sklearn.preprocessing import MinMaxScaler
import numpy as np

scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(data['Close'].values.reshape(-1, 1))
```

```
def create_sequences(data, sequence_length):
    sequences = []
    for i in range(len(data) - sequence_length):
        sequences.append(data[i:i + sequence_length])
    return np.array(sequences)

sequence_length = 60
sequences = create_sequences(scaled_data, sequence_length)

x = sequences[:, :-1]
y = sequences[:, -1]

split = int(0.8 * len(x))
x_train, x_test = x[:split], x[split:]
y_train, y_test = y[:split], y[split:]
```

6 LSTM Model Construction

The LSTM model architecture includes LSTM layers, Dropout layers to reduce overfitting, and a Dense output layer. The following code snippet demonstrates building and compiling the LSTM model:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout

model = Sequential([
    LSTM(50, return_sequences=True, input_shape=(x_train.shape[1], 1)),
    Dropout(0.2),
    LSTM(50, return_sequences=False),
    Dropout(0.2),
    Dense(1)
])

model.compile(optimizer='adam', loss='mean_squared_error')
```

7 Model Training and Evaluation

The model is trained using the training data and validated with the test data. After training, predictions are made on the test set, and the results are visualized.

```
history = model.fit(x_train, y_train, epochs=20, batch_size=32, validation_data=(x_test, y_t

predictions = model.predict(x_test)
```

```
predictions = scaler.inverse_transform(predictions)

dates = data.index[-len(predictions):]

plt.figure(figsize=(14, 7))
plt.plot(dates, scaler.inverse_transform(y_test.reshape(-1, 1)), label='Actual Price')
plt.plot(dates, predictions, label='Predicted Price')
plt.title('Stock Price Prediction')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.legend()
plt.show()
```

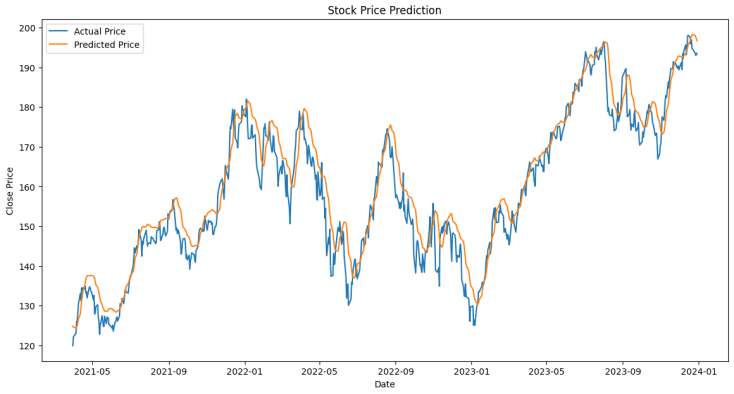

Figure 2: Predicted vs Actual Stock Prices

8 Model Performance Metrics

We evaluate the model's performance using Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).

```
from sklearn.metrics import mean_absolute_error, mean_squared_error

mae = mean_absolute_error(scaler.inverse_transform(y_test.reshape(-1, 1)), predictions)
mse = mean_squared_error(scaler.inverse_transform(y_test.reshape(-1, 1)), predictions)
rmse = np.sqrt(mse)

print(f'MAE: {mae}, MSE: {mse}, RMSE: {rmse}')
```

9 Conclusion

The LSTM model provides a method for predicting stock prices based on historical data. The results demonstrate the model's capability to capture trends, although challenges like market volatility and unforeseen events can impact its accuracy. Future work could explore improving the model by incorporating additional features or using advanced architectures.