Task-1 : BEGINNER LEVEL TASK

Task_1-2: Stock Market Prediction And Forecasting Using Stacked LSTM

Datasetlinks: : https://raw.githubusercontent.com/mwitiderrick/stockprice/master/NSE-TATAGLOBAL.csv (https://raw.githubusercontent.com/mwitiderrick/stockprice/master/NSE-TATAGLOBAL.csv)

Name: Janhavi Vijay Pawar

Importing Libraries

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.preprocessing import MinMaxScaler

import warnings
warnings.filterwarnings("ignore")
```

Reading the DataSet

In [2]:

```
#Import the data and remove rows containing NAN values
stk = pd.read_csv("stock.csv")
```

In [3]:

```
stk.head() # Prints first Five Rows
```

Out[3]:

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

In [4]:

```
stk.tail() # Prints last five rows
```

Out[4]:

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
2030	2010-07-27	117.6	119.50	112.00	118.80	118.65	586100	694.98
2031	2010-07-26	120.1	121.00	117.10	117.10	117.60	658440	780.01
2032	2010-07-23	121.8	121.95	120.25	120.35	120.65	281312	340.31
2033	2010-07-22	120.3	122.00	120.25	120.75	120.90	293312	355.17
2034	2010-07-21	122.1	123.00	121.05	121.10	121.55	658666	803.56

In [5]:

```
stk.describe()
```

Out[5]:

	Open	High	Low	Last	Close	Total Trade Quantity	Turr (I
count	2035.000000	2035.000000	2035.000000	2035.000000	2035.00000	2.035000e+03	2035.00
mean	149.713735	151.992826	147.293931	149.474251	149.45027	2.335681e+06	3899.98
std	48.664509	49.413109	47.931958	48.732570	48.71204	2.091778e+06	4570.76
min	81.100000	82.800000	80.000000	81.000000	80.95000	3.961000e+04	37.04
25%	120.025000	122.100000	118.300000	120.075000	120.05000	1.146444e+06	1427.46
50%	141.500000	143.400000	139.600000	141.100000	141.25000	1.783456e+06	2512.03
75%	157.175000	159.400000	155.150000	156.925000	156.90000	2.813594e+06	4539.01
max	327.700000	328.750000	321.650000	325.950000	325.75000	2.919102e+07	55755.08

In [6]:

```
stk.isnull()
```

Out[6]:

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
2030	False	False	False	False	False	False	False	False
2031	False	False	False	False	False	False	False	False
2032	False	False	False	False	False	False	False	False
2033	False	False	False	False	False	False	False	False
2034	False	False	False	False	False	False	False	False

2035 rows × 8 columns

Sorting the Data

In [7]:

```
stk['Date']=pd.to_datetime(stk['Date'])
print(type(stk.Date[0]))
```

```
<class 'pandas._libs.tslibs.timestamps.Timestamp'>
```

In [8]:

```
df=stk.sort_values(by='Date')
df.head()
```

Out[8]:

	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
2032	2010-07-23	121.8	121.95	120.25	120.35	120.65	281312	340.31
2031	2010-07-26	120.1	121.00	117.10	117.10	117.60	658440	780.01
2030	2010-07-27	117.6	119.50	112.00	118.80	118.65	586100	694.98

In [9]:

```
df.reset_index(inplace=True)
```

In [10]:

```
df.head()
```

Out[10]:

	index	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	2034	2010-07-21	122.1	123.00	121.05	121.10	121.55	658666	803.56
1	2033	2010-07-22	120.3	122.00	120.25	120.75	120.90	293312	355.17
2	2032	2010-07-23	121.8	121.95	120.25	120.35	120.65	281312	340.31
3	2031	2010-07-26	120.1	121.00	117.10	117.10	117.60	658440	780.01
4	2030	2010-07-27	117.6	119.50	112.00	118.80	118.65	586100	694.98

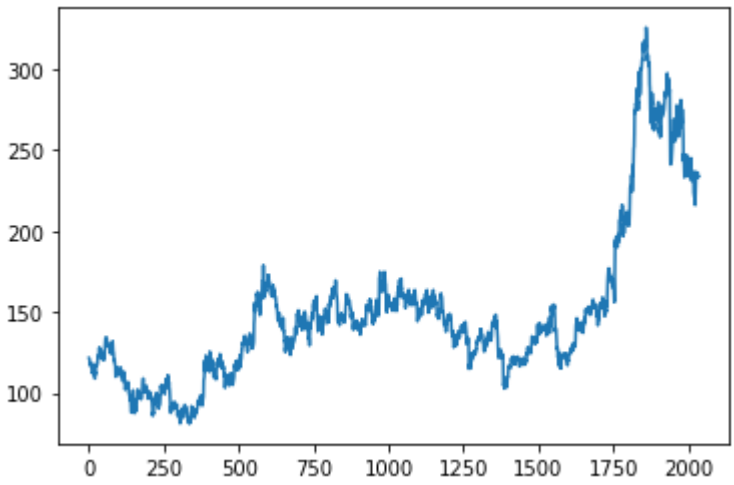
Data Visualization

In [11]:

```
plt.plot(df['Close'])
```

Out[11]:

```
[<matplotlib.lines.Line2D at 0x7f79f8ed05d0>]
```



In [12]:

```
dff=df['Close']
dff
```

Out[12]:

```
0       121.55
1       120.90
2       120.65
3       117.60
4       118.65
         ...
2030    233.30
2031    236.10
2032    234.25
2033    233.25
2034    233.75
Name: Close, Length: 2035, dtype: float64
```

Min Max Scaler

In [13]:

```
scaler=MinMaxScaler(feature_range=(0,1))
dff=scaler.fit_transform(np.array(dff).reshape(-1,1))
dff
```

Out[13]:

```
array([[0.16584967],
       [0.16319444],
       [0.1621732 ],
       ...,
       [0.62622549],
       [0.62214052],
       [0.62418301]])
```

Spliting the Dataset

In [14]:

```
training_size=int(len(dff)*0.70)
test_size=len(dff)-training_size
train_data,test_data=dff[0:training_size,:],dff[training_size:len(dff),:1]
```

Converting an Array of values into a Dataset matrix

In [15]:

```
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)
```

Spliting the Data into Train and Test

In [16]:

```
time_step = 100
X_train, y_train = create_dataset(train_data, time_step)
X_test, ytest = create_dataset(test_data, time_step)
```

In [17]:

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

```
(1323, 100)
(1323,)
```

Out[17]:

```
(None, None)
```

In [18]:

```
print(X_test.shape), print(ytest.shape)
```

```
(510, 100)
(510,)
```

Out[18]:

```
(None, None)
```

In [19]:

```
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)
```

Creating the stacked LSTM Model

In [20]:

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

In [21]:

```
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

 dense (Dense)               (None, 1)                 51

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

In [22]:

```
model.fit(X_train,y_train,validation_split=0.1,epochs=60,batch_size=64,verbose=1)
```

```
Epoch 1/60
19/19 [==============================] - 11s 298ms/step - loss: 0.0099 - v
al_loss: 0.0022
Epoch 2/60
19/19 [==============================] - 7s 349ms/step - loss: 0.0017 - va
l_loss: 0.0014
Epoch 3/60
19/19 [==============================] - 7s 387ms/step - loss: 9.0782e-04
- val_loss: 0.0012
Epoch 4/60
19/19 [==============================] - 7s 357ms/step - loss: 8.4069e-04
- val_loss: 0.0011
Epoch 5/60
19/19 [==============================] - 5s 276ms/step - loss: 8.1129e-04
- val_loss: 0.0011
Epoch 6/60
19/19 [==============================] - 3s 180ms/step - loss: 8.2129e-04
- val_loss: 0.0011
Epoch 7/60
```

Prediction and Checking Performance

In [23]:

```
test_predict=model.predict(X_test)
```

In [24]:

```
test_predicted=scaler.inverse_transform(test_predict)
test_predicted
```

```
       [226.03673 ],
       [228.11343 ],
       [228.45093 ],
       [229.2298  ],
       [222.62845 ],
       [216.06458 ],
       [212.7183  ],
       [213.67584 ],
       [215.61908 ],
       [214.01822 ],
       [215.49237 ],
       [223.54834 ],
       [231.23088 ],
       [233.15855 ],
       [231.00076 ],
       [227.71596 ],
       [224.64618 ],
       [224.62502 ],
       [224.81697 ]], dtype=float32)
```

Calculating the Performance

In [25]:

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

In [26]:

```
performance = math.sqrt(mean_squared_error(ytest,test_predict))
performance
```

Out[26]:

```
0.04607845860418348
```