Deep Learning Model to Predict Infosys Stock Prices

Code:

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

def fetch_stock_data(ticker, period='5y'):
        stock_data = yf.Ticker(ticker)
        df = stock_data.history(period=period)
        df = df[['Close']]  # Use only the closing prices
        return df

def preprocess_data(df, lookback=60):
        scaler = MinMaxScaler(feature_range=(0, 1))
        scaled_data = scaler.fit_transform(df)

        X, y = [], []
        for i in range(lookback, len(scaled_data)):
        X.append(scaled_data[i-lookback:i, 0])
        y.append(scaled_data[i, 0])

        X, y = np.array(X), np.array(y)
        X = np.reshape(X, (X.shape[0], X.shape[1], 1))
        return X, y, scaler

def create_rnn_model(input_shape):
        model = Sequential()
        model.add(SimpleRNN(units=50, return_sequences=False,
input_shape=input_shape))
        model.add(Dense(units=1))
        model.compile(optimizer='adam', loss='mean_squared_error')
        return model

# Create the LSTM model
def create_lstm_model(input_shape):
        model = Sequential()
        model.add(LSTM(units=50, return_sequences=False, input_shape=input_shape))
```

```
        model.add(Dense(units=1))
        model.compile(optimizer='adam', loss='mean_squared_error')
        return model

# Create the Bidirectional LSTM model
def create_bidirectional_lstm_model(input_shape):
        model = Sequential()
        model.add(Bidirectional(LSTM(units=50, return_sequences=False),
input_shape=input_shape))
        model.add(Dense(units=1))
        model.compile(optimizer='adam', loss='mean_squared_error')
        return model

# Predict future values using the trained model
def predict_future_values(model, data, future_steps, scaler):
        predictions = []
        last_sequence = data[-1]  # Use the last available sequence for prediction

        for _ in range(future_steps):
        prediction = model.predict(np.reshape(last_sequence, (1, last_sequence.shape[0],
1)))
        predictions.append(prediction[0, 0])
        last_sequence = np.roll(last_sequence, -1)
        last_sequence[-1] = prediction

        predictions = scaler.inverse_transform(np.array(predictions).reshape(-1, 1))
        return predictions

df = fetch_stock_data('INFY.NS')

split = int(len(df) * 0.9)
train_data, test_data = df[:split], df[split:]

lookback = 60  # We look at the past 60 days
X_train, y_train, scaler = preprocess_data(train_data, lookback)
X_test, y_test, _ = preprocess_data(test_data, lookback)

rnn_model = create_rnn_model((X_train.shape[1], 1))
lstm_model = create_lstm_model((X_train.shape[1], 1))
bi_lstm_model = create_bidirectional_lstm_model((X_train.shape[1], 1))

rnn_model.fit(X_train, y_train, epochs=10, batch_size=len(X_train))
lstm_model.fit(X_train, y_train, epochs=10, batch_size=len(X_train))
bi_lstm_model.fit(X_train, y_train, epochs=10, batch_size=len(X_train))

future_steps_1m = 30
future_steps_6m = 180
future_steps_1y = 365
```

```
rnn_predictions_1m = predict_future_values(rnn_model, X_test, future_steps_1m, scaler)
lstm_predictions_1m = predict_future_values(lstm_model, X_test, future_steps_1m, scaler)
bi_lstm_predictions_1m = predict_future_values(bi_lstm_model, X_test, future_steps_1m,
scaler)

rnn_predictions_6m = predict_future_values(rnn_model, X_test, future_steps_6m, scaler)
lstm_predictions_6m = predict_future_values(lstm_model, X_test, future_steps_6m, scaler)
bi_lstm_predictions_6m = predict_future_values(bi_lstm_model, X_test, future_steps_6m,
scaler)

rnn_predictions_1y = predict_future_values(rnn_model, X_test, future_steps_1y, scaler)
lstm_predictions_1y = predict_future_values(lstm_model, X_test, future_steps_1y, scaler)
bi_lstm_predictions_1y = predict_future_values(bi_lstm_model, X_test, future_steps_1y,
scaler)

print("Predicted closing stock value after 1 month (30 days):")
print("RNN:", rnn_predictions_1m[-1])
print("LSTM:", lstm_predictions_1m[-1])
print("Bidirectional LSTM:", bi_lstm_predictions_1m[-1])

print("\nPredicted closing stock value after 6 months (180 days):")
print("RNN:", rnn_predictions_6m[-1])
print("LSTM:", lstm_predictions_6m[-1])
print("Bidirectional LSTM:", bi_lstm_predictions_6m[-1])

print("\nPredicted closing stock value after 1 year (365 days):")
print("RNN:", rnn_predictions_1y[-1])
print("LSTM:", lstm_predictions_1y[-1])
print("Bidirectional LSTM:", bi_lstm_predictions_1y[-1])

plt.figure(figsize=(14, 7))
plt.plot(range(1, future_steps_1y + 1), rnn_predictions_1y, label='RNN Prediction (1 year)',
linestyle='--')
plt.plot(range(1, future_steps_1y + 1), lstm_predictions_1y, label='LSTM Prediction (1 year)',
linestyle='--')
plt.plot(range(1, future_steps_1y + 1), bi_lstm_predictions_1y, label='Bidirectional LSTM
Prediction (1 year)', linestyle='--')

plt.title('Stock Price Prediction for Infosys using RNN, LSTM, and Bidirectional LSTM')
plt.xlabel('Days into the Future')
plt.ylabel('Stock Price (INR)')
plt.legend()
plt.show()
```

Output:

Predicted closing stock value after 1 month (30 days):
RNN: [781.0751]
LSTM: [894.6687]
Bidirectional LSTM: [2091.6877]

Predicted closing stock value after 6 months (180 days):
RNN: [-111.55409]
LSTM: [810.9517]
Bidirectional LSTM: [2674.0408]

Predicted closing stock value after 1 year (365 days):
RNN: [-152.16139]
LSTM: [810.95026]
Bidirectional LSTM: [2787.2603]