In [34]:

```
#import the necessary libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [35]:

```
#reading our dataset

df = pd.read_csv("NSE-TATAGLOBAL.csv")
```

In [3]:

```
df
```

Out[3]:

	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 [36]:

```
#we are going to train our model on df['Open'] column. You can try this on other column as

Open = df.iloc[:,1:2].values
Open
```

Out[36]:

```
array([[234.05],
       [234.55],
       [240.  ],
       ...,
       [121.8 ],
       [120.3 ],
       [122.1 ]])
```

In [37]:

```
#we will standardise our data to avoid the problem of exploding gradient

from sklearn.preprocessing import MinMaxScaler
```

In [38]:

```
#now let's scale our data by fit_transform() method

sc = MinMaxScaler(feature_range=(0,1))
opensc = sc.fit_transform(Open)
opensc
```

Out[38]:

```
array([[0.6202352 ],
       [0.62226277],
       [0.64436334],
       ...,
       [0.16504461],
       [0.15896188],
       [0.16626115]])
```

In [7]:

```
len(opensc)
```

Out[7]:

```
2035
```

In [43]:

```
#now we will create input and output datasets for training our model.
#we will train our LSTM model on time steps of 60 datapoints and predict the next one.
#so , our single input instance will be x1=opensc.iloc[0:59] and output for that will be y1
# similarly x2=opensc.iloc[1:60] , y2=opensc.iloc[61] and so on

xtrain=[]
ytrain=[]

for i in range(60,2035):
    xtrain.append(opensc[i-60:i,0])
    ytrain.append(opensc[i,0])
xtrain , ytrain = np.array(xtrain) , np.array(ytrain)
```

In [9]:

```
xtrain.shape
```

Out[9]:

```
(1975, 60)
```

In [40]:

```
xtrain = np.reshape(xtrain, (xtrain.shape[0], xtrain.shape[1], 1))
```

In [41]:

```
xtrain.shape
```

Out[41]:

```
(1975, 60, 1)
```

In [42]:

```
#importing the necessary libraries for creating our LSTM model

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

In [46]:

```
#we are naming our model --> regressor1 , you can name it anything else also.
#than one by one we will add layers to our regressor1 .
#each LSTM layer will be followed by a Dropout Layer wit probability=0.2
#in this model we will use 4 LSTM layers and 1 Dense layer

regressor1 = Sequential()

regressor1.add(LSTM(units = 50, return_sequences = True, input_shape = (xtrain.shape[1], 1)
regressor1.add(Dropout(0.2))
```

In [17]:

```
regressor1.add(LSTM(units = 50, return_sequences = True))
regressor1.add(Dropout(0.2))
```

In [18]:

```
regressor1.add(LSTM(units = 50, return_sequences = True))
regressor1.add(Dropout(0.2))
```

In [19]:

```
regressor1.add(LSTM(units = 50))
regressor1.add(Dropout(0.2))
```

In [22]:

```
# Adding the output layer , we will take units=1 , because we have set our model to predict
regressor1.add(Dense(units = 1))

# Compiling the RNN
regressor1.compile(optimizer = 'adam', loss = 'mean_squared_error')

# Fitting the RNN to the Training set
regressor1.fit(xtrain, ytrain, epochs = 50, batch_size = 32)
```

```
Epoch 1/50
62/62 [==============================] - 10s 61ms/step - loss: 0.0562
Epoch 2/50
62/62 [==============================] - 4s 60ms/step - loss: 0.0361
Epoch 3/50
62/62 [==============================] - 4s 60ms/step - loss: 0.0359
Epoch 4/50
62/62 [==============================] - 4s 62ms/step - loss: 0.0355
Epoch 5/50
62/62 [==============================] - 4s 62ms/step - loss: 0.0306
Epoch 6/50
62/62 [==============================] - 4s 64ms/step - loss: 0.0172
Epoch 7/50
62/62 [==============================] - 4s 63ms/step - loss: 0.0058
Epoch 8/50
62/62 [==============================] - 4s 63ms/step - loss: 0.0052
Epoch 9/50
62/62 [==============================] - 4s 63ms/step - loss: 0.0029
Epoch 10/50
62/62 [==============================] - 4s 63ms/step - loss: 0.0024
Epoch 11/50
62/62 [==============================] - 4s 63ms/step - loss: 0.0021
Epoch 12/50
62/62 [==============================] - 4s 62ms/step - loss: 0.0020
Epoch 13/50
62/62 [==============================] - 4s 62ms/step - loss: 0.0023
Epoch 14/50
62/62 [==============================] - 4s 61ms/step - loss: 0.0021
Epoch 15/50
62/62 [==============================] - 4s 61ms/step - loss: 0.0017
Epoch 16/50
62/62 [==============================] - 4s 61ms/step - loss: 0.0017
Epoch 17/50
62/62 [==============================] - 4s 60ms/step - loss: 0.0018
Epoch 18/50
62/62 [==============================] - 4s 60ms/step - loss: 0.0016
Epoch 19/50
62/62 [==============================] - 4s 61ms/step - loss: 0.0015
Epoch 20/50
62/62 [==============================] - 4s 62ms/step - loss: 0.0015
Epoch 21/50
62/62 [==============================] - 4s 62ms/step - loss: 0.0014
Epoch 22/50
62/62 [==============================] - 4s 60ms/step - loss: 0.0014
Epoch 23/50
62/62 [==============================] - 4s 60ms/step - loss: 0.0015
Epoch 24/50
62/62 [==============================] - 4s 64ms/step - loss: 0.0014
```

```
Epoch 25/50
62/62 [==============================] - 4s 60ms/step - loss: 0.0012
Epoch 26/50
62/62 [==============================] - 4s 61ms/step - loss: 0.0013
Epoch 27/50
62/62 [==============================] - 4s 60ms/step - loss: 0.0014
Epoch 28/50
62/62 [==============================] - 4s 60ms/step - loss: 0.0013
Epoch 29/50
62/62 [==============================] - 4s 62ms/step - loss: 0.0013
Epoch 30/50
62/62 [==============================] - 4s 71ms/step - loss: 0.0012
Epoch 31/50
62/62 [==============================] - 4s 61ms/step - loss: 0.0011
Epoch 32/50
62/62 [==============================] - 4s 60ms/step - loss: 0.0011
Epoch 33/50
62/62 [==============================] - 4s 61ms/step - loss: 0.0011
Epoch 34/50
62/62 [==============================] - 4s 61ms/step - loss: 0.0010
Epoch 35/50
62/62 [==============================] - 4s 62ms/step - loss: 0.0011
Epoch 36/50
62/62 [==============================] - 4s 62ms/step - loss: 0.0011
Epoch 37/50
62/62 [==============================] - 4s 62ms/step - loss: 0.0010
Epoch 38/50
62/62 [==============================] - 4s 65ms/step - loss: 9.2179e-04
Epoch 39/50
62/62 [==============================] - 4s 65ms/step - loss: 0.0010
Epoch 40/50
62/62 [==============================] - 4s 65ms/step - loss: 9.6964e-04
Epoch 41/50
62/62 [==============================] - 4s 60ms/step - loss: 9.9284e-04
Epoch 42/50
62/62 [==============================] - 4s 60ms/step - loss: 8.9140e-04
Epoch 43/50
62/62 [==============================] - 4s 63ms/step - loss: 8.6000e-04
Epoch 44/50
62/62 [==============================] - 4s 60ms/step - loss: 9.2178e-04
Epoch 45/50
62/62 [==============================] - 4s 60ms/step - loss: 8.4539e-04
Epoch 46/50
62/62 [==============================] - 4s 60ms/step - loss: 8.0638e-04
Epoch 47/50
62/62 [==============================] - 4s 62ms/step - loss: 8.6092e-04
Epoch 48/50
62/62 [==============================] - 4s 61ms/step - loss: 8.7283e-04
Epoch 49/50
62/62 [==============================] - 4s 62ms/step - loss: 8.1207e-04
Epoch 50/50
62/62 [==============================] - 4s 60ms/step - loss: 9.0348e-04
```

Out[22]:

```
<keras.callbacks.History at 0x2231bce9f40>
```

In [25]:

```
pred_train=regressor1.predict(xtrain)
pred_train.shape
```

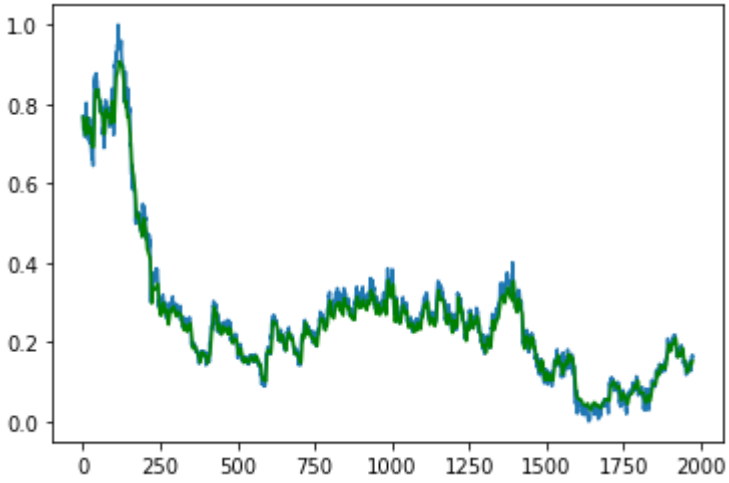
```
62/62 [==============================] - 1s 21ms/step
```

Out[25]:

```
(1975, 1)
```

In [26]:

```
plt.plot(ytrain)
plt.plot(pred_train , 'g')
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: