BHARAT DATA SCIENCE INTERNSHIP

TASK-1: Stock Prediction

Take stock price of any company you want and predicts its price by using LSTM. Use only Jupyter notebook code.

In [2]:
```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
from sklearn.preprocessing import MinMaxScaler
```

In [4]:
```
# Load the historical stock price data
df=pd.read_csv(r"C:\Users\harsh\Downloads\datasetsandcodefilesstockmarketprediction\tesla.csv")
```

In [5]:
```
df
```

Out[5]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	29-06-2010	19.000000	25.000000	17.540001	23.889999	23.889999	18766300
1	30-06-2010	25.790001	30.420000	23.299999	23.830000	23.830000	17187100
2	01-07-2010	25.000000	25.920000	20.270000	21.959999	21.959999	8218800
3	02-07-2010	23.000000	23.100000	18.709999	19.200001	19.200001	5139800
4	06-07-2010	20.000000	20.000000	15.830000	16.110001	16.110001	6866900
...	...	...	...	...	...	...	...
2188	11-03-2019	283.519989	291.279999	280.500000	290.920013	290.920013	7392300
2189	12-03-2019	286.489990	288.070007	281.059998	283.359985	283.359985	7504100
2190	13-03-2019	283.899994	291.989990	282.700012	288.959991	288.959991	6844700
2191	14-03-2019	292.450012	295.390015	288.290009	289.959991	289.959991	7074200
2192	15-03-2019	283.510010	283.723999	274.399994	275.429993	275.429993	14758243

2193 rows × 7 columns

]:

In [6

```
df.head()
```

Out[6]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	29-06-2010	19.000000	25.00	17.540001	23.889999	23.889999	18766300
1	30-06-2010	25.790001	30.42	23.299999	23.830000	23.830000	17187100
2	01-07-2010	25.000000	25.92	20.270000	21.959999	21.959999	8218800
3	02-07-2010	23.000000	23.10	18.709999	19.200001	19.200001	5139800
4	06-07-2010	20.000000	20.00	15.830000	16.110001	16.110001	6866900

In [7]:

```
df.shape
```

Out[7]: (2193, 7)

In [8]:

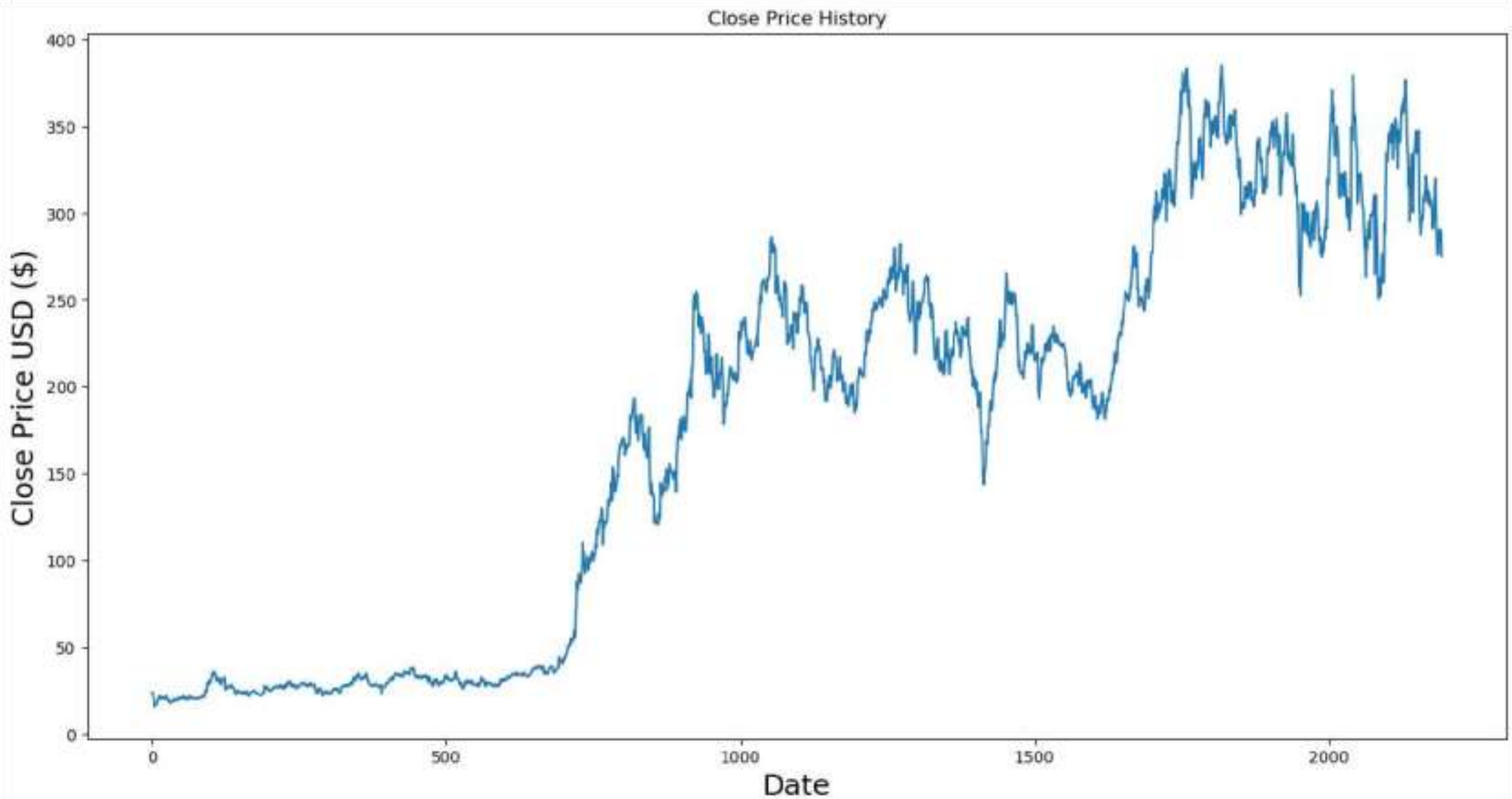
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2193 entries, 0 to 2192
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       2193 non-null   object
 1   Open       2193 non-null   float64
 2   High       2193 non-null   float64
 3   Low        2193 non-null   float64
 4   Close      2193 non-null   float64
 5   Adj Close  2193 non-null   float64
 6   Volume     2193 non-null   int64
dtypes: float64(5), int64(1), object(1)
memory usage: 120.1+ KB
```

]:

In [9

