## BEGINNER LEVEL TASK-02

# Stock Market Prediction And Forecasting Using Stacked LSTM

In [2]:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [14]:

```
import pandas as pd
import io
import requests
import datetime
```

Import Dataset

In [56]:

```
df1 = pd.read_csv("C:/Users/Dell/Desktop/@apj.csv/tatatest.csv")
df.head()
```

Out[56]:

	Open	High levels	Low levels	Last	Close	Total Trade Quantity	Turnover (Lacs)	Close: 30 Day Mean
0	234.05	235.95	230.20	233.50	233.75	3069914	7162.35	NaN
1	234.55	236.80	231.10	233.80	233.25	5082859	11859.95	NaN
2	240.00	240.00	232.50	235.00	234.25	2240909	5248.60	NaN
3	233.30	236.75	232.00	236.25	236.10	2349368	5503.90	NaN
4	233.55	239.20	230.75	234.00	233.30	3423509	7999.55	NaN

In [57]:

```
df2 =pd.read_csv("C:/Users/Dell/Desktop/@apj.csv/NSE-TATAGLOBAL.csv")
```

In [58]:

```
df.head()
```

Out[58]:

	Open	High levels	Low levels	Last	Close	Total Trade Quantity	Turnover (Lacs)	Close: 30 Day Mean
0	234.05	235.95	230.20	233.50	233.75	3069914	7162.35	NaN
1	234.55	236.80	231.10	233.80	233.25	5082859	11859.95	NaN
2	240.00	240.00	232.50	235.00	234.25	2240909	5248.60	NaN
3	233.30	236.75	232.00	236.25	236.10	2349368	5503.90	NaN
4	233.55	239.20	230.75	234.00	233.30	3423509	7999.55	NaN

Shape of data

In [17]:

```
df.shape
```

Out[17]:

```
(2035, 8)
```

```
Collaboration of  information about data
```

In [16]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2035 entries, 0 to 2034
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Date                  2035 non-null   object 
 1   Open                  2035 non-null   float64
 2   High levels           2035 non-null   float64
 3   Low levels            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
```

In [18]:

```
df.describe()
```

Out[18]:

	Open	High levels	Low levels	Last	Close	Total Trade Quantity	T
count	2035.000000	2035.000000	2035.000000	2035.000000	2035.00000	2.035000e+03	203
mean	149.713735	151.992826	147.293931	149.474251	149.45027	2.335681e+06	389
std	48.664509	49.413109	47.931958	48.732570	48.71204	2.091778e+06	457
min	81.100000	82.800000	80.000000	81.000000	80.95000	3.961000e+04	37
25%	120.025000	122.100000	118.300000	120.075000	120.05000	1.146444e+06	142
50%	141.500000	143.400000	139.600000	141.100000	141.25000	1.783456e+06	251
75%	157.175000	159.400000	155.150000	156.925000	156.90000	2.813594e+06	453
max	327.700000	328.750000	321.650000	325.950000	325.75000	2.919102e+07	5575

In [21]:

```
df.dtypes
```

Out[21]:

```
Date                     object
Open                    float64
High levels             float64
Low levels              float64
Last                    float64
Close                   float64
Total Trade Quantity      int64
Turnover (Lacs)         float64
dtype: object
```

Total percentage of data is missing

In [22]:

```
missing_values_count = df.isnull().sum()

total_cells = np.product(df.shape)

total_missing = missing_values_count.sum()

percentage_missing = (total_missing/total_cells)*100

print(percentage_missing)
```

```
0.0
```

In [25]:

