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


import pandas as pd
import io
import requests
import datetime


df = pd.read_csv("https://raw.githubusercontent.com/mwitiderrick/stockprice/master/NSE-TAT
```

```
df.head()
```

	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

```
df1 = pd.read_csv("https://raw.githubusercontent.com/mwitiderrick/stockprice/master/NSE-TA
df1.head()
```

	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

```
df.shape
```

```
(2035, 8)
```

```
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                 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	
count	2035.000000	2035.000000	2035.000000	2035.000000	2035.00000	2.035000e+03	2
mean	149.713735	151.992826	147.293931	149.474251	149.45027	2.335681e+06	3
std	48.664509	49.413109	47.931958	48.732570	48.71204	2.091778e+06	4
min	81.100000	82.800000	80.000000	81.000000	80.95000	3.961000e+04	
25%	120.025000	122.100000	118.300000	120.075000	120.05000	1.146444e+06	1
50%	141.500000	143.400000	139.600000	141.100000	141.25000	1.783456e+06	2
75%	157.175000	159.400000	155.150000	156.925000	156.90000	2.813594e+06	4
max	327.700000	328.750000	321.650000	325.950000	325.75000	2.919102e+07	55

```
df.dtypes
```

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

```
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)
```

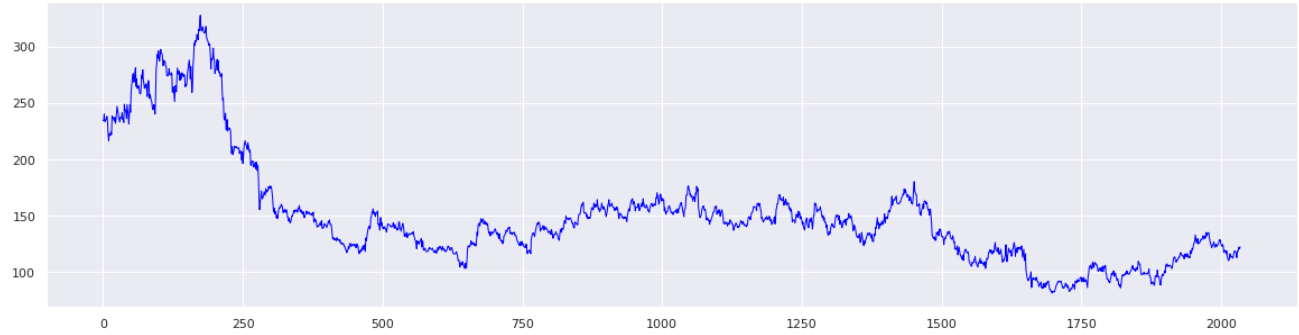
```
0.0
```

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

	column_name	percentage
0	Date	0.0
1	Open	0.0
2	High	0.0
3	Low	0.0
4	Last	0.0
5	Close	0.0
6	Total Trade Quantity	0.0
7	Turnover (Lacs)	0.0

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1dc2c19810>
```




```
df.columns
```

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

```
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

```
df["Date"]=pd.to_datetime(df.Date,format="%Y-%m-%d")
df.index=df['Date']
df
```

Date	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
2018-09-28	2018-09-28	234.05	235.95	230.20	233.50	233.75	3069914	7162.35
2018-09-27	2018-09-27	234.55	236.80	231.10	233.80	233.25	5082859	11859.95
2018-09-26	2018-09-26	240.00	240.00	232.50	235.00	234.25	2240909	5248.60
2018-09-25	2018-09-25	233.30	236.75	232.00	236.25	236.10	2349368	5503.90
2018-09-24	2018-09-24	233.55	239.20	230.75	234.00	233.30	3423509	7999.55
...	...	...	...	...	...	...	...	...
2010-07-27	2010-07-27	117.60	119.50	112.00	118.80	118.65	586100	694.98
2010-07-26	2010-07-26	120.10	121.00	117.10	117.10	117.60	658440	780.01

