Code:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense, Dropout

# Load the stock price data

data = pd.read\_csv('stock\_data.csv') # Replace 'stock\_data.csv' with your dataset

# Select the 'Close' prices as the target variable

prices = data['Close'].values.reshape(-1, 1)

# Normalize the data

scaler = MinMaxScaler(feature\_range=(0, 1))

prices\_scaled = scaler.fit\_transform(prices)

# Split the data into training and test sets

train\_size = int(len(prices\_scaled) \* 0.8)

train\_data = prices\_scaled[:train\_size]

test\_data = prices\_scaled[train\_size:]

# Create sequences of data for training

def create\_sequences(data, sequence\_length):

X, y = [], []

for i in range(len(data) - sequence\_length):

X.append(data[i:i+sequence\_length])

y.append(data[i+sequence\_length])

return np.array(X), np.array(y)

sequence\_length = 10 # You can adjust this value

X\_train, y\_train = create\_sequences(train\_data, sequence\_length)

X\_test, y\_test = create\_sequences(test\_data, sequence\_length)

# Build the LSTM model

model = Sequential()

model.add(LSTM(units=50, return\_sequences=True, input\_shape=(X\_train.shape[1], 1)))

model.add(LSTM(units=50))

model.add(Dense(units=1))

model.compile(optimizer='adam', loss='mean\_squared\_error')

# Train the model

model.fit(X\_train, y\_train, epochs=10, batch\_size=32)

# Evaluate the model

train\_loss = model.evaluate(X\_train, y\_train, verbose=0)

test\_loss = model.evaluate(X\_test, y\_test, verbose=0)

print(f"Train Loss: {train\_loss:.4f}, Test Loss: {test\_loss:.4f}")

# Make predictions

train\_predictions = model.predict(X\_train)

test\_predictions = model.predict(X\_test)

# Inverse transform the predictions to the original scale

train\_predictions = scaler.inverse\_transform(train\_predictions)

test\_predictions = scaler.inverse\_transform(test\_predictions)

# Plot the results

plt.figure(figsize=(12, 6))

plt.plot(prices, label='Actual Prices', color='b')

plt.plot(range(sequence\_length, train\_size), train\_predictions, label='Train Predictions', color='g')

plt.plot(range(train\_size + sequence\_length, len(prices)), test\_predictions, label='Test Predictions', color='r')

plt.legend()

plt.show()

Output:

Epoch 1/10

213/213 [==============================] - 8s 14ms/step - loss: 5.6418e-04

Epoch 2/10

213/213 [==============================] - 3s 13ms/step - loss: 4.2065e-05

Epoch 3/10

213/213 [==============================] - 3s 15ms/step - loss: 4.1125e-05

Epoch 4/10

213/213 [==============================] - 6s 27ms/step - loss: 4.0485e-05

Epoch 5/10

213/213 [==============================] - 6s 27ms/step - loss: 4.4396e-05

Epoch 6/10

213/213 [==============================] - 5s 26ms/step - loss: 3.7652e-05

Epoch 7/10

213/213 [==============================] - 3s 14ms/step - loss: 3.3948e-05

Epoch 8/10

213/213 [==============================] - 3s 13ms/step - loss: 3.4644e-05

Epoch 9/10

213/213 [==============================] - 3s 14ms/step - loss: 3.2606e-05

Epoch 10/10

213/213 [==============================] - 4s 18ms/step - loss: 2.8910e-05

Train Loss: 0.0000, Test Loss: 0.0008

213/213 [==============================] - 2s 4ms/step

53/53 [==============================] - 0s 5ms/step

