**MODEL BUILDING**

# Importing The Model Building Libraries

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

# Initializing The Model model

= Sequential()

# Adding LSTM Layers

model.add(LSTM(50,return\_sequences = True, input\_shape = (10,1))) model.add(LSTM(50,return\_sequences = True)) model.add(LSTM(50))

# Adding Output Layers

model.add(Dense(1)) 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

**Configure The Learning Process** model.compile(loss='mean\_squared\_error', optimizer = 'adam')

# Train The Model

model.fit(X\_train, y\_train, validation\_data = (X\_test, ytest), epochs

= 10, batch\_size = 64, verbose = 1) Epoch 1/10

84/84 [==============================]

|  |  |  |  |
| --- | --- | --- | --- |
| - 8s | 33ms/step | - loss: | 0.0019 - |
| - 1s | 17ms/step | - loss: | 1.2329e- |
| * 1s * 1s | 17ms/step  18ms/step | * loss: * loss: | 1.2151e- |
| * 1s * 1s * 1s | 17ms/step  17ms/step  17ms/step | * loss: * loss: * loss: | 1.1884e- |
| - 1s | 17ms/step | - loss: | 1.1737e- |
| - 1s | 17ms/step | - loss: | 1.1241e- |
| - 1s | 17ms/step | - loss: | 1.1236e- |

val\_loss: 9.9616e-04 Epoch 2/10

84/84 [==============================]

04 - val\_loss: 7.3913e-04 Epoch 3/10

84/84 [==============================]

1.2148e04 - val\_loss: 0.0014 Epoch 4/10

84/84 [==============================]

04 - val\_loss: 7.6063e-04 Epoch 5/10

84/84 [==============================]

1.2602e04 - val\_loss: 0.0020 Epoch 6/10

84/84 [==============================]

1.2412e04 - val\_loss: 0.0011 Epoch 7/10

84/84 [==============================]

04 - val\_loss: 7.1855e-04 Epoch 8/10

84/84 [==============================]

04 - val\_loss: 7.6043e-04 Epoch 9/10

84/84 [==============================]

04 - val\_loss: 9.7294e-04 Epoch 10/10

84/84 [==============================]

04 - val\_loss: 6.5660e-04

<keras.callbacks.History at 0x2505dbb7970>

# Model Evaluation

train\_predict=model.predict(X\_train) test\_predict=model.predict(X\_test)

167/167 [==============================] - 2s 3ms/step

90/90 [==============================] - 0s 3ms/step

train\_predict = scaler.inverse\_transform(train\_predict) test\_predict = scaler.inverse\_transform(test\_predict) import math from sklearn.metrics import mean\_squared\_error math.sqrt(mean\_squared\_error(y\_train,train\_predict)) 28.851078372476536

# Save The Model

from tensorflow.keras.models import load\_model model.save("Crude\_oil.h5")

# Test The Model

look\_back = 0

trainPredictPlot = np.empty\_like(data\_oil) trainPredictPlot[:,

:] = np.nan

trainPredictPlot[look\_back:len(train\_predict) + look\_back, :] = train\_predict

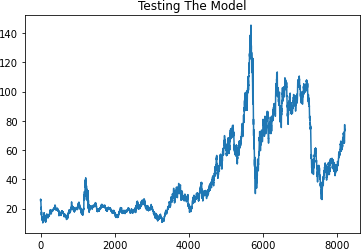
testPredictPlot = np.empty\_like(data\_oil) testPredictPlot[:,:]

= np.nan

testPredictPlot[len(train\_predict)+(look\_back\*2)+1: len(data\_oil)-1,

:] = test\_predict

plt.plot(scaler.inverse\_transform(data\_oil)) plt.plot(trainPredictPlot) plt.plot(testPredictPlot) plt.title("Testing The Model") plt.show()



len(test\_data) 2876

x\_input = test\_data[2866:].reshape(1,-1) x\_input.shape (1, 10)

temp\_input = list(x\_input) temp\_input = temp\_input[0].tolist() temp\_input [0.44172960165852215,

0.48111950244335855,

0.49726047682511476,

0.4679401747371539,

0.4729749740855915,

0.47119798608026064,

0.47341922108692425,

0.4649785280616022,

0.4703835332444839,

0.47149415074781587]

lst\_output = [] n\_steps = 10 i=0 **while**(i<10): **if**(len(temp\_input)>10): x\_input = np.array(temp\_input[1:]) print("{} day input {}".format(i,x\_input)) x\_input = x\_input.reshape(1,-1) x\_input = x\_input.reshape((1,n\_steps, 1))

yhat = model.predict(x\_input, verbose = 0) print("{} day output {}".format(i,yhat)) temp\_input.extend(yhat[0].tolist())

temp\_input = temp\_input[1:] lst\_output.extend(yhat.tolist()) i=i+1 **else**: x\_input = x\_input.reshape((1, n\_steps,1)) yhat = model.predict(x\_input, verbose = 0) print(yhat[0])

temp\_input.extend(yhat[0].tolist()) print(len(temp\_input)) lst\_output.extend(yhat.tolist()) i=i+1 [0.47125974]

11

1 day input [0.4811195 0.49726048 0.46794017 0.47297497 0.47119799

0.47341922

0.46497853 0.47038353 0.47149415 0.47125974]

1 day output [[0.47265336]]

2 day input [0.49726048 0.46794017 0.47297497 0.47119799 0.47341922

0.46497853

0.47038353 0.47149415 0.47125974 0.47265336]

2 day output [[0.4715367]]

3 day input [0.46794017 0.47297497 0.47119799 0.47341922 0.46497853

0.47038353

0.47149415 0.47125974 0.47265336 0.4715367 ]

3 day output [[0.46978694]]

4 day input [0.47297497 0.47119799 0.47341922 0.46497853 0.47038353

0.47149415

0.47125974 0.47265336 0.4715367 0.46978694]

4 day output [[0.4700314]]

5 day input [0.47119799 0.47341922 0.46497853 0.47038353 0.47149415

0.47125974

0.47265336 0.4715367 0.46978694 0.47003141]

5 day output [[0.4699089]]

6 day input [0.47341922 0.46497853 0.47038353 0.47149415 0.47125974

0.47265336

0.4715367 0.46978694 0.47003141 0.46990889]

6 day output [[0.46986535]]

7 day input [0.46497853 0.47038353 0.47149415 0.47125974 0.47265336

0.4715367

0.46978694 0.47003141 0.46990889 0.46986535]

7 day output [[0.46965963]]

8 day input [0.47038353 0.47149415 0.47125974 0.47265336 0.4715367

0.46978694

0.47003141 0.46990889 0.46986535 0.46965963]

8 day output [[0.4699126]]

9 day input [0.47149415 0.47125974 0.47265336 0.4715367 0.46978694

0.47003141

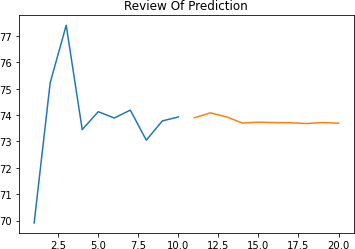
0.46990889 0.46986535 0.46965963 0.46991259]

9 day output [[0.46976325]]

day\_new = np.arange(1,11) day\_pred

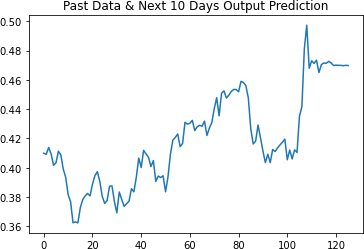
= np.arange(11,21) len(data\_oil) 8216

plt.plot(day\_new,scaler.inverse\_transform(data\_oil[8206:])) plt.title("Review Of Prediction") plt.plot(day\_pred,scaler.inverse\_transform(lst\_output)) plt.show()



df3 = data\_oil.tolist() df3.extend(lst\_output) plt.title("Past Data & Next 10 Days Output Prediction") plt.plot(df3[8100:])

[<matplotlib.lines.Line2D at 0x250696187c0>]



df3 = scaler.inverse\_transform(df3).tolist()

plt.title("Past Data & Next 10 Days Output Prediction After Reversing The Scaled Values") plt.plot(df3)

[<matplotlib.lines.Line2D at 0x25069758a30>]

