```
import numpy as np
import pandas as pd
import datetime
import math
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense , LSTM
from sklearn.metrics import mean_squared_error
```

```
Dataset_link='https://raw.githubusercontent.com/mwitiderrick/stockprice/master/NSE-TATAGLOBAL.csv'
```

```
df= pd.read_csv(Dataset_link, parse_dates=True,)
df.reset_index()
df.head(11)
```

	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
5	2018-09-21	235.00	237.00	227.95	233.75	234.60	5395319	12589.59
6	2018-09-19	235.95	237.20	233.45	234.60	234.90	1362058	3202.78
7	2018-09-18	237.90	239.25	233.50	235.50	235.05	2614794	6163.70
8	2018-09-17	233.15	238.00	230.25	236.40	236.60	3170894	7445.41
9	2018-09-14	223.45	236.70	223.30	234.00	233.95	6377909	14784.50
10	2018-09-12	216.35	223.70	212.65	221.65	222.65	4570939	10002.01

```
df.sample(11)
```

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
1151	2014-01-29	140.00	144.40	139.60	141.80	142.15	2441955	3477.95
2004	2010-09-01	121.65	123.65	120.20	122.70	123.00	2274887	2781.63
1643	2012-02-10	123.05	124.75	117.80	118.85	118.95	5100633	6184.93
943	2014-12-09	158.75	158.85	154.10	154.70	154.85	1339740	2092.18
2014	2010-08-18	110.15	111.20	108.10	108.50	108.85	959932	1052.23
1468	2012-10-22	156.00	161.85	156.00	159.40	159.60	2532870	4049.41
1097	2014-04-21	158.25	160.00	155.55	156.50	156.15	2939902	4642.58
537	2016-07-29	140.90	144.70	139.20	142.50	142.15	3056272	4351.61
934	2014-12-22	147.55	148.25	145.60	147.70	147.50	1657451	2437.44
495	2016-09-30	136.95	140.90	136.20	139.55	139.60	1471713	2041.23
1132	2014-02-25	143.20	143.55	141.00	141.10	141.20	1034913	1469.54

```
df.columns
```

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

```
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.isnull().sum()
```

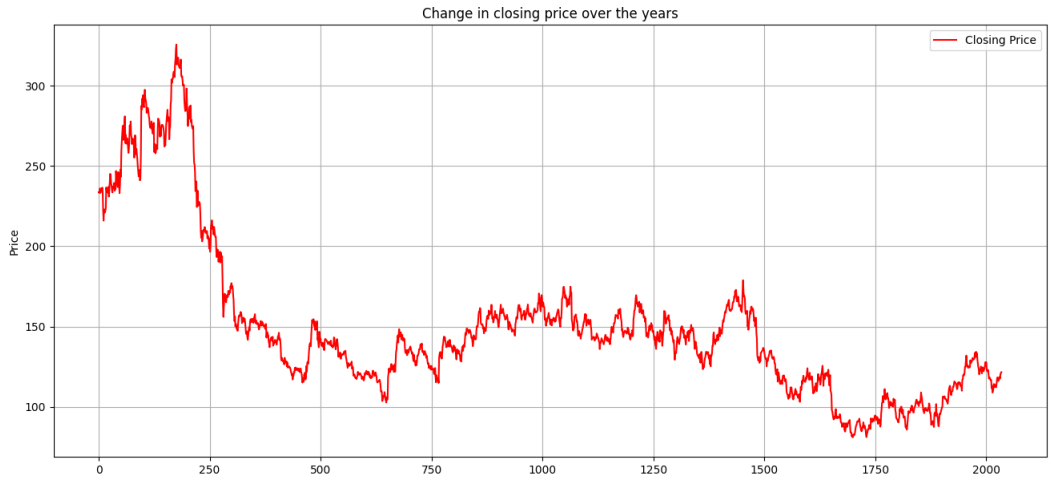
```
Date                   0
Open                   0
High                   0
Low                    0
Last                   0
Close                  0
Total Trade Quantity   0
Turnover (Lacs)        0
dtype: int64
```

```
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
min	81.100000	82.800000	80.000000	81.000000	80.95000	3.961000e+04	37.040000
25%	120.025000	122.100000	118.300000	120.075000	120.05000	1.146444e+06	1427.460000
50%	141.500000	143.400000	139.600000	141.100000	141.25000	1.783456e+06	2512.030000
75%	157.175000	159.400000	155.150000	156.925000	156.90000	2.813594e+06	4539.015000
max	327.700000	328.750000	321.650000	325.950000	325.75000	2.919102e+07	55755.080000

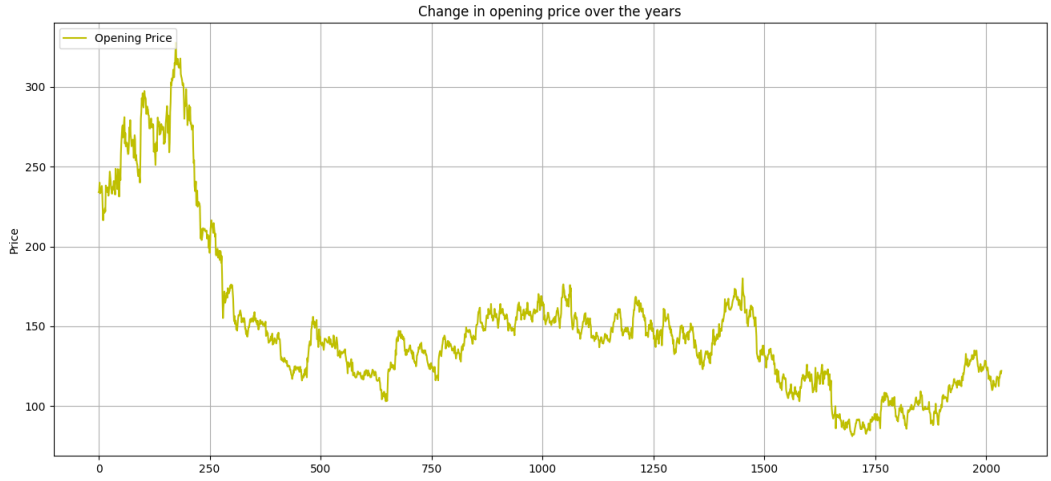
```
plt.figure(figsize=(10,6))
df['Close'].plot(kind='line',figsize=(16,7),color='r',label="Closing Price")

