```
Beginner Level: Task 2 Stock Market Prediction and Forecasting using stacked LSTM

We have to use NSE- TATAGLOBAL dataset and perform data analysis to predict and forecast the outcomes using LSTM.

Long Short-Term Memory (LSTM) is one type of recurrent neural network which is used to learn order dependence in sequence
prediction problems. Due to its capability of storing past information, LSTM is very useful in predicting stock prices.
```

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

In [2]:
```
data=pd.read_csv("https://raw.githubusercontent.com/mwitiderrick/stockprice/master/NSE-TATAGLOBAL.csv")
data
```

Out[2]:

	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
...	...	...	...	...	...	...	...	...
2030	2010-07-27	117.60	119.50	112.00	118.80	118.65	586100	694.98
2031	2010-07-26	120.10	121.00	117.10	117.10	117.60	658440	780.01
2032	2010-07-23	121.80	121.95	120.25	120.35	120.65	281312	340.31
2033	2010-07-22	120.30	122.00	120.25	120.75	120.90	293312	355.17
2034	2010-07-21	122.10	123.00	121.05	121.10	121.55	658666	803.56

2035 rows × 8 columns

In [55]:
```
#Converting Date to datatime format and sorting by date
data.Date=pd.to_datetime(data.Date,dayfirst=True)
data=data.sort_values('Date',ascending=True)
```

In [56]:
```
data.head()
```

Out[56]:

	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 [57]:
```
data.tail()
```

Out[57]:

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

In [58]:
```
data.dtypes
```

Out[58]:
```
Date                    datetime64[ns]
Open                           float64
High                           float64
Low                            float64
Last                           float64
Close                          float64
Total Trade Quantity             int64
Turnover (Lacs)                float64
dtype: object
```

In [59]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2035 entries, 2034 to 0
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Date                  2035 non-null   datetime64[ns]
 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: datetime64[ns](1), float64(6), int64(1)
memory usage: 143.1 KB
```

In [60]:
```
data.isnull().sum()
```

Out[60]:
```
Date                    0
Open                    0
High                    0
Low                     0
Last                    0
Close                   0
Total Trade Quantity    0
Turnover (Lacs)         0
dtype: int64
```

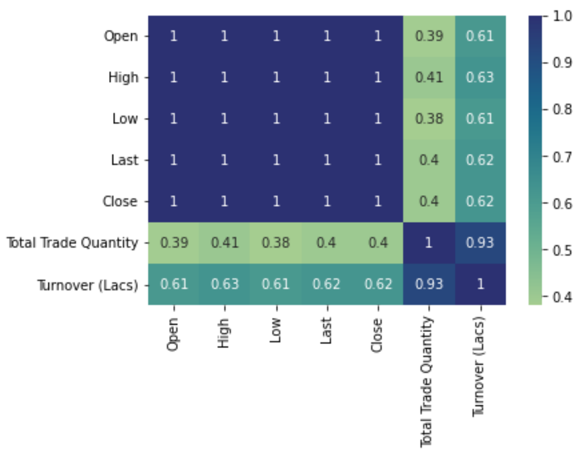
In [61]:
```
data.corr()
```

Out[61]:

	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
Open	1.000000	0.999015	0.998825	0.997781	0.997840	0.385951	0.612877
High	0.999015	1.000000	0.998773	0.999155	0.999194	0.406405	0.630589
Low	0.998825	0.998773	1.000000	0.999051	0.999119	0.380621	0.608502
Last	0.997781	0.999155	0.999051	1.000000	0.999961	0.399328	0.624584
Close	0.997840	0.999194	0.999119	0.999961	1.000000	0.398911	0.624213
Total Trade Quantity	0.385951	0.406405	0.380621	0.399328	0.398911	1.000000	0.926931
Turnover (Lacs)	0.612877	0.630589	0.608502	0.624584	0.624213	0.926931	1.000000

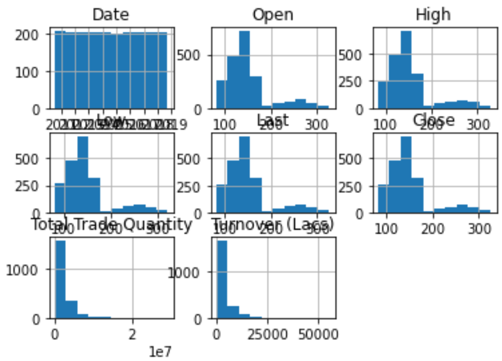
In [62]:
```
sns.heatmap(data.corr(),annot= True, cmap='crest')
```

Out[62]: <AxesSubplot:>



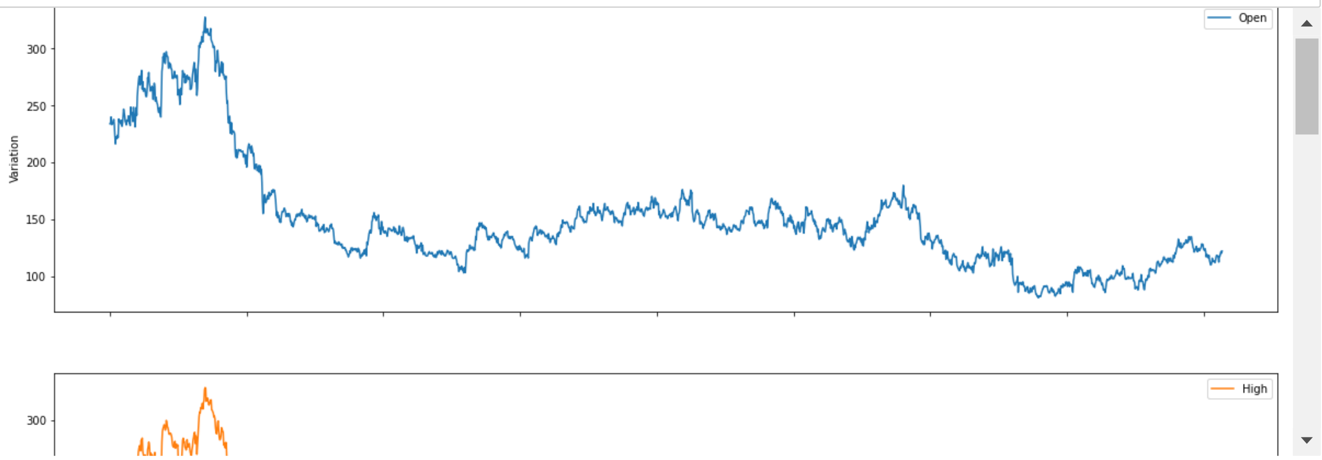
In [63]:
```python
data.hist()
```

Out[63]:
```
array([[<AxesSubplot:title={'center':'Date'}>,
        <AxesSubplot:title={'center':'Open'}>,
        <AxesSubplot:title={'center':'High'}>],
       [<AxesSubplot:title={'center':'Low'}>,
        <AxesSubplot:title={'center':'Last'}>,
        <AxesSubplot:title={'center':'Close'}>],
       [<AxesSubplot:title={'center':'Total Trade Quantity'}>,
        <AxesSubplot:title={'center':'Turnover (Lacs)'}>, <AxesSubplot:>]],
      dtype=object)
```



In [64]:
```python
cols_plot = ['Open','High','Low','Last','Close']
axes = data[cols_plot].plot(alpha = 1, figsize=(20, 30), subplots = True)

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



In [65]:
```
sns.pairplot(data,hue="Turnover (Lacs)")
```

Out[65]:
```
<seaborn.axisgrid.PairGrid at 0x218d6973b50>
```

In [66]:
```
df=data[['Close']]
```

In [67]:
```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range=(0,1))
df= sc.fit_transform(np.array(df).reshape(-1,1))
```

In [68]:
```
df
```

Out[68]:
```
array([[0.16584967],
       [0.16319444],
       [0.1621732 ],
       ...,
       [0.62622549],
       [0.62214052],
       [0.62418301]])
```

In [69]:
```
#Training and testing
training_size=int(len(df)*0.65)
test_size=len(df)-training_size
train_data,test_data=df[0:training_size,:],df[training_size:len(df):1]
```

In [70]:
```
training_size,test_size
```

Out[70]:
```
(1322, 713)
```

Converting an array of values into dataset matrix

In [71]:
```
import numpy
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] ### 0 0,1,2,3,...,99   100

        dataX.append(a)
        dataY.append(dataset[i+time_step,0])

    return numpy.array(dataX),numpy.array(dataY)
```

In [72]:
```
#reshape into X=t, t+1,t+2,t+3 and Y=t+4

time_step=100

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

In [73]:
```
print(X_train)
```

```
[[0.16584967 0.16319444 0.1621732  ... 0.14011438 0.13848039 0.12479575]
 [0.16319444 0.1621732  0.14971405 ... 0.13848039 0.12479575 0.12254902]
 [0.1621732  0.14971405 0.15400327 ... 0.12479575 0.12254902 0.13010621]
 ...
 [0.19669118 0.19505719 0.20996732 ... 0.20751634 0.20751634 0.19219771]
 [0.19505719 0.20996732 0.21098856 ... 0.20751634 0.19219771 0.18341503]
 [0.20996732 0.21098856 0.21568627 ... 0.19219771 0.18341503 0.19546569]]
```

In [74]:
```
print(X_test.shape), print(y_test.shape)
```

```
(612, 100)
(612,)
```

Out[74]:
```
(None, None)
```

In [75]:
```
#reshape input(samples,timestep,features) required for LSTM model
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 LSTM Model

In [76]:
```
!pip install keras

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

```
Requirement already satisfied: keras in c:\users\hp\anaconda3\lib\site-packages (2.11.0)
```

In [77]:
```
model = Sequential()

model.add(LSTM(units=50,return_sequences=True,input_shape=(100, 1)))
model.add(Dropout(0.2))

model.add(LSTM(units=50,return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(units=50,return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(units=50))
model.add(Dropout(0.2))

model.add(Dense(units=1))

model.compile(optimizer='adam',loss='mean_squared_error')

model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=50,batch_size=32,verbose=1)
```

```
Epoch 1/50
39/39 [==============================] - 33s 371ms/step - loss: 0.0074 - val_loss: 0.0066
Epoch 2/50
39/39 [==============================] - 11s 293ms/step - loss: 0.0019 - val_loss: 0.0072
Epoch 3/50
39/39 [==============================] - 11s 289ms/step - loss: 0.0015 - val_loss: 0.0083
Epoch 4/50
39/39 [==============================] - 11s 292ms/step - loss: 0.0016 - val_loss: 0.0063
Epoch 5/50
39/39 [==============================] - 12s 300ms/step - loss: 0.0014 - val_loss: 0.0051
Epoch 6/50
39/39 [==============================] - 11s 294ms/step - loss: 0.0013 - val_loss: 0.0095
Epoch 7/50
39/39 [==============================] - 11s 293ms/step - loss: 0.0012 - val_loss: 0.0051
Epoch 8/50
39/39 [==============================] - 11s 293ms/step - loss: 0.0013 - val_loss: 0.0087
Epoch 9/50
39/39 [==============================] - 11s 292ms/step - loss: 0.0011 - val_loss: 0.0094
Epoch 10/50
39/39 [==============================] - 11s 291ms/step - loss: 0.0011 - val_loss: 0.0148
Epoch 11/50
39/39 [==============================] - 11s 294ms/step - loss: 0.0011 - val_loss: 0.0100
Epoch 12/50
39/39 [==============================] - 11s 289ms/step - loss: 0.0011 - val_loss: 0.0125
Epoch 13/50
39/39 [==============================] - 11s 293ms/step - loss: 9.6631e-04 - val_loss: 0.0115
Epoch 14/50
39/39 [==============================] - 11s 294ms/step - loss: 9.8161e-04 - val_loss: 0.0097
Epoch 15/50
39/39 [==============================] - 11s 293ms/step - loss: 9.3110e-04 - val_loss: 0.0092
Epoch 16/50
39/39 [==============================] - 11s 291ms/step - loss: 8.9092e-04 - val_loss: 0.0089
Epoch 17/50
39/39 [==============================] - 11s 292ms/step - loss: 8.8136e-04 - val_loss: 0.0050
Epoch 18/50
39/39 [==============================] - 11s 291ms/step - loss: 8.4461e-04 - val_loss: 0.0063
Epoch 19/50
39/39 [==============================] - 11s 292ms/step - loss: 8.9430e-04 - val_loss: 0.0063
Epoch 20/50
39/39 [==============================] - 12s 296ms/step - loss: 8.2578e-04 - val_loss: 0.0050
Epoch 21/50
39/39 [==============================] - 11s 292ms/step - loss: 8.9410e-04 - val_loss: 0.0068
Epoch 22/50
39/39 [==============================] - 11s 292ms/step - loss: 8.5433e-04 - val_loss: 0.0072
Epoch 23/50
39/39 [==============================] - 11s 292ms/step - loss: 7.6908e-04 - val_loss: 0.0069
Epoch 24/50
39/39 [==============================] - 11s 292ms/step - loss: 9.3290e-04 - val_loss: 0.0052
Epoch 25/50
39/39 [==============================] - 11s 293ms/step - loss: 7.7685e-04 - val_loss: 0.0093
Epoch 26/50
39/39 [==============================] - 11s 292ms/step - loss: 7.0732e-04 - val_loss: 0.0097
Epoch 27/50
39/39 [==============================] - 11s 294ms/step - loss: 7.3500e-04 - val_loss: 0.0078
Epoch 28/50
39/39 [==============================] - 11s 292ms/step - loss: 6.9466e-04 - val_loss: 0.0028
Epoch 29/50
39/39 [==============================] - 11s 291ms/step - loss: 6.6322e-04 - val_loss: 0.0045
Epoch 30/50
39/39 [==============================] - 11s 292ms/step - loss: 6.4973e-04 - val_loss: 0.0065
Epoch 31/50
39/39 [==============================] - 11s 294ms/step - loss: 6.5927e-04 - val_loss: 0.0039
Epoch 32/50
39/39 [==============================] - 12s 313ms/step - loss: 6.5846e-04 - val_loss: 0.0063
Epoch 33/50
39/39 [==============================] - 11s 291ms/step - loss: 6.5454e-04 - val_loss: 0.0028
Epoch 34/50
39/39 [==============================] - 11s 293ms/step - loss: 6.7720e-04 - val_loss: 0.0041
Epoch 35/50
39/39 [==============================] - 11s 291ms/step - loss: 5.6233e-04 - val_loss: 0.0038
Epoch 36/50
39/39 [==============================] - 11s 291ms/step - loss: 5.8999e-04 - val_loss: 0.0051
Epoch 37/50
39/39 [==============================] - 12s 297ms/step - loss: 5.3070e-04 - val_loss: 0.0027
Epoch 38/50
39/39 [==============================] - 14s 371ms/step - loss: 5.4505e-04 - val_loss: 0.0052
Epoch 39/50
39/39 [==============================] - 12s 312ms/step - loss: 5.3821e-04 - val_loss: 0.0026
Epoch 40/50
39/39 [==============================] - 12s 311ms/step - loss: 5.2316e-04 - val_loss: 0.0054
Epoch 41/50
39/39 [==============================] - 12s 299ms/step - loss: 5.7232e-04 - val_loss: 0.0062
Epoch 42/50
39/39 [==============================] - 12s 311ms/step - loss: 5.2391e-04 - val_loss: 0.0047
Epoch 43/50
39/39 [==============================] - 12s 309ms/step - loss: 5.1087e-04 - val_loss: 0.0031
```

```
Epoch 44/50
39/39 [==============================] - 12s 299ms/step - loss: 4.9855e-04 - val_loss: 0.0034
Epoch 45/50
39/39 [==============================] - 12s 309ms/step - loss: 5.5969e-04 - val_loss: 0.0053
Epoch 46/50
39/39 [==============================] - 11s 290ms/step - loss: 4.7888e-04 - val_loss: 0.0056
Epoch 47/50
39/39 [==============================] - 11s 290ms/step - loss: 4.6109e-04 - val_loss: 0.0034
Epoch 48/50
39/39 [==============================] - 12s 298ms/step - loss: 4.7615e-04 - val_loss: 0.0052
Epoch 49/50
39/39 [==============================] - 12s 298ms/step - loss: 4.3014e-04 - val_loss: 0.0034
Epoch 50/50
39/39 [==============================] - 11s 289ms/step - loss: 3.8748e-04 - val_loss: 0.0049
```

Out[77]:
```
<keras.callbacks.History at 0x218d996b8e0>
```

In [98]:
```python
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_4 (LSTM)               (None, 100, 50)           10400

