Crude Oil Price Prediction Using IBM Watson Studio

INTRODUCTION :

Prediction of crude oil prices has been a wide topic for ages. People use their intuition and lot of techniques to guess the prices of crude oil. It takes a lot of knowledge about the crude oil to accurately predict it. Predicting the crude oil price is very significant in various economic, political and industrial areas, both for crude oil importer and exporter countries [1]. Since crude oil is the most important strategic resource around the globe; it has become the crucial commodity for the world's economy. Thus, prediction of prices of crude oil has always been considered as a very exciting and challenging task which drew the curiosity of professionals, researchers and organizations all over the world [2]. Moreover, crude oil volatility has a critical impact on macroeconomic parameters such as such as inflation, unemployment, exchange rate, economic growth of countries whose economy rely heavily on crude oil export or import. Thus, crude oil price prediction can help governments of countries of the world in economic policy-making and make quick and operative economic decisions to hedge against probable risk in these economic parameters [3].

LITERATURE SURVEY :

Crude oil forecasting is an important topic in financial and economic studies. Many studies have been performed to forecast the prices of crude oil. After performing several tests, in 2005, Moshiri and Foroutan [4] concluded that future prices time series is stochastic and non-linear. They compared ARMA and GARCH techniques to ANN and found that ANN performed better for crude oil price forecasting. Kulkarni and Haidar [5] presented a model for forecasting crude oil spot price direction in the short-term, up to three days ahead based on multilayered feedforward neural network. They tested the relation between crude oil future prices and spot price. They found the evidence that future prices of crude oil contain new information about oil spot price detection

This Project mainly focuses on applying Neural Networks to predict the Crude Oil Price. This decision helps us to buy crude oil at the proper time. Time series analysis is the best option for this kind of prediction because we are using the Previous history of crude oil prices to predict

future crude oil. So we would be implementing RNN(Recurrent Neural Network) with LSTM(Long Short Term Memory) to achieve the task.

THEORITICAL ANALYSIS :

Software Requirements-
Python 3.9
Tensorflow
Keras
Flask
IBM Watson

EXPERIMENTAL INVESTIGATIONS :

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

```
In [3]: data.head()
```

Out[3]:

	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

```
In [4]: data.tail()
```

Out[4]:

	Date	Closing Value
8218	2018-07-03	74.19
8219	2018-07-04	NaN
8220	2018-07-05	73.05
8221	2018-07-06	73.78
8222	2018-07-09	73.93