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

```
df.dtypes
```

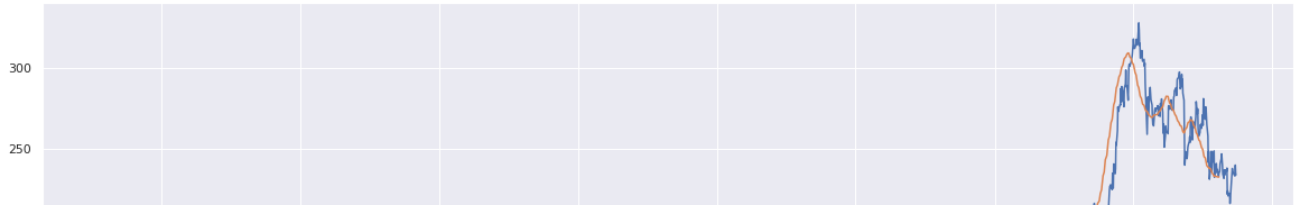
```
Open                    float64
High                    float64
Low                     float64
Last                    float64
Close                   float64
Total Trade Quantity      int64
Turnover (Lacs)         float64
dtype: object
```

```
df.rolling(7).mean().head(10)
```

Date	Open	High	Low	Last	Close	Total Trade Quantity	Turr (L
2018-09-28	NaN	NaN	NaN	NaN	NaN	NaN	
2018-09-27	NaN	NaN	NaN	NaN	NaN	NaN	
2018-09-26	NaN	NaN	NaN	NaN	NaN	NaN	
2018-09-25	NaN	NaN	NaN	NaN	NaN	NaN	
2018-09-24	NaN	NaN	NaN	NaN	NaN	NaN	
2018-09-21	NaN	NaN	NaN	NaN	NaN	NaN	
2018							

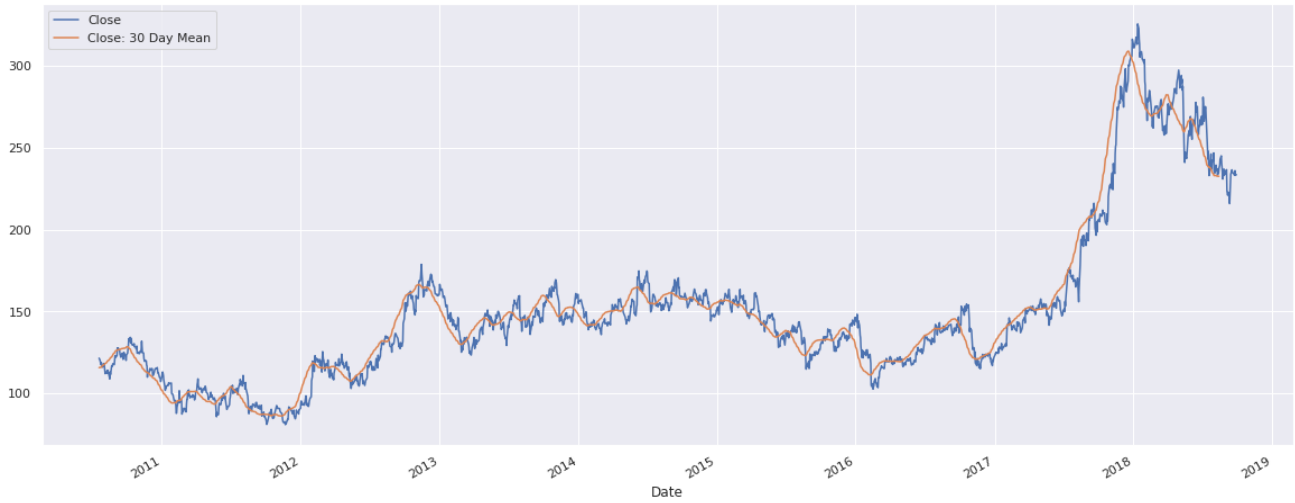
```
df['Open'].plot(figsize=(20,8),alpha = 1)
df.rolling(window=30).mean()['Close'].plot(alpha = 1)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1dc1f09150>
```



