FINALCODE

TeamID	PNT2022TMID47290
ProjectName	CrudeOilPricePrediction

SourceCode

Buildingthemodel:

```
import numpy as
npimportpandasasp
d
importmatplotlib.pyplotasplt

data = pd.read_excel("Crude Oil Prices
Daily.xlsx")data.head()

data.isnull().any()

data.isnull().sum()

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

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

from sklearn.preprocessing import
MinMaxScalerscaler=MinMaxScaler(feature_ran
ge=(0,1) )
data_oil=scaler.fit_transform(np.array(data_oil).reshape(-1,1))

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

training_size =
int(len(data_oil)*0.65)test_size=len(d
ata_oil)-training_size
```

```
train_data,test_data=data_oil[0:training_size,:],data_oil[training_size:len(data_oil),:1]
```

```
training_size,

test_sizetrain_data.sha

pe

import numpy
def create_dataset(dataset,
    time_step=1):dataX,dataY =[], []
    foriinrange(len(dataset)-time_step-1):a
        = dataset[i:(i+time_step),
        0]dataX.append(a)dataY.append(datas
        et[i+time_step,0])
    returnnp.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_s
tep)

print(X_train.shape),

print(y_train.shape)print(X_test.shape),

print(ytest.shape)X_train

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
Sequentialfromtensorflow.keras.layersimport
Dense
from tensorflow.keras.layers import
```

```
LSTMmodel=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.summary()

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

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=sc
aler.inverse_transform(test_predict)

importmath
from sklearn.metrics import
mean_squared_errormath.sqrt(mean_squared_error(y
_train,train_predict))

from tensorflow.keras.models import
load_modelmodel.save("Crude_oil.h5")

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_predictplt.plot(scaler.inverse_transform(data_oil))
```

```
plt.plot(trainPredictPlot)plt.pl
ot(testPredictPlot)plt.title("Te
sting The Model")plt.show()

len(test_data)

x_input = test_data[2866:].reshape(1,-
1)x_input.shape

temp_input =
list(x_input)temp_input =
temp_input[0].tolist()temp_input

lst_output=[]n_
steps =
10i=0while(i<1
0):
    if(len(temp_input)>10):
        x_input =
        np.array(temp_input[1:])print("{} day
        input
        {}".format(i,x_input))x_input=x_input.re
        shape(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,verb
```

ose=0)

```
    print(yhat[0])temp_input.extend(y
    hat[0].tolist())print(len(temp_input
    ))lst_output.extend(yhat.tolist())i=i
    +1

day_new =
np.arange(1,11)day_pred=np.aran
ge(11,21)

len(data_oil)

plt.plot(day_new,scaler.inverse_transform(data_oil[8206:]))
plt.title("Review Of
Prediction")plt.plot(day_pred,scaler.inverse_transform(lst_o
utput))plt.show()

df3 =
data_oil.tolist()df3.ext
end(lst_output)
plt.title("Past Data & Next 10 Days Output
Prediction")plt.plot(df3[8100:])

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

DeployingonIBMCloud:

```
get_ipython().system('pipinstallibm_watson_machine_learning')

from ibm_watson_machine_learning import
APIClientwml_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("SpaceUID=" +space_uid)

client.set.default_space(space_uid)

client.software_specifications.list()

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

model.save('crude.h5')

get_ipython().system('tar -zcvfcrude-oil.tgzCrude.h5')

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

model_details =
client.repository.store_model(model='crude.tgz',meta_props={client.repository.M
odelMetaNames.NAME:"crude_oil_model",client.repository.ModelMetaNames.T
YPE:"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')
```

INTEGRATEFLASKWITH SCORING ENDPOINT

App.py

```
from flask import
Flask,render_template,request,redirectimportpandas
aspd
importnumpy as np
from flask import Flask, render_template, Response,
requestimport pickle
from sklearn.preprocessing import
LabelEncoderimportrequests

# NOTE: you must manually set API_KEY below using information retrieved from
yourIBMCloud 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"])
defindex():
   returnrender_template('index.html')



@app.route('/predict',methods=["POST","GET"])
defpredict():
   if request.method ==
      "POST":string=
```

request.form['val']

```
    string= string.split(',')
    temp_input=[eval(i) for iin string]

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

    lst_output =
    []n_steps =
    10i=0while(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_ou
            tput.extend(yhat.tolist())
            i=i+1



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

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

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



    ifrequest.method=="GET":
        returnrender_template('web.html')

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

INDEX.HTML

```
<!DOCTYPEhtml>
<head>
    <title>CrudeOilPricePrediction</title>
    <linkrel="stylesheet"href="{{url_for('static',filename='css/index.css')}}">
</head>
<body>
    <h1>CrudeOilPricePrediction</h1>
    <p> Demand for oil is inelastic, therefore the rise in price is good
     newsfor producers because they will see an increase in their revenue.
     Oilimporters, however, will experience increased costs of purchasing
     oil.Because oil is the largest traded commodity, the effects are
     quitesignificant.Arising oilpricecaneven shift economic/political
     power from oil importers to oil exporters. The crude oil price
     movementsaresubject to diverseinfluencing factors.
    </p><br><br>
    <ahref="{{url_for('predict')}}">
    PredictFuturePrice</a>
</body>
```

WEB.HTML

```
<!DOCTYPEhtml>
<head>
    <title>CrudeOilPricePrediction</title>
    <linkrel="stylesheet"href="{{url_for('static',filename='css/web.css')}}">
</head>
<body>
    <h1>
    CrudeOil PricePrediction </h1>
    <formaction="/predict"method="POST"enctype="multipart/form-data">
        <inputtype="text"name="val"placeholder="Enterthecrudeoilprice forfirst10days"
>
         <br><br><br>
        <inputtype="submit"/>
    </form><br><br>
    <div>
        {{prediction}}
    </div>

</body>
```