Stock Market Prediction-LGM(Task-2)

September 1, 2022

Let's Grow More(LGMVIP)-"DATA SCIENCE INTERN"

LGMVIP August-22

SRAVANI BANDIRAJULA

BEGINNER LEVEL TASK

TASK-2-Stock Market Prediction And Forecasting Using Stacked LSTM

DatasetLink:https://raw.githubusercontent.com/mwitiderrick/stockprice/master/NSE-TATAGLOBAL.csv

Importing Libraries

[1]:
```
import numpy as np
import math
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

[2]:
```
pwd
```

[2]:
```
'C:\\Users\\Sravani'
```

[3]:
```
df=pd.read_csv('C:\\Users\\Sravani\StockMarketPrediction.csv')
```

[4]:
```
df
```

[4]:
```
            Date    Open    High     Low    Last   Close  \
0     2018-09-28  234.05  235.95  230.20  233.50  233.75
1     2018-09-27  234.55  236.80  231.10  233.80  233.25
2     2018-09-26  240.00  240.00  232.50  235.00  234.25
3     2018-09-25  233.30  236.75  232.00  236.25  236.10
4     2018-09-24  233.55  239.20  230.75  234.00  233.30
…            …       …      …       …      …      …
2030  2010-07-27  117.60  119.50  112.00  118.80  118.65
2031  2010-07-26  120.10  121.00  117.10  117.10  117.60
2032  2010-07-23  121.80  121.95  120.25  120.35  120.65
2033  2010-07-22  120.30  122.00  120.25  120.75  120.90
2034  2010-07-21  122.10  123.00  121.05  121.10  121.55
```

```
      Total Trade Quantity  Turnover (Lacs)
0                  3069914          7162.35
1                  5082859         11859.95
2                  2240909          5248.60
3                  2349368          5503.90
4                  3423509          7999.55
…                      …              …
2030                586100           694.98
2031                658440           780.01
2032                281312           340.31
2033                293312           355.17
2034                658666           803.56

[2035 rows x 8 columns]
```

[5]:

```
df.describe()
```

[5]:

```
              Open         High          Low         Last        Close  \
count  2035.000000  2035.000000  2035.000000  2035.000000  2035.00000
mean    149.713735   151.992826   147.293931   149.474251   149.45027
std      48.664509    49.413109    47.931958    48.732570    48.71204
min      81.100000    82.800000    80.000000    81.000000    80.95000
25%     120.025000   122.100000   118.300000   120.075000   120.05000
50%     141.500000   143.400000   139.600000   141.100000   141.25000
75%     157.175000   159.400000   155.150000   156.925000   156.90000
max     327.700000   328.750000   321.650000   325.950000   325.75000

       Total Trade Quantity  Turnover (Lacs)
count          2.035000e+03      2035.000000
mean           2.335681e+06      3899.980565
std            2.091778e+06      4570.767877
min            3.961000e+04        37.040000
25%            1.146444e+06      1427.460000
50%            1.783456e+06      2512.030000
75%            2.813594e+06      4539.015000
max            2.919102e+07     55755.080000
```

[6]:

```
df.tail()
```

[6]:

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

```
      Turnover (Lacs)
2030           694.98
2031           780.01
2032           340.31
2033           355.17
2034           803.56
```

