Coding Sathi Data Science Internship july 2023

Task 2: Stock Market Prediction and Forecasting using stacked LSTM

Author-Vivek Kumar

In [11]:
```
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import pandas as pd
import numpy as np
import math
warnings.filterwarnings('ignore')
```

In [12]:
```
d = pd.read_csv('https://raw.githubusercontent.com/mwitiderrick/stockprice/master/NSE-TATAGLOBAL.csv')
d.head()
```

Out[12]:

	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

In [13]:
```
d.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
```

In [14]:
```
d.describe()
```

Out[14]:

	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

In [15]:
```
d['Date'] = pd.to_datetime(d['Date'])
d.dtypes
```

Out[15]:
```
Date                    datetime64[ns]
Open                           float64
High                           float64
Low                            float64
Last                           float64
Close                          float64
Total Trade Quantity             int64
Turnover (Lacs)                float64
dtype: object
```

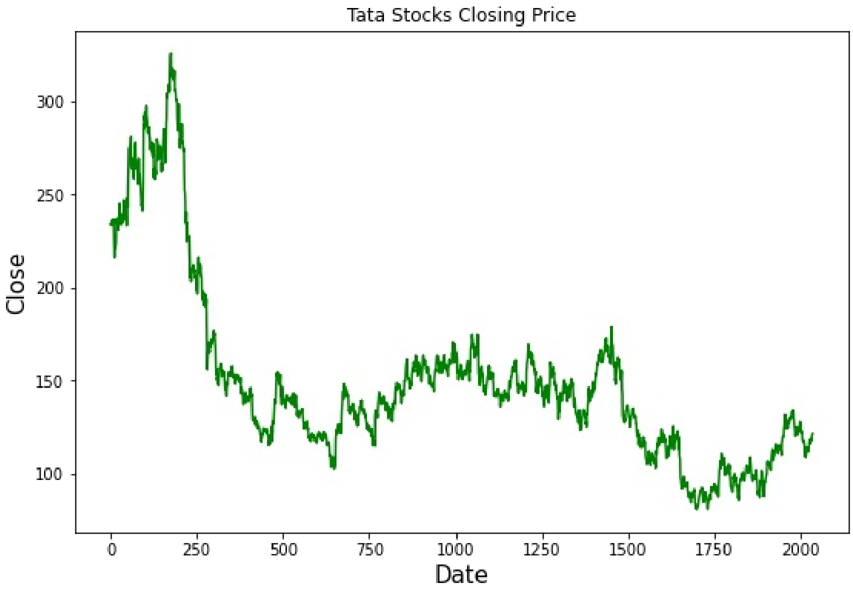
In [16]:
```
d = d.sort_values('Date')
d.head()
```

Out[16]:

	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

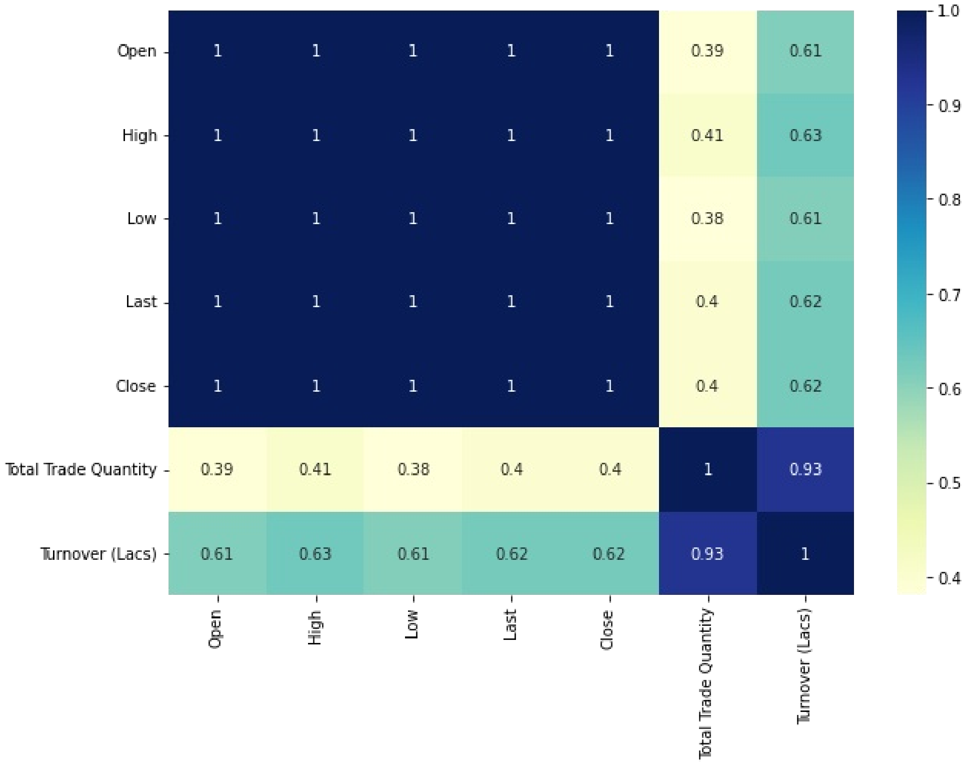
In [17]:
```
plt.figure(figsize = (9,6))
plt.title('Tata Stocks Closing Price')
plt.plot(d['Close'],'g')
plt.xlabel('Date',fontsize=15)
plt.ylabel('Close',fontsize=15)
```