```
#Ploting the closing price of the stock to visualize the trend
plt.figure(figsize=(16,8))
plt.title('Close Price History')
plt.plot(df['Close'])
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.show()
```



In [10]:

```
#we need to preprocess the data before feeding it into the LSTM.
data = df.filter(['Close']).values
```

]:

In [11

```
#we will normalize the data between 0 and 1 using the MinMaxScaler:
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(data)
```

In [12]:

```
#To Train the Dataset
train_data = scaled_data[:int(len(scaled_data)*0.8)]
x_train = []
y_train = []

for i in range(60, len(train_data)):
    x_train.append(train_data[i-60:i, 0])
    y_train.append(train_data[i, 0])

x_train, y_train = np.array(x_train), np.array(y_train)
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
```

]:

In [13

```
#Build the LSTM model
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(x_train.shape[1], 1)))
model.add(Dropout(0.2))
model.add(LSTM(50, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(50))
model.add(Dropout(0.2))
model.add(Dense(1))

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

model.fit(x_train, y_train, epochs=50, batch_size=32)
```

```
Epoch 1/50
53/53 [==============================] - 20s 108ms/step - loss: 0.0236
Epoch 2/50
53/53 [==============================] - 5s 101ms/step - loss: 0.0036
Epoch 3/50
53/53 [==============================] - 6s 104ms/step - loss: 0.0030
Epoch 4/50
53/53 [==============================] - 6s 108ms/step - loss: 0.0030
Epoch 5/50
53/53 [==============================] - 6s 106ms/step - loss: 0.0031
Epoch 6/50
53/53 [==============================] - 5s 100ms/step - loss: 0.0025
Epoch 7/50
53/53 [==============================] - 6s 105ms/step - loss: 0.0026
Epoch 8/50
53/53 [==============================] - 6s 107ms/step - loss: 0.0025
Epoch 9/50
53/53 [==============================] - 6s 115ms/step - loss: 0.0023
Epoch 10/50
53/53 [==============================] - 6s 107ms/step - loss: 0.0022
Epoch 11/50
53/53 [==============================] - 6s 109ms/step - loss: 0.0022
Epoch 12/50
53/53 [==============================] - 3s 63ms/step - loss: 0.0023
Epoch 13/50
53/53 [==============================] - 3s 54ms/step - loss: 0.0020
Epoch 14/50
53/53 [==============================] - 3s 58ms/step - loss: 0.0022
Epoch 15/50
53/53 [==============================] - 6s 105ms/step - loss: 0.0018
Epoch 16/50
53/53 [==============================] - 6s 110ms/step - loss: 0.0019
Epoch 17/50
53/53 [==============================] - 6s 104ms/step - loss: 0.0018
Epoch 18/50
53/53 [==============================] - 6s 109ms/step - loss: 0.0020
Epoch 19/50
53/53 [==============================] - 6s 111ms/step - loss: 0.0018
Epoch 20/50
53/53 [==============================] - 5s 100ms/step - loss: 0.0018
Epoch 21/50
53/53 [==============================] - 6s 111ms/step - loss: 0.0017
Epoch 22/50
```