```
[7]: df.dtypes
```

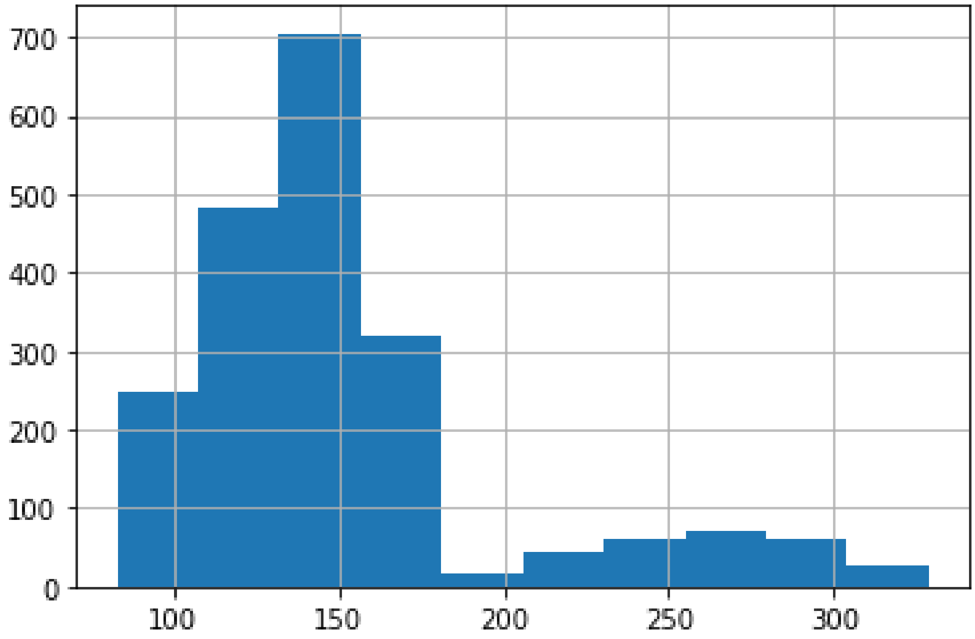
```
[7]: Date                        object
     Open                       float64
     High                       float64
     Low                        float64
     Last                       float64
     Close                      float64
     Total Trade Quantity         int64
     Turnover (Lacs)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

```
[8]: df['Date'].value_counts()
```

```
[8]: 2018-09-28    1
     2013-04-10    1
     2013-03-20    1
     2013-03-21    1
     2013-03-22    1
                  ..
     2016-01-11    1
     2016-01-12    1
     2016-01-13    1
     2016-01-14    1
     2010-07-21    1
     Name: Date, Length: 2035, dtype: int64
```

```
[9]: df['High'].hist()
```

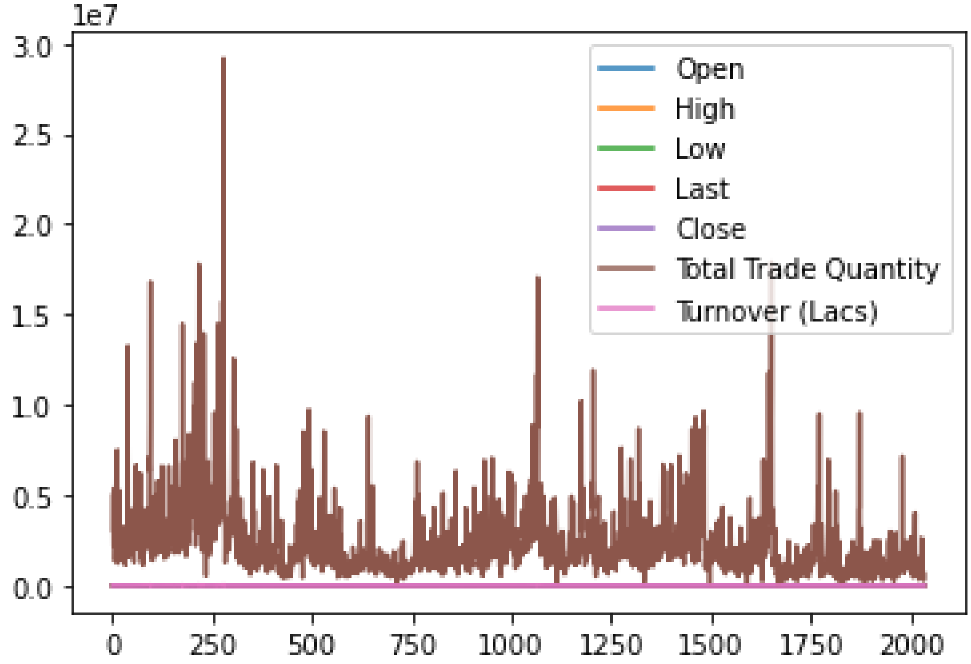
```
[9]: <AxesSubplot:>
```



[10]:
```
plt.figure(figsize=(18,6))
df.plot()
```

[10]:
```
<AxesSubplot:>
```

```
<Figure size 1296x432 with 0 Axes>
```



```
[11]: data_set = df.filter(['Close'])
      dataset = df.values
      training_data_len=math.ceil(len(df) * 8)
      training_data_len
```

```
[11]: 16280
```

```
[12]: dataset
```

```
[12]: array([['2018-09-28', 234.05, 235.95, …, 233.75, 3069914, 7162.35],
             ['2018-09-27', 234.55, 236.8, …, 233.25, 5082859, 11859.95],
             ['2018-09-26', 240.0, 240.0, …, 234.25, 2240909, 5248.6],
             …,
             ['2010-07-23', 121.8, 121.95, …, 120.65, 281312, 340.31],
             ['2010-07-22', 120.3, 122.0, …, 120.9, 293312, 355.17],
             ['2010-07-21', 122.1, 123.0, …, 121.55, 658666, 803.56]],
            dtype=object)
```

```
[13]: df = df.iloc[:, 0:5]
      df
```

```
[13]:             Date    Open    High     Low    Last
      0     2018-09-28  234.05  235.95  230.20  233.50
      1     2018-09-27  234.55  236.80  231.10  233.80
      2     2018-09-26  240.00  240.00  232.50  235.00
      3     2018-09-25  233.30  236.75  232.00  236.25
      4     2018-09-24  233.55  239.20  230.75  234.00
      …            …       …       …       …       …
      2030  2010-07-27  117.60  119.50  112.00  118.80
      2031  2010-07-26  120.10  121.00  117.10  117.10
      2032  2010-07-23  121.80  121.95  120.25  120.35
      2033  2010-07-22  120.30  122.00  120.25  120.75
      2034  2010-07-21  122.10  123.00  121.05  121.10

      [2035 rows x 5 columns]
```

```
[14]: training_set = df.iloc[:, 1:2].values
      training_set
```

```
[14]: array([[234.05],
             [234.55],
             [240.  ],
             …,
             [121.8 ],
             [120.3 ],
        [122.1 ]])
```

[15]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range = (0, 1))
data_training_scaled = scaler.fit_transform(training_set)
```

[16]:
```
features_set = []
labels = []
for i in range(60, 586):
  features_set.append(data_training_scaled[i - 60:i, 0])
  labels.append(data_training_scaled[i, 0])
```

[17]:
```
features_set, labels = np.array(features_set), np.array(labels)
```

[18]:
```
features_set = np.reshape(features_set, (features_set.shape[0], features_set.
 ↪shape[1], 1))
features_set.shape
```

[18]:
```
(526, 60, 1)
```

[19]:
```
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
```

[20]:
```
model = Sequential()
```

[21]:
```
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(features_set, labels, epochs=25, batch_size=10)
```

```
Epoch 1/25
53/53 [==============================] - 0s 1ms/step - loss: 0.0118
Epoch 2/25
53/53 [==============================] - 0s 592us/step - loss: 0.0118
Epoch 3/25
53/53 [==============================] - 0s 481us/step - loss: 0.0118
Epoch 4/25
53/53 [==============================] - 0s 513us/step - loss: 0.0118
Epoch 5/25
53/53 [==============================] - 0s 478us/step - loss: 0.0118
Epoch 6/25
53/53 [==============================] - 0s 524us/step - loss: 0.0118
Epoch 7/25
53/53 [==============================] - 0s 533us/step - loss: 0.0118
Epoch 8/25
53/53 [==============================] - 0s 519us/step - loss: 0.0118
Epoch 9/25
53/53 [==============================] - 0s 554us/step - loss: 0.0118
```

