```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the inpu

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output whe
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current se
```

```
!pip install yfinance
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting yfinance
  Downloading yfinance-0.1.90-py2.py3-none-any.whl (29 kB)
Requirement already satisfied: pandas>=1.3.0 in /usr/local/lib/python3.8/dist-packages (from yfinance) (1.3.5)
Requirement already satisfied: numpy>=1.16.5 in /usr/local/lib/python3.8/dist-packages (from yfinance) (1.21.6)
Requirement already satisfied: multitasking>=0.0.7 in /usr/local/lib/python3.8/dist-packages (from yfinance) (0.0.11)
Requirement already satisfied: lxml>=4.9.1 in /usr/local/lib/python3.8/dist-packages (from yfinance) (4.9.2)
Collecting requests>=2.26
  Downloading requests-2.28.1-py3-none-any.whl (62 kB)
     |████████████████████████████████| 62 kB 1.1 MB/s
Requirement already satisfied: appdirs>=1.4.4 in /usr/local/lib/python3.8/dist-packages (from yfinance) (1.4.4)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.8/dist-packages (from pandas>=1.3.0->yfinance) (2.8
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.8/dist-packages (from pandas>=1.3.0->yfinance) (2022.6)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-packages (from python-dateutil>=2.7.3->pandas>=1.3.0->yfin
Requirement already satisfied: charset-normalizer<3,>=2 in /usr/local/lib/python3.8/dist-packages (from requests>=2.26->yfinance) (
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests>=2.26->yfinance) (2022.1
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests>=2.26->yfinance) (1.2
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests>=2.26->yfinance) (2.10)
Installing collected packages: requests, yfinance
  Attempting uninstall: requests
    Found existing installation: requests 2.23.0
    Uninstalling requests-2.23.0:
      Successfully uninstalled requests-2.23.0
Successfully installed requests-2.28.1 yfinance-0.1.90
```

```
import yfinance as yf
```

```
stock_symbol = 'GAIL.NS'
```

```
data = yf.download(tickers = stock_symbol,period ='5y',interval ='1d')
```

```
[*********************100%***********************]  1 of 1 completed
```

```
type(data)
```

```
pandas.core.frame.DataFrame
```

```
data.head()
```

Date	Open	High	Low	Close	Adj Close	Volume
2017-12-18	121.975029	126.112534	118.962532	124.000031	100.664841	25651817
2017-12-19	124.125031	125.737534	122.400032	124.137527	100.776451	11376621
2017-12-20	124.525032	127.375031	124.250031	124.850029	101.354866	17112567
2017-12-21	125.500031	126.500031	124.587532	125.375031	101.781067	9272425
2017-12-22	125.425034	127.175034	125.150032	125.775032	102.105804	12435376

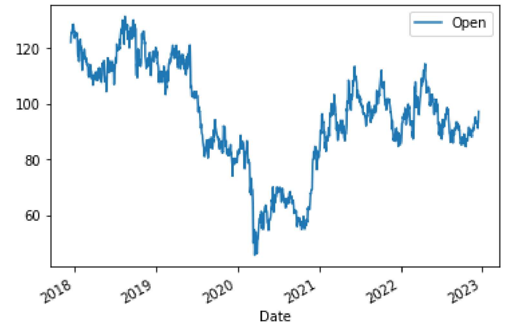
```
data.tail()
```

Date	Open	High	Low	Close	Adj Close	Volume
2022-12-12	91.099998	93.400002	90.199997	92.800003	92.800003	15920076
2022-12-13	93.150002	94.000000	92.699997	93.250000	93.250000	11215454
2022-12-14	93.250000	96.300003	93.150002	96.000000	96.000000	22598875
2022-12-15	96.150002	98.400002	95.449997	97.349998	97.349998	28006772
2022-12-16	97.199997	100.199997	96.599998	96.750000	96.750000	35484499

```
opn = data[['Open']]
```

```
opn.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f436bf7f100>
```



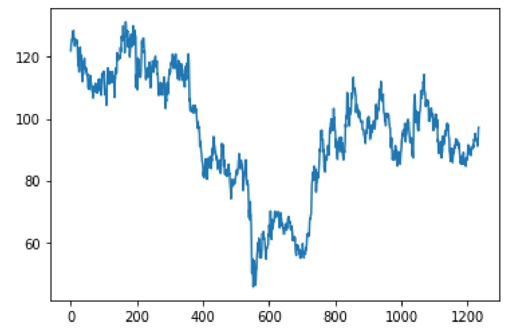

```
import matplotlib.pyplot as plt
```

```
ds = opn.values
```

```
ds
```

```
array([[121.97502899],
       [124.12503052],
       [124.52503204],
       ...,
       [ 93.25      ],
       [ 96.15000153],
       [ 97.19999695]])
```

```
plt.plot(ds);
```




```
import numpy as np
```

```
from sklearn.preprocessing import MinMaxScaler
```

```
normalizer = MinMaxScaler(feature_range=(0,1))
ds_scaled = normalizer.fit_transform(np.array(ds).reshape(-1,1))
```

```
len(ds_scaled),len(ds)
```

```
(1235, 1235)
```

```
train_size = int(len(ds_scaled)*0.70)
test_size = len(ds_scaled)-train_size
```

```
train_size,test_size
```

```
(864, 371)
```

```
ds_train,ds_test = ds_scaled[0:train_size,:],ds_scaled[train_size:len(ds_scaled),:1]
```

```
len(ds_train),len(ds_test)
```

```
(864, 371)
```

```
#creating dataset in time series for LSTM
def create_ds(dataset,step):
    Xtrain,Ytrain = [],[]
    for i in range(len(dataset)-step-1):
        a = dataset[i:(i+step),0]
        Xtrain.append(a)
        Ytrain.append(dataset[i+step,0])
    return np.array(Xtrain),np.array(Ytrain)
```

```
#taking 100 days price as one record for training
time_stamp = 100
x_train,y_train = create_ds(ds_train,time_stamp)
x_test,y_test = create_ds(ds_test,time_stamp)
```

```
x_train.shape,y_train.shape
```

```
((763, 100), (763,))
```

```
x_test.shape,y_test.shape
```

```
((270, 100), (270,))
```

```
#reshaping data to fit into 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)

```

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

```
#creating LSTM model using keras
model = Sequential()
model.add(LSTM(units=50,return_sequences=True,input_shape=(x_train.shape[1],1)))
model.add(LSTM(units=50,return_sequences=True))
model.add(LSTM(units=50))
model.add(Dense(units=1,activation='linear'))
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
_________________________________________________________________
```

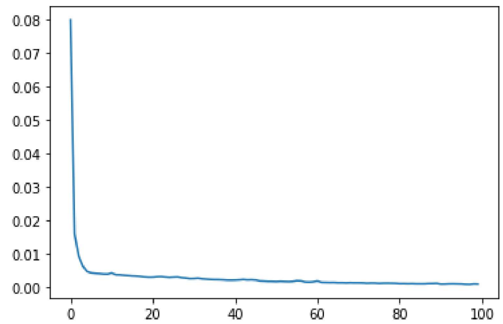
```
#Training model with adam optimizer and mean squared error loss function
```

```
model.compile(loss='mean_squared_error',optimizer = 'adam')
model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=100,batch_size=64)
```

```
Epoch 5/100
12/12 [==============================] - 2s 201ms/step - loss: 0.0048 - val_loss: 0.0034
Epoch 6/100
12/12 [==============================] - 2s 202ms/step - loss: 0.0044 - val_loss: 0.0032
Epoch 7/100
12/12 [==============================] - 2s 209ms/step - loss: 0.0043 - val_loss: 0.0030
Epoch 8/100
12/12 [==============================] - 2s 204ms/step - loss: 0.0042 - val_loss: 0.0029
Epoch 9/100
12/12 [==============================] - 2s 204ms/step - loss: 0.0040 - val_loss: 0.0029
Epoch 10/100
12/12 [==============================] - 2s 201ms/step - loss: 0.0040 - val_loss: 0.0030
Epoch 11/100
12/12 [==============================] - 2s 201ms/step - loss: 0.0044 - val_loss: 0.0033
Epoch 12/100
12/12 [==============================] - 2s 200ms/step - loss: 0.0038 - val_loss: 0.0027
Epoch 13/100
12/12 [==============================] - 2s 204ms/step - loss: 0.0038 - val_loss: 0.0027
Epoch 14/100
12/12 [==============================] - 2s 203ms/step - loss: 0.0037 - val_loss: 0.0026
Epoch 15/100
12/12 [==============================] - 2s 201ms/step - loss: 0.0036 - val_loss: 0.0027
Epoch 16/100
12/12 [==============================] - 2s 201ms/step - loss: 0.0035 - val_loss: 0.0026
Epoch 17/100
12/12 [==============================] - 2s 204ms/step - loss: 0.0034 - val_loss: 0.0026
Epoch 18/100
12/12 [==============================] - 2s 204ms/step - loss: 0.0033 - val_loss: 0.0025
Epoch 19/100
12/12 [==============================] - 2s 202ms/step - loss: 0.0032 - val_loss: 0.0025
Epoch 20/100
12/12 [==============================] - 2s 201ms/step - loss: 0.0031 - val_loss: 0.0024
Epoch 21/100
12/12 [==============================] - 2s 209ms/step - loss: 0.0031 - val_loss: 0.0024
Epoch 22/100
12/12 [==============================] - 2s 202ms/step - loss: 0.0032 - val_loss: 0.0024
Epoch 23/100
12/12 [==============================] - 2s 201ms/step - loss: 0.0033 - val_loss: 0.0026
Epoch 24/100
12/12 [==============================] - 2s 204ms/step - loss: 0.0032 - val_loss: 0.0024
Epoch 25/100
12/12 [==============================] - 2s 204ms/step - loss: 0.0031 - val_loss: 0.0024
Epoch 26/100
12/12 [==============================] - 2s 206ms/step - loss: 0.0031 - val_loss: 0.0028
Epoch 27/100
12/12 [==============================] - 2s 204ms/step - loss: 0.0032 - val_loss: 0.0024
Epoch 28/100
12/12 [==============================] - 2s 202ms/step - loss: 0.0030 - val_loss: 0.0024
Epoch 29/100
12/12 [==============================] - 2s 203ms/step - loss: 0.0028 - val_loss: 0.0021
Epoch 30/100
12/12 [==============================] - 2s 202ms/step - loss: 0.0027 - val_loss: 0.0021
Epoch 31/100
12/12 [==============================] - 2s 203ms/step - loss: 0.0027 - val_loss: 0.0021
Epoch 32/100
12/12 [==============================] - 2s 202ms/step - loss: 0.0028 - val_loss: 0.0024
Epoch 33/100
12/12 [==============================] - 2s 202ms/step - loss: 0.0026 - val_loss: 0.0020
Epoch 34/100
```

```
#plotting loss,it showsthat loss has decreased significantly and model trained well
loss = model.history.history['loss']
plt.plot(loss)
```

```
[<matplotlib.lines.Line2D at 0x7f4303201640>]
```



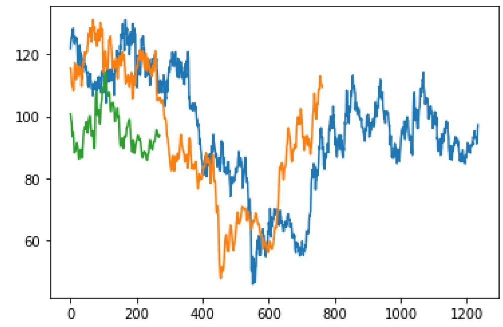

```
#Predicting on train and test data
train_predict = model.predict(x_train)
test_predict = model.predict(x_test)
```

```
24/24 [==============================] - 2s 36ms/step
9/9 [==============================] - 0s 36ms/step
```

```
#inverse transforn to get actual value
train_predict = normalizer.inverse_transform(train_predict)
test_predict = normalizer.inverse_transform(test_predict)
```

```
#comparing using visuals
plt.plot(normalizer.inverse_transform(ds_scaled))
plt.plot(train_predict)
plt.plot(test_predict)
```

```
[<matplotlib.lines.Line2D at 0x7f43097d48e0>]
```



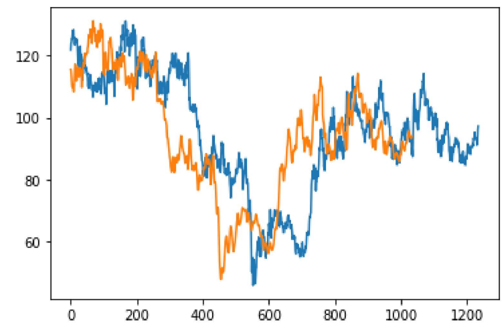

```
type(train_predict)
```

```
numpy.ndarray
```

```
test = np.vstack((train_predict,test_predict))
```

```
#combining the predicted data to creatye uniforn data visualization
plt.plot(normalizer.inverse_transform(ds_scaled))
plt.plot(test)
```

```
[<matplotlib.lines.Line2D at 0x7f43097f2970>]
```




```
len(ds_test)
```

```
371
```

```
#getting  the last 100 days records
fut_inp = ds_test[270:]
```

```
fut_inp = fut_inp.reshape(1,-1)
```

```
tmp_inp = list(fut_inp)
```

```
#creating list of the last 100 days
tmp_inp = tmp_inp[0].tolist()
```

```
#Predicting next 30 days price suing the current data
#It will predict in sliding window manner (algorithm) with stride 1
lst_output=[]
n_steps=100
```

```
i=0
while(i<30):

    if(len(tmp_inp)>100):
        fut_inp = np.array(tmp_inp[1:])
        fut_inp=fut_inp.reshape(1,-1)
        fut_inp = fut_inp.reshape((1, n_steps, 1))
        yhat = model.predict(fut_inp, verbose=0)
        tmp_inp.extend(yhat[0].tolist())
        tmp_inp = tmp_inp[1:]
        lst_output.extend(yhat.tolist())
        i=i+1
    else:
        fut_inp = fut_inp.reshape((1, n_steps,1))
        yhat = model.predict(fut_inp, verbose=0)
        tmp_inp.extend(yhat[0].tolist())
        lst_output.extend(yhat.tolist())
        i=i+1


print(lst_output)

```

```
[[0.60503751039505], [0.6185694336891174], [0.6282083988189697], [0.6361008882522583], [0.6428959369659424], [0.6489161849021912],
```

```
len(ds_scaled)

```

```
1235
```

```
#Creating a dummy plane to plot graph one after another
plot_new=np.arange(1,101)
plot_pred=np.arange(101,131)
```

```

ds_new = ds_scaled.tolist()
```

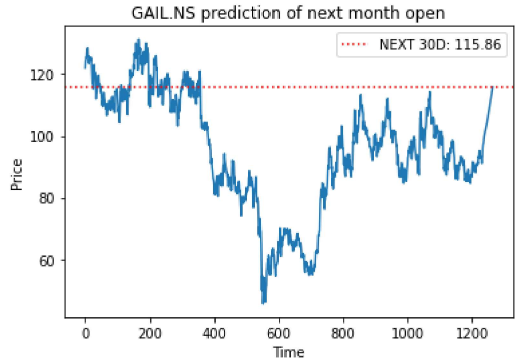
```
len(ds_new)

```

```
1235
```

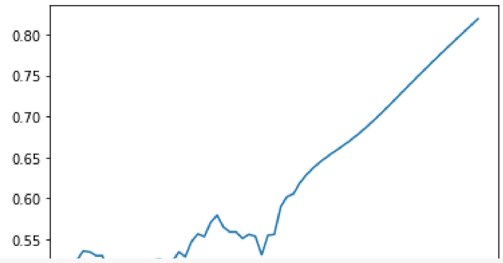
```
#Plotting final results with predicted value after 30 Days
plt.plot(final_graph,)
plt.ylabel("Price")
plt.xlabel("Time")
plt.title("{0} prediction of next month open".format(stock_symbol))
plt.axhline(y=final_graph[len(final_graph)-1], color = 'red', linestyle = ':', label = 'NEXT 30D: {0}'.
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f43063e6ee0>
```



```
#Entends helps us to fill the missing value with approx value
ds_new.extend(lst_output)
plt.plot(ds_new[1200:])
```

[<matplotlib.lines.Line2D at 0x7f430653a6d0>]




```
#Creating final data for plotting
final_graph = normalizer.inverse_transform(ds_new).tolist()
```

```
#Plotting final results with predicted value after 30 Days
plt.plot(final_graph,)
plt.ylabel("Price")
plt.xlabel("Time")
plt.title("{0} prediction of next month open".format(stock_symbol))
plt.axhline(y=final_graph[len(final_graph)-1], color = 'red', linestyle = ':', label = 'NEXT 30D: {0}'.·
plt.legend()
```

<matplotlib.legend.Legend at 0x7f43062a3ca0>