```
53/53 [==============================] - 6s 109ms/step - loss: 0.0018
Epoch 23/50
53/53 [==============================] - 5s 102ms/step - loss: 0.0018
Epoch 24/50
53/53 [==============================] - 6s 111ms/step - loss: 0.0015
Epoch 25/50
53/53 [==============================] - 6s 107ms/step - loss: 0.0017
Epoch 26/50
53/53 [==============================] - 6s 104ms/step - loss: 0.0014
Epoch 27/50
53/53 [==============================] - 6s 111ms/step - loss: 0.0016
Epoch 28/50
53/53 [==============================] - 6s 106ms/step - loss: 0.0016
Epoch 29/50
53/53 [==============================] - 6s 106ms/step - loss: 0.0014
Epoch 30/50
53/53 [==============================] - 6s 111ms/step - loss: 0.0017
Epoch 31/50
53/53 [==============================] - 5s 103ms/step - loss: 0.0017
Epoch 32/50
53/53 [==============================] - 3s 50ms/step - loss: 0.0014
Epoch 33/50
53/53 [==============================] - 2s 45ms/step - loss: 0.0013
Epoch 34/50
53/53 [==============================] - 2s 47ms/step - loss: 0.0013
Epoch 35/50
53/53 [==============================] - 3s 59ms/step - loss: 0.0014
Epoch 36/50
53/53 [==============================] - 3s 48ms/step - loss: 0.0014
Epoch 37/50
53/53 [==============================] - 2s 42ms/step - loss: 0.0013
Epoch 38/50
53/53 [==============================] - 3s 52ms/step - loss: 0.0014
Epoch 39/50
53/53 [==============================] - 3s 62ms/step - loss: 0.0013
Epoch 40/50
53/53 [==============================] - 3s 64ms/step - loss: 0.0014
Epoch 41/50
53/53 [==============================] - 3s 63ms/step - loss: 0.0012
Epoch 42/50
53/53 [==============================] - 2s 44ms/step - loss: 0.0013
Epoch 43/50
53/53 [==============================] - 3s 48ms/step - loss: 0.0014
```

```
Epoch 44/50
53/53 [==============================] - 4s 67ms/step - loss: 0.0011
Epoch 45/50
53/53 [==============================] - 4s 68ms/step - loss: 0.0013
Epoch 46/50
53/53 [==============================] - 4s 70ms/step - loss: 0.0011
Epoch 47/50
53/53 [==============================] - 3s 47ms/step - loss: 0.0012
Epoch 48/50
53/53 [==============================] - 3s 49ms/step - loss: 0.0011
Epoch 49/50
53/53 [==============================] - 4s 73ms/step - loss: 0.0011
Epoch 50/50
53/53 [==============================] - 4s 66ms/step - loss: 0.0011
```

Out[13]:

```
<keras.src.callbacks.History at 0x1cdbb2535d0>
```

In [14]:

```python
#predictions on the test data
test_data = scaled_data[int(len(scaled_data)*0.8) - 60:]
x_test = []
y_test = data[int(len(data)*0.8):, :]

for i in range(60, len(test_data)):
    x_test.append(test_data[i-60:i, 0])

x_test = np.array(x_test)
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))

predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
```

```
14/14 [==============================] - 2s 16ms/step
```

```
In [15]: #Visualizing the Predicted price as compared to Actual price
plt.figure(figsize=(16,8))
plt.title('Predicted vs Actual Stock Price')
plt.plot(y_test, label='Actual Price')
plt.plot(predictions, label='Predicted Price')
plt.xlabel('Time', fontsize=18)
plt.ylabel('Stock Price USD ($)', fontsize=18)
plt.legend()
plt.show()
```