Out[17]:
```
Text(0, 0.5, 'Close')
```



In [18]:
```
dcorr = d.corr()
top_corr_features = dcorr.index
plt.figure(figsize=(10,7))
sns.heatmap(d[top_corr_features].corr(), annot=True, cmap="YlGnBu")
```

Out[18]:
```
<AxesSubplot:>
```



MinMaxScaler

From the original dataset, we can tell that each of our target value are in close proximity to one another. So, we will use MinMaxScaler to scale down all the target variables in the range of (0, 1) for the ease of computation.

In [19]:
```
data_close = d.reset_index()['Close']
data_close.head()
scaler = MinMaxScaler(feature_range = (0, 1))
data_close = scaler.fit_transform(np.array(data_close).reshape(-1, 1))
```

Splitting train, Test data

In [20]:
```
train_size = int(len(data_close)*0.70)
test_size = len(data_close) - train_size
train, test = data_close[0 : train_size, :], data_close[train_size : len(data_close), :1]
```

In [21]:
```
def create_matrix(ds, time_step=1):
    dataX, dataY = [], []
    for i in range(len(ds)-time_step-1):
        a = ds[i:(i+time_step),0]
        dataX.append(a)
        dataY.append(ds[i+time_step,0])
    return np.array(dataX), np.array(dataY)
```

In [22]:
```
step=100
```

```
X_train, y_train = create_matrix(train, step)
X_test, y_test = create_matrix(test, step)
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
(1323, 100) (1323,)
(510, 100) (510,)
```

In [23]:

```
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)
```

LSTM Model

In [24]:

```
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')
```

In [25]:

```
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: 50851 (198.64 KB)
Trainable params: 50851 (198.64 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [26]:

```
history = model.fit(X_train, y_train, validation_split=0.1, epochs=77, batch_size=64, verbose=1, shuffle=True).
```

