Stock Market Prediction And Forecasting Using Stacked LSTM

Performed by Piyush Borhade

Problem Statement:

We need to predict Stock market by using LSTMs!

Steps to solve this problem

1. Data Collection
2. Preprocess the data (train and test)
3. Create an LSTM model
4. Predict test data and plot the output
5. Predict the future 30 days and plot the output

```
import pandas_datareader as pdr
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import datetime as dt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, LSTM
```

```
df = pd.read_csv('/content/AAPL.csv')
df.head()
```

	Unnamed: 0	symbol	date	close	high	low	open	volume	adjClose
0	0	AAPL	2015-05-27 00:00:00+00:00	132.045	132.260	130.05	130.34	45833246	121.682558
1	1	AAPL	2015-05-28 00:00:00+00:00	131.780	131.950	131.10	131.86	30733309	121.438354
2	2	AAPL	2015-05-29 00:00:00+00:00	130.280	131.450	129.90	131.23	50884452	120.056069
3	3	AAPL	2015-06-01 00:00:00+00:00	130.535	131.390	130.05	131.20	32112797	120.291057
4	4	AAPL	2015-06-02 00:00:00+00:00	129.960	130.655	129.32	129.86	33667627	119.761181

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.tail()
```

	Unnamed: 0	symbol	date	close	high	low	open	volume	adjClos
1253	1253	AAPL	2020-05-18 00:00:00+00:00	314.96	316.50	310.3241	313.17	33843125	314.9
1254	1254	AAPL	2020-05-19 00:00:00+00:00	313.14	318.52	313.0100	315.03	25432385	313.1
1255	1255	AAPL	2020-05-20 00:00:00+00:00	319.23	319.52	316.2000	316.68	27876215	319.2
1256	1256	AAPL	2020-05-21 00:00:00+00:00	316.85	320.89	315.8700	318.66	25672211	316.8
1257	1257	AAPL	2020-05-22 00:00:00+00:00	318.89	319.23	315.3500	315.77	20450754	318.8

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

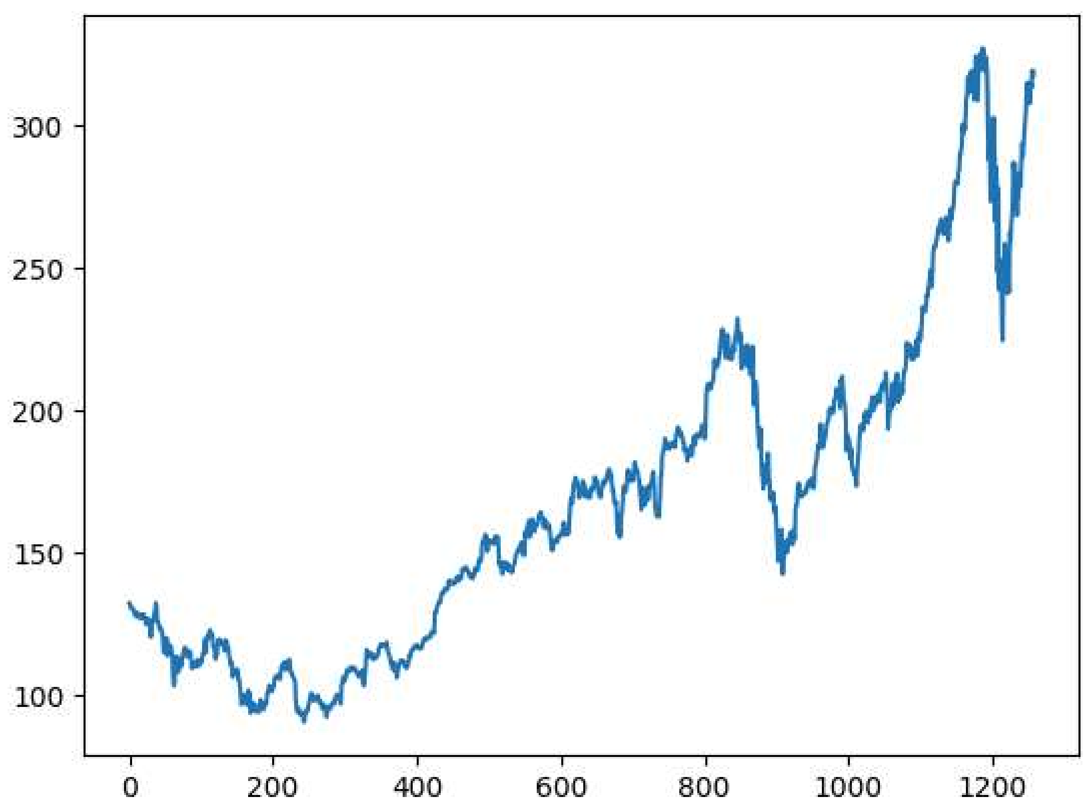
	close
0	132.045
1	131.780
2	130.280
3	130.535
4	129.960
...	...
1253	314.960
1254	313.140
1255	319.230
1256	316.850
1257	318.890

1258 rows × 1 columns

dtype: float64

