A.Y.: 2023-24 **Class/ Sem: T.Y.B.Tech/ Sem-VI** **Sub:** FMC

Experiment 4

Bhuvi Ghosh
60009210191

Aim: Use LSTM to analyse historical stock

Objective:

- Apply Long Short-Term Memory (LSTM) networks for the analysis of historical stock data.

Theory:

1. Long Short-Term Memory (LSTM): LSTM is a type of recurrent neural network (RNN) architecture designed to remember information for long periods. It addresses the vanishing gradient problem of traditional RNNs, allowing it to learn and remember over long sequences.
2. Stock Price Prediction: Stock price prediction is a common application of machine learning in finance. LSTM networks can be used to predict future stock prices based on historical price data. By training the LSTM on historical stock prices, it can learn patterns and trends that may help predict future price movements.
3. Data Preparation: Historical stock price data is typically prepared by creating a time series dataset. Each data point in the dataset consists of features (e.g., past stock prices, trading volume) and the target variable (e.g., future stock price).
4. Model Training: The LSTM model is trained on the historical stock price dataset. During training, the model learns the patterns and relationships in the data.
5. Model Evaluation: After training, the model is evaluated using a separate test dataset to assess its performance in predicting future stock prices. Common evaluation metrics include mean squared error (MSE) and mean absolute error (MAE).
6. Interpretation: The results of the LSTM analysis can be used to understand the predictive power of the model and gain insights into the behaviour of stock prices over time.

Conclusion

Hands-on experience with LSTM networks and their application in financial analysis.

Lab Experiment to be done by students:

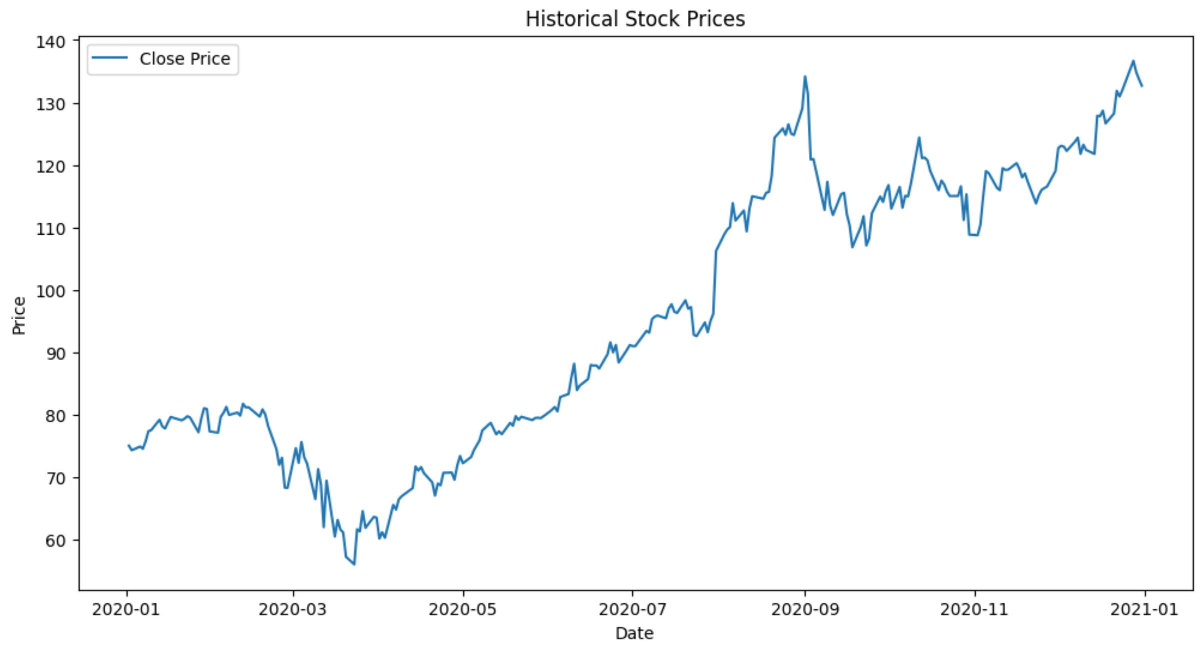
1. Import data from Yfinance
2. Plot Historic data for specific stock in a specific duration.
3. Create the training data set
4. Create the scaled training data set
5. Build the LSTM model for prediction
6. Create the testing data set
7. Plot and Visualize the data
8. Evaluate the model's performance using metrics such as Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE)

```
import yfinance as yf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import LSTM, Dense
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

```
stock_symbol = 'AAPL'
start_date = '2020-01-01'
end_date = '2021-01-01'
```

```
stock_data = yf.download(stock_symbol, start=start_date, end=end_date)
```

```
plt.figure(figsize=(12, 6))
plt.plot(stock_data['Close'], label='Close Price')
plt.title('Historical Stock Prices')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.show()
```



```
def create_dataset(data, time_step=1):
    X, y = [], []

        X.append(data[i:(i + time_step), 0])
        y.append(data[i + time_step, 0])
    return np.array(X), np.array(y)
```

```
time_step = 60
data = stock_data['Close'].values.reshape(-1, 1)
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(data)
```

```
X_train, y_train = create_dataset(scaled_data, time_step)

X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
```

```
model = Sequential()
model.add(LSTM(units=50, return_sequences=True,
input_shape=(X_train.shape[1], 1)))
model.add(LSTM(units=50, return_sequences=False))
model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mean_squared_error')
```

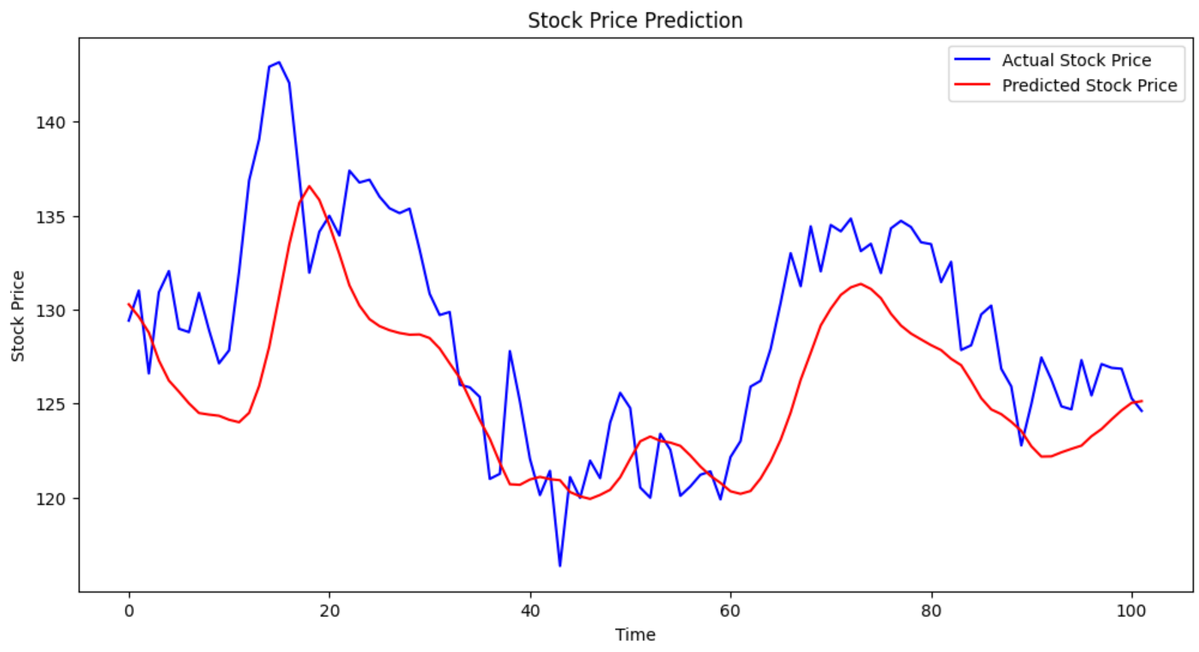
```
model.fit(X_train, y_train, epochs=100, batch_size=32)

test_data = yf.download(stock_symbol, start='2021-01-01', end='2021-06-
01')
actual_prices = test_data['Close'].values.reshape(-1, 1)
total_data = pd.concat((stock_data['Close'], test_data['Close']),
axis=0)
inputs = total_data[len(total_data) - len(test_data) -
time_step:].values
inputs = inputs.reshape(-1, 1)
inputs = scaler.transform(inputs)
```

```
X_test = []
for i in range(time_step, len(inputs)):
    X_test.append(inputs[i - time_step:i, 0])
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
```

```
predicted_prices = model.predict(X_test)
predicted_prices = scaler.inverse_transform(predicted_prices)
```

```
plt.figure(figsize=(12, 6))
plt.plot(actual_prices, color='blue', label='Actual Stock Price')
plt.plot(predicted_prices, color='red', label='Predicted Stock Price')
plt.title('Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('Stock Price')
plt.legend()
plt.show()
```



```
mae = mean_absolute_error(actual_prices, predicted_prices)
rmse = mean_squared_error(actual_prices, predicted_prices,
squared=False)

print("Mean Absolute Error (MAE):", mae)
print("Root Mean Squared Error (RMSE):", rmse)
```

```
Mean Absolute Error (MAE): 3.716269848393459
```

```
Root Mean Squared Error (RMSE): 4.7161629042408375
```