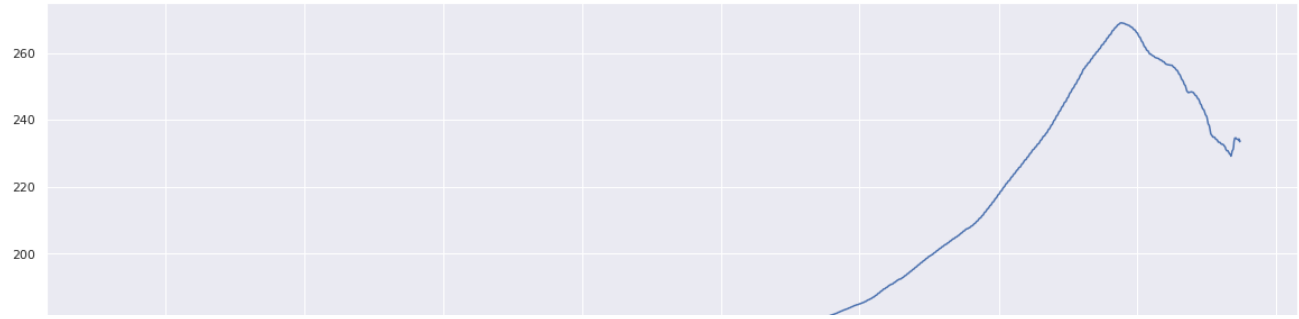

```
df['Close: 30 Day Mean'] = df['Close'].rolling(window=30).mean()
df[['Close','Close: 30 Day Mean']].plot(figsize=(20,8),alpha = 1)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1dc1e81110>
```




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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1dc1f09ed0>
```

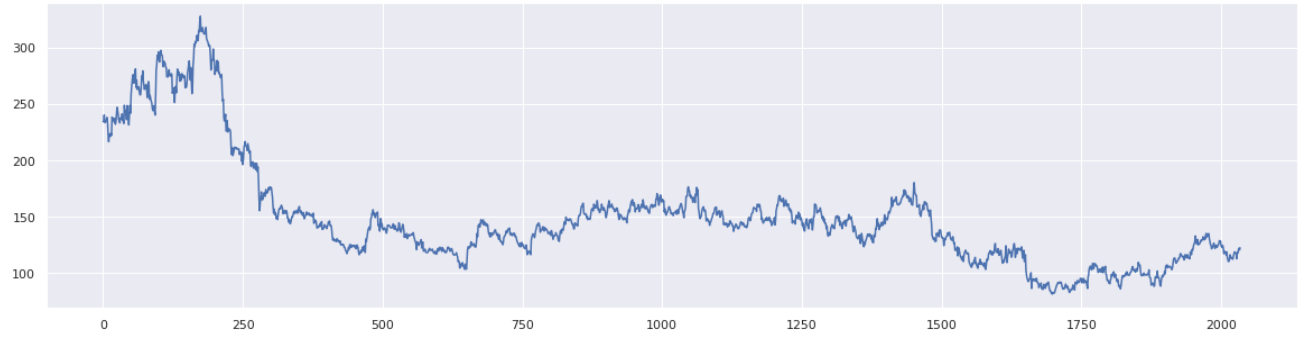



```
df2=df1.reset_index()['Open']
df2
```

```
0       234.05
1       234.55
2       240.00
3       233.30
4       233.55
         ...
2030    117.60
2031    120.10
2032    121.80
2033    120.30
2034    122.10
Name: Open, Length: 2035, dtype: float64
```

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

```
[<matplotlib.lines.Line2D at 0x7f1dc1d10990>]
```




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

```
[[0.6202352 ]
 [0.62226277]
 [0.64436334]
 ...
```

```
       [0.16504461]
       [0.15896188]
       [0.16626115]]
```

```
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]
```

```
train_size,test_size
```

```
     (1526, 509)
```

```
train_data,test_data
```

```
            [0.19870235],
            [0.21796431],
            [0.21553122],
            [0.20600162],
            [0.21654501],
            [0.21654501],
            [0.2175588 ],
            [0.19870235],
            [0.19018654],
            [0.17802109],
            [0.175588  ],
            [0.16301703],
            [0.16707218],
            [0.17112733],
            [0.17639903],
            [0.18349554],
            [0.1717356 ],
            [0.16423358],
            [0.16991079],
```

```
            [0.17619627],
            [0.16788321],
            [0.16909976],
            [0.17396594],
            [0.17741281],
            [0.18268451],
            [0.19221411],
            [0.18896999],
            [0.19018654],
            [0.17396594],
            [0.17092457],
            [0.16788321],
            [0.17477697],
            [0.16443633],
            [0.14557989],
            [0.15287916],
            [0.15369019],
            [0.15044607],
            [0.14152474],
            [0.15145985],
            [0.13341444],
            [0.12530414],
            [0.11719384],
            [0.11780211],
            [0.12489862],
            [0.14132198]
```

```
[0.14132198],
[0.13098135],
[0.12935929],
[0.13240065],
[0.12895377],
[0.12530414],
[0.13381995],
[0.14557989],
[0.15166261],
[0.15085158],
[0.14679643],
[0.14355231],
[0.12733171],
[0.14963504],
[0.14801298],
```

```
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)
```

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

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

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

```
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)
```

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

```
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_2"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_6 (LSTM)                (None, 100, 50)           10400
_________________________________________________________________
lstm_7 (LSTM)                (None, 100, 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
_________________________________________________________________
```

```
model.fit(X_train,y_train,validation_data=(X_test,ytest),epochs=100,batch_size=64,verbose=
```

```
Epoch 71/100
23/23 [==============================] - 4s 194ms/step - loss: 3.6984e-04 - val_lo
Epoch 72/100
23/23 [==============================] - 4s 192ms/step - loss: 4.2137e-04 - val_lo
Epoch 73/100
23/23 [==============================] - 4s 189ms/step - loss: 3.5805e-04 - val_lo
Epoch 74/100
23/23 [==============================] - 4s 194ms/step - loss: 3.4126e-04 - val_lo
Epoch 75/100
23/23 [==============================] - 4s 193ms/step - loss: 3.9137e-04 - val_lo
Epoch 76/100
23/23 [==============================] - 4s 193ms/step - loss: 3.7434e-04 - val_lo
Epoch 77/100
23/23 [==============================] - 4s 193ms/step - loss: 4.2114e-04 - val_lo
Epoch 78/100
23/23 [==============================] - 4s 194ms/step - loss: 3.7205e-04 - val_lo
Epoch 79/100
23/23 [==============================] - 4s 196ms/step - loss: 3.1387e-04 - val_lo
Epoch 80/100
23/23 [==============================] - 4s 193ms/step - loss: 3.0322e-04 - val_lo
Epoch 81/100
23/23 [==============================] - 4s 193ms/step - loss: 3.4725e-04 - val_lo
Epoch 82/100
23/23 [==============================] - 4s 196ms/step - loss: 2.8825e-04 - val_lo
Epoch 83/100
23/23 [==============================] - 4s 196ms/step - loss: 2.9167e-04 - val_lo
Epoch 84/100
23/23 [==============================] - 4s 196ms/step - loss: 2.8828e-04 - val_lo
Epoch 85/100
23/23 [==============================] - 4s 195ms/step - loss: 2.7689e-04 - val_lo
Epoch 86/100
23/23 [==============================] - 4s 194ms/step - loss: 2.8874e-04 - val_lo
Epoch 87/100
23/23 [==============================] - 4s 194ms/step - loss: 2.9837e-04 - val_lo
Epoch 88/100
23/23 [==============================] - 4s 193ms/step - loss: 2.7817e-04 - val_lo
Epoch 89/100
23/23 [==============================] - 4s 196ms/step - loss: 2.6157e-04 - val_lo
Epoch 90/100
23/23 [==============================] - 4s 195ms/step - loss: 3.1714e-04 - val_lo
Epoch 91/100
23/23 [==============================] - 5s 197ms/step - loss: 2.7386e-04 - val_lo
Epoch 92/100
23/23 [==============================] - 5s 196ms/step - loss: 2.6693e-04 - val_lo
```

```
Epoch 93/100
23/23 [==============================] - 4s 195ms/step - loss: 3.2088e-04 - val_lo
Epoch 94/100
23/23 [==============================] - 4s 196ms/step - loss: 3.3713e-04 - val_lo
Epoch 95/100
23/23 [==============================] - 4s 195ms/step - loss: 3.4057e-04 - val_lo
Epoch 96/100
23/23 [==============================] - 4s 195ms/step - loss: 3.1105e-04 - val_lo
Epoch 97/100
23/23 [==============================] - 4s 195ms/step - loss: 2.6496e-04 - val_lo
Epoch 98/100
23/23 [==============================] - 4s 196ms/step - loss: 2.5973e-04 - val_lo
Epoch 99/100
23/23 [==============================] - 4s 193ms/step - loss: 2.5126e-04 - val_lo
Epoch 100/100
```

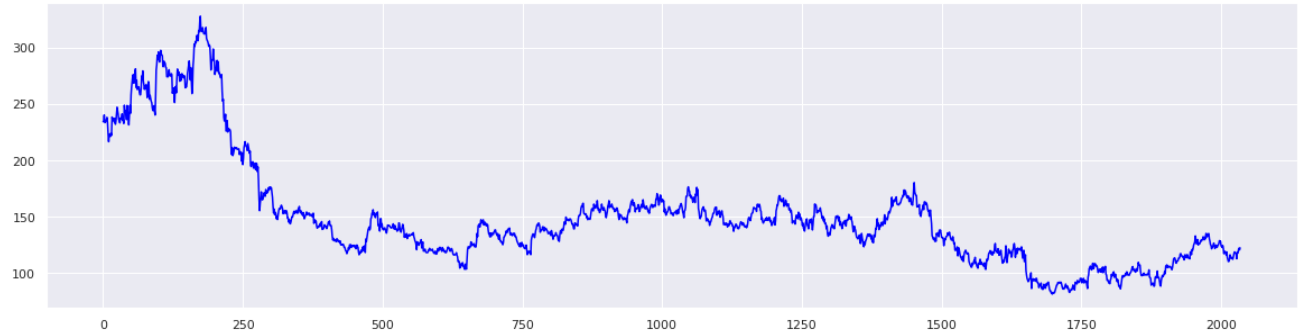
```
import tensorflow as tf


train_predict=model.predict(X_train)
test_predict=model.predict(X_test)


train_predict=scaler.inverse_transform(train_predict)
test_predict=scaler.inverse_transform(test_predict)


import math
from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(y_train,train_predict))
```

```
161.54842460530494
```

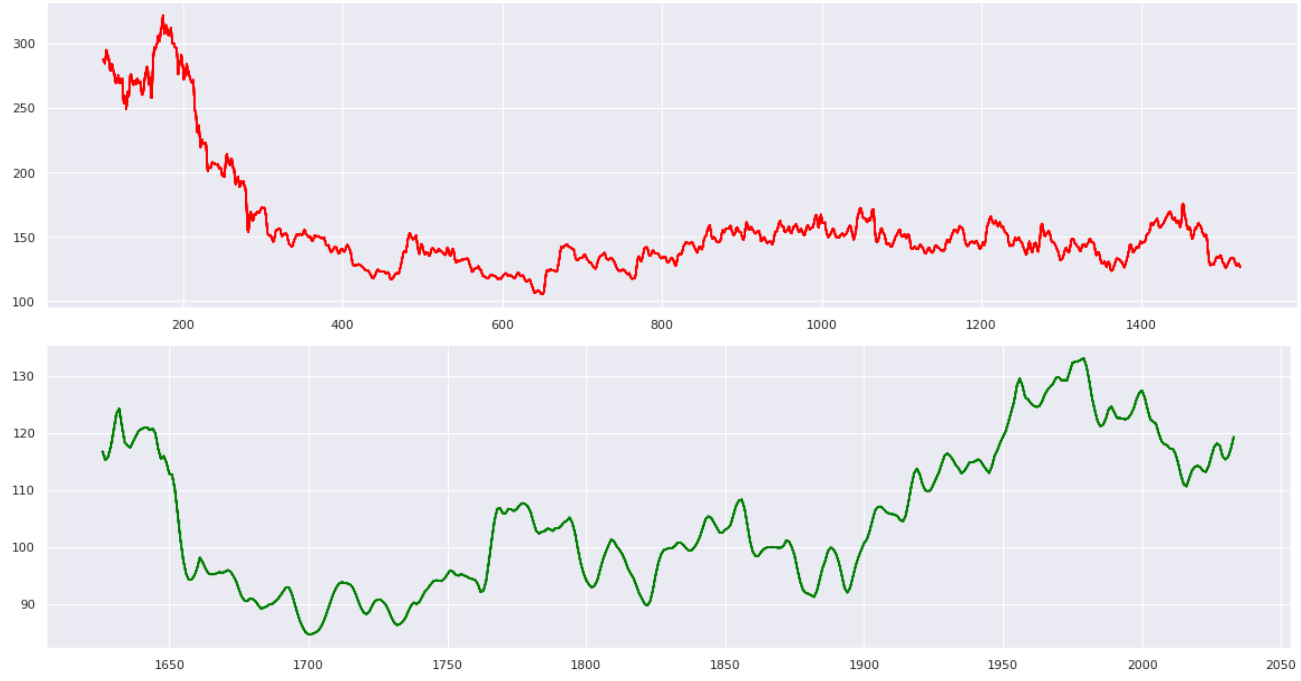
```
math.sqrt(mean_squared_error(ytest,test_predict))
```

```
105.48500700182166
```

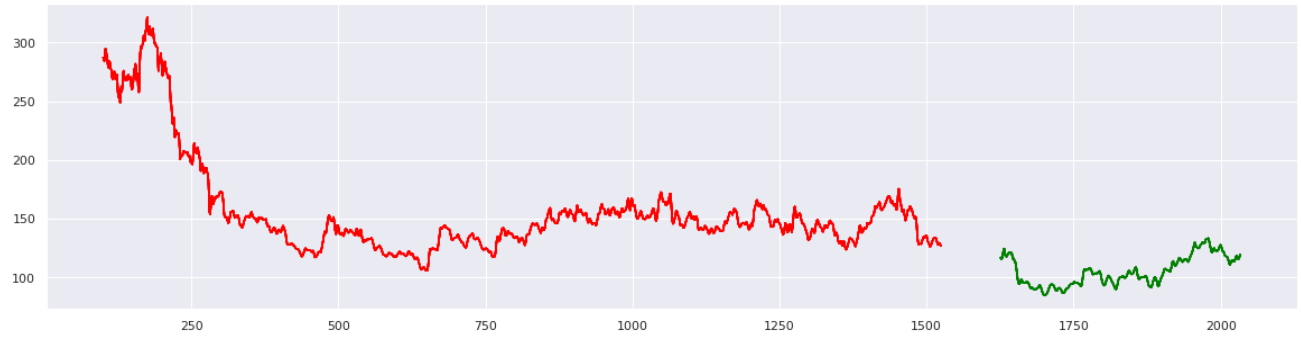
```
look_back=100
trainPredictPlot = numpy.empty_like(df1)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[look_back:len(train_predict)+look_back, :] = train_predict


testPredictPlot = numpy.empty_like(df1)
testPredictPlot[:, :] = numpy.nan
testPredictPlot[len(train_predict)+(look_back*2)+1:len(df1)-1, :] = test_predict


pred  = scaler.inverse_transform(df2)
plt.plot(pred,color='blue')
plt.show()
```



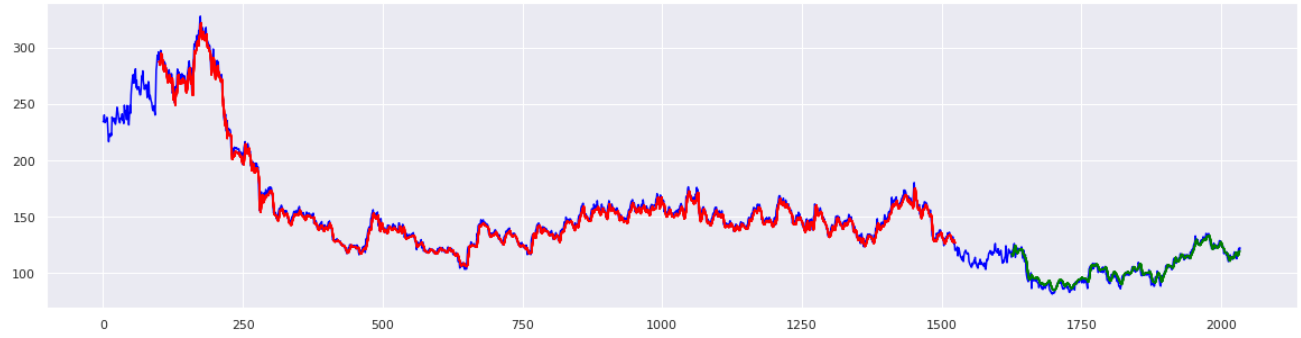

```
plt.plot(trainPredictPlot,color='red')
plt.show()
plt.plot(testPredictPlot,color='green')
plt.show()
```




```
plt.plot(trainPredictPlot,color='red')
plt.plot(testPredictPlot,color='green')
plt.show()
```




```
plt.plot(pred,color='blue')
plt.plot(trainPredictPlot,color='red')
plt.plot(testPredictPlot,color='green')
plt.show()
```




```
len(test_data)
```

```
509
```

```
x_input=test_data[341:].reshape(1,-1)
x_input.shape
```

```
(1, 168)
```