```
Epoch 10/25
53/53 [==============================] - 0s 531us/step - loss: 0.0118
Epoch 11/25
53/53 [==============================] - 0s 522us/step - loss: 0.0118
Epoch 12/25
53/53 [==============================] - 0s 539us/step - loss: 0.0118
Epoch 13/25
53/53 [==============================] - 0s 529us/step - loss: 0.0118
Epoch 14/25
53/53 [==============================] - 0s 542us/step - loss: 0.0118
Epoch 15/25
53/53 [==============================] - 0s 629us/step - loss: 0.0118
Epoch 16/25
53/53 [==============================] - 0s 517us/step - loss: 0.0118
Epoch 17/25
53/53 [==============================] - 0s 634us/step - loss: 0.0118
Epoch 18/25
53/53 [==============================] - 0s 482us/step - loss: 0.0118
Epoch 19/25
53/53 [==============================] - 0s 499us/step - loss: 0.0118
Epoch 20/25
53/53 [==============================] - 0s 487us/step - loss: 0.0118
Epoch 21/25
53/53 [==============================] - 0s 495us/step - loss: 0.0118
Epoch 22/25
53/53 [==============================] - 0s 490us/step - loss: 0.0118
Epoch 23/25
53/53 [==============================] - 0s 529us/step - loss: 0.0118
Epoch 24/25
53/53 [==============================] - 0s 504us/step - loss: 0.0118
Epoch 25/25
53/53 [==============================] - 0s 511us/step - loss: 0.0118
```

[21]:
```
<tensorflow.python.keras.callbacks.History at 0x2541d0bfdf0>
```

[22]:
```python
data_total = pd.concat((df['Open'], df['Open']), axis=0)
```

[23]:
```python
test_inputs = data_total[len(data_total) - len(df) - 20:].values
test_inputs.shape
```

[23]:
```
(2055,)
```

[24]:
```python
test_inputs = test_inputs.reshape(-1, 1)
test_inputs = scaler.transform(test_inputs)
```

[25]:
```python
test_feature = []
for i in range(60, 89):
  test_feature.append(test_inputs[i-60:i, 0])
```

[26]:
```
test_feature = np.array(test_feature)
test_feature = np.reshape(test_feature, (test_feature.shape[0] - test_feature.
 ↪shape[1], 1))
test_feature.shape
```

[26]:
```
(1740, 1)
```

[27]:
```
predictions = model.predict(test_feature)
```

[28]:
```
predictions
```

[28]:
```
array([[0.12489862],
       [0.14132197],
       [0.13098134],
       …,
       [0.72587186],
       [0.71695054],
       [0.7175588 ]], dtype=float32)
```

[29]:
```
x_train = df[0:1256]
y_train = df[1:1257]
print(x_train.shape)
print(y_train.shape)
```

```
(1256, 5)
(1256, 5)
```

[30]:
```
x_train
```

[30]:
```
            Date    Open    High     Low    Last
0     2018-09-28  234.05  235.95  230.20  233.50
1     2018-09-27  234.55  236.80  231.10  233.80
2     2018-09-26  240.00  240.00  232.50  235.00
3     2018-09-25  233.30  236.75  232.00  236.25
4     2018-09-24  233.55  239.20  230.75  234.00
…            …       …       …       …       …
1251  2013-09-04  142.00  145.35  140.65  143.60
1252  2013-09-03  144.10  145.20  140.70  141.80
1253  2013-09-02  139.40  144.40  139.35  144.00
1254  2013-08-30  138.10  140.65  136.70  139.20
1255  2013-08-29  137.00  140.40  137.00  137.10

[1256 rows x 5 columns]
```

[31]:
```
np.random.seed(1)
np.random.randn(4, 4)
```

[31]:

```
array([[ 1.62434536, -0.61175641, -0.52817175, -1.07296862],
       [ 0.86540763, -2.3015387 ,  1.74481176, -0.7612069 ],
       [ 0.3190391 , -0.24937038,  1.46210794, -2.06014071],
       [-0.3224172 , -0.38405435,  1.13376944, -1.09989127]])
```

[32]:

```
np.random.normal(1)
```

[32]:

```
0.8275717924495642
```

[33]:

```
np.random.normal(4)
```

[33]:

```
3.122141582078628
```

[34]:

```
np.random.seed(40)
```

[35]:

```
np.random.normal(size=1000, scale=100).std()
```

[35]:

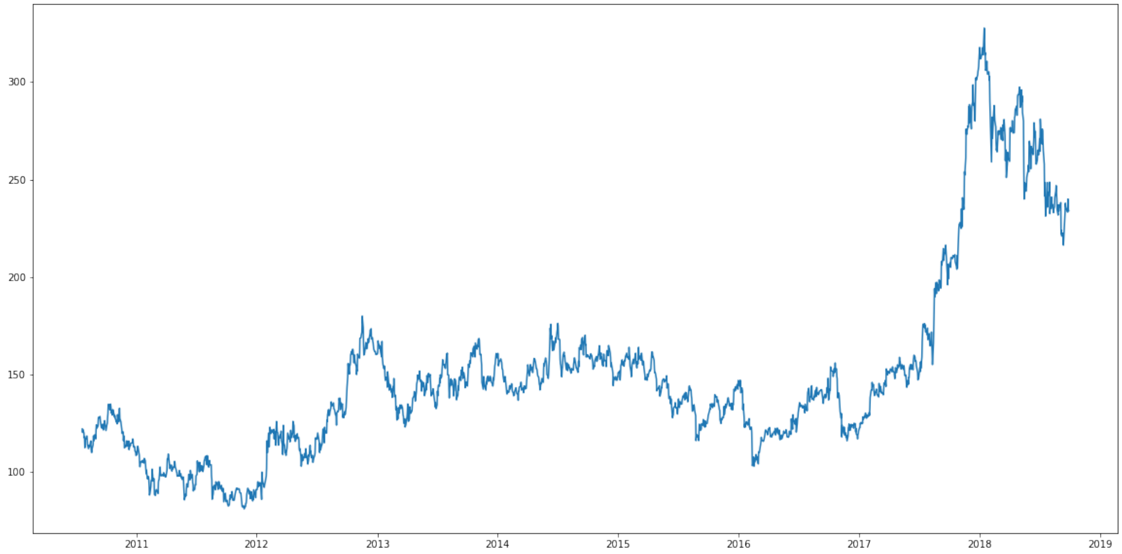
```
99.40257120628782
```

[36]:

```
df["Date"] = pd.to_datetime(df.Date)
df.index = df['Date']

plt.figure(figsize=(20, 10))
plt.plot(df["Open"], label='ClosePriceHist')
```

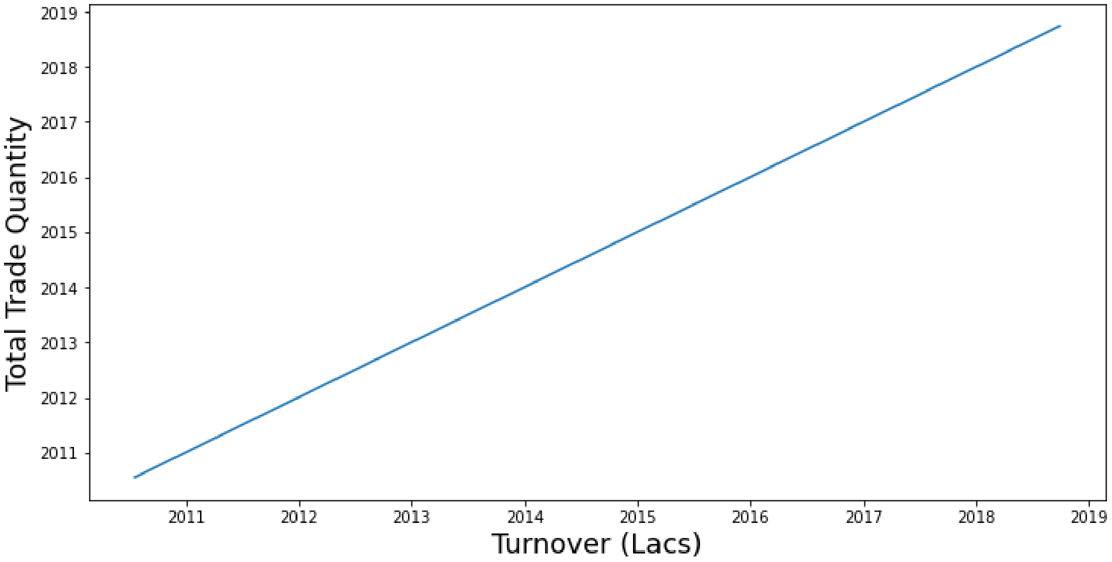
[36]:

```
[<matplotlib.lines.Line2D at 0x2541e467220>]
```



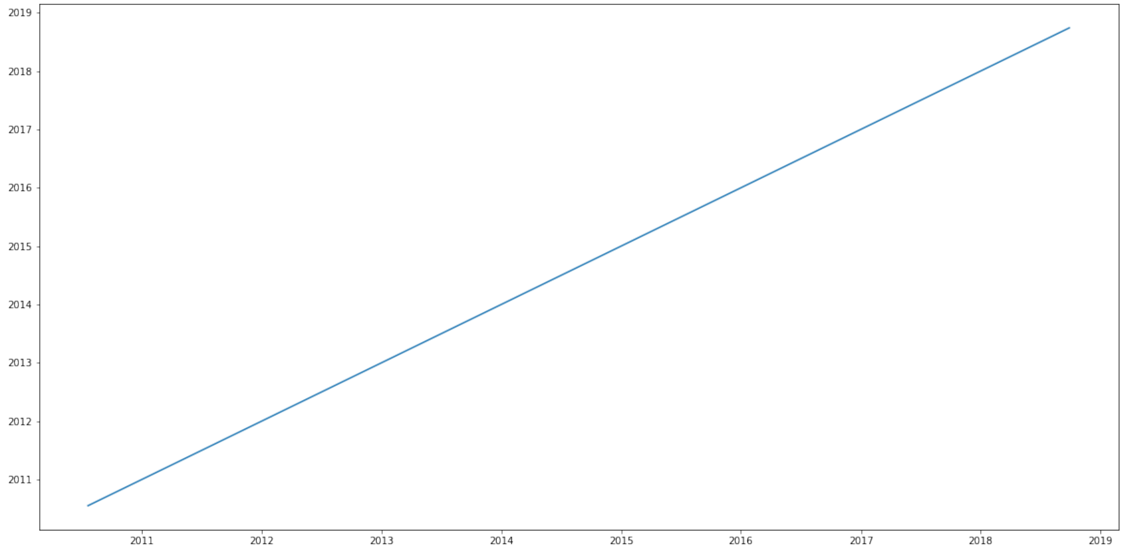
[37]:

```
plt.figure(figsize=(12,6))
plt.plot(df['Date'])
plt.xlabel('Turnover (Lacs)', fontsize=18)
plt.ylabel('Total Trade Quantity', fontsize=18)
plt.show()
```



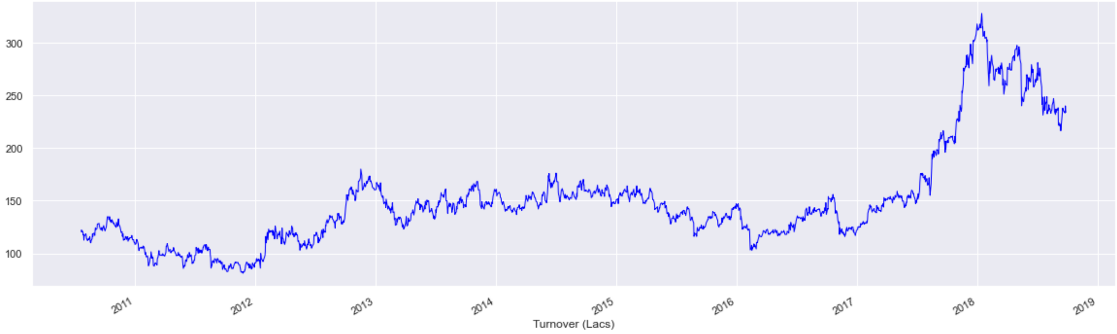
```
[38]: df["Turnover (Lacs)"] = pd.to_datetime(df.Date)
      df.index = df['Turnover (Lacs)']

      plt.figure(figsize=(20, 10))
      plt.plot(df["Turnover (Lacs)"], label='ClosePriceHist')
```

```
[38]: [<matplotlib.lines.Line2D at 0x2541e549d00>]
```



```
[39]: sns.set(rc = {'figure.figsize': (20, 6)})
      df['Open'].plot(linewidth = 1,color='blue')
```

```
[39]: <AxesSubplot:xlabel='Turnover (Lacs)'>
```



```
[40]: df.columns
```

```
[40]: Index(['Date', 'Open', 'High', 'Low', 'Last', 'Turnover (Lacs)'],
      dtype='object')
```

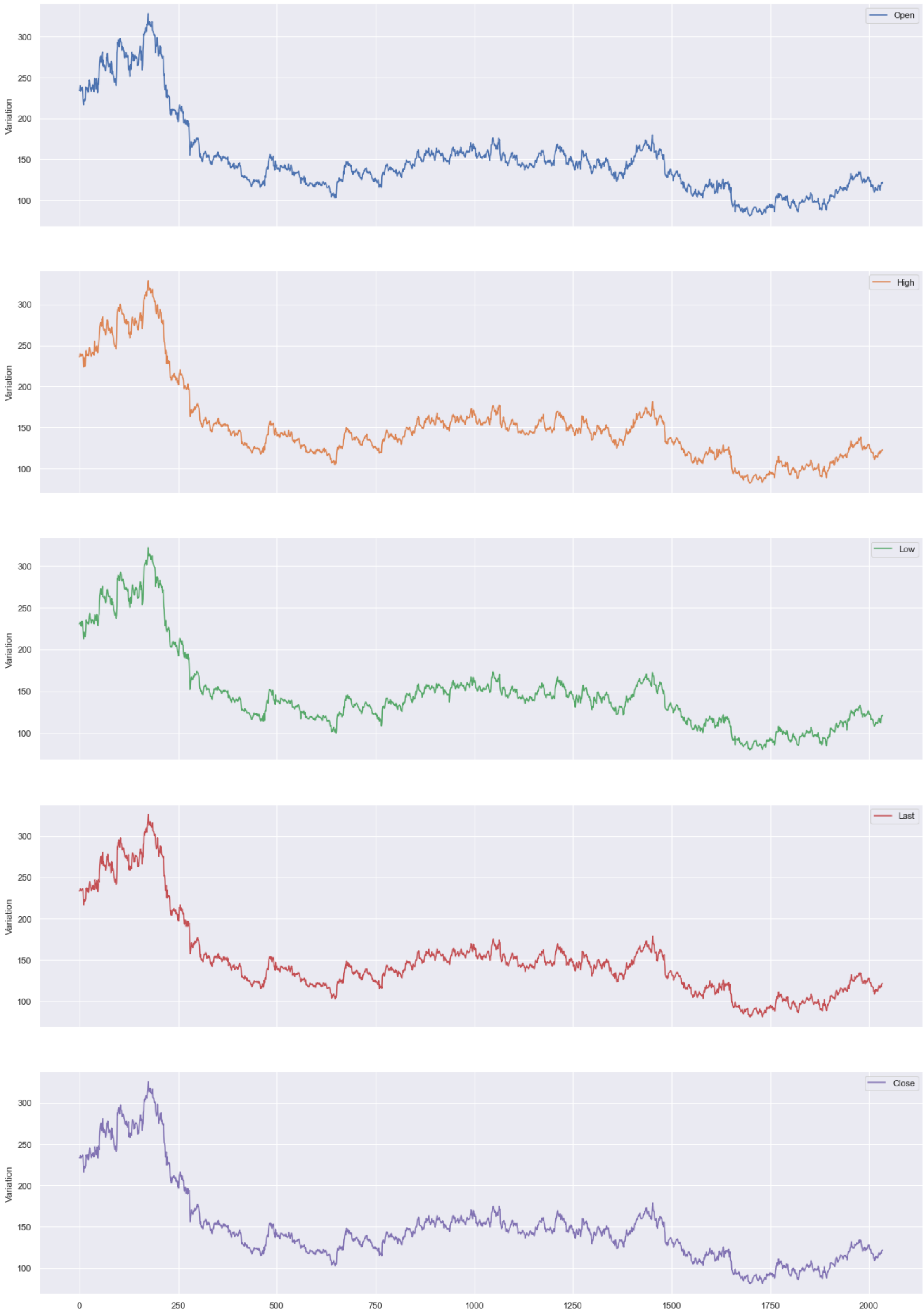
```
[41]: df=pd.read_csv('C:\\Users\\Sravani\StockMarketPrediction.csv')
      df
```

```
[41]:             Date    Open    High     Low    Last   Close  \
      0     2018-09-28  234.05  235.95  230.20  233.50  233.75
      1     2018-09-27  234.55  236.80  231.10  233.80  233.25
      2     2018-09-26  240.00  240.00  232.50  235.00  234.25
      3     2018-09-25  233.30  236.75  232.00  236.25  236.10
      4     2018-09-24  233.55  239.20  230.75  234.00  233.30
      …            …       …       …       …       …       …
      2030  2010-07-27  117.60  119.50  112.00  118.80  118.65
      2031  2010-07-26  120.10  121.00  117.10  117.10  117.60
      2032  2010-07-23  121.80  121.95  120.25  120.35  120.65
      2033  2010-07-22  120.30  122.00  120.25  120.75  120.90
      2034  2010-07-21  122.10  123.00  121.05  121.10  121.55

            Total Trade Quantity  Turnover (Lacs)
      0                  3069914          7162.35
      1                  5082859         11859.95
      2                  2240909          5248.60
      3                  2349368          5503.90
      4                  3423509          7999.55
```

```
…                   …                 …
2030                586100            694.98
2031                658440            780.01
2032                281312            340.31
2033                293312            355.17
2034                658666            803.56

[2035 rows x 8 columns]
```

[42]:

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

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


Open
High
Low
Last
Close
Variation
300
250
200
150
100
0
250
500
750
1000
1250
1500
1750
2000

[ ]:

```
```