plt.ylabel("Price")
plt.legend(loc="upper right")
plt.title("Change in closing price over the years")
plt.grid()
```



```
plt.figure(figsize=(10,6))
df['Open'].plot(kind='line',figsize=(16,7),color='y',label="Opening Price")

plt.ylabel("Price")
plt.legend(loc="upper left")
plt.title("Change in opening price over the years")
plt.grid()
```

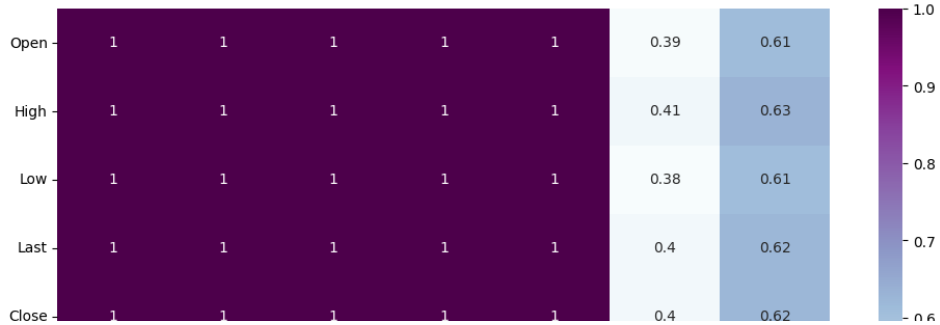



```
df1=df.reset_index()['Close']
df1
```

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

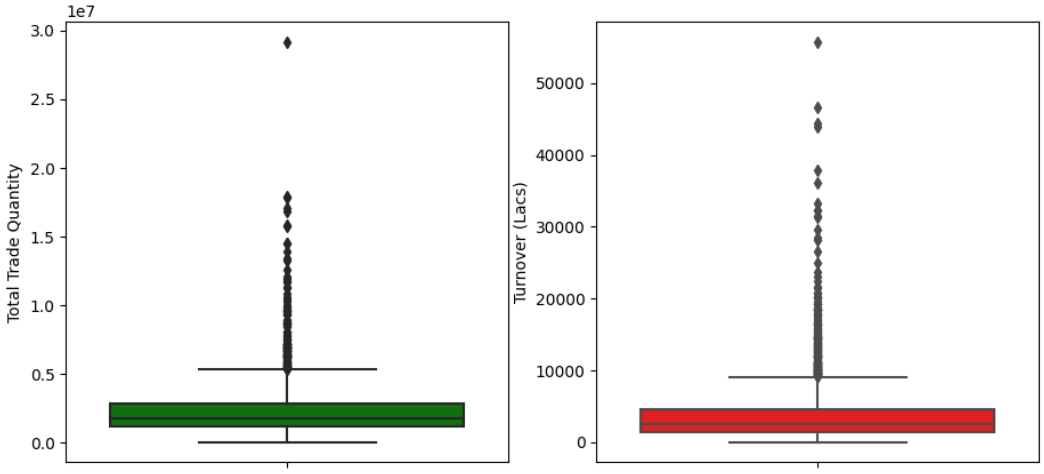
```
plt.figure(figsize=(12,6))
sns.heatmap(df.corr(),annot=True,cmap='BuPu')
```

```
<ipython-input-32-146f16eb7eae>:2: FutureWarning: The default value of numeric_only in DataFrame.corr i
  sns.heatmap(df.corr(),annot=True,cmap='BuPu')
<Axes: >
```

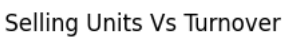



```
plt.figure(figsize=(11,5))
plt.subplot(1,2,1)
sns.boxplot(data=df,y='Total Trade Quantity',color='green')
plt.subplot(1,2,2)
sns.boxplot(data=df,y='Turnover (Lacs)',color='red')
```

```
<Axes: ylabel='Turnover (Lacs)'>
```




```
fig=plt.figure(figsize=(7,6))
plt.scatter(df['Total Trade Quantity'],df['Turnover (Lacs)'], alpha=0.5, edgecolor='r', color='cyan')
plt.xlabel("Trade Quantity (in 100000)")
plt.ylabel("Turnover (in lacs)")
plt.title(" Selling Units Vs Turnover")
plt.show()
```




```
training_set= df[['Open']]
training_set=pd.DataFrame(training_set)
training_set
```

	Open
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

2035 rows × 1 columns


```
scaler=MinMaxScaler(feature_range=(0,1))
training_set_scaler=scaler.fit_transform(np.array(df1).reshape(-1,1))
training_set_scaler
```

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

```
train_size1= int(len(training_set_scaler)*0.65)
test_size1=int(len(training_set_scaler))-train_size1
train_data1,test_data1=training_set_scaler[0:train_size1,:],training_set_scaler[train_size1:len(df),:1]
train_size1
```

```
1322
```

```
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]   ###i=0, 0,1,2,3-----99   100
        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_data1, time_step)
x_test, y_test= create_dataset(test_data1, time_step)
print(x_train.shape,y_train.shape)
```

```
(1221, 100) (1221,)
```

```
x_test.shape
```

```
(612, 100)
```

```
y_test.shape
```

```
(612,)
```

```
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)
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(100,1)))
model.add(LSTM(50, return_sequences=True, input_shape=(100,1)))
model.add(LSTM(50))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam', metrics='acc')
```

```
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
_________________________________________________________________
```

```
model.fit(x_train, y_train, validation_data = (x_test, y_test), epochs = 75, batch_size = 64, verbose = 1)
```

```
train_predict1=model.predict(x_train)
test_predict1=model.predict(x_test)
#Transformback to original form
train_predict1=scaler.inverse_transform(train_predict1)
test_predict1=scaler.inverse_transform(test_predict1)
math.sqrt(mean_squared_error(y_train,train_predict1))
```

```
39/39 [==============================] - 3s 37ms/step
20/20 [==============================] - 1s 36ms/step
169.78752519734851
```

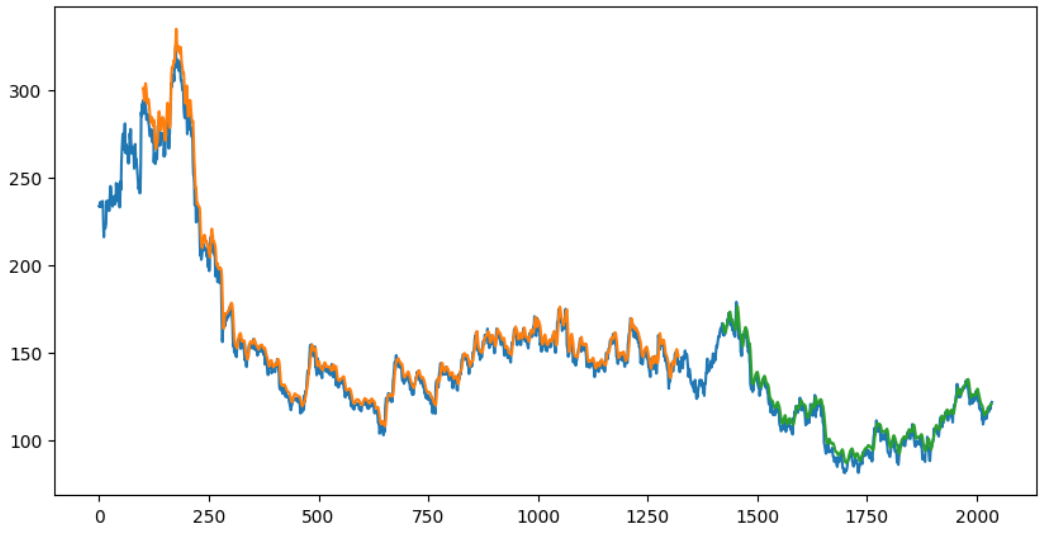
```
math.sqrt(mean_squared_error(y_test,test_predict1))
```

```
117.71104585498023
```

```
### Plotting
# shift train predictions for plotting
look_back=100
trainPredictPlot = np.empty_like(training_set_scaler)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[look_back:len(train_predict1)+look_back, :] = train_predict1

# shift test predictions for plotting
testPredictPlot = np.empty_like(training_set_scaler)
testPredictPlot[:, :] = np.nan
testPredictPlot[len(train_predict1)+(look_back*2)+1:len(df1)-1, :] = test_predict1

# plot baseline and predictions
plt.figure(figsize=(10,5))
plt.plot(scaler.inverse_transform(training_set_scaler))
plt.plot(trainPredictPlot)
plt.plot(testPredictPlot)
plt.show()
```



Colab paid products - Cancel contracts here

✓ 0s completed at 9:47 PM