

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

In [4]:

```
data=pd.read_excel("Crude Oil Prices Daily.xlsx")
```

In [5]:

```
data.isnull().any()
```

Out[5]:

```
Date                False
Closing Value        True
dtype: bool
```

In [6]:

```
data.isnull().sum()
```

Out[6]:

```
Date                0
Closing Value        7
dtype: int64
```

In [7]:

```
data.dropna(axis=0,inplace=True)
```

In [8]:

```
data.isnull().sum()
```

Out[8]:

```
Date                0
Closing Value        0
dtype: int64
```

In [9]:

```
data_oil=data.reset_index()['Closing Value']
data_oil
```

Out[9]:

```
0      25.56
1      26.00
2      26.53
3      25.85
4      25.87
...
8211   73.89
8212   74.19
8213   73.05
8214   73.78
8215   73.93
Name: Closing Value, Length: 8216, dtype: float64
```

In [10]:

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler(feature_range=(0,1))
data_oil=scaler.fit_transform(np.array(data_oil).reshape(-1,1))
```

In [11]:

```
data_oil
```

```
array([[0.11335703],  
       [0.11661484],  
       [0.12053902],  
       ...,  
       [0.46497853],  
       [0.47038353],  
       [0.47149415]])
```

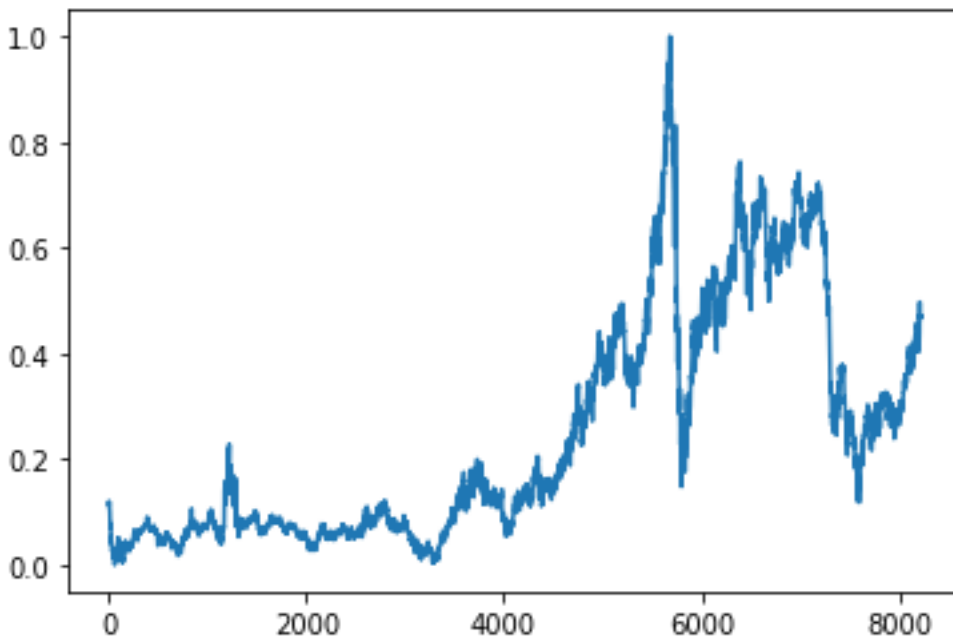
Out[11]:

```
plt.plot(data_oil)
```

In [12]:

Out[12]:

```
[]
```



In [13]:

```
training_size=int(len(data_oil)*0.65)  
test_size=len(data_oil)-training_size  
train_data,test_data=data_oil[0:training_size:],data_oil[training_size:len(d  
ata_oil),:1]
```

In [14]:

```
training_size,test_size
```

Out[14]:

```
(5340, 2876)
```

In [15]:

```
train_data.shape
```

Out[15]:

```
(5340, 1)
```

In [16]:

```
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]
        dataX.append(a)
        dataY.append(dataset[i+time_step,0])
    return np.array(dataX),np.array(dataY)

```

In [17]:

```

time_step=10
x_train,y_train=create_dataset(train_data,time_step)
x_test,y_test=create_dataset(test_data,time_step)

```

In [18]:

```

print(x_train.shape),print(y_train.shape)
(5329, 10)
(5329,)

```

Out[18]:

```
(None, None)
```

In [19]:

```

print(x_test.shape),print(y_test.shape)
(2865, 10)
(2865,)

```

Out[19]:

```
(None, None)
```

In [20]:

```
x_train
```

Out[20]:

```

array([[0.11335703, 0.11661484, 0.12053902, ..., 0.10980305, 0.1089886 ,
        0.11054346],
       [0.11661484, 0.12053902, 0.11550422, ..., 0.1089886 , 0.11054346,
        0.10165852],
       [0.12053902, 0.11550422, 0.1156523 , ..., 0.11054346, 0.10165852,
        0.09906708],
       ...,
       [0.36731823, 0.35176958, 0.36080261, ..., 0.36391234, 0.37042796,
        0.37042796],
       [0.35176958, 0.36080261, 0.35354657, ..., 0.37042796, 0.37042796,
        0.37879461],
       [0.36080261, 0.35354657, 0.35295424, ..., 0.37042796, 0.37879461,
        0.37916482]])

```

In [21]:

```

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)

```

In [22]:

```

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM

```

In [23]:

```
model=Sequential()
```

In [24]:

```

model.add(LSTM(50,return_sequences=True,input_shape=(10,1)))
model.add(LSTM(50,return_sequences=True))

```

```
model.add(LSTM(50))
```

In [25]:

```
model.add(Dense(1))
```

In [26]:

```
model.summary()
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 10, 50)	10400
lstm_1 (LSTM)	(None, 10, 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

In [27]:

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

In [28]:

```
model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=3,batch_size=64,verbose=1)
```

```
Epoch 1/3
```

```
84/84 [=====] - 12s 49ms/step - loss: 0.0020 - val_loss: 9.1990e-04
```

```
Epoch 2/3
```

```
84/84 [=====] - 2s 30ms/step - loss: 1.2791e-04 - val_loss: 8.4157e-04
```

```
Epoch 3/3
```

```
84/84 [=====] - 3s 31ms/step - loss: 1.2276e-04 - val_loss: 7.7963e-04
```

Out[28]:

In [29]:

```
train_predict=scaler.inverse_transform(train_data)
```

```
test_predict=scaler.inverse_transform(test_data)
```

```
### Calculate RMSE performance metrics
```

```
import math
```

```
from sklearn.metrics import mean_squared_error
```

```
math.sqrt(mean_squared_error(train_data,train_predict))
```

Out[29]:

```
29.347830443269938
```

In [30]:

```
from tensorflow.keras.models import load_model
```

In [32]:

```
model.save("crude_oil.hs")
```

```
WARNING:absl:Found untraced functions such as lstm_cell_layer_call_fn, lstm_cell_layer_call_and_return_conditional_losses, lstm_cell_1_layer_call_fn, lstm_cell_1_layer_call_and_return_conditional_losses, lstm_cell_2_layer_call_fn while saving (showing 5 of 6). These functions will not be directly callable after loading.
```