

## MODEL BUILDING- MODEL EVALUATION

Team ID	PNT2022TMID43576
Project Name	Crude Oil Price Prediction

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
In [2]: data=pd.read_excel("/content/Crude Oil Prices Daily.xlsx")
```

```
In [3]: data.isnull().any()
```

```
Out[3]: Date          False
Closing Value      True
dtype: bool
```

```
In [4]: data.isnull().sum()
```

```
Out[4]: Date          0
Closing Value      7
dtype: int64
```

```
In [5]: data.dropna(axis=0,inplace=True)
```

```
In [6]: data.isnull().sum()
```

```
Out[6]: Date          0
Closing Value      0
dtype: int64
```

```
In [7]: data_oil=data.reset_index()['Closing Value']
data_oil
```

```
Out[7]: 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
```

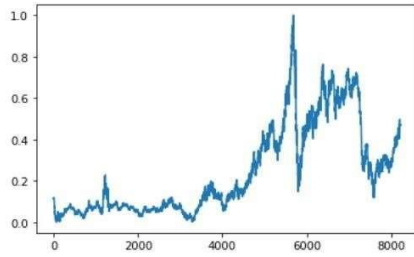
```
In [8]: from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler(feature_range=(0,1))
data_oil=scaler.fit_transform(np.array(data_oil).reshape(-1,1))
```

```
In [9]: data_oil
```

```
Out[9]: array([[0.11335703],
[0.11661484],
[0.12053902],
...,
[0.46497853],
[0.47038353],
[0.47149415]])
```

```
In [10]: plt.plot(data_oil)
```

Out[10]: []



```
In [11]: 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),:]
```

```
In [12]: training_size,test_size
```

Out[12]: (5340, 2876)

```
In [13]: train_data.shape
```

Out[13]: (5340, 1)

```
In [14]: 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 [15]: time_step=10
x_train,y_train=create_dataset(train_data,time_step)
x_test,y_test=create_dataset(test_data,time_step)
```

```
In [16]: print(x_train.shape),print(y_train.shape)
```

(5329, 10)  
(5329,)

Out[16]: (None, None)

```
In [17]: print(x_test.shape),print(y_test.shape)
```

(2865, 10)  
(2865,)

Out[17]: (None, None)

```
In [18]: x_train
```

```
Out[18]: 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 [19]: 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 [20]: from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import Dense
         from tensorflow.keras.layers import LSTM
```

```
In [22]: model=Sequential()
```

```
In [23]: model.add(LSTM(50,return_sequences=True,input_shape=(10,1)))
         model.add(LSTM(50,return_sequences=True))
         model.add(LSTM(50))
```

```
In [24]: model.add(Dense(1))
```

```
In [25]: model.summary()
```

Model: "sequential\_1"

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

```
In [27]: model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=3,batch_size=64,verbose=1)

Epoch 1/3
84/84 [=====] - 11s 48ms/step - loss: 0.0023 - val_loss: 0.0010
Epoch 2/3
84/84 [=====] - 2s 28ms/step - loss: 1.2599e-04 - val_loss: 8.0346e-04
Epoch 3/3
84/84 [=====] - 2s 30ms/step - loss: 1.2479e-04 - val_loss: 9.4227e-04
```

```
Out[27]:
```

```
In [28]: 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[28]: 29.347830443269938
```