```
df1.plot()
```

```
<Axes: >
```



```
df1.shape
```

```
(1258,)
```

```
sc = MinMaxScaler(feature_range = (0,1))
```

```
df1 = sc.fit_transform(np.array(df1).reshape(-1,1))
```

```
df1
```

```
array([[0.17607447],
       [0.17495567],
       [0.16862282],
       ...,
       [0.96635143],
       [0.9563033 ],
       [0.96491598]])
```

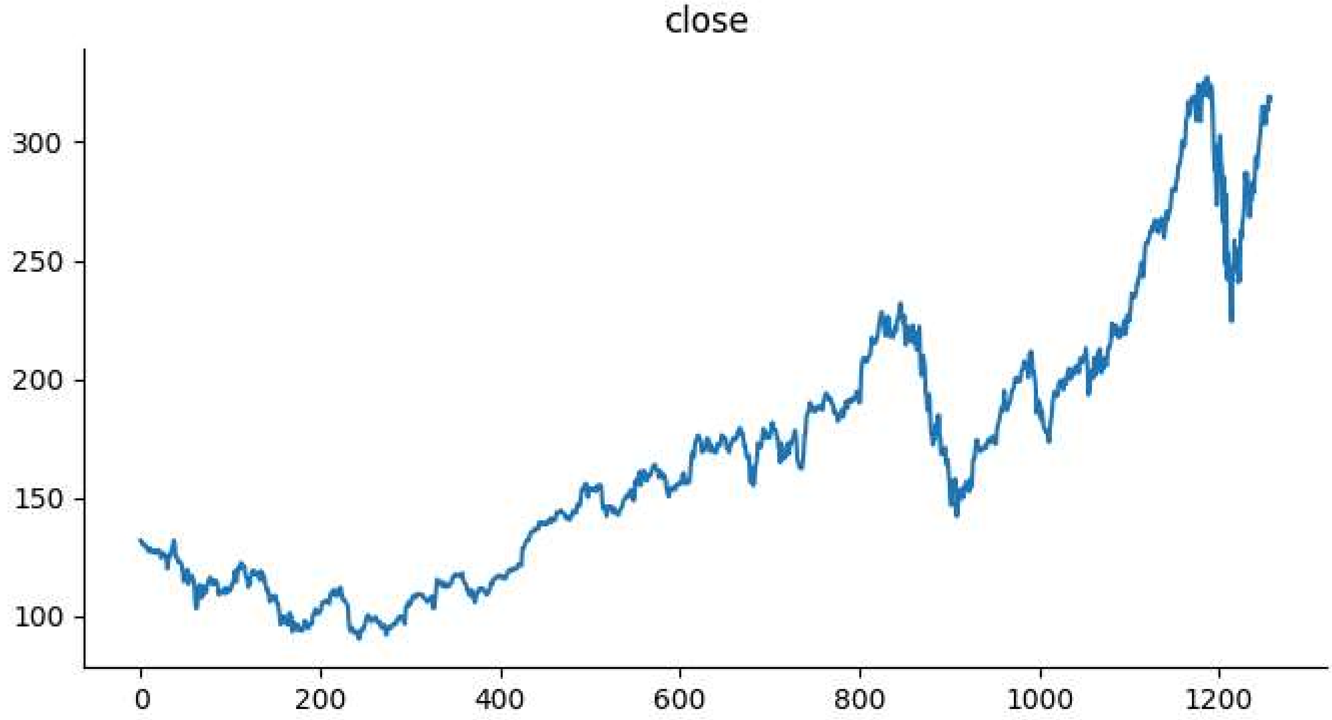
```
df1.shape
```