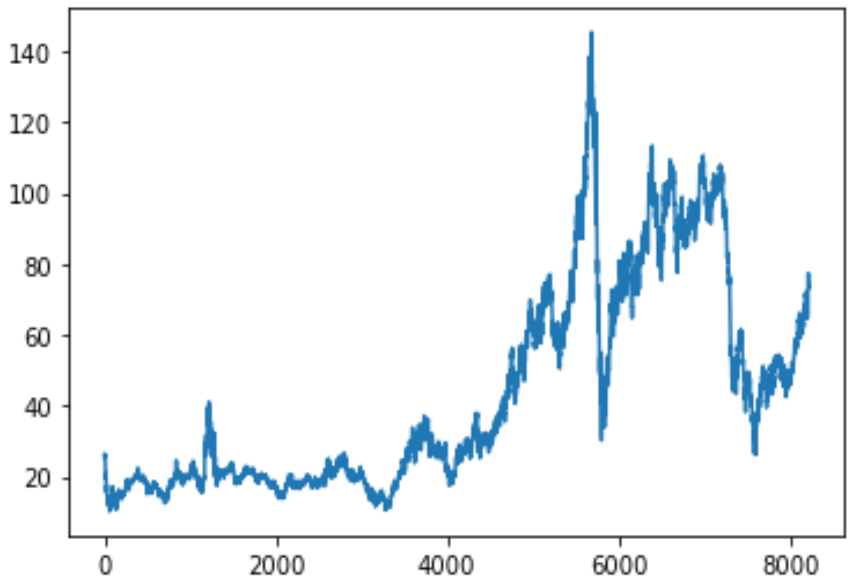
In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8223 entries, 0 to 8222
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Date           8223 non-null   datetime64[ns]
 1   Closing Value  8216 non-null   float64
dtypes: datetime64[ns](1), float64(1)
memory usage: 128.6 KB
```

In [6]:
```
data.describe()
```

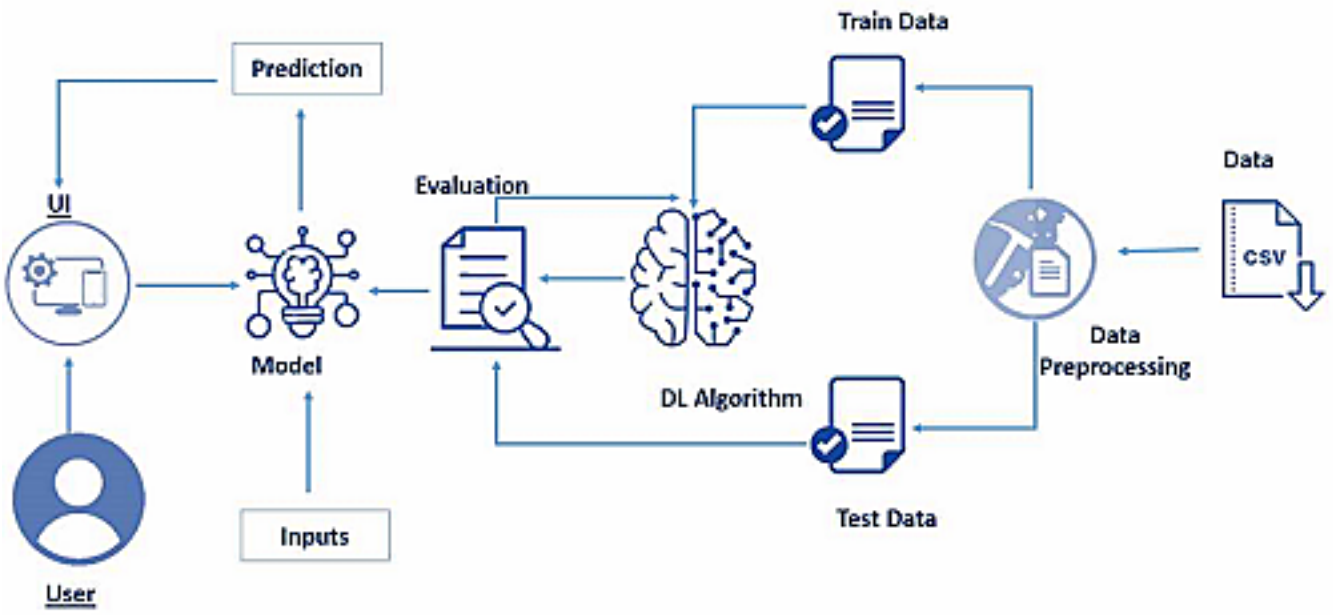
Out[6]:

	Closing Value
count	8216.000000
mean	43.492139
std	29.616804
min	10.250000
25%	19.577500
50%	29.610000
75%	63.402500
max	145.310000



Graph plot of OIL data

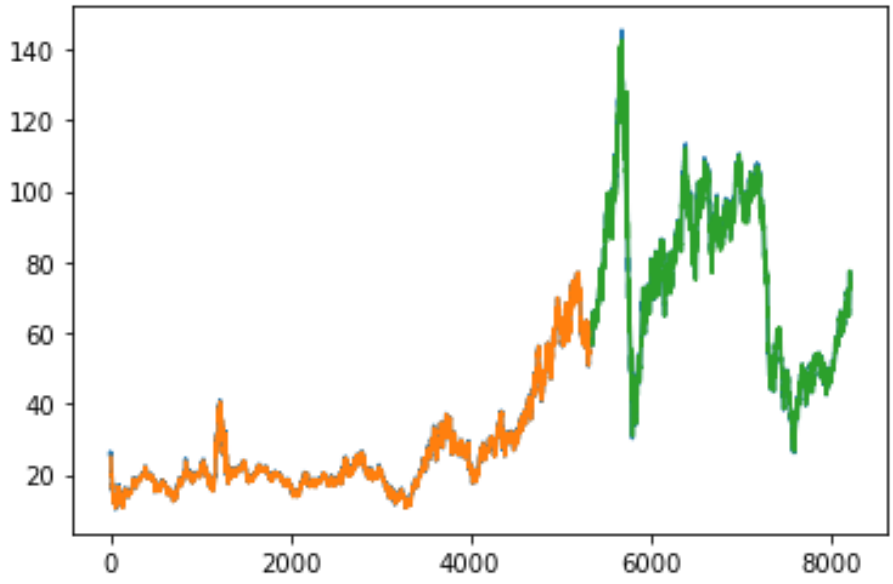
FLOWCHART :

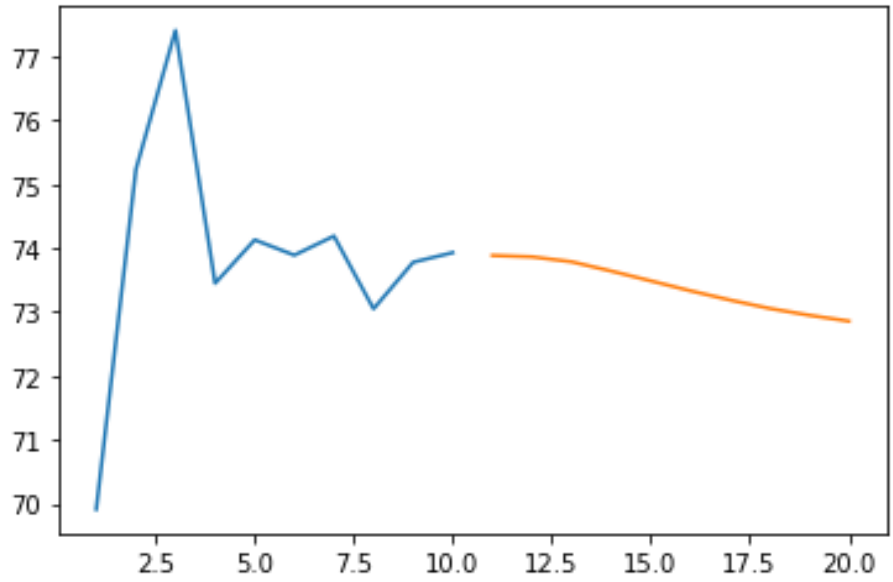


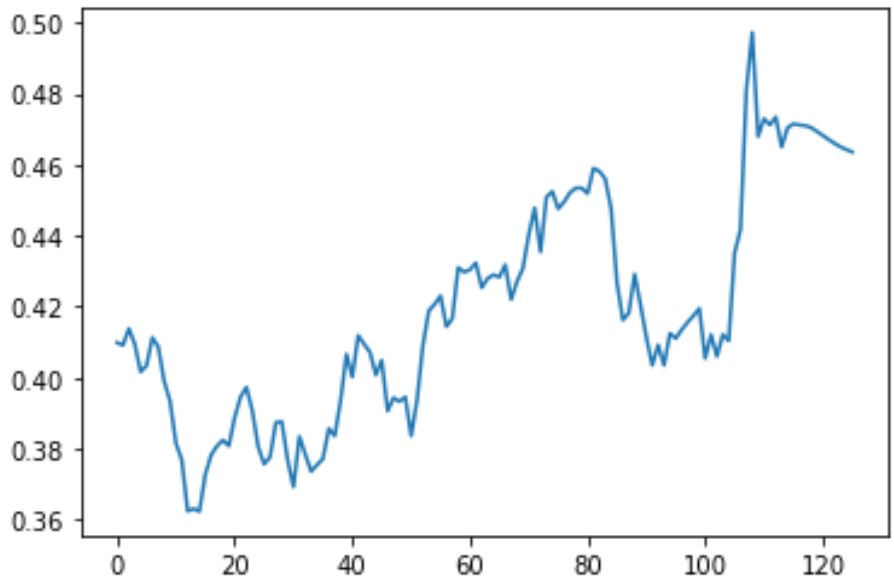
RESULT

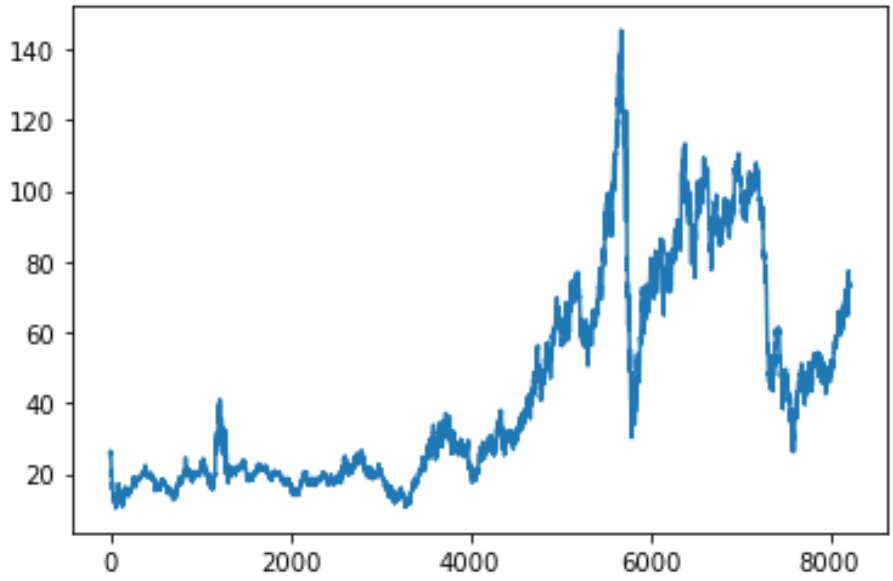
```
[0.4711648]
11
1 day input [0.4811195  0.49726048 0.46794017 0.47297497 0.47119799 0.47341922
 0.46497853 0.47038353 0.47149415 0.47116479]
1 day output [[0.4710071]]
2 day input [0.49726048 0.46794017 0.47297497 0.47119799 0.47341922 0.46497853
 0.47038353 0.47149415 0.47116479 0.47100711]
2 day output [[0.47043177]]