```
Epoch 1/77
19/19 [==============================] - 19s 466ms/step - loss: 0.0095 - val_loss: 0.0011
Epoch 2/77
19/19 [==============================] - 6s 312ms/step - loss: 0.0017 - val_loss: 0.0014
Epoch 3/77
19/19 [==============================] - 5s 242ms/step - loss: 8.8285e-04 - val_loss: 0.0011
Epoch 4/77
19/19 [==============================] - 6s 306ms/step - loss: 8.3423e-04 - val_loss: 0.0011
Epoch 5/77
19/19 [==============================] - 6s 313ms/step - loss: 7.5993e-04 - val_loss: 0.0010
Epoch 6/77
19/19 [==============================] - 6s 319ms/step - loss: 7.3531e-04 - val_loss: 9.5006e-04
Epoch 7/77
19/19 [==============================] - 6s 309ms/step - loss: 7.1640e-04 - val_loss: 9.8613e-04
Epoch 8/77
19/19 [==============================] - 6s 311ms/step - loss: 7.2356e-04 - val_loss: 9.1564e-04
Epoch 9/77
19/19 [==============================] - 6s 314ms/step - loss: 6.8091e-04 - val_loss: 8.4356e-04
Epoch 10/77
19/19 [==============================] - 6s 320ms/step - loss: 6.6459e-04 - val_loss: 8.2614e-04
Epoch 11/77
19/19 [==============================] - 6s 305ms/step - loss: 7.0201e-04 - val_loss: 7.6497e-04
Epoch 12/77
19/19 [==============================] - 6s 308ms/step - loss: 6.5266e-04 - val_loss: 7.3140e-04
Epoch 13/77
19/19 [==============================] - 6s 310ms/step - loss: 6.5251e-04 - val_loss: 6.9356e-04
Epoch 14/77
19/19 [==============================] - 6s 302ms/step - loss: 6.1944e-04 - val_loss: 6.7393e-04
Epoch 15/77
19/19 [==============================] - 6s 309ms/step - loss: 5.7875e-04 - val_loss: 6.5108e-04
Epoch 16/77
19/19 [==============================] - 6s 309ms/step - loss: 5.5924e-04 - val_loss: 6.1736e-04
Epoch 17/77
19/19 [==============================] - 5s 281ms/step - loss: 5.6897e-04 - val_loss: 7.1954e-04
Epoch 18/77
19/19 [==============================] - 6s 312ms/step - loss: 5.4055e-04 - val_loss: 6.1696e-04
Epoch 19/77
19/19 [==============================] - 6s 314ms/step - loss: 5.1237e-04 - val_loss: 5.5446e-04
Epoch 20/77
19/19 [==============================] - 6s 311ms/step - loss: 5.5356e-04 - val_loss: 5.4329e-04
Epoch 21/77
19/19 [==============================] - 6s 302ms/step - loss: 5.2250e-04 - val_loss: 5.3143e-04
Epoch 22/77
19/19 [==============================] - 6s 304ms/step - loss: 4.8748e-04 - val_loss: 5.0588e-04
```