```
(1258, 1)
```

```
df1
```

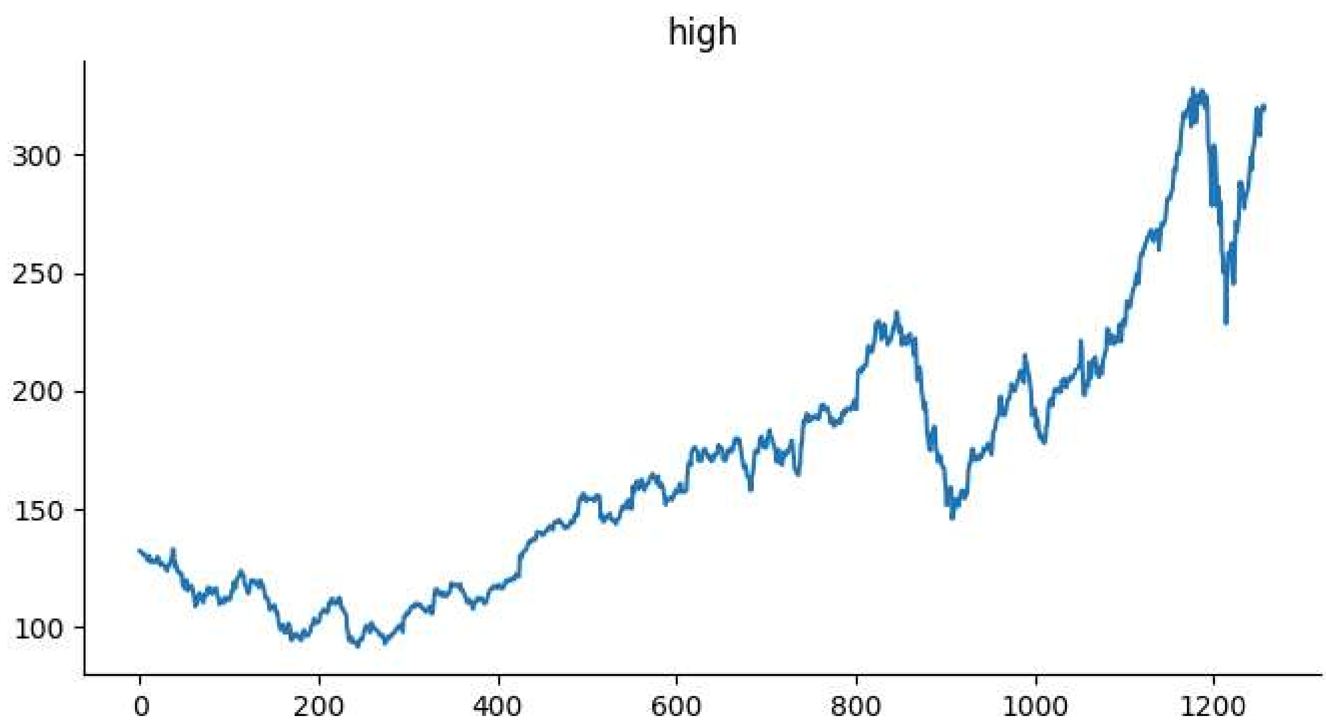
```
array([[0.17607447],
       [0.17495567],
       [0.16862282],
       ...,
       [0.96635143],
       [0.9563033 ],
       [0.96491598]])
```

```
df['close'].plot(kind='line', figsize=(8, 4), title='close')
plt.gca().spines[['top', 'right']].set_visible(False)
```



Closing Price The closing price is the last price at which the stock is traded during the regular trading day. A stock's closing price is the standard benchmark used by investors to track its performance over time.

```
from matplotlib import pyplot as plt
df['high'].plot(kind='line', figsize=(8, 4), title='high')
plt.gca().spines[['top', 'right']].set_visible(False)
```



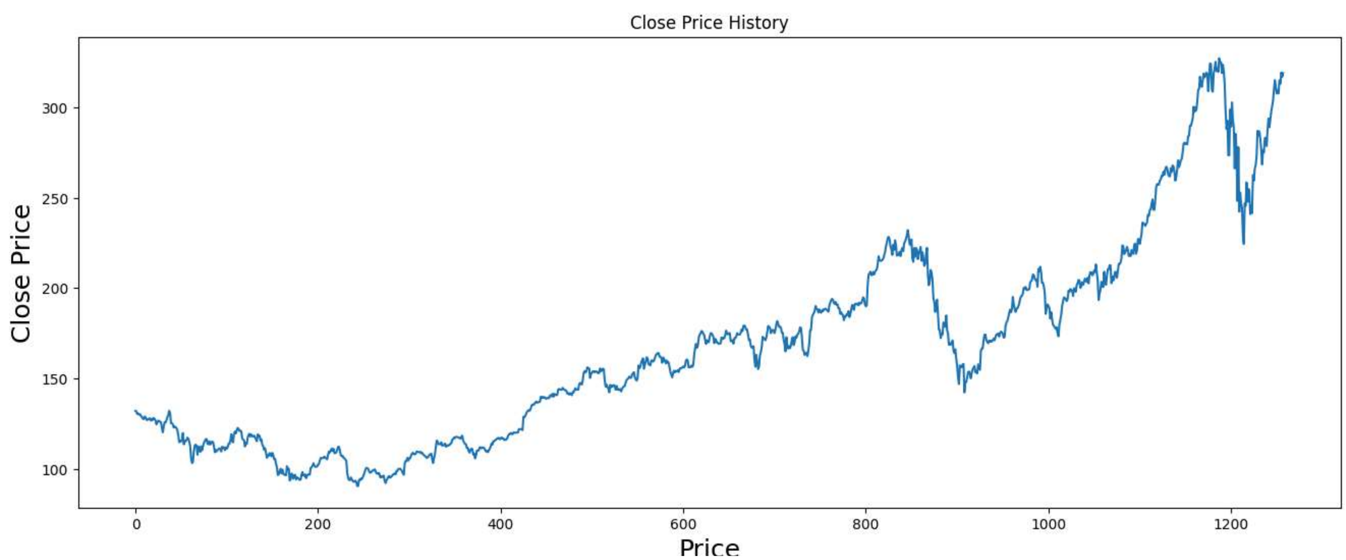
```
df1.describe()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-43-784441b173b6> in <cell line: 1>()
----> 1 df1.describe()

AttributeError: 'numpy.ndarray' object has no attribute 'describe'
```

Next steps: Explain error

```
plt.figure(figsize=(16,6))
plt.title('Close Price History')
plt.plot(df['close'])
plt.xlabel('Price', fontsize=18)
plt.ylabel('Close Price', fontsize=18)
plt.show()
```




```
training_size=int(len(df1)*0.65)
training_size
```

```
817
```

```
test_size=len(df1)-training_size
test_size
```

```
441
```

```
train_data,test_data=df1[0:training_size,:],df1[training_size:len(df1),:1]
```

```
training_size,test_size
```

```
(817, 441)
```

```
train_data
```

```
           [0.50042219],
           [0.50413747],
           [0.5062062 ],
           [0.51920966],
           [0.53719497],
           [0.52824453],
           [0.52647133]])
```

```
import numpy
# convert an array of values into a dataset matrix
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 numpy.array(dataX), numpy.array(dataY)
```

```
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)
```

```
(716, 100)
(716,)
(None, None)
```

```
print(X_test.shape), print(ytest.shape)
```

```
(340, 100)
(340,)
(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_1"

Layer (type)	Output Shape	Param #
lstm_3 (LSTM)	(None, 100, 50)	10,400
lstm_4 (LSTM)	(None, 100, 50)	20,200
lstm_5 (LSTM)	(None, 50)	20,200
dense_1 (Dense)	(None, 1)	51

Total params: 50,851 (198.64 KB)
Trainable params: 50,851 (198.64 KB)
Non-trainable params: 0 (0.00 B)

```
model.fit(X_train,y_train,validation_data=(X_test,ytest),epochs=100,batch_size=64,verbose=1)
```

```
Epoch 88/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 3s 210ms/step - loss: 2.0650e-04 - val_loss: 0.0014
Epoch 89/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 3s 230ms/step - loss: 2.1511e-04 - val_loss: 9.9373e-04
Epoch 90/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 2s 162ms/step - loss: 1.9305e-04 - val_loss: 9.3673e-04
Epoch 91/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 2s 166ms/step - loss: 1.8922e-04 - val_loss: 0.0012
Epoch 92/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 3s 165ms/step - loss: 2.0922e-04 - val_loss: 9.3541e-04
Epoch 93/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 3s 163ms/step - loss: 1.6130e-04 - val_loss: 9.2957e-04
Epoch 94/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 4s 261ms/step - loss: 1.6379e-04 - val_loss: 9.1518e-04
Epoch 95/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 2s 163ms/step - loss: 1.5981e-04 - val_loss: 8.9743e-04
Epoch 96/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 2s 164ms/step - loss: 1.4234e-04 - val_loss: 0.0012
Epoch 97/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 3s 166ms/step - loss: 1.9458e-04 - val_loss: 8.7867e-04
Epoch 98/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 2s 164ms/step - loss: 1.5569e-04 - val_loss: 8.5700e-04
Epoch 99/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 3s 246ms/step - loss: 1.4725e-04 - val_loss: 9.4421e-04
Epoch 100/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 4s 164ms/step - loss: 1.5349e-04 - val_loss: 0.0010
<keras.src.callbacks.history.History at 0x79d413ad1540>
```

```
import tensorflow as tf
```

```
tf.__version__
```

```
'2.17.0'
```

```
### Lets Do the prediction and check performance metrics
train_predict=model.predict(X_train)
test_predict=model.predict(X_test)
```

```
23/23 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
```

```
##Transformback to original form
train_predict=sc.inverse_transform(train_predict)
test_predict=sc.inverse_transform(test_predict)
```

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

```
143.50636833170182
```