 dropout_4 (Dropout)         (None, 100, 50)           0

 lstm_5 (LSTM)               (None, 100, 50)           20200

 dropout_5 (Dropout)         (None, 100, 50)           0

 lstm_6 (LSTM)               (None, 100, 50)           20200

 dropout_6 (Dropout)         (None, 100, 50)           0

 lstm_7 (LSTM)               (None, 50)                20200

 dropout_7 (Dropout)         (None, 50)                0

 dense_1 (Dense)             (None, 1)                 51

=================================================================
Total params: 71,051
Trainable params: 71,051
Non-trainable params: 0
_________________________________________________________________
```

In [99]:
```python
# Lets Do the prediction and check performance metrics
train_predict=model.predict(X_train)
test_predict=model.predict(X_test)
```

```
39/39 [==============================] - 4s 91ms/step
20/20 [==============================] - 2s 91ms/step
```

In [100]:
```python
#Transformback to original form
train_predict=sc.inverse_transform(train_predict)
test_predict=sc.inverse_transform(test_predict)
```

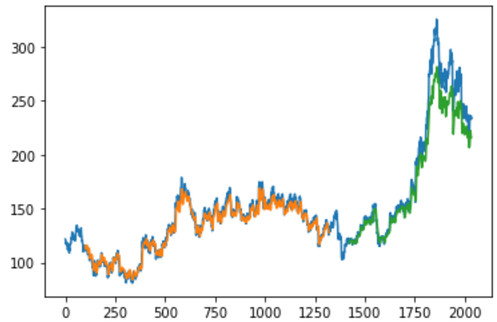
In [101]:
```python
# Calculate RMSE performance metrics
import math
from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(y_train,train_predict))
```

Out[101]:
```
132.62609235298586
```

In [102]:
```python
# Test Data RMSE
math.sqrt(mean_squared_error(y_test,test_predict))
```

Out[102]:
```
187.31137215338686
```

In [103]:
```
# Plotting
# shift train predictions for plotting
look_back=100
trainPredictPlot = numpy.empty_like(df)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[look_back:len(train_predict)+look_back, :] = train_predict
# shift test predictions for plotting
testPredictPlot = numpy.empty_like(df)
testPredictPlot[:, :] = numpy.nan
testPredictPlot[len(train_predict)+(look_back*2)+1:len(df)-1, :] = test_predict
# plot baseline and predictions
plt.plot(sc.inverse_transform(df))
plt.plot(trainPredictPlot)
plt.plot(testPredictPlot)
plt.show()
```



In [104]:
```
len(test_data)
```

Out[104]: 713

In [105]:
```
x_input=test_data[613:].reshape(1,-1)
x_input.shape
```

Out[105]: (1, 100)

In [106]:
```
temp_input=list(x_input)
temp_input=temp_input[0].tolist()
```

In [107]:
```
# demonstrate prediction for next days
from numpy import array

lst_output=[]
n_steps=100
i=0
while(i<30):

    if(len(temp_input)>100):
        #print(temp_input)
        x_input=np.array(temp_input[1:])
        print("{} day input {}".format(i,x_input))
        x_input=x_input.reshape(1,-1)
        x_input = x_input.reshape((1, n_steps, 1))
        #print(x_input)
        yhat = model.predict(x_input, verbose=0)
        print("{} day output {}".format(i,yhat))
        temp_input.extend(yhat[0].tolist())
        temp_input=temp_input[1:]
        #print(temp_input)
        lst_output.extend(yhat.tolist())
        i=i+1
    else:
        x_input = x_input.reshape((1, n_steps,1))
        yhat = model.predict(x_input, verbose=0)
        print(yhat[0])
        temp_input.extend(yhat[0].tolist())
        print(len(temp_input))
        lst_output.extend(yhat.tolist())
        i=i+1

        print(lst_output)
```

```
[0.5535315]
101
[[0.5535315275192261]]
1 day input [0.86213235 0.83455882 0.84375    0.71466503 0.66053922 0.65420752
 0.68035131 0.6689134  0.66421569 0.67892157 0.69138072 0.70894608
 0.72263072 0.73406863 0.72120098 0.74979575 0.7689951  0.74468954
 0.7120098  0.74673203 0.75490196 0.76041667 0.75980392 0.74652778
 0.76245915 0.77062908 0.80412582 0.78104575 0.79411765 0.76021242
 0.73488562 0.72406046 0.73999183 0.74979575 0.76143791 0.75428922
 0.74754902 0.76960784 0.7495915  0.81719771 0.80208333 0.77736928
 0.75714869 0.79370915 0.79166667 0.77144608 0.75265523 0.72691993
 0.6629902  0.68198529 0.64419935 0.62152778 0.6439951  0.66319444
 0.67544935 0.63705065 0.64603758 0.65890523 0.65870098 0.67790033
 0.63194444 0.63112745 0.62785948 0.64767157 0.63991013 0.63480392
 0.63868464 0.62336601 0.62745098 0.63970588 0.64705882 0.66196895
 0.67075163 0.65196078 0.63112745 0.61294935 0.6376634  0.62254902
 0.62745098 0.63337418 0.62642974 0.63623366 0.58312908 0.57781863
 0.57230392 0.58006536 0.57618464 0.55167484 0.57883987 0.625
 0.63582516 0.62949346 0.62888072 0.62765523 0.62234477 0.63378268
```

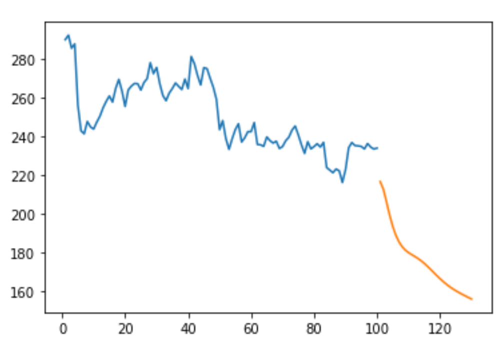
In [108]:
```
day_new=np.arange(1,101)
day_pred=np.arange(101,131)
```

In [109]:
```
len(df)
```

Out[109]: 2035

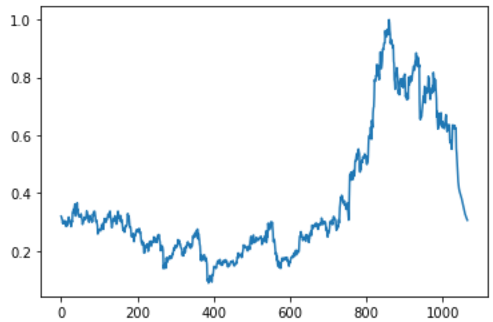
In [110]:
```
plt.plot(day_new,sc.inverse_transform(df[1935:]))
plt.plot(day_pred,sc.inverse_transform(lst_output))
```

Out[110]: [<matplotlib.lines.Line2D at 0x218e781ea30>]



In [111]:
```
df2=df.tolist()
df2.extend(lst_output)
plt.plot(df2[1000:])
```

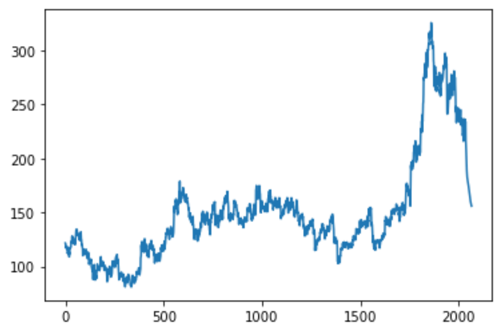
Out[111]: [<matplotlib.lines.Line2D at 0x218e78487f0>]



In [112]:
```
df2=sc.inverse_transform(df2).tolist()
```

In [113]:
```
plt.plot(df2)
```

Out[113]: [<matplotlib.lines.Line2D at 0x218e790de80>]



In [ ]:

In [ ]: