Stock Market Prediction HACKATHON

NIFTY 50

In [165…

```
### Data Collection
import pandas_datareader as pdr
import pandas as pd
key=""
```

In [166…

```
df_nifty = pdr.get_data_yahoo('^NSEI')
df = df_nifty
```

In [167…

```
df_nifty
```

Out[167]:

Date	High	Low	Open	Close	Volume	Adj Close
2017-11-07	10485.750000	10340.799805	10477.150391	10350.150391	286900.0	10350.150391
2017-11-08	10384.250000	10285.500000	10361.950195	10303.150391	274500.0	10303.150391
2017-11-09	10368.450195	10266.950195	10358.650391	10308.950195	240200.0	10308.950195
2017-11-10	10344.950195	10254.099609	10304.349609	10321.750000	279400.0	10321.750000
2017-11-13	10334.150391	10216.250000	10322.000000	10224.950195	210300.0	10224.950195
...	...	...	...	...	...	...
2022-10-31	18022.800781	17899.900391	17910.199219	18012.199219	227200.0	18012.199219
2022-11-01	18175.800781	18060.150391	18130.699219	18145.400391	349900.0	18145.400391
2022-11-02	18178.750000	18048.650391	18177.900391	18082.849609	270900.0	18082.849609
2022-11-03	18106.300781	17959.199219	17968.349609	18052.699219	213000.0	18052.699219
2022-11-04	18135.099609	18017.150391	18053.400391	18117.150391	267900.0	18117.150391

1231 rows × 6 columns

In [168…

```
df
```

Out[168]:

Date	High	Low	Open	Close	Volume	Adj Close
2017-11-07	10485.750000	10340.799805	10477.150391	10350.150391	286900.0	10350.150391
2017-11-08	10384.250000	10285.500000	10361.950195	10303.150391	274500.0	10303.150391
2017-11-09	10368.450195	10266.950195	10358.650391	10308.950195	240200.0	10308.950195
2017-11-10	10344.950195	10254.099609	10304.349609	10321.750000	279400.0	10321.750000
2017-11-13	10334.150391	10216.250000	10322.000000	10224.950195	210300.0	10224.950195
...	...	...	...	...	...	...
2022-10-31	18022.800781	17899.900391	17910.199219	18012.199219	227200.0	18012.199219
2022-11-01	18175.800781	18060.150391	18130.699219	18145.400391	349900.0	18145.400391
2022-11-02	18178.750000	18048.650391	18177.900391	18082.849609	270900.0	18082.849609
2022-11-03	18106.300781	17959.199219	17968.349609	18052.699219	213000.0	18052.699219
2022-11-04	18135.099609	18017.150391	18053.400391	18117.150391	267900.0	18117.150391

1231 rows × 6 columns

In [169…

```
df.head()
```

Out[169]:

Date	High	Low	Open	Close	Volume	Adj Close
2017-11-07	10485.750000	10340.799805	10477.150391	10350.150391	286900.0	10350.150391
2017-11-08	10384.250000	10285.500000	10361.950195	10303.150391	274500.0	10303.150391
2017-11-09	10368.450195	10266.950195	10358.650391	10308.950195	240200.0	10308.950195
2017-11-10	10344.950195	10254.099609	10304.349609	10321.750000	279400.0	10321.750000
2017-11-13	10334.150391	10216.250000	10322.000000	10224.950195	210300.0	10224.950195

In [170…

```
df.tail()
```

Out[170]:

Date	High	Low	Open	Close	Volume	Adj Close
2022-10-31	18022.800781	17899.900391	17910.199219	18012.199219	227200.0	18012.199219
2022-11-01	18175.800781	18060.150391	18130.699219	18145.400391	349900.0	18145.400391
2022-11-02	18178.750000	18048.650391	18177.900391	18082.849609	270900.0	18082.849609
2022-11-03	18106.300781	17959.199219	17968.349609	18052.699219	213000.0	18052.699219
2022-11-04	18135.099609	18017.150391	18053.400391	18117.150391	267900.0	18117.150391

In [171…

```
df1=df.reset_index()['Close']
```

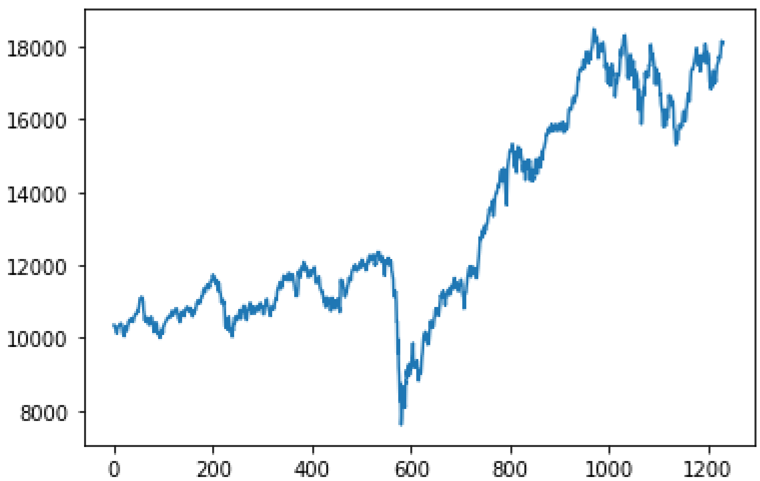
In [172…

```
df1
```

```
Out[172]: 0       10350.150391
1       10303.150391
2       10308.950195
3       10321.750000
4       10224.950195
            ...
1226    18012.199219
1227    18145.400391
1228    18082.849609
1229    18052.699219
1230    18117.150391
Name: Close, Length: 1231, dtype: float64
```

In [173…
```
import matplotlib.pyplot as plt
plt.plot(df1)
```

Out[173]: [<matplotlib.lines.Line2D at 0x17331906740>]



In [174…
```
### LSTM are sensitive to the scale of the data. so we apply MinMax scaler
```

In [175…
```
import numpy as np
```

In [176…
```
df1
```

```
Out[176]: 0       10350.150391
1       10303.150391
2       10308.950195
3       10321.750000
4       10224.950195
            ...
1226    18012.199219
1227    18145.400391
1228    18082.849609
1229    18052.699219
1230    18117.150391
Name: Close, Length: 1231, dtype: float64
```

In [177…
```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler(feature_range=(0,1))
df1=scaler.fit_transform(np.array(df1).reshape(-1,1))
```

In [178…
```
print(df1)
```

```
[[0.25213496]
 [0.24780986]
 [0.24834358]
 ...
 [0.96372427]
 [0.96094972]
 [0.96688074]]
```

In [179…

```
df1
```

Out[179]:

```
array([[0.25213496],
       [0.24780986],
       [0.24834358],
       ...,
       [0.96372427],
       [0.96094972],
       [0.96688074]])
```

In [180…

```
import numpy
# convert an array of values into a dataset matrix
def create_dataset(dataset, time_step=1):
        dataX, dataY = [], []
        for i in range(len(dataset)-time_step-1):
                a = dataset[i:(i+time_step), 0]   ###i=0, 0,1,2,3-----99   100
                dataX.append(a)
                dataY.append(dataset[i + time_step, 0])
        return numpy.array(dataX), numpy.array(dataY)
```

In [181…

```
time_step = 50
X_train, y_train = create_dataset(df1, time_step)
```

In [182…

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
```

In [183…

```
import tensorflow as tf
```

In [184…

```
import math
from sklearn.metrics import mean_squared_error
```

In [185…

```
print(X_train.shape)
```

```
(1180, 50)
```

In [186…

```
# reshape input to be [samples, time steps, features] which is required for LSTM
X_train =X_train.reshape(X_train.shape[0],X_train.shape[1] , 1)
```

In [187…

```
### Create the Stacked LSTM model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
```

In [188…

```
model=Sequential()
```

```python
model.add(LSTM(50,return_sequences=True,input_shape=(50,1)))
model.add(LSTM(50,return_sequences=True))
model.add(LSTM(50))
model.add(Dense(1))
model.compile(loss='mean_squared_error',optimizer='adam')
```

In [189…

```python
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 lstm_9 (LSTM)               (None, 50, 50)            10400     
                                                                 
 lstm_10 (LSTM)              (None, 50, 50)            20200     
                                                                 
 lstm_11 (LSTM)              (None, 50)                20200     
                                                                 
 dense_3 (Dense)             (None, 1)                 51        
                                                                 
=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```

In [190…

```python
model.summary()
```


```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 lstm_9 (LSTM)               (None, 50, 50)            10400     
                                                                 
 lstm_10 (LSTM)              (None, 50, 50)            20200     
                                                                 
 lstm_11 (LSTM)              (None, 50)                20200     
                                                                 
 dense_3 (Dense)             (None, 1)                 51        
                                                                 
=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```


In [ ]:

In [191…

