

## MODEL BUILDING-MODEL EVALUATION

Team ID	PNT2022TMID52922
Project Name	Crude Oil Price Prediction

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

```
[ ] data = pd.read_excel("/content/drive/MyDrive/Crude Oil Prices Daily.xlsx")
data.head()
```

	Date	Closing Value
0	1986-01-02	25.56
1	1986-01-03	26.00
2	1986-01-06	26.53
3	1986-01-07	25.85
4	1986-01-08	25.87

```
[ ] data.isnull().any()
```

Date	False
Closing Value	True
dtype: bool	

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

```
[ ] data.isnull().sum()
```

Date	0
Closing Value	0
dtype: int64	

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

```
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
```

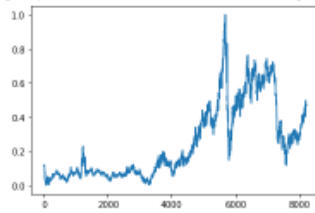
```
[ ] from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler(feature_range=(0,1))
data_oil=scaler.fit_transform(np.array(data_oil).reshape(-1,1))
```

```
[ ] data_oil
```

```
array([[0.11335703],
       [0.11661484],
       [0.12053902],
       ...,
       [0.46407853],
       [0.47038353],
       [0.47149415]])
```

```
[ ] plt.plot(data_oil)
```

1 [matplotlib.lines.Line2D at 0x7f05a724c690]



```
[ ] 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(data_oil):-1]
```

```
[ ] training_size,test_size
```

```
(5348, 2876)
```

```
[ ] train_data.shape
```

```
(5348, 1)
```

```
[ ] 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)
```

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

2 print(x\_train.shape),print(y\_train.shape)

```
(5329, 10)
(5329,)
(None, None)
```

```
[ ] print(x_test.shape),print(y_test.shape)
```

```
(2865, 10)
(2865,)
(None, None)
```

```
[ ] x_train
```

```
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, 0.36880261, ..., 0.36391234, 0.37042796,
        0.37042796],
       [0.35176958, 0.36880261, 0.35354657, ..., 0.37042796, 0.37042796,
        0.37879461],
       [0.36880261, 0.35354657, 0.35295424, ..., 0.37042796, 0.37879461,
        0.37916482]])
```

```
[ ] 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=(10,1)))
model.add(LSTM(50,return_sequences=True))
model.add(LSTM(50))
```

3 model.add(Dense(1))

4 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
```

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

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

```
Epoch 1/3
84/84 [=====] - 8s 30ms/step - loss: 0.0020 - val_loss: 0.0012
Epoch 2/3
84/84 [=====] - 1s 17ms/step - loss: 1.3052e-04 - val_loss: 8.6409e-04
Epoch 3/3
84/84 [=====] - 1s 17ms/step - loss: 1.3058e-04 - val_loss: 7.7004e-04
<keras.callbacks.History at 0x7f0548caa6d0>
```

```
[ ] 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))
```

29.347830443269938