LetsGrowMore Task-3

bold text

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
df=pd.read_csv("https://raw.githubusercontent.com/mwitiderrick/stockprice/master/NSE-TATAGLOBAL.csv")
df=df.iloc[::-1]
df.head()
```

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
2034	2010-07-21	122.1	123.00	121.05	121.10	121.55	658666	803.56
2033	2010-07-22	120.3	122.00	120.25	120.75	120.90	293312	355.17

Automatic saving failed. This file was updated remotely or in another tab. Show diff

340.31

```
df.tail()
```

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
4	2018-09-24	233.55	239.20	230.75	234.00	233.30	3423509	7999.55
3	2018-09-25	233.30	236.75	232.00	236.25	236.10	2349368	5503.90
2	2018-09-26	240.00	240.00	232.50	235.00	234.25	2240909	5248.60
1	2018-09-27	234.55	236.80	231.10	233.80	233.25	5082859	11859.95
0	2018-09-28	234.05	235.95	230.20	233.50	233.75	3069914	7162.35

```
df.shape
```

```
(2035, 8)
```

```
df.columns
```

```
Index(['Date', 'Open', 'High', 'Low', 'Last', 'Close', 'Total Trade Quantity',
       'Turnover (Lacs)'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2035 entries, 2034 to 0
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Date                  2035 non-null   object 
 1   Open                  2035 non-null   float64
 2   High                  2035 non-null   float64
 3   Low                   2035 non-null   float64
 4   Last                  2035 non-null   float64
 5   Close                 2035 non-null   float64
 6   Total Trade Quantity  2035 non-null   int64  
 7   Turnover (Lacs)       2035 non-null   float64
dtypes: float64(6), int64(1), object(1)
memory usage: 127.3+ KB
```

```
df.describe()
```

	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
count	2035.000000	2035.000000	2035.000000	2035.000000	2035.00000	2.035000e+03	2035.000000
mean	149.713735	151.992826	147.293931	149.474251	149.45027	2.335681e+06	3899.980565
std	48.664509	49.413109	47.931958	48.732570	48.71204	2.091778e+06	4570.767877

```
#Data Preprocessing
df.isnull().sum()
```

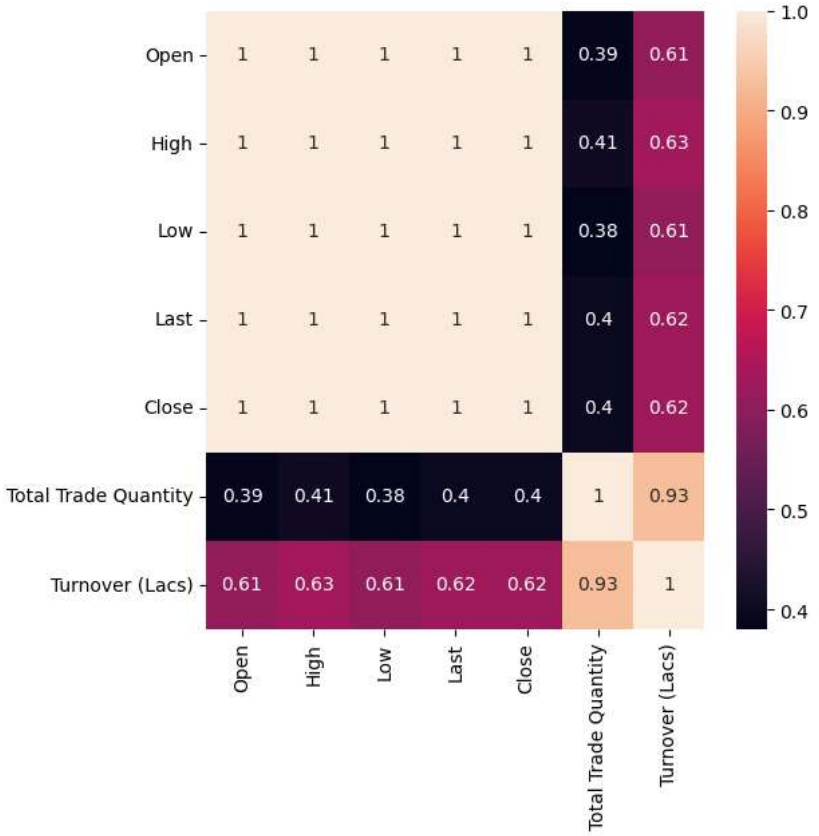
```
Date                  0
Open                  0
High                  0
Low                   0
Last                  0
Close                 0
Total Trade Quantity  0
Turnover (Lacs)       0
dtype: int64
```

```
duplicates= df.duplicated()
duplicates.value_counts() # no duplicates present
```

```
False    2035
dtype: int64
```



```
<Axes: >
```




```
df_high=df.reset_index()['High']
plt.plot(df_high)
```

```
[<matplotlib.lines.Line2D at 0x7f97dee9a050>]
```


```
## As LSTM are not robust to the scale of the data, so we apply MinMax Scaler to transform our values in the range of 0 and 1.##
```

bold text

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range = (0,1))
df_high = scaler.fit_transform(np.array(df_high).reshape(-1,1))
```

```
df_high.shape
```

```
(2035, 1)
```

```
df_high
```



```
[0.15938199],
[0.15917869],
...,
[0.6391543 ],
[0.62614353],
[0.62268754]])
```

```
#Split the data into train and test split
training_size = int(len(df_high) * 0.75)
test_size = len(df_high) - training_size
train_data,test_data = df_high[0:training_size,:], df_high[training_size:len(df_high),:1]
```

```
training_size, test_size
```

```
(1526, 509)
```

```
# 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]
        dataX.append(a)
        dataY.append(dataset[i + time_step, 0])
    return np.array(dataX), np.array(dataY)
```

```
time_step = 100
x_train, y_train = create_dataset(train_data, time_step)
x_test, y_test = create_dataset(test_data, time_step)
```

```
#Reshape the input to be [samples, time steps, features] which is the requirement of LSTM
x_train = x_train.reshape(x_train.shape[0], x_train.shape[1], 1)
x_test = x_test.reshape(x_test.shape[0], x_test.shape[1], 1)
```

```
print(x_train.shape), print(y_train.shape)
```

```
(1425, 100, 1)
(1425,)
(None, None)
```

```
print(x_test.shape), print(y_test.shape)
```

```
(408, 100, 1)
(408,)
(None, None)
```

```
import tensorflow as tf
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense
from tensorflow.python.keras.layers import LSTM


### Create the Stacked LSTM model
model = Sequential()
model.add(LSTM(50, return_sequences = True, input_shape = (100,1)))
model.add(LSTM(50, return_sequences = True))
model.add(LSTM(50))
model.add(Dense(1))
model.compile(loss = 'mean_squared_error', optimizer = 'adam')


model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 100, 50)           10400
_________________________________________________________________
lstm_1 (LSTM)                (None, 100, 50)           20200
_________________________________________________________________
lstm_2 (LSTM)                (None, 50)                20200
_________________________________________________________________
```



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

```
model.fit(x_train, y_train, validation_data = (x_test, y_test), epochs = 100, batch_size = 64, verbose = 1)
```

```
23/23 [==============================] - 10s 421ms/step - loss: 1.1150e-04 - val_loss: 6.0017
Epoch 98/100
23/23 [==============================] - 12s 513ms/step - loss: 1.2381e-04 - val_loss: 7.8035e-04
Epoch 99/100
23/23 [==============================] - 10s 450ms/step - loss: 1.1862e-04 - val_loss: 9.8941e-04
Epoch 100/100
23/23 [==============================] - 10s 449ms/step - loss: 1.2630e-04 - val_loss: 5.0547e-04
<tensorflow.python.keras.callbacks.History at 0x7f9786d9b730>
```

```
#predictions
#Lets predict and check performance metrics
train_predict = model.predict(x_train)
test_predict = model.predict(x_test)


#Transform back to original form
train_predict = scaler.inverse_transform(train_predict)
test_predict = scaler.inverse_transform(test_predict)


#Calculate RMSE performance metrics
import math
from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(y_train, train_predict))
```

```
136.00024491956285
```

#Test Data RMSE

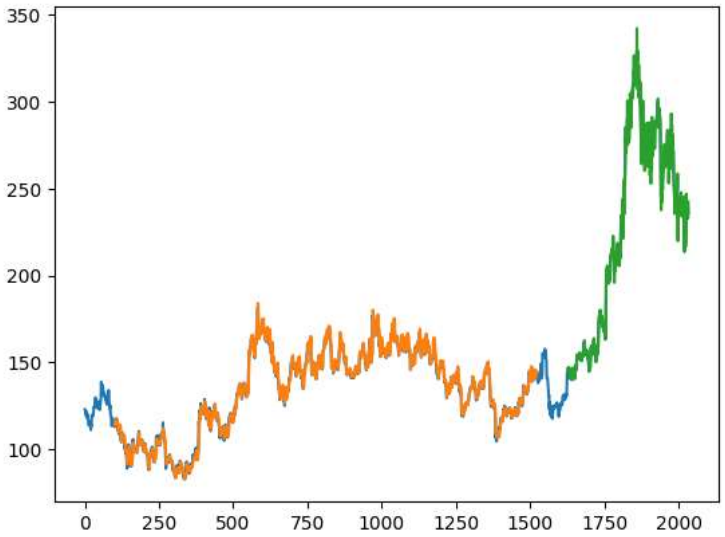

```
232.108189091383
```

```
#Plotting
#Shift train prediction for plotting
look_back = 100
trainPredictPlot = np.empty_like(df_high)
trainPredictPlot[:,:] = np.nan
trainPredictPlot[look_back:len(train_predict) + look_back, :] = train_predict

#Shift test prediction for plotting
testPredictPlot = np.empty_like(df_high)
testPredictPlot[:,:] = np.nan
testPredictPlot[len(train_predict) + (look_back * 2)+1:len(df_high) - 1, :] = test_predict


#Plot baseline and predictions
plt.plot(scaler.inverse_transform(df_high))
plt.plot(trainPredictPlot)
plt.plot(testPredictPlot)
plt.show()

print("Green indicates the Predicted Data")
print("Blue indicates the Complete Data")
print("Orange indicates the Train Data")
```




```
Green indicates the Predicted Data
Blue indicates the Complete Data
Orange indicates the Train Data
```

```
#Predict the next 28 days Stock Price
len(test_data), x_test.shape
```

```
(509, (408, 100, 1))
```

```
x_input = test_data[409:].reshape(1,-1)
x_input.shape
```

```
(1, 100)
```

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

lst_output=[]
n_steps=100
nextNumberOfDays = 28
i=0

while(i<nextNumberOfDays):



    if(len(temp_input)>100):
        x_input=np.array(temp_input[1:])
        print("{} day input {}".format(i,x_input))
        x_input=x_input.reshape(1,-1)
```



```
        print("{} day output {}".format(i,yhat))
        temp_input.extend(yhat[0].tolist())
        temp_input=temp_input[1:]
        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)
```

```
 0.67818662 0.65257166 0.64301687 0.65643423 0.67656028 0.67371417
 0.65114861 0.65521447 0.66761537 0.67838992 0.69993901 0.63122586
 0.63508843 0.64098394 0.64545639 0.64118723 0.63854442 0.63081927
 0.62472047 0.6330555  0.64728603 0.6574507  0.66761537 0.66720878
 0.64159382 0.62776987 0.63651149 0.63630819 0.62980281 0.62817646
 0.63813783 0.65358813 0.63183574 0.57653995 0.57816629 0.57613336
 0.61943484 0.57979264 0.57288067 0.62573694 0.63102257 0.6361049
 0.62776987 0.6269567  0.63590161 0.62594023 0.6391543  0.62614353
 0.62268754 0.61578542 0.60856575 0.59583825 0.58454323 0.57332188
 0.56321543 0.55462968 0.54750437 0.54174709 0.53720558 0.5336929
 0.53098476 0.52899045 0.52761322 0.52676195 0.52635247 0.52629787
 0.52646524 0.52681202 0.52725804 0.52772969 0.52816713 0.52851886
 0.52874333 0.52881092 0.52870315 0.52841264]
27 day output [[0.5279415]]
```

```
day_new = np.arange(1,101)
day_pred = np.arange(101,129)
```

```
day_new.shape
```

```
(100,)
```

```
day_pred.shape
```

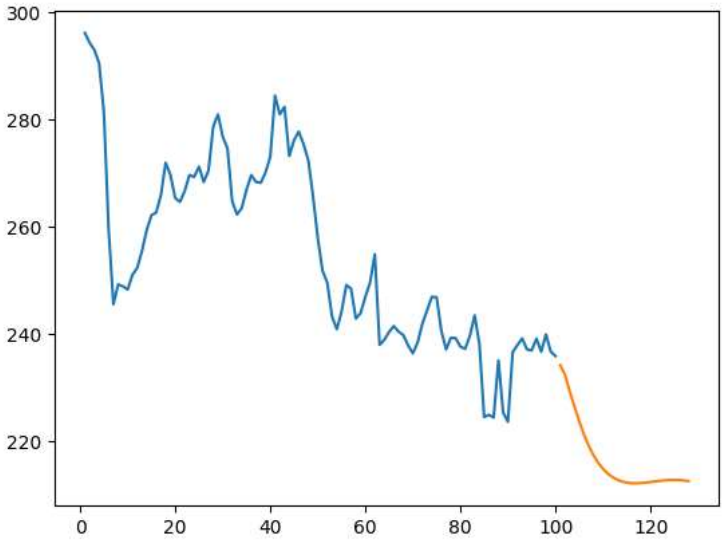
```
(28,)
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
2035
```

```
plt.plot(day_new, scaler.inverse_transform(df_high[1935:]))
plt.plot(day_pred, scaler.inverse_transform(lst_output))
```

```
[<matplotlib.lines.Line2D at 0x7f97871a6e30>]
```



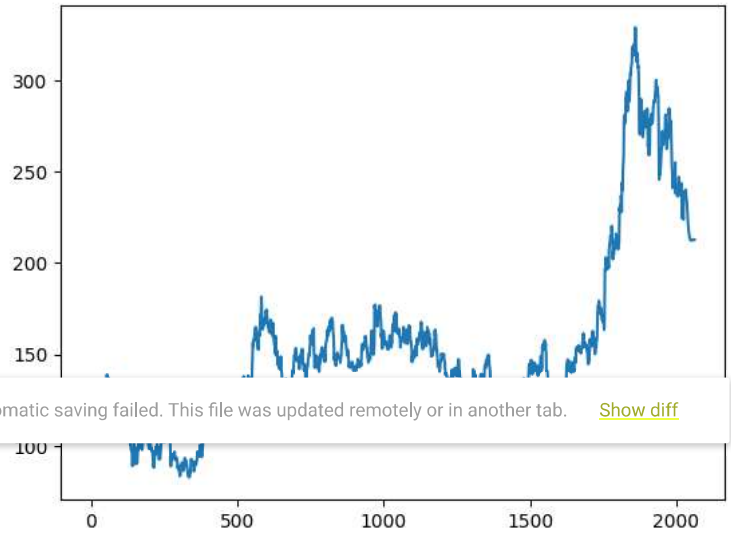

```
data_new = df_high.tolist()
data_new.extend(lst_output)
plt.plot(data_new[2000:])
```

```
[<matplotlib.lines.Line2D at 0x7f978711b340>]
```


```
data_new =scaler.inverse_transform(data_new).tolist()
```

```
plt.plot(data_new)
```

```
[<matplotlib.lines.Line2D at 0x7f978707c880>]
```



✓ 0s completed at 1:12 PM