```
NAN = [(c, df[c].isnull().mean()*100) for c in df]
NAN = pd.DataFrame(NAN, columns=['column_name', 'percentage'])
df
```

Out[25]:

	Date	Open	High levels	Low levels	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	28-09-2018	234.05	235.95	230.20	233.50	233.75	3069914	7162.35
1	27-09-2018	234.55	236.80	231.10	233.80	233.25	5082859	11859.95
2	26-09-2018	240.00	240.00	232.50	235.00	234.25	2240909	5248.60
3	25-09-2018	233.30	236.75	232.00	236.25	236.10	2349368	5503.90
4	24-09-2018	233.55	239.20	230.75	234.00	233.30	3423509	7999.55
...	...	...	...	...	...	...	...	...
2030	27-07-2010	117.60	119.50	112.00	118.80	118.65	586100	694.98
2031	26-07-2010	120.10	121.00	117.10	117.10	117.60	658440	780.01
2032	23-07-2010	121.80	121.95	120.25	120.35	120.65	281312	340.31
2033	22-07-2010	120.30	122.00	120.25	120.75	120.90	293312	355.17
2034	21-07-2010	122.10	123.00	121.05	121.10	121.55	658666	803.56

2035 rows × 8 columns

In [ ]:

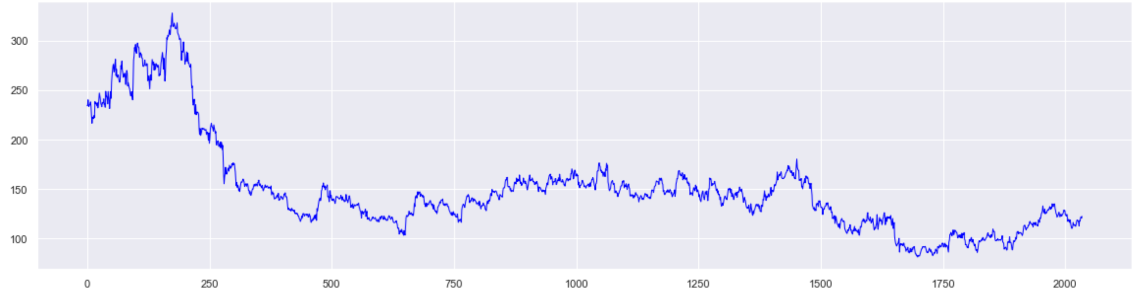
```
Data visuales
```

In [26]:

```
sns.set(rc = {'figure.figsize': (20, 5)})
df['Open'].plot(linewidth = 1,color='blue')
```

Out[26]:

```
<AxesSubplot:>
```



In [27]:

```
df.columns
```

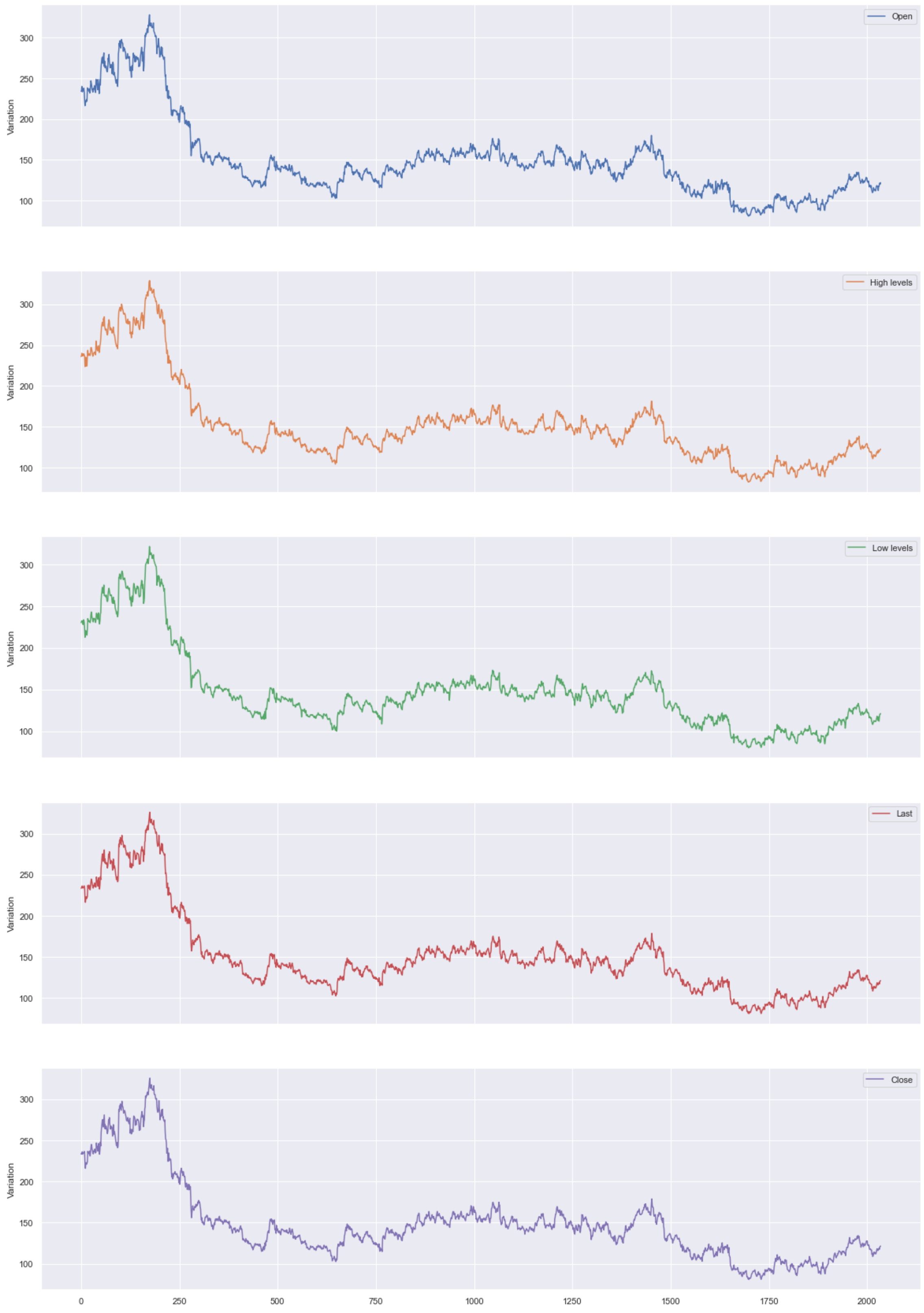
Out[27]:

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

In [32]:

```
cols_plot = ['Open','High levels','Low levels','Last','Close']
axes = df[cols_plot].plot(alpha = 1, figsize=(20, 30), subplots = True)

for ax in axes:
    ax.set_ylabel('Variation')
```



In [39]:

```
del df["Date"]
```

In [40]:

```
df.dtypes
```

Out[40]:

```
Open                      float64
High levels               float64
Low levels                float64
Last                      float64
Close                     float64
Total Trade Quantity        int64
Turnover (Lacs)           float64
dtype: object
```

7 day rolling mean

In [43]:

```
df.rolling(7).mean().head(25)
```

Out[43]:

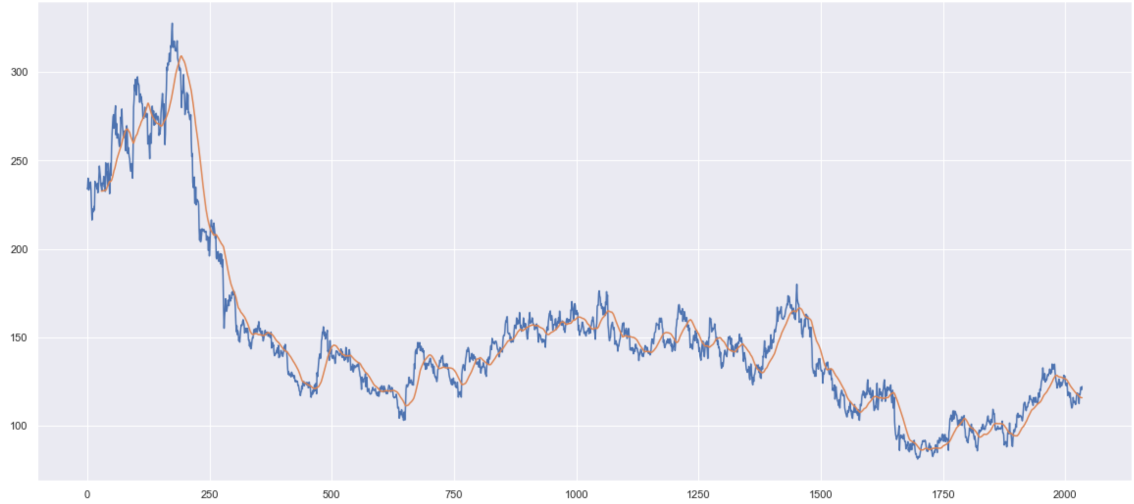
	Open	High levels	Low levels	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	235.200000	237.557143	231.135714	234.414286	234.307143	3.274848e+06	7652.388571
7	235.750000	238.028571	231.607143	234.700000	234.492857	3.209831e+06	7509.724286
8	235.550000	238.200000	231.485714	235.071429	234.971429	2.936693e+06	6879.075714
9	233.185714	237.728571	230.171429	234.928571	234.928571	3.527693e+06	8241.347143
10	230.764286	235.864286	227.407143	232.842857	233.007143	3.845060e+06	8883.934286
11	229.185714	233.892857	225.135714	230.321429	230.535714	3.857272e+06	8846.257143
12	227.400000	233.628571	224.092857	228.507143	228.735714	4.159956e+06	9494.928571
13	225.264286	231.814286	222.042857	226.871429	227.028571	4.141448e+06	9429.222857
14	223.278571	229.778571	219.857143	224.792857	225.028571	4.016310e+06	9099.654286
15	221.685714	227.864286	217.707143	222.750000	223.000000	3.995196e+06	8989.585714
16	223.792857	228.078571	217.607143	221.242857	221.535714	3.591903e+06	8043.774286
17	226.600000	230.914286	220.807143	223.414286	223.542857	3.687891e+06	8406.114286
18	228.671429	232.964286	223.392857	226.028571	226.157143	3.665725e+06	8431.457143
19	230.507143	233.271429	225.028571	228.350000	228.157143	2.866757e+06	6629.497143
20	232.342857	235.157143	226.971429	229.928571	229.814286	2.889922e+06	6706.281429
21	234.200000	237.200000	228.842857	231.635714	231.571429	2.980773e+06	6951.288571
22	235.600000	239.307143	231.107143	233.735714	233.664286	2.833191e+06	6667.718571
23	235.071429	239.164286	232.192857	234.885714	234.707143	2.587985e+06	6114.222857
24	235.685714	238.742857	231.914286	234.692857	234.528571	2.060999e+06	4846.218571

In [50]:

```
df['Open'].plot(figsize=(20,9),alpha = 1)
df.rolling(window=30).mean()['Close'].plot(alpha = 1)
```

Out[50]:

```
<AxesSubplot:>
```

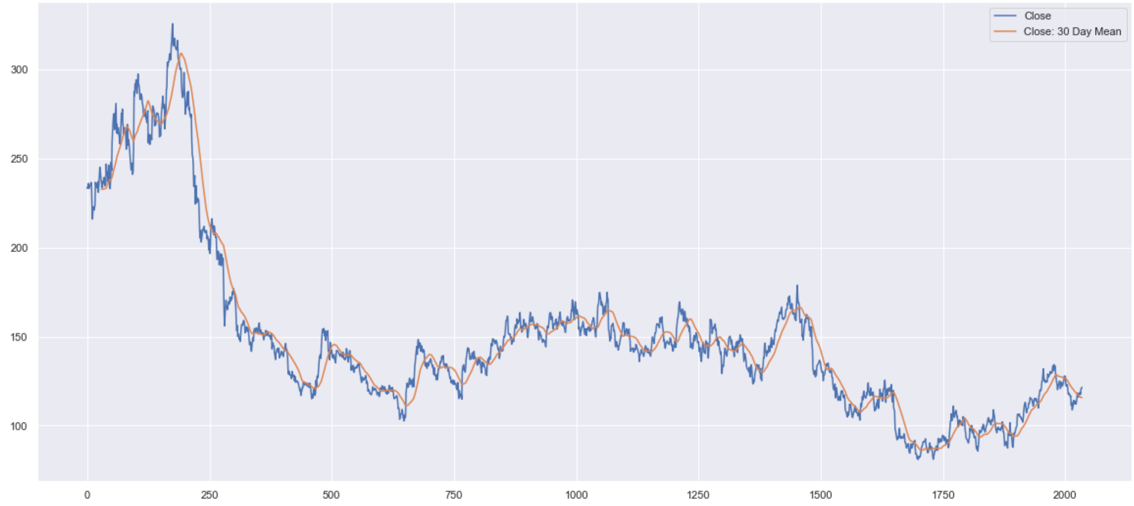


In [51]:

```
df['Close: 30 Day Mean'] = df['Close'].rolling(window=30).mean()
df[['Close','Close: 30 Day Mean']].plot(figsize=(20,9),alpha = 1)
```

Out[51]:

```
<AxesSubplot:>
```

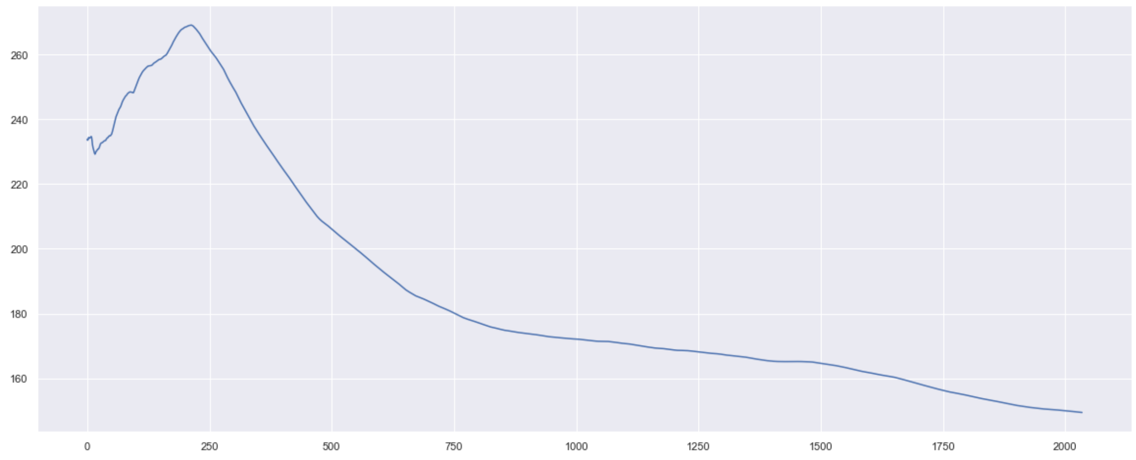


In [54]:

```
df['Close'].expanding(min_periods=1).mean().plot(figsize=(20,8),alpha = 1)
```

Out[54]:

```
<AxesSubplot:>
```



Optional specify a minimum numbe2of periods

In [59]:

```
df2=df1.reset_index()['Open']
df2
```

Out[59]:

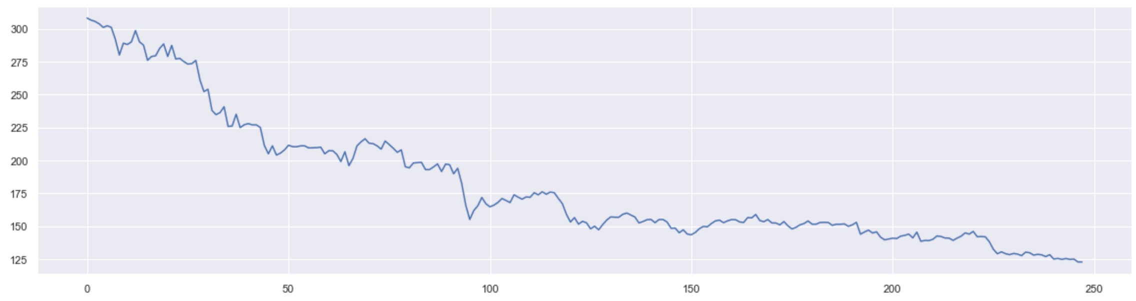
```
0      308.05
1      306.50
2      305.50
3      303.70
4      301.00
        ...  
243    125.40
244    124.75
245    125.00
246    122.80
247    122.80
Name: Open, Length: 248, dtype: float64
```

In [60]:

```
plt.plot(df2)
```

Out[60]:

```
[<matplotlib.lines.Line2D at 0x2980b9c0af0>]
```



LSTM are sensitive to the scale of the data using MinMax scaler

In [61]:

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler(feature_range=(0,1))
df2=scaler.fit_transform(np.array(df2).reshape(-1,1))
print(df2)
```

```
 [0.17381916]
 [0.16005398]
 [0.15978408]
 [0.15222672]
 [0.165722  ]
 [0.14817814]
 [0.13549258]
 [0.14197031]
 [0.15222672]
 [0.15762483]
 [0.16815115]
 [0.15492578]
 [0.15492578]
 [0.16194332]
 [0.16248313]
 [0.16194332]
 [0.15033738]
 [0.15465587]
 [0.15465587]
 [0.1560054 ]
```

In [63]:

```
train_size=int(len(df2)*0.75)
test_size=len(df2)-train_size
train_data,test_data=df2[0:train_size,:],df2[train_size:len(df2),:1]
```

In [64]:

```
train_size,test_size
```

Out[64]:

```
(186, 62)
```

In [65]:

```
train_data,test_data
```

Out[65]:

```
(array([[1.        ],
        [0.99163293],
        [0.98623482],
        [0.97651822],
        [0.96194332],
        [0.96869096],
        [0.96194332],
        [0.91336032],
        [0.848583  ],
        [0.89716599],
        [0.89176788],
        [0.9025641 ],
        [0.94898785],
        [0.9025641 ],
        [0.88933873],
        [0.82699055],
        [0.84264507],
```

In [75]:

```
def create_dataset(dataset, time_step=1):
    train_X, train_Y = [], []
    for i in range(len(dataset)-time_step-1):
        a = dataset[i:(i+time_step), 0]   ###i=0, 0,1,2,3-----99   100 
        train_X.append(a)
        train_Y.append(dataset[i + time_step, 0])
    return numpy.array(train_X), numpy.array(train_Y)
```

In [76]:

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

In [77]:

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

```
(85, 100)
(85,)
```

Out[77]:

```
(None, None)
```

In [84]:

```
X_train =X_train.reshape(X_train.shape[0],X_train.shape[1] , 1)
```

Thank you