3 day input [0.46794017 0.47297497 0.47119799 0.47341922 0.46497853 0.47038353
 0.47149415 0.47116479 0.47100711 0.47043177]
3 day output [[0.46935746]]
4 day input [0.47297497 0.47119799 0.47341922 0.46497853 0.47038353 0.47149415
 0.47116479 0.47100711 0.47043177 0.46935746]
4 day output [[0.46821904]]
5 day input [0.47119799 0.47341922 0.46497853 0.47038353 0.47149415 0.47116479
 0.47100711 0.47043177 0.46935746 0.46821904]
5 day output [[0.46706894]]
6 day input [0.47341922 0.46497853 0.47038353 0.47149415 0.47116479 0.47100711
 0.47043177 0.46935746 0.46821904 0.46706894]
6 day output [[0.4659978]]
7 day input [0.46497853 0.47038353 0.47149415 0.47116479 0.47100711 0.47043177
 0.46935746 0.46821904 0.46706894 0.46599779]
7 day output [[0.46502292]]
8 day input [0.47038353 0.47149415 0.47116479 0.47100711 0.47043177 0.46935746
 0.46821904 0.46706894 0.46599779 0.46502292]
8 day output [[0.4642337]]
9 day input [0.47149415 0.47116479 0.47100711 0.47043177 0.46935746 0.46821904
 0.46706894 0.46599779 0.46502292 0.4642337 ]
9 day output [[0.4635509]]
[[0.4711647927761078], [0.4710071086883545], [0.4704317485466003], [0.4693574607372284], [0.4682190418243408], [0.467068940401
07727], [0.4659977853298187], [0.4650229215621948], [0.4642336964607239], [0.4635508954524994]]
```

Prediction of the prices for 10 days


140
120
100
80
60
40
20
0
2000
4000
6000
8000

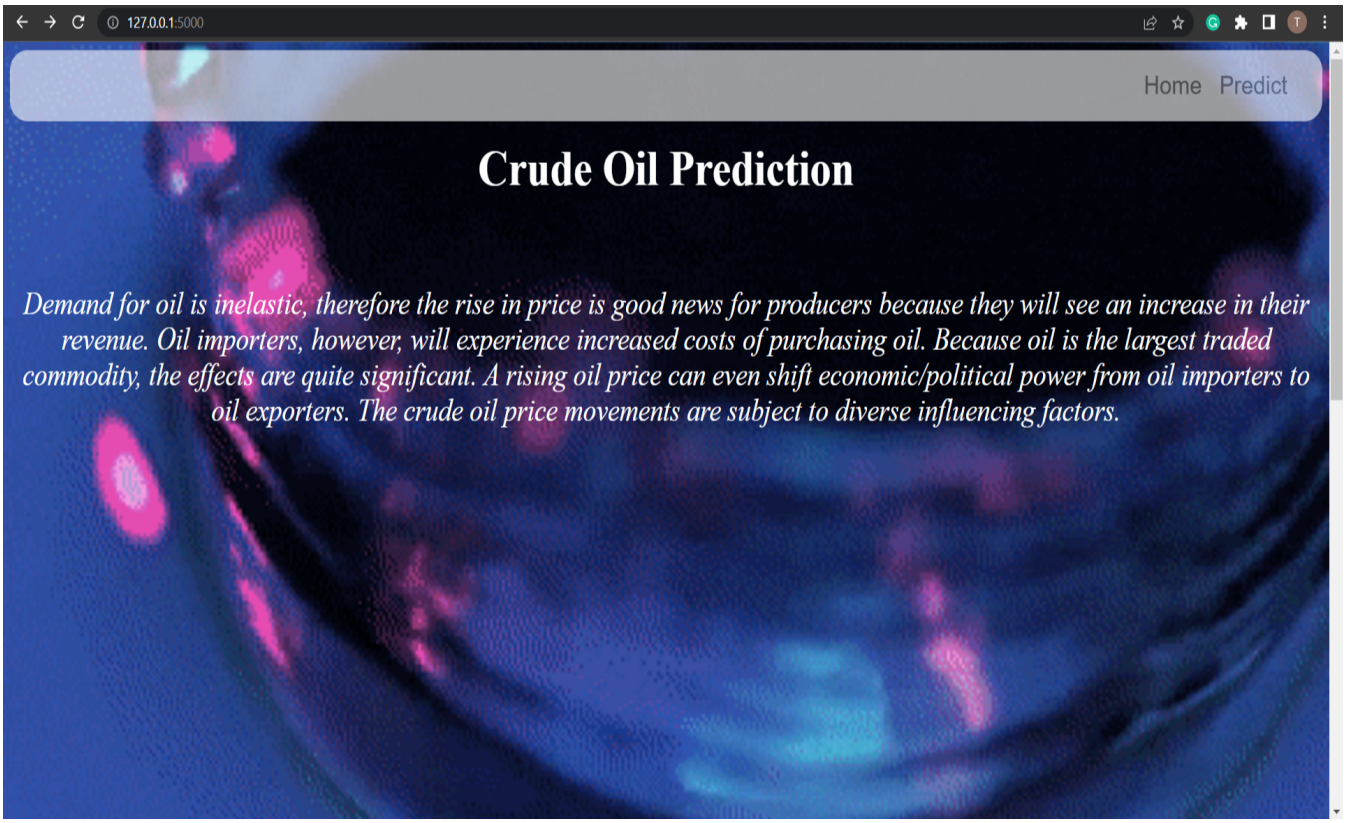

77
76
75
74
73
72
71
70
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


0.50
0.48
0.46
0.44
0.42
0.40
0.38
0.36
0
20
40
60
80
100
120

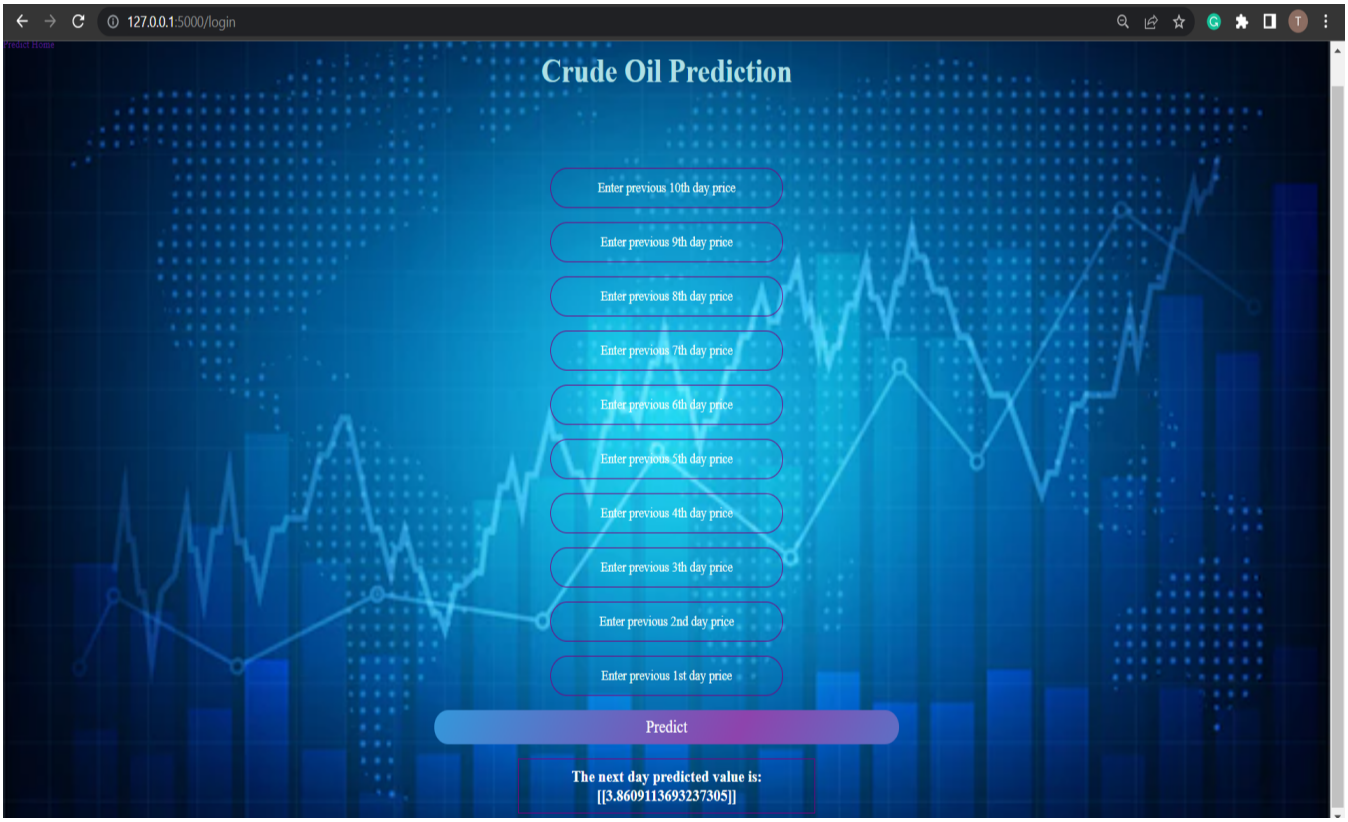


Plot for the prices of crude oil

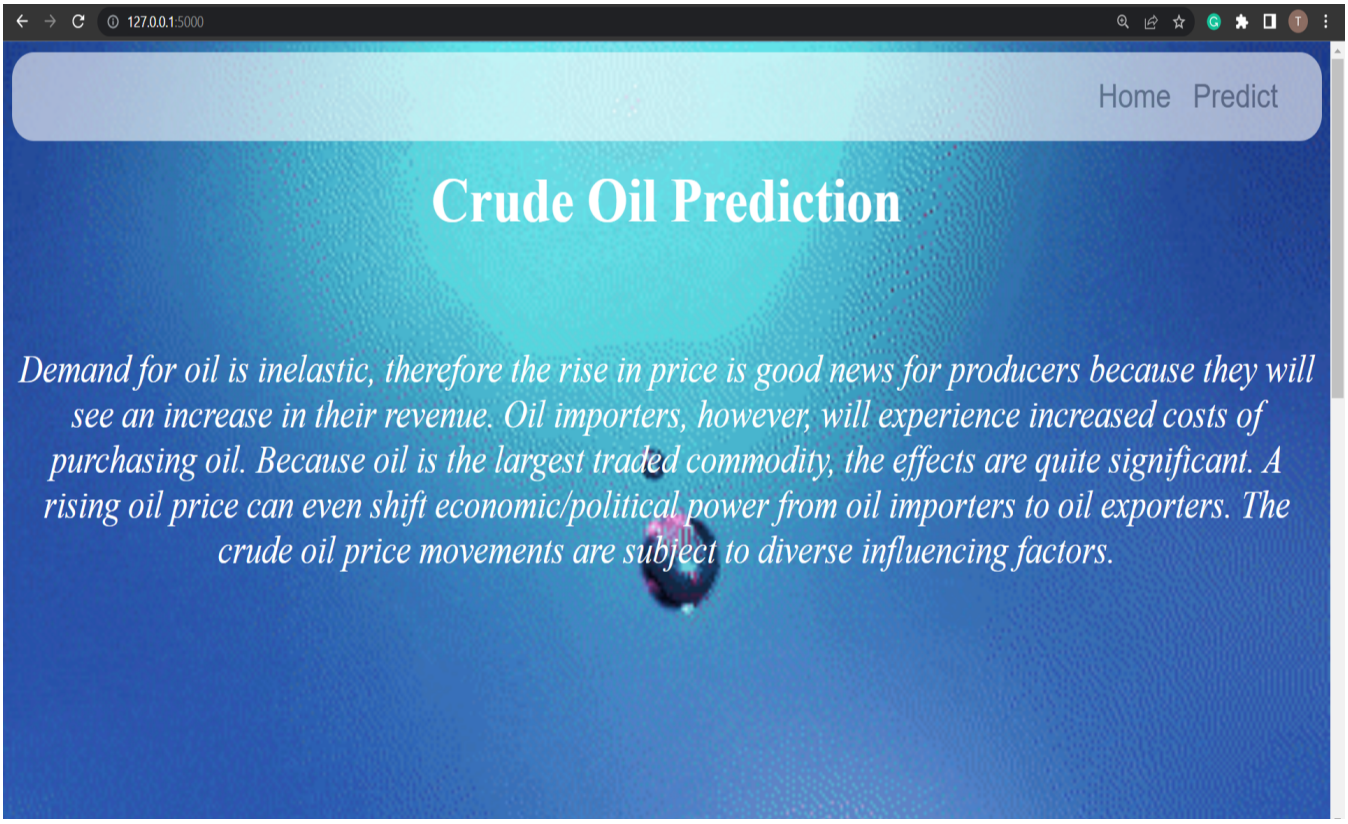
Flask App




127.0.0.1:5000/predict
Predict Home
Crude Oil Prediction
74.99
73.66
74.56
74.89
75
74.62
76.98
77
78
77.56
Predict


127.0.0.1:5000/login
Predict Home
Crude Oil Prediction
Enter previous 10th day price
Enter previous 9th day price
Enter previous 8th day price
Enter previous 7th day price
Enter previous 6th day price
Enter previous 5th day price
Enter previous 4th day price
Enter previous 3th day price
Enter previous 2nd day price
Enter previous 1st day price
Predict
The next day predicted value is:
[[3.8609113693237305]]

Flask App with IBM Integration



ADVANTAGES & DISADVANTAGES

Advantages

- Can predict the prices for next 10 days
- Deployed through IBM Watson
- Can be easily integrated

Disadvantages

- Doesn't account for market conditions and political factors while predicting
- Doesn't monitor real time prices

APPLICATIONS

Crude oil price forecasting plays a significant role in world economy and its accurate prediction has significant benefits for the economic conditions of a country. In this direction, an effort has been in this paper. This project can be applied to predict the prices of crude oil.

CONCLUSION

Oil demand is inelastic, therefore the rise in price is good news for producers because they will see an increase in their revenue. Oil importers, however, will experience increased costs of purchasing oil. Because oil is the largest traded commodity, the effects are quite significant. A rising oil price can even shift economic/political power from oil importers to oil exporters. The crude oil price movements are subject to diverse influencing factors.

FUTURE SCOPE

This project can be further developed to create a more robust and accurate system which can monitor and predict the prices of Crude Oil in real time.

BIBILOGRAPHY

[1] Mohammad Reza Mahdiani and Ehsan Khamehchi, “A modified neural network model for predicting the crude oil price”, Intellectual Economics, vol. 10, no. 2, pp. 71-77, Aug. 2016.
[2] Manel Hamdi and Chaker Aloui, "Forecasting Crude Oil Price Using Artificial Neural Networks: A Literature Survey," Economics Bulletin, AccessEcon, vol. 35, no. 2, pp. 1339-1359, 2015.
[3] Yu Runfang, Du Jiangze and Liu Xiaotao, “Improved Forecast Ability of Oil Market Volatility Based on combined Markov Switching and GARCH-cla
[4] S. Moshiri, and F. Foroutan, “Forecasting nonlinear crude oil futures prices,” The Energy Journal vol. 27, pp. 81-95, 2005.
[5] Siddhi Vinayak Kulkarni and Imad Haidar, Forecasting Model for Crude Oil Price Using Artificial Neural Networks and Commodity Futures Prices. International Journal of Computer Science and Information Security, vol. 2, no.1, June 2009.

APPENDIX

Model training

```
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]   ###i=0, 0,1,2,3-----99   100
            dataX.append(a)
            dataY.append(dataset[i + time_step, 0])
      return np.array(dataX), np.array(dataY)

model=Sequential()
model.add(LSTM(50,return_sequences=True,input_shape=(10,1)))
model.add(LSTM(50,return_sequences=True))
model.add(LSTM(50))
model.add(Dense(1))
model.compile(loss='mean_squared_error',optimizer='adam')
model.fit(X_train,y_train,validation_data=(X_test,ytest),epochs=50,batch_size=
64,verbose=1)

from numpy import array

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

print(lst_output)
```

Flask app.py

```
import numpy as np
from flask import Flask,render_template,request
from tensorflow.keras.models import load_model

app = Flask(__name__)
model = load_model('crude_oil.h5',)

@app.route('/')
def home() :
    return render_template("index.html")
@app.route('/about')
def home1() :
    return render_template("index.html")
@app.route('/predict')
def home2() :
    return render_template("web.html")

@app.route('/login',methods = ['POST'])
def login() :

    a=request.form['year1']
    b=request.form['year2']
    c=request.form['year3']
    d=request.form['year4']
    e=request.form['year5']
    f=request.form['year6']
    g=request.form['year7']
    h=request.form['year8']
    i=request.form['year9']
    j=request.form['year10']
    x_input = [a,b,c,d,e,f,g,h,i,j]
    for i in range(0, len(x_input)):
        x_input[i] = float(x_input[i])
    print(x_input)
    x_input=np.array(x_input).reshape(1,-1)
    temp_input=list(x_input)
```

```
    temp_input=temp_input[0].tolist()
    lst_output=[]
    n_steps=10
    i=0
    while(i<1):

        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

    print(lst_output)
    return render_template("web.html",showcase = 'The next day predicted
value is:'+str(lst_output))

if __name__ == '__main__' :
    app.run(debug = True,port=5000)
```

IBM flaskintegration.py

```
import requests
import numpy as np
from flask import Flask, request, jsonify,render_template
API_KEY = "tP0KjjEf6DX0Co2H4Ifbi88zwKPKVEQKQX-ga1GkbYYR"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey": API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-
type:apikey'})
mltoken = token_response.json()["access_token"]
```

```
header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' +
mltoken}
app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/about')
def home1():
    return render_template('index.html')
@app.route('/predict')
def home2():
    return render_template('web.html')

@app.route('/login',methods = ['POST'])
def login():
    a=request.form['year1']
    b=request.form['year2']
    c=request.form['year3']
    d=request.form['year4']
    e=request.form['year5']
    f=request.form['year6']
    g=request.form['year7']
    h=request.form['year8']
    i=request.form['year9']
    j=request.form['year10']
    x_input = [a,b,c,d,e,f,g,h,i,j]

    for i in range(0, len(x_input)):
        x_input[i] = float(x_input[i])
    x_input=np.array(x_input).reshape(1,-1)
    n_steps=10
    i=0
    while(i<1):
        x_input = x_input.reshape((n_steps,1))
        payload_scoring = {"input_data": [{"fields": [["Closing Value"]],
"values": [x_input.tolist()]}]}
        response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/8c1f578c-e1b5-4d71-bec5-
c60cc7f7b61c/predictions?version=2021-11-06', json=payload_scoring,
headers={'Authorization': 'Bearer ' + mltoken})
        yhat =response_scoring.json()
        i=i+1
        yhat=yhat['predictions'][0]['values'][0][0]
        return render_template('web.html',showcase = 'The next day predicted
```

```
value is : '+str(yhat))
if __name__ =="__main__":
    app.run(debug = True)
    app.run(host = '0.0.0.0', port = 5000)
```

IBM Scoring endpoint.py

```
import requests
import numpy as np
API_KEY = "tP0KjjEf6DX0Co2H4Ifbi88zwKPKVEQKQX-ga1GkbYYR"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey": API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-
type:apikey'})
mltoken = token_response.json()["access_token"]
header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' +
mltoken}
a = 11
b = 22
c = 33
d = 44
e = 55
f = 66
g = 77
h = 88
i = 99
j = 84
k=int(j)
#print()
count=0
while(k!=0):
    k=k/10
    k=int(k)
    count = count+1
x_input = [a,b,c,d,e,f,g,h,i,j]
for i in range(0, len(x_input)):
    x_input[i] = float(x_input[i])
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler(feature_range=(0,1))
x_input=scaler.fit_transform(np.array(x_input).reshape(-1,1))
temp_input=list(x_input)
temp_input=temp_input[0].tolist()
```

```
lst_output=[]
n_steps=10
i=0
while(i<1):
    x_input = x_input.reshape((n_steps,1))
    payload_scoring = {"input_data": [{"fields": [["Closing value"]],
"values": [x_input.tolist()]}]}
    response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/8c1f578c-e1b5-4d71-bec5-
c60cc7f7b61c/predictions?version=2021-11-06', json=payload_scoring,
headers={'Authorization': 'Bearer ' + mltoken})
    print("Scoring response")
    predictions = response_scoring.json()
    print(predictions)
```

IBM Training file

```
from ibm_watson_machine_learning import APIClient
wml_credentials = {
                   "url": "https://us-south.ml.cloud.ibm.com",
                   "apikey":"tP0KjjEf6DX0Co2H4Ifbi88zwKPKVEQKQX-ga1GkbYYR"
}
client = APIClient(wml_credentials)
def guid_from_space_name(client,space_name):
    space = client.spaces.get_details()
    return(next(item for item in space['resources'] if item['entity']["name"]
== space_name) ['metadata']['id'])
space_uid = guid_from_space_name(client,'Models')
print("Space UID = " +space_uid)
client.set.default_space(space_uid)
software_spec_uid =
client.software_specifications.get_uid_by_name("default_py3.7_opence")
software_spec_uid
!tar -zcvf Crudeoil1.tgz crude_oil.h5
model_details =
client.repository.store_model(model='Crudeoil1.tgz',meta_props={
    client.repository.ModelMetaNames.NAME:"Crudeprice",
    client.repository.ModelMetaNames.TYPE:"Tensorflow_2.4",
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid}
                                            )
model_id = client.repository.get_model_uid(model_details)
```