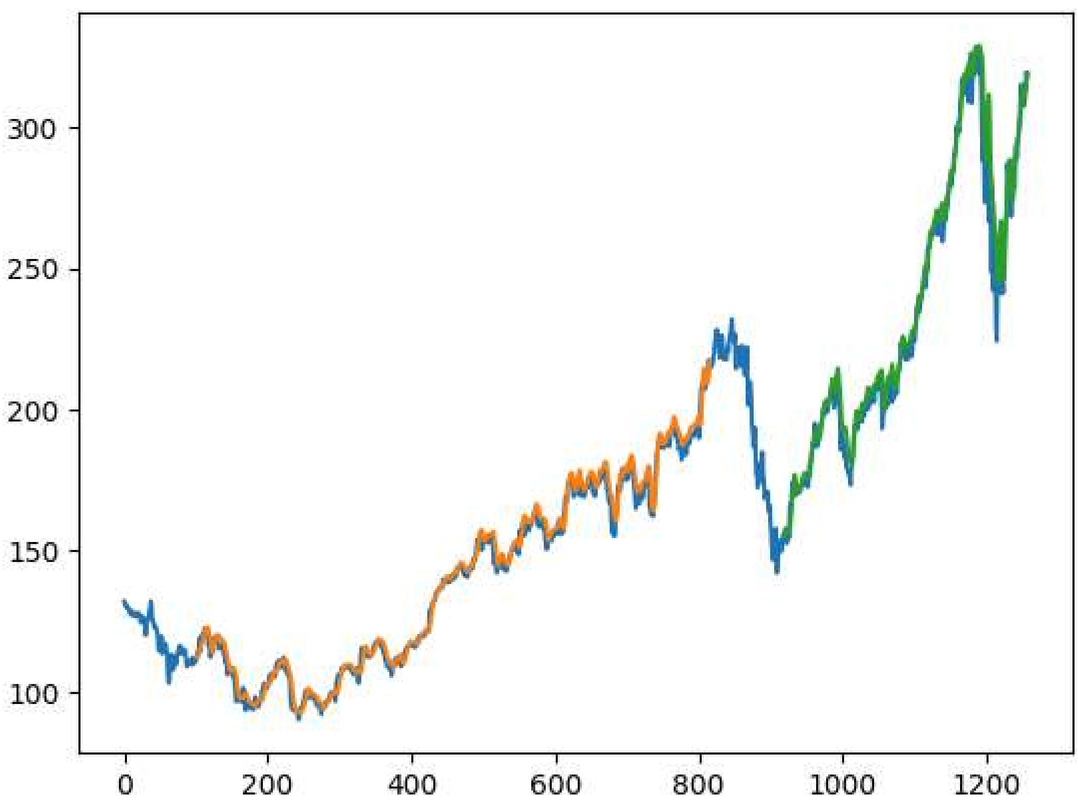
```
math.sqrt(mean_squared_error(ytest,test_predict))
```

```
241.5633092082751
```

```
### Plotting
# shift train predictions for plotting
look_back=100
trainPredictPlot = numpy.empty_like(df1)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[look_back:len(train_predict)+look_back, :] = train_predict

# shift test predictions for plotting
testPredictPlot = numpy.empty_like(df1)
testPredictPlot[:, :] = numpy.nan
testPredictPlot[len(train_predict)+(look_back*2)+1:len(df1)-1, :] = test_predict

# plot baseline and predictions
plt.plot(sc.inverse_transform(df1))
plt.plot(trainPredictPlot)
plt.plot(testPredictPlot)
plt.show()
```



```
len(test_data)
```

```
441
```

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

```
(1, 100)
```

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

```
temp_input
```

```
0.814109600079542,
0.7947310647639958,
0.8333614793548934,
0.8589884319851391,
0.8390188296884238,
0.8562864139153934,
0.8748627881448958,
0.887824031073208,
0.9009541501308793,
0.927932111795997 8,
0.9485349995778098,
0.9333361479354896,
0.9174617917757326,
0.925441188887951,
0.9177151059697712,
0.9483239044161109,
0.9406400405302711,
0.9663514312251966,
0.9563033015283293,
0.964915984125644]
```

```
# demonstrate prediction for next 10 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
```

```
 0.6665963  0.7921557  0.64118044 0.68614371 0.66001013 0.65203074
```

```
print(lst_output)
```

```
[[0.9803187847137451], [0.985552966594696], [0.991817057132721], [0.9982050061225891], [
```

```
day_new = np.arange(1,101)
day_pred = np.arange(101,131)
```

```
len(df1)
```

```
1258
```

```
plt.plot(day_new,sc.inverse_transform(df1[1158:]))
plt.plot(day_pred,sc.inverse_transform(lst_output))
```

```
[<matplotlib.lines.Line2D at 0x79d410578550>]
```