Delivery of Sprint-4

Date	**12 November 2022**
Team ID	**PNT2022TMID21805**
Project Name	**Crude Oil Price Prediction**

REGISTER FOR IBM CLOUD

The process of registering for IBM Cloud account for all the members of the team was successfully completed.

TRAIN THE ML MODEL ON IBM

 # # DATA PREPROCESSING

 # # Importing The Libraries

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

 # # Importing The Dataset

```
import os, types import
pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook. cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='DniKDOiBzjciVYi0lFC0XLbDwNMPgaL7RkoNT-y7NhQ2',
```

```
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
config=Config(signature_version='oauth'),     endpoint_url='https://s3.private.us.cloud-object-
storage.appdomain.cloud')

bucket = 'crudeoilpricepredictionusinglstm-donotdelete-pr-cscxajvuz8ywfj' object_key
= 'Crude Oil Prices Daily.xlsx'

body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']

data = pd.read_excel(body.read()) data.head()

# # Handling Missing Values

data.isnull().any()

data.isnull().sum()

data.dropna(axis=0,inplace=True)
data.isnull().sum()

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

 # # Feature Scaling

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

 # # Data Visualization

```
plt.title('Crude OIl Price') plt.plot(data_oil)
```

 # # Splitting Data Into Train and Test

```
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
```

 # # Creating A Dataset With Sliding Windows

```
import numpy 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, ytest = create_dataset(test_data, time_step)

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)
```

 # # 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()
```

 # # Configure The Learning Process

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

 # # Fitting The Model

```
model.fit(X_train, y_train, validation_data = (X_test, ytest), epochs = 10, batch_size = 64,
verbose = 1)

train_predict=model.predict(X_train)
test_predict=model.predict(X_test)
```

```
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))
```

 # # Save The Model

```
from tensorflow.keras.models import load_model model.save("Crude_oil.h5")

get_ipython().system('tar -zcvf crude-oil-predict-model.tgz Crude_oil.h5')
```

 # # Training the model on IBM cloud

```
get_ipython().system('pip install ibm_watson_machine_learning')

from ibm_watson_machine_learning import APIClient wml_credentials
= {
   "url": "https://us-south.ml.cloud.ibm.com",
   "apikey": "uVEty-CB4dYcccQ_Jq9V-atVXmL1dByE_wiDm95lcyTQ"
}

client = APIClient(wml_credentials)

def guid_from_space_name(client, NewSpace):
   space = client.spaces.get_details()
   return(next(item for item in space['resources'] if item['entity']["name"] ==
NewSpace)['metadata']['id'])

space_uid = guid_from_space_name(client, 'NewSpace') print("Space
UID = " + space_uid)

client.set.default_space(space_uid)

client.software_specifications.list()

software_spec_id =  client.software_specifications.get_id_by_name('tensorflow_rt22.1py3.9')
print(software_spec_id)

model.save('crude.h5')

get_ipython().system('tar -zcvf crude-oil.tgz Crude.h5')
```

```
software_space_uid =
client.software_specifications.get_uid_by_name('tensorflow_rt22.1py3.9')
software_space_uid

model_details = client.repository.store_model(model='crude.tgz',meta_props={
client.repository.ModelMetaNames.NAME:"crude_oil_model",
client.repository.ModelMetaNames.TYPE:"tensorflow_2.7",
client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_id }
                                )
model_id = client.repository.get_model_uid(model_details) model_id

client.repository.download(model_id,'crude_oil_model.tar.gb')
```

INTEGRATE FLASK WITH SCORING END POINT

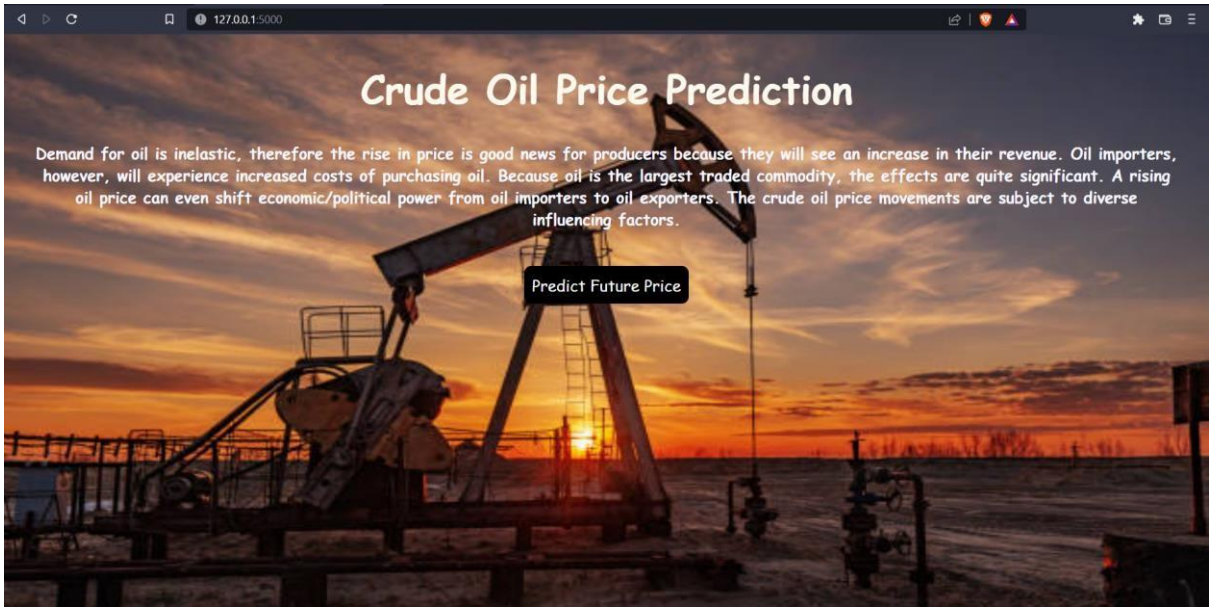
App.py

```
from flask import Flask,render_template,request,redirect
import pandas as pd import numpy as np
from flask import Flask, render_template, Response, request import
pickle
from sklearn.preprocessing import LabelEncoder import
requests

# NOTE: you must manually set API_KEY below using information retrieved from your IBM
Cloud account.
API_KEY = "uVEty-CB4dYcccQ_Jq9V-atVXmL1dByE_wiDm95lcyTQ"
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('/',methods=["GET"]) def
index():     return
render_template('index.html')
@app.route('/predict',methods=["POS
T","GET"]) def predict():     if
request.method == "POST":
string = request.form['val']         string
= string.split(',')
```

```
        temp_input = [eval(i) for i in string]

        x_input = np.zeros(shape=(1, 10))
        x_input.shape

        lst_output = []
n_steps = 10
i=0         while(i<10):
            if(len(temp_input)>10):
                x_input = np.array(temp_input[1:])
x_input = x_input.reshape(1,-1)                 x_input =
x_input.reshape((1,n_steps, 1))                 yhat =
model.predict(x_input, verbose = 0)
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)
temp_input.extend(yhat[0].tolist())
lst_output.extend(yhat.tolist())                i=i+1


            # NOTE: manually define and pass the array(s) of values to be scored in the next line
payload_scoring = {"input_data": [{    "values": [[x_input]]    }]}

            response_scoring =
requests.post('https://ussouth.ml.cloud.ibm.com/ml/v4/deployments/7f67cbed-6222-413b-
9901b2a72807ac82/predictions?version=2022-10-30', json=payload_scoring,
headers={'Authorization': 'Bearer ' + mltoken})
predictions = response_scoring.json()
            print(response_scoring.json())

            val = lst_output[9]
            return render_template('web.html' , prediction = val)


    if request.method=="GET":
        return render_template('web.html')

if __name__=="__main__":
    model = load_model('C:/Users/rkara/IBM/Sprint - 4/Crude_oil.tar.gz')
app.run(debug=True)
```

OUTPUT:

Home Page



Prediction Page

If no values are entered a error message is displayed

Entering the crude oil price for ten days

The predicted result is displayed below