```
Epoch 23/77
19/19 [==============================] - 6s 313ms/step - loss: 4.8471e-04 - val_loss: 4.8191e-04
Epoch 24/77
19/19 [==============================] - 6s 301ms/step - loss: 4.6727e-04 - val_loss: 4.5961e-04
Epoch 25/77
19/19 [==============================] - 6s 308ms/step - loss: 4.2883e-04 - val_loss: 4.4935e-04
Epoch 26/77
19/19 [==============================] - 6s 303ms/step - loss: 4.6596e-04 - val_loss: 4.9181e-04
Epoch 27/77
19/19 [==============================] - 6s 301ms/step - loss: 4.3406e-04 - val_loss: 4.1582e-04
Epoch 28/77
19/19 [==============================] - 6s 302ms/step - loss: 3.9079e-04 - val_loss: 3.8903e-04
Epoch 29/77
19/19 [==============================] - 6s 306ms/step - loss: 3.7504e-04 - val_loss: 4.6897e-04
Epoch 30/77
19/19 [==============================] - 6s 314ms/step - loss: 3.6913e-04 - val_loss: 3.5128e-04
Epoch 31/77
19/19 [==============================] - 6s 308ms/step - loss: 3.5813e-04 - val_loss: 3.3327e-04
Epoch 32/77
19/19 [==============================] - 6s 306ms/step - loss: 3.3935e-04 - val_loss: 3.3347e-04
Epoch 33/77
19/19 [==============================] - 6s 304ms/step - loss: 3.4492e-04 - val_loss: 3.0228e-04
Epoch 34/77
19/19 [==============================] - 6s 305ms/step - loss: 3.5085e-04 - val_loss: 3.5436e-04
Epoch 35/77
19/19 [==============================] - 6s 301ms/step - loss: 3.3567e-04 - val_loss: 2.8558e-04
Epoch 36/77
19/19 [==============================] - 6s 302ms/step - loss: 2.9934e-04 - val_loss: 2.9267e-04
Epoch 37/77
19/19 [==============================] - 6s 307ms/step - loss: 2.8918e-04 - val_loss: 2.5104e-04
Epoch 38/77
19/19 [==============================] - 6s 302ms/step - loss: 2.9202e-04 - val_loss: 3.3358e-04
Epoch 39/77
19/19 [==============================] - 6s 310ms/step - loss: 2.7510e-04 - val_loss: 2.7438e-04
Epoch 40/77
19/19 [==============================] - 6s 307ms/step - loss: 2.7610e-04 - val_loss: 2.2728e-04
Epoch 41/77
19/19 [==============================] - 6s 311ms/step - loss: 2.5894e-04 - val_loss: 2.1416e-04
Epoch 42/77
19/19 [==============================] - 6s 305ms/step - loss: 2.6017e-04 - val_loss: 2.2252e-04
Epoch 43/77
19/19 [==============================] - 6s 303ms/step - loss: 2.7456e-04 - val_loss: 2.4486e-04
Epoch 44/77
19/19 [==============================] - 6s 310ms/step - loss: 2.5814e-04 - val_loss: 2.2163e-04
Epoch 45/77
19/19 [==============================] - 6s 314ms/step - loss: 2.5313e-04 - val_loss: 2.5874e-04
Epoch 46/77
19/19 [==============================] - 6s 306ms/step - loss: 2.2761e-04 - val_loss: 2.4506e-04
Epoch 47/77
19/19 [==============================] - 6s 304ms/step - loss: 2.2094e-04 - val_loss: 1.8435e-04
Epoch 48/77
19/19 [==============================] - 6s 308ms/step - loss: 2.2782e-04 - val_loss: 1.8110e-04
Epoch 49/77
19/19 [==============================] - 6s 306ms/step - loss: 2.3707e-04 - val_loss: 1.8098e-04
Epoch 50/77
19/19 [==============================] - 6s 309ms/step - loss: 2.4272e-04 - val_loss: 2.0720e-04
Epoch 51/77
19/19 [==============================] - 6s 304ms/step - loss: 2.5417e-04 - val_loss: 1.7808e-04
Epoch 52/77
19/19 [==============================] - 6s 311ms/step - loss: 2.1095e-04 - val_loss: 1.7075e-04
Epoch 53/77
19/19 [==============================] - 6s 309ms/step - loss: 2.1497e-04 - val_loss: 2.2800e-04
Epoch 54/77
19/19 [==============================] - 6s 312ms/step - loss: 1.9633e-04 - val_loss: 1.5867e-04
Epoch 55/77
19/19 [==============================] - 6s 306ms/step - loss: 1.9405e-04 - val_loss: 1.5750e-04
Epoch 56/77
19/19 [==============================] - 6s 305ms/step - loss: 1.8291e-04 - val_loss: 1.5191e-04
Epoch 57/77
19/19 [==============================] - 6s 318ms/step - loss: 1.7643e-04 - val_loss: 1.4763e-04
Epoch 58/77
19/19 [==============================] - 6s 304ms/step - loss: 1.9593e-04 - val_loss: 1.5243e-04
Epoch 59/77
19/19 [==============================] - 6s 317ms/step - loss: 1.8519e-04 - val_loss: 1.4589e-04
Epoch 60/77
19/19 [==============================] - 6s 321ms/step - loss: 1.7811e-04 - val_loss: 1.4403e-04
Epoch 61/77
19/19 [==============================] - 6s 321ms/step - loss: 1.7554e-04 - val_loss: 1.5382e-04
Epoch 62/77
19/19 [==============================] - 6s 319ms/step - loss: 1.6123e-04 - val_loss: 1.5165e-04
Epoch 63/77
19/19 [==============================] - 6s 320ms/step - loss: 1.6884e-04 - val_loss: 1.7173e-04
Epoch 64/77
19/19 [==============================] - 6s 320ms/step - loss: 1.7212e-04 - val_loss: 1.5789e-04
Epoch 65/77
19/19 [==============================] - 6s 320ms/step - loss: 1.7205e-04 - val_loss: 1.3889e-04
Epoch 66/77
19/19 [==============================] - 6s 322ms/step - loss: 1.5475e-04 - val_loss: 1.3328e-04
Epoch 67/77
```

```
19/19 [==============================] - 6s 322ms/step - loss: 1.5964e-04 - val_loss: 1.3603e-04
Epoch 68/77
19/19 [==============================] - 6s 322ms/step - loss: 1.4715e-04 - val_loss: 1.3006e-04
Epoch 69/77
19/19 [==============================] - 6s 325ms/step - loss: 1.4506e-04 - val_loss: 1.3063e-04
Epoch 70/77
19/19 [==============================] - 6s 326ms/step - loss: 1.4956e-04 - val_loss: 1.2533e-04
Epoch 71/77
19/19 [==============================] - 6s 328ms/step - loss: 1.4491e-04 - val_loss: 1.2328e-04
Epoch 72/77
19/19 [==============================] - 6s 319ms/step - loss: 1.4210e-04 - val_loss: 1.3301e-04
Epoch 73/77
19/19 [==============================] - 6s 321ms/step - loss: 1.5111e-04 - val_loss: 1.2929e-04
Epoch 74/77
19/19 [==============================] - 6s 320ms/step - loss: 1.4872e-04 - val_loss: 1.6713e-04
Epoch 75/77
19/19 [==============================] - 6s 321ms/step - loss: 1.4340e-04 - val_loss: 1.2040e-04
Epoch 76/77
19/19 [==============================] - 6s 324ms/step - loss: 1.5255e-04 - val_loss: 1.4013e-04
Epoch 77/77
19/19 [==============================] - 6s 328ms/step - loss: 1.9157e-04 - val_loss: 1.3463e-04
```

In [27]:
```
train_predict = model.predict(X_train)
test_predict = model.predict(X_test)
```

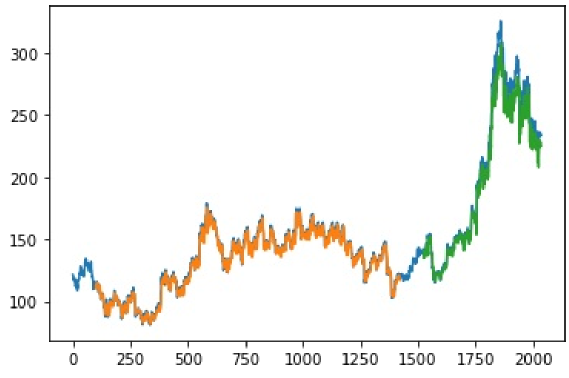
```
42/42 [==============================] - 5s 64ms/step
16/16 [==============================] - 1s 62ms/step
```

In [28]:
```
# Reversing the MinMax Scaler
train_predict = scaler.inverse_transform(train_predict)
test_predict = scaler.inverse_transform(test_predict)
```

In [29]:
```
math.sqrt(mean_squared_error(y_train, train_predict))
math.sqrt(mean_squared_error(y_test,test_predict))
```

Out[29]:
```
204.41385942429943
```

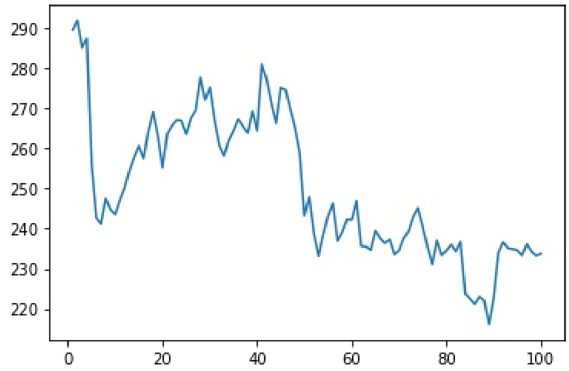
In [30]:
```
### Visualise the Predictions
look_back = 100
train_num_pyredict_plot = np.empty_like(data_close)
train_num_pyredict_plot[:, :] = np.nan
train_num_pyredict_plot[look_back : len(train_predict) + look_back, :] = train_predict
test_predict_plot = np.empty_like(data_close)
test_predict_plot[:, :] = np.nan
test_predict_plot[len(train_predict) + (look_back * 2) + 1 : len(data_close) - 1, :] = test_predict
plt.plot(scaler.inverse_transform(data_close))
plt.plot(train_num_pyredict_plot)
plt.plot(test_predict_plot)
plt.show()
```



In [31]:
```
x_inum_pyut=test[307:].reshape(1, -1)
x_inum_pyut.shape
temp_inum_pyut = list(x_inum_pyut)
temp_inum_pyut = temp_inum_pyut[0].tolist()
temp_inum_pyut = list(x_inum_pyut)
temp_inum_pyut = temp_inum_pyut[0].tolist()
```

In [32]:
```
day_new = np.arange(1, 101)
day_pred = np.arange(101, 131)
plt.plot(day_new, scaler.inverse_transform(data_close[1935 : ]))
```

Out[32]:
```
[<matplotlib.lines.Line2D at 0x1eeb1891430>]
```



In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js