```python
model.fit(X_train,y_train,epochs=95,batch_size=64,verbose=1)
```

```
Epoch 1/95
19/19 [==============================] - 31s 66ms/step - loss: 0.0634
Epoch 2/95
19/19 [==============================] - 1s 53ms/step - loss: 0.0076
Epoch 3/95
19/19 [==============================] - 1s 59ms/step - loss: 0.0030
Epoch 4/95
19/19 [==============================] - 1s 53ms/step - loss: 0.0024
Epoch 5/95
19/19 [==============================] - 2s 120ms/step - loss: 0.0021
Epoch 6/95
19/19 [==============================] - 2s 105ms/step - loss: 0.0020
Epoch 7/95
19/19 [==============================] - 1s 69ms/step - loss: 0.0019
Epoch 8/95
19/19 [==============================] - 1s 53ms/step - loss: 0.0019
Epoch 9/95
19/19 [==============================] - 2s 103ms/step - loss: 0.0019
Epoch 10/95
19/19 [==============================] - 2s 89ms/step - loss: 0.0018
Epoch 11/95
19/19 [==============================] - 1s 58ms/step - loss: 0.0017
Epoch 12/95
19/19 [==============================] - 1s 67ms/step - loss: 0.0017
Epoch 13/95
19/19 [==============================] - 2s 117ms/step - loss: 0.0016
Epoch 14/95
19/19 [==============================] - 1s 62ms/step - loss: 0.0016
Epoch 15/95
19/19 [==============================] - 1s 72ms/step - loss: 0.0016
Epoch 16/95
19/19 [==============================] - 1s 56ms/step - loss: 0.0015
Epoch 17/95
19/19 [==============================] - 2s 121ms/step - loss: 0.0015
Epoch 18/95
19/19 [==============================] - 2s 82ms/step - loss: 0.0014
Epoch 19/95
19/19 [==============================] - 1s 58ms/step - loss: 0.0013
Epoch 20/95
19/19 [==============================] - 2s 87ms/step - loss: 0.0013
Epoch 21/95
19/19 [==============================] - 2s 121ms/step - loss: 0.0013
Epoch 22/95
19/19 [==============================] - 1s 54ms/step - loss: 0.0014
Epoch 23/95
19/19 [==============================] - 1s 54ms/step - loss: 0.0012
Epoch 24/95
19/19 [==============================] - 2s 105ms/step - loss: 0.0012
Epoch 25/95
19/19 [==============================] - 2s 102ms/step - loss: 0.0012
Epoch 26/95
19/19 [==============================] - 1s 54ms/step - loss: 0.0012
Epoch 27/95
19/19 [==============================] - 1s 63ms/step - loss: 0.0012
Epoch 28/95
19/19 [==============================] - 2s 124ms/step - loss: 0.0014
Epoch 29/95
19/19 [==============================] - 2s 86ms/step - loss: 0.0012
Epoch 30/95
19/19 [==============================] - 1s 53ms/step - loss: 0.0011
Epoch 31/95
19/19 [==============================] - 2s 97ms/step - loss: 0.0011
Epoch 32/95
19/19 [==============================] - 2s 110ms/step - loss: 0.0010
```

```
Epoch 33/95
19/19 [==============================] - 1s 52ms/step - loss: 9.8459e-04
Epoch 34/95
19/19 [==============================] - 1s 53ms/step - loss: 9.5258e-04
Epoch 35/95
19/19 [==============================] - 2s 119ms/step - loss: 9.2632e-04
Epoch 36/95
19/19 [==============================] - 2s 91ms/step - loss: 8.8624e-04
Epoch 37/95
19/19 [==============================] - 1s 71ms/step - loss: 8.9555e-04
Epoch 38/95
19/19 [==============================] - 2s 120ms/step - loss: 8.6953e-04
Epoch 39/95
19/19 [==============================] - 1s 65ms/step - loss: 0.0010
Epoch 40/95
19/19 [==============================] - 2s 98ms/step - loss: 9.2777e-04
Epoch 41/95
19/19 [==============================] - 2s 109ms/step - loss: 9.0445e-04
Epoch 42/95
19/19 [==============================] - 1s 55ms/step - loss: 8.6800e-04
Epoch 43/95
19/19 [==============================] - 2s 118ms/step - loss: 7.7234e-04
Epoch 44/95
19/19 [==============================] - 2s 83ms/step - loss: 7.9726e-04
Epoch 45/95
19/19 [==============================] - 2s 85ms/step - loss: 9.5005e-04
Epoch 46/95
19/19 [==============================] - 2s 119ms/step - loss: 7.8229e-04
Epoch 47/95
19/19 [==============================] - 1s 54ms/step - loss: 7.4864e-04
Epoch 48/95
19/19 [==============================] - 2s 109ms/step - loss: 7.1492e-04
Epoch 49/95
19/19 [==============================] - 2s 98ms/step - loss: 7.9809e-04
Epoch 50/95
19/19 [==============================] - 1s 68ms/step - loss: 8.3725e-04
Epoch 51/95
19/19 [==============================] - 2s 121ms/step - loss: 7.8676e-04
Epoch 52/95
19/19 [==============================] - 2s 78ms/step - loss: 6.8271e-04
Epoch 53/95
19/19 [==============================] - 1s 53ms/step - loss: 6.7073e-04
Epoch 54/95
19/19 [==============================] - 2s 95ms/step - loss: 7.0047e-04
Epoch 55/95
19/19 [==============================] - 2s 111ms/step - loss: 6.6189e-04
Epoch 56/95
19/19 [==============================] - 1s 53ms/step - loss: 6.1252e-04
Epoch 57/95
19/19 [==============================] - 2s 120ms/step - loss: 6.7730e-04
Epoch 58/95
19/19 [==============================] - 2s 86ms/step - loss: 6.3764e-04
Epoch 59/95
19/19 [==============================] - 1s 76ms/step - loss: 5.9660e-04
Epoch 60/95
19/19 [==============================] - 2s 118ms/step - loss: 5.8040e-04
Epoch 61/95
19/19 [==============================] - 1s 61ms/step - loss: 6.1030e-04
Epoch 62/95
19/19 [==============================] - 2s 101ms/step - loss: 5.6770e-04
Epoch 63/95
19/19 [==============================] - 2s 106ms/step - loss: 5.5067e-04
Epoch 64/95
19/19 [==============================] - 1s 60ms/step - loss: 6.1562e-04
```

```
Epoch 65/95
19/19 [==============================] - 2s 118ms/step - loss: 6.5192e-04
Epoch 66/95
19/19 [==============================] - 2s 80ms/step - loss: 5.8258e-04
Epoch 67/95
19/19 [==============================] - 2s 85ms/step - loss: 5.3014e-04
Epoch 68/95
19/19 [==============================] - 2s 120ms/step - loss: 5.3399e-04
Epoch 69/95
19/19 [==============================] - 2s 87ms/step - loss: 6.1689e-04
Epoch 70/95
19/19 [==============================] - 1s 78ms/step - loss: 5.8128e-04
Epoch 71/95
19/19 [==============================] - 2s 118ms/step - loss: 4.9175e-04
Epoch 72/95
19/19 [==============================] - 1s 66ms/step - loss: 4.6666e-04
Epoch 73/95
19/19 [==============================] - 2s 122ms/step - loss: 4.7283e-04
Epoch 74/95
19/19 [==============================] - 2s 126ms/step - loss: 4.8113e-04
Epoch 75/95
19/19 [==============================] - 1s 53ms/step - loss: 4.7078e-04
Epoch 76/95
19/19 [==============================] - 2s 112ms/step - loss: 5.5953e-04
Epoch 77/95
19/19 [==============================] - 2s 119ms/step - loss: 4.4551e-04
Epoch 78/95
19/19 [==============================] - 1s 60ms/step - loss: 5.0078e-04
Epoch 79/95
19/19 [==============================] - 2s 103ms/step - loss: 4.5631e-04
Epoch 80/95
19/19 [==============================] - 2s 119ms/step - loss: 5.2089e-04
Epoch 81/95
19/19 [==============================] - 1s 70ms/step - loss: 5.0533e-04
Epoch 82/95
19/19 [==============================] - 2s 97ms/step - loss: 4.5970e-04
Epoch 83/95
19/19 [==============================] - 2s 119ms/step - loss: 4.2343e-04
Epoch 84/95
19/19 [==============================] - 2s 76ms/step - loss: 4.1014e-04
Epoch 85/95
19/19 [==============================] - 2s 92ms/step - loss: 4.8267e-04
Epoch 86/95
19/19 [==============================] - 3s 132ms/step - loss: 4.1879e-04
Epoch 87/95
19/19 [==============================] - 2s 110ms/step - loss: 4.8556e-04
Epoch 88/95
19/19 [==============================] - 1s 58ms/step - loss: 6.2825e-04
Epoch 89/95
19/19 [==============================] - 2s 130ms/step - loss: 4.5675e-04
Epoch 90/95
19/19 [==============================] - 1s 74ms/step - loss: 4.2981e-04
Epoch 91/95
19/19 [==============================] - 2s 90ms/step - loss: 4.0608e-04
Epoch 92/95
19/19 [==============================] - 2s 119ms/step - loss: 3.8676e-04
Epoch 93/95
19/19 [==============================] - 2s 82ms/step - loss: 4.2434e-04
Epoch 94/95
19/19 [==============================] - 2s 83ms/step - loss: 3.9256e-04
Epoch 95/95
19/19 [==============================] - 2s 119ms/step - loss: 3.8783e-04
```

```
Out[191]: <keras.callbacks.History at 0x17331cd73a0>
```

In [192…

```
import tensorflow as tf
```

In [193…

```
tf.__version__
```

Out[193]:

```
'2.10.0'
```

In [194…

```
### Lets Do the prediction and check performance metrics
train_predict=model.predict(X_train)
```

```
37/37 [==============================] - 3s 14ms/step
```

In [195…

```
##Transformback to original form
train_predict=scaler.inverse_transform(train_predict)
```

In [196…

```
### Calculate RMSE performance metrics
import math
ytrain  = scaler.inverse_transform(y_train.reshape(-1,1))
from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(ytrain,train_predict))
```

Out[196]:

```
207.70889226705145
```

In [ ]:

In [197…

```
len(df1)
```

Out[197]:

```
1231
```

In [198…

```
x_input=df1[len(df1)-50:].reshape(1,-1)
x_input.shape
```

Out[198]:

```
(1, 50)
```

In [ ]:

In [ ]:

In [199…

```
temp_input=list(x_input)
temp_input=temp_input[0].tolist()
```

In [200…

```
# demonstrate prediction for next 10 days
from numpy import array

lst_output=[]
n_steps=50
i=0
while(i<5):

    if(len(temp_input)>50):
        #print(temp_input)
```

Loading [MathJax]/extensions/Safe.js

```
        x_input=np.array(temp_input[1:])
        print("{} day input {}".format(i,x_input))
        x_input=x_input.reshape(1,-1)
        x_input = x_input.reshape((1, n_steps, 1))
        #print(x_input)
        yhat = model.predict(x_input, verbose=0)
        print("{} day output {}".format(i,yhat))
        temp_input.extend(yhat[0].tolist())
        temp_input=temp_input[1:]
        #print(temp_input)
        lst_output.extend(yhat.tolist())
        i=i+1
    else:
        x_input = x_input.reshape((1, n_steps,1))
        yhat = model.predict(x_input, verbose=0)
        print(yhat[0])
        temp_input.extend(yhat[0].tolist())
        print(len(temp_input))
        lst_output.extend(yhat.tolist())
        i=i+1


print(lst_output)
```

```
[0.9667334]
51
1 day input [0.91215431 0.91550868 0.89287092 0.9339502  0.91402713 0.91371871
 0.92534601 0.92440727 0.92153621 0.93758045 0.94076443 0.95024284
 0.96254647 0.95644525 0.94481813 0.91292735 0.92133832 0.93919086
 0.93018174 0.92203317 0.89420058 0.86557681 0.86475777 0.85106461
 0.84733767 0.87275913 0.85371029 0.88931885 0.8946102  0.89303196
 0.8862544  0.86256305 0.87545082 0.86539726 0.88116543 0.89276973
 0.90888748 0.91121575 0.91597329 0.91710992 0.93132286 0.92447628
 0.93189333 0.93648085 0.95722278 0.9694804  0.96372427 0.96094972
 0.96688074 0.9667334 ]
1 day output [[0.9665176]]
2 day input [0.91550868 0.89287092 0.9339502  0.91402713 0.91371871 0.92534601
 0.92440727 0.92153621 0.93758045 0.94076443 0.95024284 0.96254647
 0.95644525 0.94481813 0.91292735 0.92133832 0.93919086 0.93018174
 0.92203317 0.89420058 0.86557681 0.86475777 0.85106461 0.84733767
 0.87275913 0.85371029 0.88931885 0.8946102  0.89303196 0.8862544
 0.86256305 0.87545082 0.86539726 0.88116543 0.89276973 0.90888748
 0.91121575 0.91597329 0.91710992 0.93132286 0.92447628 0.93189333
 0.93648085 0.95722278 0.9694804  0.96372427 0.96094972 0.96688074
 0.9667334  0.96651763]
2 day output [[0.96598506]]
3 day input [0.89287092 0.9339502  0.91402713 0.91371871 0.92534601 0.92440727
 0.92153621 0.93758045 0.94076443 0.95024284 0.96254647 0.95644525
 0.94481813 0.91292735 0.92133832 0.93919086 0.93018174 0.92203317
 0.89420058 0.86557681 0.86475777 0.85106461 0.84733767 0.87275913
 0.85371029 0.88931885 0.8946102  0.89303196 0.8862544  0.86256305
 0.87545082 0.86539726 0.88116543 0.89276973 0.90888748 0.91121575
 0.91597329 0.91710992 0.93132286 0.92447628 0.93189333 0.93648085
 0.95722278 0.9694804  0.96372427 0.96094972 0.96688074 0.9667334
 0.96651763 0.96598506]
3 day output [[0.96540946]]
4 day input [0.9339502  0.91402713 0.91371871 0.92534601 0.92440727 0.92153621
 0.93758045 0.94076443 0.95024284 0.96254647 0.95644525 0.94481813
 0.91292735 0.92133832 0.93919086 0.93018174 0.92203317 0.89420058
 0.86557681 0.86475777 0.85106461 0.84733767 0.87275913 0.85371029
 0.88931885 0.8946102  0.89303196 0.8862544  0.86256305 0.87545082
 0.86539726 0.88116543 0.89276973 0.90888748 0.91121575 0.91597329
 0.91710992 0.93132286 0.92447628 0.93189333 0.93648085 0.95722278
 0.9694804  0.96372427 0.96094972 0.96688074 0.9667334  0.96651763
 0.96598506 0.96540946]
4 day output [[0.9649588]]
[[0.9667333960533142], [0.9665176272392273], [0.9659850597381592], [0.9654094576835632],
[0.9649587869644165]]
```

In [201…

```
future_5 = scaler.inverse_transform(lst_output)
future_5
```

Out[201]:

```
array([[18115.54922349],
       [18113.20450678],
       [18107.41720184],
       [18101.16224898],
       [18096.26490006]])
```

In [202…

```
future_5_upper = future_5 + 0.05*future_5
future_5_upper
```

Out[202]:

```
array([[19021.32668467],
       [19018.86473211],
       [19012.78806193],
       [19006.22036143],
       [19001.07814506]])
```

In [203…

```
future_5_lower = future_5 - 0.05*future_5
future_5_lower
```

Out[203]:

```
array([[17209.77176232],
       [17207.54428144],
       [17202.04634175],
       [17196.10413653],
       [17191.45165506]])
```

In [204…

```
future_nifty = pd.DataFrame(future_5,columns=['future_nifty'])
future_nifty['future_upper'] = future_5_upper
future_nifty['future_lower'] = future_5_lower
future_nifty
```

Out[204]:

	future_nifty	future_upper	future_lower
0	18115.549223	19021.326685	17209.771762
1	18113.204507	19018.864732	17207.544281
2	18107.417202	19012.788062	17202.046342
3	18101.162249	19006.220361	17196.104137
4	18096.264900	19001.078145	17191.451655

In [209…

```
future_nifty.to_csv('stock_pred_nifty.csv')
```

In [ ]:

In [ ]:

Stock Market Prediction HACKATHON

FTSE

In [107…

```
### Data Collection
import pandas_datareader as pdr
import pandas as pd
key=""
```

In [108…

```
df_nifty = pd.read_csv("C:\\Users\\Intel\\Downloads\\FTSE100.csv")
```

In [109…

```
df_nifty
```

Out[109]:

	Date	Open Price	Close Price	High Price	Low Price	Volume
0	06-Nov-2017	7560.35	7562.28	7572.90	7544.20	633452032
1	07-Nov-2017	7562.28	7513.11	7582.85	7507.83	747728576
2	08-Nov-2017	7513.11	7529.72	7534.49	7504.77	808188160
3	09-Nov-2017	7529.72	7484.10	7532.20	7476.89	860767680
4	10-Nov-2017	7484.10	7432.99	7500.29	7421.73	694860992
...	...	...	...	...	...	...
1253	31-Oct-2022	7047.67	7094.53	7132.85	7030.12	863121664
1254	01-Nov-2022	7094.53	7186.16	7221.32	7094.53	876203072
1255	02-Nov-2022	7186.16	7144.14	7205.07	7130.50	685693888
1256	03-Nov-2022	7144.14	7188.63	7188.63	7076.47	709517568
1257	04-Nov-2022	7188.63	7334.84	7376.23	7188.63	892653632

1258 rows × 6 columns

In [110…

```
df = df_nifty
```

In [111…

```
df.head()
```

Out[111]:

	Date	Open Price	Close Price	High Price	Low Price	Volume
0	06-Nov-2017	7560.35	7562.28	7572.90	7544.20	633452032
1	07-Nov-2017	7562.28	7513.11	7582.85	7507.83	747728576
2	08-Nov-2017	7513.11	7529.72	7534.49	7504.77	808188160
3	09-Nov-2017	7529.72	7484.10	7532.20	7476.89	860767680
4	10-Nov-2017	7484.10	7432.99	7500.29	7421.73	694860992

In [112…

```
df.tail()
```

Out[112]:

	Date	Open Price	Close Price	High Price	Low Price	Volume
1253	31-Oct-2022	7047.67	7094.53	7132.85	7030.12	863121664
1254	01-Nov-2022	7094.53	7186.16	7221.32	7094.53	876203072
1255	02-Nov-2022	7186.16	7144.14	7205.07	7130.50	685693888
1256	03-Nov-2022	7144.14	7188.63	7188.63	7076.47	709517568
1257	04-Nov-2022	7188.63	7334.84	7376.23	7188.63	892653632

In [113…

```
df1=df.reset_index()["Close Price"]
```

In [114…

```
df1
```

Out[114]:

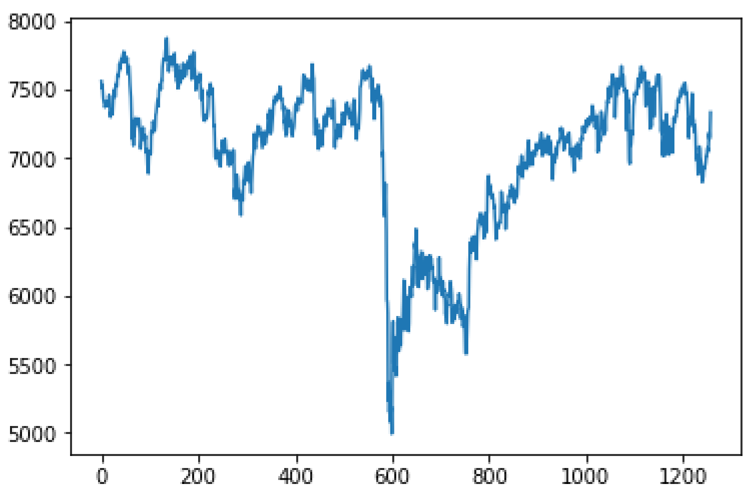
```
0       7562.28
1       7513.11
2       7529.72
3       7484.10
4       7432.99
          ...
1253    7094.53
1254    7186.16
1255    7144.14
1256    7188.63
1257    7334.84
Name: Close Price, Length: 1258, dtype: float64
```

In [115…

```
import matplotlib.pyplot as plt
plt.plot(df1)
```

Out[115]:

```
[<matplotlib.lines.Line2D at 0x23f3efa6770>]
```



In [116…

```
### LSTM are sensitive to the scale of the data. so we apply MinMax scaler
```

In [117…

```
import numpy as np
```

In [118…

```
df1
```

Out[118]:
```
0       7562.28
1       7513.11
2       7529.72
3       7484.10
4       7432.99
          ...
1253    7094.53
1254    7186.16
1255    7144.14
1256    7188.63
1257    7334.84
Name: Close Price, Length: 1258, dtype: float64
```

In [119…
```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler(feature_range=(0,1))
df1=scaler.fit_transform(np.array(df1).reshape(-1,1))
```

In [120…
```
print(df1)
```

```
[[0.89070108]
 [0.87364924]
 [0.87940948]
 ...
 [0.74569282]
 [0.76112167]
 [0.81182635]]
```

In [121…
```
df1
```

Out[121]:
```
array([[0.89070108],
       [0.87364924],
       [0.87940948],
       ...,
       [0.74569282],
       [0.76112167],
       [0.81182635]])
```

In [122…
```
import numpy
# convert an array of values into a dataset matrix
def create_dataset(dataset, time_step=1):
        dataX, dataY = [], []
        for i in range(len(dataset)-time_step-1):
                a = dataset[i:(i+time_step), 0]   ###i=0, 0,1,2,3-----99   100
                dataX.append(a)
                dataY.append(dataset[i + time_step, 0])
        return numpy.array(dataX), numpy.array(dataY)
```

In [123…
```
time_step = 50
X_train, y_train = create_dataset(df1, time_step)
```

In [124…
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
```

In [125…
```
import tensorflow as tf
```

In [126…
```
import math
from sklearn.metrics import mean_squared_error
```

In [127…
```
print(X_train.shape)
```

```
(1207, 50)
```

In [128…
```
# reshape input to be [samples, time steps, features] which is required for LSTM
X_train =X_train.reshape(X_train.shape[0],X_train.shape[1] , 1)
```

In [129…
```
### Create the Stacked LSTM model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
```

In [130…
```
model=Sequential()
model.add(LSTM(50,return_sequences=True,input_shape=(50,1)))
model.add(LSTM(50,return_sequences=True))
model.add(LSTM(50))
model.add(Dense(1))
model.compile(loss='mean_squared_error',optimizer='adam')
```

In [131…
```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 lstm_6 (LSTM)               (None, 50, 50)            10400     
                                                                 
 lstm_7 (LSTM)               (None, 50, 50)            20200     
                                                                 
 lstm_8 (LSTM)               (None, 50)                20200     
                                                                 
 dense_2 (Dense)             (None, 1)                 51        
                                                                 
=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```

In [132…
```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 lstm_6 (LSTM)               (None, 50, 50)            10400     
                                                                 
 lstm_7 (LSTM)               (None, 50, 50)            20200     
                                                                 
 lstm_8 (LSTM)               (None, 50)                20200     
                                                                 
 dense_2 (Dense)             (None, 1)                 51        
                                                                 
=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

```
```

In [133…

```
model.fit(X_train,y_train,epochs=95,batch_size=64,verbose=1)
```

```
Epoch 1/95
19/19 [==============================] - 14s 72ms/step - loss: 0.0965
Epoch 2/95
19/19 [==============================] - 1s 76ms/step - loss: 0.0100
Epoch 3/95
19/19 [==============================] - 2s 82ms/step - loss: 0.0072
Epoch 4/95
19/19 [==============================] - 2s 91ms/step - loss: 0.0063
Epoch 5/95
19/19 [==============================] - 1s 55ms/step - loss: 0.0055
Epoch 6/95
19/19 [==============================] - 1s 59ms/step - loss: 0.0052
Epoch 7/95
19/19 [==============================] - 1s 71ms/step - loss: 0.0051
Epoch 8/95
19/19 [==============================] - 2s 103ms/step - loss: 0.0047
Epoch 9/95
19/19 [==============================] - 1s 54ms/step - loss: 0.0044
Epoch 10/95
19/19 [==============================] - 1s 62ms/step - loss: 0.0043
Epoch 11/95
19/19 [==============================] - 2s 123ms/step - loss: 0.0041
Epoch 12/95
19/19 [==============================] - 2s 77ms/step - loss: 0.0040
Epoch 13/95
19/19 [==============================] - 1s 58ms/step - loss: 0.0037
Epoch 14/95
19/19 [==============================] - 2s 95ms/step - loss: 0.0037
Epoch 15/95
19/19 [==============================] - 2s 113ms/step - loss: 0.0034
Epoch 16/95
19/19 [==============================] - 1s 53ms/step - loss: 0.0033
Epoch 17/95
19/19 [==============================] - 1s 56ms/step - loss: 0.0032
Epoch 18/95
19/19 [==============================] - 2s 119ms/step - loss: 0.0032
Epoch 19/95
19/19 [==============================] - 2s 90ms/step - loss: 0.0031
Epoch 20/95
19/19 [==============================] - 1s 53ms/step - loss: 0.0030
Epoch 21/95
19/19 [==============================] - 1s 74ms/step - loss: 0.0029
Epoch 22/95
19/19 [==============================] - 2s 117ms/step - loss: 0.0030
Epoch 23/95
19/19 [==============================] - 1s 66ms/step - loss: 0.0030
Epoch 24/95
19/19 [==============================] - 1s 57ms/step - loss: 0.0031
Epoch 25/95
19/19 [==============================] - 2s 106ms/step - loss: 0.0033
Epoch 26/95
19/19 [==============================] - 2s 104ms/step - loss: 0.0027
Epoch 27/95
19/19 [==============================] - 1s 57ms/step - loss: 0.0027
Epoch 28/95
19/19 [==============================] - 2s 118ms/step - loss: 0.0025
Epoch 29/95
19/19 [==============================] - 2s 81ms/step - loss: 0.0024
Epoch 30/95
19/19 [==============================] - 2s 84ms/step - loss: 0.0024
Epoch 31/95
19/19 [==============================] - 2s 118ms/step - loss: 0.0024
Epoch 32/95
19/19 [==============================] - 1s 65ms/step - loss: 0.0024
```

```
Epoch 33/95
19/19 [==============================] - 1s 54ms/step - loss: 0.0025
Epoch 34/95
19/19 [==============================] - 2s 114ms/step - loss: 0.0023
Epoch 35/95
19/19 [==============================] - 2s 92ms/step - loss: 0.0021
Epoch 36/95
19/19 [==============================] - 1s 71ms/step - loss: 0.0021
Epoch 37/95
19/19 [==============================] - 2s 118ms/step - loss: 0.0020
Epoch 38/95
19/19 [==============================] - 1s 68ms/step - loss: 0.0020
Epoch 39/95
19/19 [==============================] - 2s 99ms/step - loss: 0.0021
Epoch 40/95
19/19 [==============================] - 2s 109ms/step - loss: 0.0020
Epoch 41/95
19/19 [==============================] - 1s 60ms/step - loss: 0.0018
Epoch 42/95
19/19 [==============================] - 2s 128ms/step - loss: 0.0018
Epoch 43/95
19/19 [==============================] - 1s 76ms/step - loss: 0.0018
Epoch 44/95
19/19 [==============================] - 1s 53ms/step - loss: 0.0018
Epoch 45/95
19/19 [==============================] - 2s 92ms/step - loss: 0.0017
Epoch 46/95
19/19 [==============================] - 2s 117ms/step - loss: 0.0015
Epoch 47/95
19/19 [==============================] - 1s 69ms/step - loss: 0.0017
Epoch 48/95
19/19 [==============================] - 2s 119ms/step - loss: 0.0017
Epoch 49/95
19/19 [==============================] - 2s 77ms/step - loss: 0.0016
Epoch 50/95
19/19 [==============================] - 2s 87ms/step - loss: 0.0015
Epoch 51/95
19/19 [==============================] - 2s 119ms/step - loss: 0.0022
Epoch 52/95
19/19 [==============================] - 1s 55ms/step - loss: 0.0016
Epoch 53/95
19/19 [==============================] - 2s 115ms/step - loss: 0.0014
Epoch 54/95
19/19 [==============================] - 2s 119ms/step - loss: 0.0013
Epoch 55/95
19/19 [==============================] - 1s 57ms/step - loss: 0.0013
Epoch 56/95
19/19 [==============================] - 2s 107ms/step - loss: 0.0013
Epoch 57/95
19/19 [==============================] - 2s 100ms/step - loss: 0.0013
Epoch 58/95
19/19 [==============================] - 1s 64ms/step - loss: 0.0012
Epoch 59/95
19/19 [==============================] - 2s 120ms/step - loss: 0.0012
Epoch 60/95
19/19 [==============================] - 2s 111ms/step - loss: 0.0013
Epoch 61/95
19/19 [==============================] - 1s 67ms/step - loss: 0.0011
Epoch 62/95
19/19 [==============================] - 2s 122ms/step - loss: 0.0012
Epoch 63/95
19/19 [==============================] - 1s 74ms/step - loss: 0.0011
Epoch 64/95
19/19 [==============================] - 1s 61ms/step - loss: 0.0012
```

```
Epoch 65/95
19/19 [==============================] - 2s 95ms/step - loss: 0.0015
Epoch 66/95
19/19 [==============================] - 2s 119ms/step - loss: 0.0012
Epoch 67/95
19/19 [==============================] - 2s 113ms/step - loss: 0.0010
Epoch 68/95
19/19 [==============================] - 1s 56ms/step - loss: 0.0010
Epoch 69/95
19/19 [==============================] - 2s 119ms/step - loss: 0.0011
Epoch 70/95
19/19 [==============================] - 2s 119ms/step - loss: 9.6225e-04
Epoch 71/95
19/19 [==============================] - 1s 55ms/step - loss: 9.8483e-04
Epoch 72/95
19/19 [==============================] - 2s 120ms/step - loss: 9.5864e-04
Epoch 73/95
19/19 [==============================] - 2s 122ms/step - loss: 9.6855e-04
Epoch 74/95
19/19 [==============================] - 1s 53ms/step - loss: 9.4694e-04
Epoch 75/95
19/19 [==============================] - 2s 114ms/step - loss: 0.0011
Epoch 76/95
19/19 [==============================] - 2s 123ms/step - loss: 9.1690e-04
Epoch 77/95
19/19 [==============================] - 1s 58ms/step - loss: 8.8044e-04
Epoch 78/95
19/19 [==============================] - 2s 108ms/step - loss: 9.5615e-04
Epoch 79/95
19/19 [==============================] - 2s 119ms/step - loss: 8.6304e-04
Epoch 80/95
19/19 [==============================] - 4s 189ms/step - loss: 9.3515e-04
Epoch 81/95
19/19 [==============================] - 1s 54ms/step - loss: 9.2644e-04
Epoch 82/95
19/19 [==============================] - 2s 86ms/step - loss: 0.0010
Epoch 83/95
19/19 [==============================] - 2s 119ms/step - loss: 0.0011
Epoch 84/95
19/19 [==============================] - 2s 88ms/step - loss: 9.6089e-04
Epoch 85/95
19/19 [==============================] - 1s 79ms/step - loss: 8.8404e-04
Epoch 86/95
19/19 [==============================] - 2s 120ms/step - loss: 8.4263e-04
Epoch 87/95
19/19 [==============================] - 2s 93ms/step - loss: 8.2915e-04
Epoch 88/95
19/19 [==============================] - 1s 75ms/step - loss: 8.2844e-04
Epoch 89/95
19/19 [==============================] - 2s 119ms/step - loss: 9.0050e-04
Epoch 90/95
19/19 [==============================] - 2s 99ms/step - loss: 9.6384e-04
Epoch 91/95
19/19 [==============================] - 1s 69ms/step - loss: 8.6329e-04
Epoch 92/95
19/19 [==============================] - 2s 119ms/step - loss: 8.5187e-04
Epoch 93/95
19/19 [==============================] - 2s 105ms/step - loss: 9.4429e-04
Epoch 94/95
19/19 [==============================] - 1s 61ms/step - loss: 7.7267e-04
Epoch 95/95
19/19 [==============================] - 2s 121ms/step - loss: 8.2764e-04
```

Out[133]:

```
<keras.callbacks.History at 0x23f46af2e60>
```

In [134…
```
import tensorflow as tf
```

In [135…
```
tf.__version__
```

Out[135]:
```
'2.10.0'
```

In [136…
```
### Lets Do the prediction and check performance metrics
train_predict=model.predict(X_train)
```

```
38/38 [==============================] - 2s 14ms/step
```

In [137…
```
##Transformback to original form
train_predict=scaler.inverse_transform(train_predict)
```

In [138…
```
### Calculate RMSE performance metrics
import math
ytrain  = scaler.inverse_transform(y_train.reshape(-1,1))
from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(ytrain,train_predict))
```

Out[138]:
```
79.46343852811971
```

In [ ]:
```

```

In [139…
```
len(df1)
```

Out[139]:
```
1258
```

In [140…
```
x_input=df1[len(df1)-50:].reshape(1,-1)
x_input.shape
```

Out[140]:
```
(1, 50)
```

In [ ]:
```

```

In [ ]:
```

```

In [141…
```
temp_input=list(x_input)
temp_input=temp_input[0].tolist()
```

In [142…
```
# demonstrate prediction for next 10 days
from numpy import array

lst_output=[]
n_steps=50
i=0
while(i<5):

    if(len(temp_input)>50):
        #print(temp_input)
```

```
        x_input=np.array(temp_input[1:])
        print("{} day input {}".format(i,x_input))
        x_input=x_input.reshape(1,-1)
        x_input = x_input.reshape((1, n_steps, 1))
        #print(x_input)
        yhat = model.predict(x_input, verbose=0)
        print("{} day output {}".format(i,yhat))
        temp_input.extend(yhat[0].tolist())
        temp_input=temp_input[1:]
        #print(temp_input)
        lst_output.extend(yhat.tolist())
        i=i+1
    else:
        x_input = x_input.reshape((1, n_steps,1))
        yhat = model.predict(x_input, verbose=0)
        print(yhat[0])
        temp_input.extend(yhat[0].tolist())
        print(len(temp_input))
        lst_output.extend(yhat.tolist())
        i=i+1


print(lst_output)
```

```
[0.8068342]
51
1 day input [0.84389435 0.82111695 0.79424739 0.74720484 0.79322088 0.79538487
 0.79989666 0.77818391 0.78658672 0.81745481 0.85974975 0.82951976
 0.79187185 0.79352606 0.7777851  0.76251925 0.77811802 0.75102651
 0.70215636 0.70297133 0.69036191 0.69757522 0.65464218 0.65888
 0.66406456 0.72568977 0.71395428 0.69475926 0.69261607 0.68159497
 0.65590451 0.63541594 0.6437806  0.64673529 0.66804575 0.67376784
 0.66969302 0.67625435 0.68520856 0.70055764 0.70038078 0.71515072
 0.72126122 0.71223765 0.7284884  0.76026509 0.74569282 0.76112167
 0.81182635 0.80683422]
1 day output [[0.82088023]]
2 day input [0.82111695 0.79424739 0.74720484 0.79322088 0.79538487 0.79989666
 0.77818391 0.78658672 0.81745481 0.85974975 0.82951976 0.79187185
 0.79352606 0.7777851  0.76251925 0.77811802 0.75102651 0.70215636
 0.70297133 0.69036191 0.69757522 0.65464218 0.65888    0.66406456
 0.72568977 0.71395428 0.69475926 0.69261607 0.68159497 0.65590451
 0.63541594 0.6437806  0.64673529 0.66804575 0.67376784 0.66969302
 0.67625435 0.68520856 0.70055764 0.70038078 0.71515072 0.72126122
 0.71223765 0.7284884  0.76026509 0.74569282 0.76112167 0.81182635
 0.80683422 0.82088023]
2 day output [[0.8288827]]
3 day input [0.79424739 0.74720484 0.79322088 0.79538487 0.79989666 0.77818391
 0.78658672 0.81745481 0.85974975 0.82951976 0.79187185 0.79352606
 0.7777851  0.76251925 0.77811802 0.75102651 0.70215636 0.70297133
 0.69036191 0.69757522 0.65464218 0.65888    0.66406456 0.72568977
 0.71395428 0.69475926 0.69261607 0.68159497 0.65590451 0.63541594
 0.6437806  0.64673529 0.66804575 0.67376784 0.66969302 0.67625435
 0.68520856 0.70055764 0.70038078 0.71515072 0.72126122 0.71223765
 0.7284884  0.76026509 0.74569282 0.76112167 0.81182635 0.80683422
 0.82088023 0.82888269]
3 day output [[0.8375417]]
4 day input [0.74720484 0.79322088 0.79538487 0.79989666 0.77818391 0.78658672
 0.81745481 0.85974975 0.82951976 0.79187185 0.79352606 0.7777851
 0.76251925 0.77811802 0.75102651 0.70215636 0.70297133 0.69036191
 0.69757522 0.65464218 0.65888    0.66406456 0.72568977 0.71395428
 0.69475926 0.69261607 0.68159497 0.65590451 0.63541594 0.6437806
 0.64673529 0.66804575 0.67376784 0.66969302 0.67625435 0.68520856
 0.70055764 0.70038078 0.71515072 0.72126122 0.71223765 0.7284884
 0.76026509 0.74569282 0.76112167 0.81182635 0.80683422 0.82088023
 0.82888269 0.8375417 ]
4 day output [[0.846503]]
[[0.8068342208862305], [0.8208802342414856], [0.8288826942443848], [0.8375416994094849],
[0.8465030193328857]]
```

In [144…

```
future_5 = scaler.inverse_transform(lst_output)
future_5
```

Out[144]:

```
array([[7320.44488598],
       [7360.94740825],
       [7384.02298182],
       [7408.99174275],
       [7434.83224643]])
```

In [145…

```
future_5_upper = future_5 + 0.05*future_5
future_5_upper
```

Out[145]:

```
array([[7686.46713028],
       [7728.99477866],
       [7753.22413091],
       [7779.44132989],
       [7806.57385875]])
```

In [146…
```
future_5_lower = future_5 - 0.05*future_5
future_5_lower
```

Out[146]:
```
array([[6954.42264168],
       [6992.90003784],
       [7014.82183272],
       [7038.54215561],
       [7063.09063411]])
```

In [158…
```
future_ftse = pd.DataFrame(future_5,columns=['future_ftse'])
future_ftse['future_upper'] = future_5_upper
future_ftse['future_lower'] = future_5_lower
future_ftse
```

Out[158]:

	future_ftse	future_upper	future_lower
0	7320.444886	7686.467130	6954.422642
1	7360.947408	7728.994779	6992.900038
2	7384.022982	7753.224131	7014.821833
3	7408.991743	7779.441330	7038.542156
4	7434.832246	7806.573859	7063.090634

In [160…
```
future_ftse.to_csv('stock_pred_ftse.csv')
```

In [ ]:

In [ ]:

Stock Market Prediction HACKATHON

NASDAQ 100

In [80]:
```
### Data Collection
import pandas_datareader as pdr
import pandas as pd
key=""
```

In [81]:
```
df_nifty = pdr.get_data_yahoo('^NDX')
df = df_nifty
```

In [82]:
```
df.head()
```

Out[82]:

Date	High	Low	Open	Close	Volume	Adj Close
2017-11-06	6318.580078	6291.839844	6292.149902	6313.609863	2166470000	6313.609863
2017-11-07	6328.580078	6299.529785	6314.689941	6320.779785	2205270000	6320.779785
2017-11-08	6346.979980	6308.620117	6319.029785	6345.810059	2110990000	6345.810059
2017-11-09	6315.160156	6248.290039	6295.290039	6312.209961	2235180000	6312.209961
2017-11-10	6313.169922	6284.220215	6297.149902	6309.069824	1973970000	6309.069824

In [83]:
```
df.tail()
```

Out[83]:

Date	High	Low	Open	Close	Volume	Adj Close
2022-10-31	11482.990234	11331.259766	11465.209961	11405.570312	4753740000	11405.570312
2022-11-01	11574.389648	11278.280273	11571.530273	11288.950195	4677520000	11288.950195
2022-11-02	11410.910156	10903.480469	11300.269531	10906.339844	5436420000	10906.339844
2022-11-03	10852.179688	10680.830078	10769.429688	10690.599609	5102190000	10690.599609
2022-11-04	10934.620117	10632.389648	10911.980469	10857.030273	5453750000	10857.030273

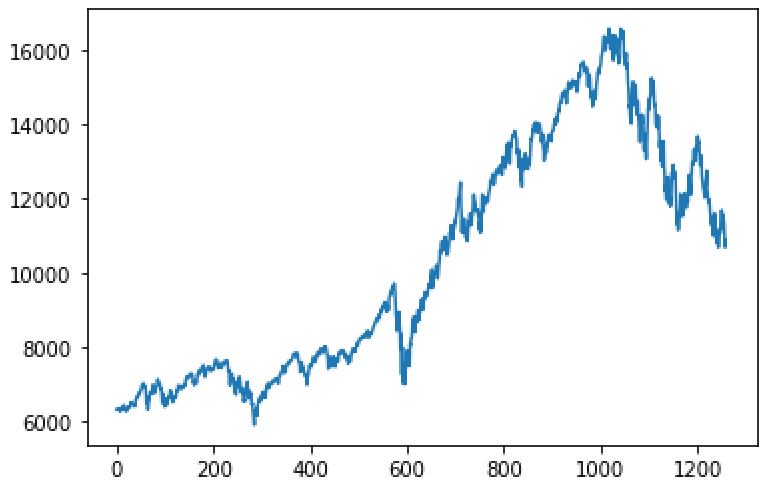
In [84]:
```
df1=df.reset_index()['Close']
```

In [85]:
```
df1
```

Out[85]:
```
0        6313.609863
1        6320.779785
2        6345.810059
3        6312.209961
4        6309.069824
            ...
1254    11405.570312
1255    11288.950195
1256    10906.339844
1257    10690.599609
1258    10857.030273
Name: Close, Length: 1259, dtype: float64
```

In [86]:
```
import matplotlib.pyplot as plt
plt.plot(df1)
```

Out[86]:
```
[<matplotlib.lines.Line2D at 0x1eeebcb2a70>]
```



In [87]:
```
### LSTM are sensitive to the scale of the data. so we apply MinMax scaler
```

In [88]:
```
import numpy as np
```

In [89]:
```
df1
```

Out[89]:
```
0        6313.609863
1        6320.779785
2        6345.810059
3        6312.209961
4        6309.069824
            ...
1254    11405.570312
1255    11288.950195
1256    10906.339844
1257    10690.599609
1258    10857.030273
Name: Close, Length: 1259, dtype: float64
```

In [90]:
```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler(feature_range=(0,1))
df1=scaler.fit_transform(np.array(df1).reshape(-1,1))
```

In [91]:
```
print(df1)
```

```
[[0.03881021]
 [0.03948193]
 [0.04182691]
 ...
 [0.46908324]
 [0.44887147]
 [0.46446364]]
```

In [92]:

```
df1
```

Out[92]:

```
array([[0.03881021],
       [0.03948193],
       [0.04182691],
       ...,
       [0.46908324],
       [0.44887147],
       [0.46446364]])
```

In [93]:

```
import numpy
# convert an array of values into a dataset matrix
def create_dataset(dataset, time_step=1):
	dataX, dataY = [], []
	for i in range(len(dataset)-time_step-1):
		a = dataset[i:(i+time_step), 0]   ###i=0, 0,1,2,3-----99   100
		dataX.append(a)
		dataY.append(dataset[i + time_step, 0])
	return numpy.array(dataX), numpy.array(dataY)
```

In [94]:

```
time_step = 50
X_train, y_train = create_dataset(df1, time_step)
```

In [95]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
```

In [96]:

```
import tensorflow as tf
```

In [97]:

```
import math
from sklearn.metrics import mean_squared_error
```

In [98]:

```
print(X_train.shape)
```

```
(1208, 50)
```

In [99]:

```
# reshape input to be [samples, time steps, features] which is required for LSTM
X_train =X_train.reshape(X_train.shape[0],X_train.shape[1] , 1)
```

In [100…

```
### Create the Stacked LSTM model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
```

In [101…

```
model=Sequential()
```

```
model.add(LSTM(50,return_sequences=True,input_shape=(50,1)))
model.add(LSTM(50,return_sequences=True))
model.add(LSTM(50))
model.add(Dense(1))
model.compile(loss='mean_squared_error',optimizer='adam')
```

In [102…

```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_6 (LSTM)               (None, 50, 50)            10400

 lstm_7 (LSTM)               (None, 50, 50)            20200

 lstm_8 (LSTM)               (None, 50)                20200

 dense_2 (Dense)             (None, 1)                 51

=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```

In [103…

```
model.summary()
```


```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_6 (LSTM)               (None, 50, 50)            10400

 lstm_7 (LSTM)               (None, 50, 50)            20200

 lstm_8 (LSTM)               (None, 50)                20200

 dense_2 (Dense)             (None, 1)                 51

=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```


In [ ]:

In [104…

```
model.fit(X_train,y_train,epochs=95,batch_size=64,verbose=1)
```

```
Epoch 1/95
19/19 [==============================] - 17s 62ms/step - loss: 0.0501
Epoch 2/95
19/19 [==============================] - 1s 63ms/step - loss: 0.0046
Epoch 3/95
19/19 [==============================] - 2s 90ms/step - loss: 0.0022
Epoch 4/95
19/19 [==============================] - 2s 94ms/step - loss: 0.0019
Epoch 5/95
19/19 [==============================] - 1s 52ms/step - loss: 0.0018
Epoch 6/95
19/19 [==============================] - 1s 59ms/step - loss: 0.0017
Epoch 7/95
19/19 [==============================] - 2s 92ms/step - loss: 0.0017
Epoch 8/95
19/19 [==============================] - 2s 116ms/step - loss: 0.0016
Epoch 9/95
19/19 [==============================] - 1s 51ms/step - loss: 0.0016
Epoch 10/95
19/19 [==============================] - 1s 52ms/step - loss: 0.0015
Epoch 11/95
19/19 [==============================] - 2s 102ms/step - loss: 0.0015
Epoch 12/95
19/19 [==============================] - 2s 102ms/step - loss: 0.0014
Epoch 13/95
19/19 [==============================] - 1s 62ms/step - loss: 0.0013
Epoch 14/95
19/19 [==============================] - 1s 58ms/step - loss: 0.0014
Epoch 15/95
19/19 [==============================] - 2s 115ms/step - loss: 0.0012
Epoch 16/95
19/19 [==============================] - 2s 80ms/step - loss: 0.0012
Epoch 17/95
19/19 [==============================] - 1s 51ms/step - loss: 0.0012
Epoch 18/95
19/19 [==============================] - 1s 69ms/step - loss: 0.0011
Epoch 19/95
19/19 [==============================] - 2s 115ms/step - loss: 0.0010
Epoch 20/95
19/19 [==============================] - 1s 69ms/step - loss: 0.0011
Epoch 21/95
19/19 [==============================] - 2s 106ms/step - loss: 9.6882e-04
Epoch 22/95
19/19 [==============================] - 1s 74ms/step - loss: 9.4774e-04
Epoch 23/95
19/19 [==============================] - 2s 129ms/step - loss: 9.2700e-04
Epoch 24/95
19/19 [==============================] - 1s 58ms/step - loss: 0.0010
Epoch 25/95
19/19 [==============================] - 1s 53ms/step - loss: 8.5717e-04
Epoch 26/95
19/19 [==============================] - 2s 127ms/step - loss: 8.4566e-04
Epoch 27/95
19/19 [==============================] - 2s 93ms/step - loss: 9.4281e-04
Epoch 28/95
19/19 [==============================] - 1s 52ms/step - loss: 9.2005e-04
Epoch 29/95
19/19 [==============================] - 1s 66ms/step - loss: 7.8747e-04
Epoch 30/95
19/19 [==============================] - 2s 115ms/step - loss: 7.5774e-04
Epoch 31/95
19/19 [==============================] - 1s 77ms/step - loss: 7.3893e-04
Epoch 32/95
19/19 [==============================] - 2s 81ms/step - loss: 7.5640e-04
```

```
Epoch 33/95
19/19 [==============================] - 2s 118ms/step - loss: 7.8859e-04
Epoch 34/95
19/19 [==============================] - 1s 54ms/step - loss: 7.7699e-04
Epoch 35/95
19/19 [==============================] - 2s 102ms/step - loss: 7.1571e-04
Epoch 36/95
19/19 [==============================] - 2s 102ms/step - loss: 6.9527e-04
Epoch 37/95
19/19 [==============================] - 1s 57ms/step - loss: 7.5580e-04
Epoch 38/95
19/19 [==============================] - 2s 116ms/step - loss: 7.1604e-04
Epoch 39/95
19/19 [==============================] - 2s 81ms/step - loss: 6.5537e-04
Epoch 40/95
19/19 [==============================] - 1s 78ms/step - loss: 6.0984e-04
Epoch 41/95
19/19 [==============================] - 2s 119ms/step - loss: 6.2901e-04
Epoch 42/95
19/19 [==============================] - 1s 57ms/step - loss: 6.2075e-04
Epoch 43/95
19/19 [==============================] - 2s 101ms/step - loss: 5.9890e-04
Epoch 44/95
19/19 [==============================] - 2s 104ms/step - loss: 5.6933e-04
Epoch 45/95
19/19 [==============================] - 1s 74ms/step - loss: 6.1502e-04
Epoch 46/95
19/19 [==============================] - 1s 75ms/step - loss: 5.4951e-04
Epoch 47/95
19/19 [==============================] - 3s 134ms/step - loss: 6.3382e-04
Epoch 48/95
19/19 [==============================] - 2s 86ms/step - loss: 6.6143e-04
Epoch 49/95
19/19 [==============================] - 1s 76ms/step - loss: 5.6370e-04
Epoch 50/95
19/19 [==============================] - 2s 117ms/step - loss: 5.2483e-04
Epoch 51/95
19/19 [==============================] - 1s 62ms/step - loss: 5.2583e-04
Epoch 52/95
19/19 [==============================] - 2s 101ms/step - loss: 4.9037e-04
Epoch 53/95
19/19 [==============================] - 2s 106ms/step - loss: 4.9321e-04
Epoch 54/95
19/19 [==============================] - 1s 58ms/step - loss: 4.7870e-04
Epoch 55/95
19/19 [==============================] - 2s 125ms/step - loss: 5.2847e-04
Epoch 56/95
19/19 [==============================] - 1s 76ms/step - loss: 4.8487e-04
Epoch 57/95
19/19 [==============================] - 2s 85ms/step - loss: 4.7596e-04
Epoch 58/95
19/19 [==============================] - 2s 117ms/step - loss: 4.8281e-04
Epoch 59/95
19/19 [==============================] - 1s 53ms/step - loss: 4.9676e-04
Epoch 60/95
19/19 [==============================] - 2s 111ms/step - loss: 4.4281e-04
Epoch 61/95
19/19 [==============================] - 2s 123ms/step - loss: 5.2368e-04
Epoch 62/95
19/19 [==============================] - 1s 66ms/step - loss: 5.0522e-04
Epoch 63/95
19/19 [==============================] - 2s 115ms/step - loss: 4.5894e-04
Epoch 64/95
19/19 [==============================] - 2s 117ms/step - loss: 4.6029e-04
```

```
Epoch 65/95
19/19 [==============================] - 1s 61ms/step - loss: 4.2935e-04
Epoch 66/95
19/19 [==============================] - 2s 104ms/step - loss: 4.7348e-04
Epoch 67/95
19/19 [==============================] - 2s 117ms/step - loss: 4.0701e-04
Epoch 68/95
19/19 [==============================] - 1s 68ms/step - loss: 4.6175e-04
Epoch 69/95
19/19 [==============================] - 2s 97ms/step - loss: 4.6685e-04
Epoch 70/95
19/19 [==============================] - 2s 118ms/step - loss: 3.9204e-04
Epoch 71/95
19/19 [==============================] - 2s 76ms/step - loss: 3.8300e-04
Epoch 72/95
19/19 [==============================] - 2s 87ms/step - loss: 3.8878e-04
Epoch 73/95
19/19 [==============================] - 2s 118ms/step - loss: 4.1850e-04
Epoch 74/95
19/19 [==============================] - 1s 53ms/step - loss: 3.7204e-04
Epoch 75/95
19/19 [==============================] - 2s 110ms/step - loss: 3.9283e-04
Epoch 76/95
19/19 [==============================] - 2s 117ms/step - loss: 4.1450e-04
Epoch 77/95
19/19 [==============================] - 2s 98ms/step - loss: 3.9620e-04
Epoch 78/95
19/19 [==============================] - 1s 67ms/step - loss: 4.5779e-04
Epoch 79/95
19/19 [==============================] - 2s 118ms/step - loss: 4.6948e-04
Epoch 80/95
19/19 [==============================] - 2s 105ms/step - loss: 4.0982e-04
Epoch 81/95
19/19 [==============================] - 1s 59ms/step - loss: 3.8901e-04
Epoch 82/95
19/19 [==============================] - 2s 117ms/step - loss: 3.9532e-04
Epoch 83/95
19/19 [==============================] - 2s 116ms/step - loss: 4.3881e-04
Epoch 84/95
19/19 [==============================] - 1s 53ms/step - loss: 3.4141e-04
Epoch 85/95
19/19 [==============================] - 2s 117ms/step - loss: 3.7571e-04
Epoch 86/95
19/19 [==============================] - 2s 118ms/step - loss: 4.1112e-04
Epoch 87/95
19/19 [==============================] - 2s 89ms/step - loss: 4.0634e-04
Epoch 88/95
19/19 [==============================] - 1s 73ms/step - loss: 3.2959e-04
Epoch 89/95
19/19 [==============================] - 2s 118ms/step - loss: 3.3771e-04
Epoch 90/95
19/19 [==============================] - 2s 99ms/step - loss: 3.9536e-04
Epoch 91/95
19/19 [==============================] - 1s 68ms/step - loss: 3.4241e-04
Epoch 92/95
19/19 [==============================] - 2s 123ms/step - loss: 3.2186e-04
Epoch 93/95
19/19 [==============================] - 2s 125ms/step - loss: 3.2292e-04
Epoch 94/95
19/19 [==============================] - 2s 101ms/step - loss: 3.1682e-04
Epoch 95/95
19/19 [==============================] - 1s 79ms/step - loss: 3.9235e-04
```

```
Out[104]: <keras.callbacks.History at 0x1eeec076fe0>
```

In [105…

```python
import tensorflow as tf
```

In [106…

```python
tf.__version__
```

Out[106]:

```
'2.10.0'
```

In [107…

```python
### Lets Do the prediction and check performance metrics
train_predict=model.predict(X_train)
```

```
38/38 [==============================] - 2s 14ms/step
```

In [108…

```python
##Transformback to original form
train_predict=scaler.inverse_transform(train_predict)
```

In [109…

```python
### Calculate RMSE performance metrics
import math
ytrain  = scaler.inverse_transform(y_train.reshape(-1,1))
from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(ytrain,train_predict))
```

Out[109]:

```
200.96305287715938
```

In [ ]:

In [110…

```python
len(df1)
```

Out[110]:

```
1259
```

In [111…

```python
x_input=df1[len(df1)-50:].reshape(1,-1)
x_input.shape
```

Out[111]:

```
(1, 50)
```

In [ ]:

In [ ]:

In [112…

```python
temp_input=list(x_input)
temp_input=temp_input[0].tolist()
```

In [113…

```python
# demonstrate prediction for next 10 days
from numpy import array

lst_output=[]
n_steps=50
i=0
while(i<5):

    if(len(temp_input)>50):
        #print(temp_input)
```

```
        x_input=np.array(temp_input[1:])
        print("{} day input {}".format(i,x_input))
        x_input=x_input.reshape(1,-1)
        x_input = x_input.reshape((1, n_steps, 1))
        #print(x_input)
        yhat = model.predict(x_input, verbose=0)
        print("{} day output {}".format(i,yhat))
        temp_input.extend(yhat[0].tolist())
        temp_input=temp_input[1:]
        #print(temp_input)
        lst_output.extend(yhat.tolist())
        i=i+1
    else:
        x_input = x_input.reshape((1, n_steps,1))
        yhat = model.predict(x_input, verbose=0)
        print(yhat[0])
        temp_input.extend(yhat[0].tolist())
        print(len(temp_input))
        lst_output.extend(yhat.tolist())
        i=i+1


print(lst_output)
```

```
[0.4500739]
51
1 day input [0.61691742 0.60364964 0.59702888 0.59727151 0.580766   0.57260309
 0.59584464 0.60163449 0.62665789 0.64084469 0.57469327 0.58413493
 0.56475041 0.55855682 0.56716557 0.55763497 0.53760965 0.52485532
 0.50701661 0.50166436 0.50331694 0.52412267 0.49329541 0.47516156
 0.49938032 0.53243352 0.53155659 0.52334226 0.48155561 0.47101597
 0.45831031 0.45777354 0.48100383 0.44900825 0.48371605 0.491699
 0.48754308 0.48223391 0.50693135 0.51816704 0.54062635 0.51588492
 0.49581084 0.52902991 0.51585399 0.50492836 0.46908324 0.44887147
 0.46446364 0.4500739 ]
1 day output [[0.45221782]]
2 day input [0.60364964 0.59702888 0.59727151 0.580766   0.57260309 0.59584464
 0.60163449 0.62665789 0.64084469 0.57469327 0.58413493 0.56475041
 0.55855682 0.56716557 0.55763497 0.53760965 0.52485532 0.50701661
 0.50166436 0.50331694 0.52412267 0.49329541 0.47516156 0.49938032
 0.53243352 0.53155659 0.52334226 0.48155561 0.47101597 0.45831031
 0.45777354 0.48100383 0.44900825 0.48371605 0.491699   0.48754308
 0.48223391 0.50693135 0.51816704 0.54062635 0.51588492 0.49581084
 0.52902991 0.51585399 0.50492836 0.46908324 0.44887147 0.46446364
 0.4500739  0.45221782]
2 day output [[0.4518387]]
3 day input [0.59702888 0.59727151 0.580766   0.57260309 0.59584464 0.60163449
 0.62665789 0.64084469 0.57469327 0.58413493 0.56475041 0.55855682
 0.56716557 0.55763497 0.53760965 0.52485532 0.50701661 0.50166436
 0.50331694 0.52412267 0.49329541 0.47516156 0.49938032 0.53243352
 0.53155659 0.52334226 0.48155561 0.47101597 0.45831031 0.45777354
 0.48100383 0.44900825 0.48371605 0.491699   0.48754308 0.48223391
 0.50693135 0.51816704 0.54062635 0.51588492 0.49581084 0.52902991
 0.51585399 0.50492836 0.46908324 0.44887147 0.46446364 0.4500739
 0.45221782 0.4518387 ]
3 day output [[0.44893545]]
4 day input [0.59727151 0.580766   0.57260309 0.59584464 0.60163449 0.62665789
 0.64084469 0.57469327 0.58413493 0.56475041 0.55855682 0.56716557
 0.55763497 0.53760965 0.52485532 0.50701661 0.50166436 0.50331694
 0.52412267 0.49329541 0.47516156 0.49938032 0.53243352 0.53155659
 0.52334226 0.48155561 0.47101597 0.45831031 0.45777354 0.48100383
 0.44900825 0.48371605 0.491699   0.48754308 0.48223391 0.50693135
 0.51816704 0.54062635 0.51588492 0.49581084 0.52902991 0.51585399
 0.50492836 0.46908324 0.44887147 0.46446364 0.4500739  0.45221782
 0.4518387  0.44893545]
4 day output [[0.44493628]]
[[0.45007389783859253], [0.45221781730651855], [0.4518387019634247], [0.4489354491233825
7], [0.44493627548217773]]
```

In [114…

```
future_5 = scaler.inverse_transform(lst_output)
future_5
```

Out[114]:

```
array([[10703.43426817],
       [10726.31844259],
       [10722.2717693 ],
       [10691.28247826],
       [10648.59533982]])
```

In [115…

```
future_5_upper = future_5 + 0.05*future_5
future_5_upper
```

Out[115]:

```
array([[11238.60598158],
       [11262.63436472],
       [11258.38535777],
       [11225.84660217],
       [11181.02510681]])
```

Loading [MathJax]/extensions/Safe.js

In [116…

```
future_5_lower = future_5 - 0.05*future_5
future_5_lower
```

Out[116]:

```
array([[10168.26255476],
       [10190.00252046],
       [10186.15818084],
       [10156.71835434],
       [10116.16557283]])
```

In [118…

```
future_nasdaq = pd.DataFrame(future_5,columns=['future_naasdaq'])
future_nasdaq['future_upper'] = future_5_upper
future_nasdaq['future_lower'] = future_5_lower
future_nasdaq
```

Out[118]:

	future_naasdaq	future_upper	future_lower
0	10703.434268	11238.605982	10168.262555
1	10726.318443	11262.634365	10190.002520
2	10722.271769	11258.385358	10186.158181
3	10691.282478	11225.846602	10156.718354
4	10648.595340	11181.025107	10116.165573

In [120…

```
future_nasdaq.to_csv('stock_pred_nasdaq.csv')
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: