BANNARI AMMAN INSTITUTE OF TECHNOLOGY

SATHYAMANGALAM-638403

CRUDE OIL PRICE PREDICTION

Team Id: PNT2022TMID01610

Team Members Name:

SELVABALA V,

PARTHASARATHY VENKIDUSAMY M,

PRAVEEN S,

HARI BHARATHI S

Project Report Format

1. **INTRODUCTION**
   1.1 Project Overview
   1.2 Purpose
2. **LITERATURE SURVEY**
   2.1 Existing problem
   2.2 References
   2.3 Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
   3.1 Empathy Map Canvas
   3.2 Ideation & Brainstorming
   3.3 Proposed Solution
   3.4 Problem Solution fit
4. **REQUIREMENT ANALYSIS**
   4.1 Functional requirement
   4.2 Non-Functional requirements
5. **PROJECT DESIGN**
   5.1 Data Flow Diagrams
   5.2 Solution & Technical Architecture
   5.3 User Stories
6. **PROJECT PLANNING & SCHEDULING**
   6.1 Sprint Planning & Estimation
   6.2 Sprint Delivery Schedule
   6.3 Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
   7.1 Feature 1
   7.2 Feature 2
   7.3 Database Schema (if Applicable)
8. **TESTING**
   8.1 Test Cases
   8.2 User Acceptance Testing
9. **RESULTS**
   9.1 Performance Metrics
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**

Source Code

GitHub & Project Demo Link

1. INTRODUCTION

1.1 PROJECT OVERREVIEW

Crude oil is amongst the most important resources in today's world, it is the chief fuel and its cost has a direct effect on the global habitat, our economy and oil exploration, exploitation and other activities. Prediction of oil prices has become the need of the hour; it is a boon to many large and small industries, individuals, and the government. The evaporative nature of crude oil, its price prediction becomes extremely difficult and it is hard to be precise with the same. Several different factors that affect crude oil prices. The main advantage of this crude oil price prediction using artificial intelligence is continuously captures the unstable pattern of the crude oil prices which have been incorporated by finding out the optimal lag and number of the delay effect that controls the prices of crude oil.

1.2 PURPOSE

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.

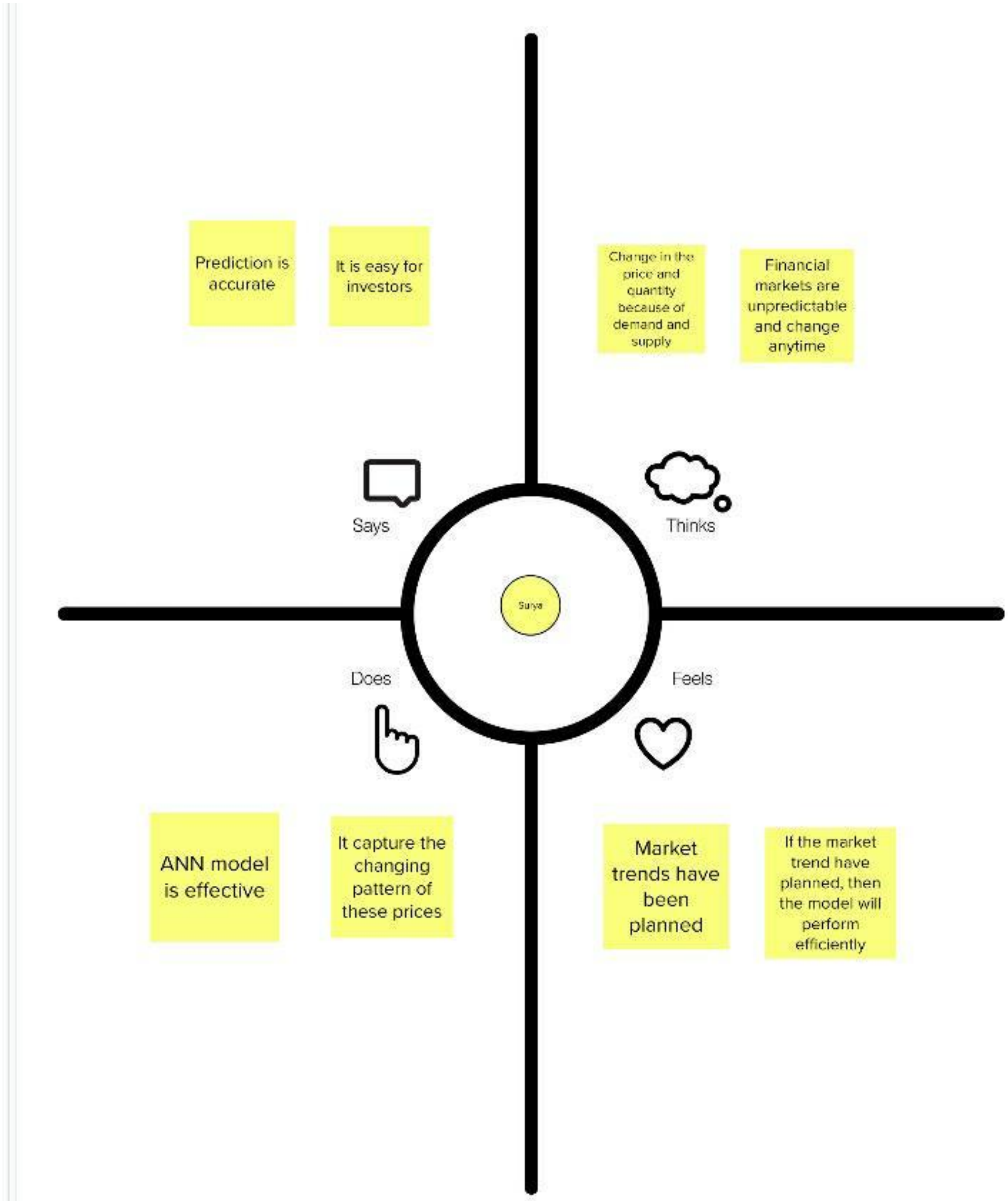
This Project mainly focuses on applying Neural Networks to predict the Crude Oil Price. This decision helps us to buy crude oil at the proper time. Time series analysis is the best Option for this kind of prediction because we are using the previous history of crude oil prices to predict future crude oil. So we would be implementing RNN (Recurrent Neural Network) with LSTM (Long Short Term Memory) to achieve the task.

2. LITERATURE SURVEY

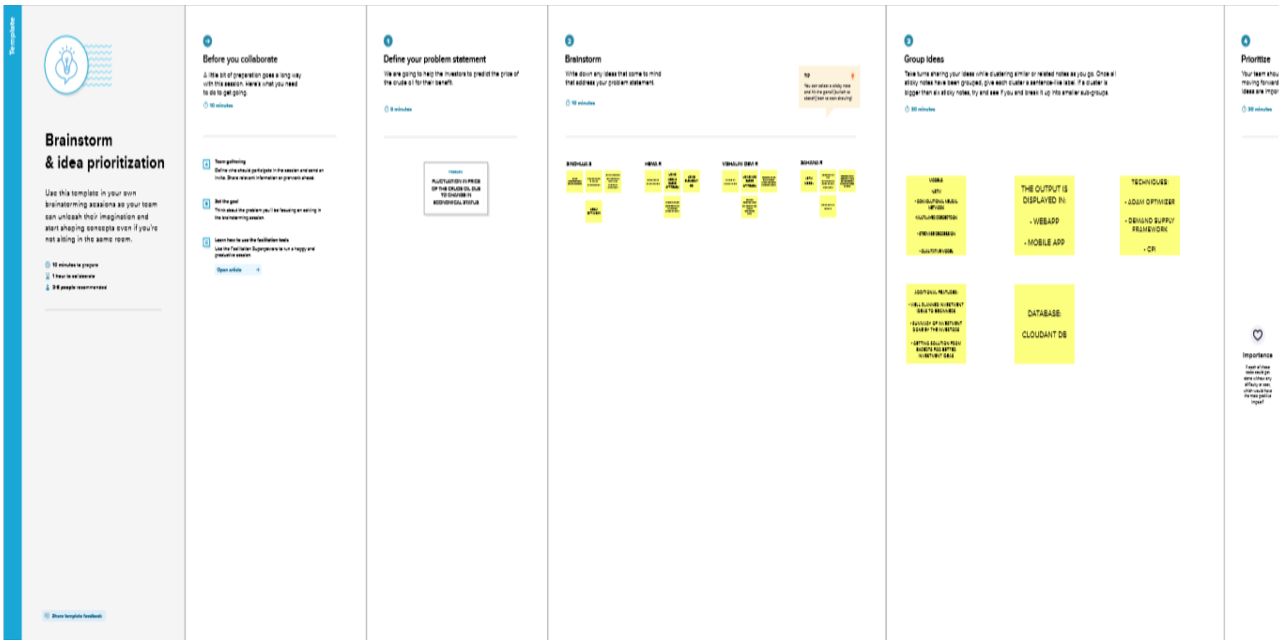
S.NO	Title	Authors	Publication Date	Methodology	Merits	Demerits
1	Brent crude oil price forecast utilizing Deep Neural Network Architectures	Amir Daneshvar and Maryam Ebrahimi	05 May 2022	Artificial Neural Network, Deep Learning	The LSTM layers results in more accurate result.	Crude oil price signals exhibit highly nonlinear and complex behavior.
2.	Crude oil prices and volatility prediction by a hybrid model based on kernel extreme learning machine	Hongli Niu and Yazhi Zhao	17 September 2021	VMD-KELM	The VMD-KELM model shows a more powerful ability than other models in improving the precision of forecasting crude oil volatility.	-
3.	Crude oil price prediction using ANN	Nalini Gupta and Shobhit Nigam	January 2020	Artificial Neural Network	ANN model is effective. This capture the changing pattern of prices. Prediction is accurate.	Market trends have to be planned, then the ANN model will perform.
4.	Crude oil price prediction using complex network and deep learning algorithms	Makumbonori Bristone, Rajesh Prasad, Adamu Ali Abubakar	19 June 2019	Artificial Neural Network, Deep Learning	The appropriate number of LSTM layers can effectively improve the model.	The other factors that affect the crude oil price volatilities such as economic growth, exchange rate demand are not considered.
5.	Daily crude oil price forecasting using Hybridizing wavelet and Artificial Neural Network Model	Ani Shabri and Ruhaidah Samsudin	16 July 2014	Artificial Neural Network	The hybrid model showed a great improvement	-

					in crude oil price modeling and produced better forecasts than ANN model alone.	
6.	Machine Learning Approach for crude oil price prediction with Artificial Neural Networks-Quantitative (ANN-Q) model	Abdullah	-	Artificial Neural Network	Returns function had successfully proved to cleanse and uniform the data from errors and noises hence, the crisp prediction result.	
7.	A novel look back N feature approach towards prediction of crude oil price	Rudra Kalyan Nayak	-	ARIMA, LBNF Algorithm	Attained better training and accuracy by shifting the dataset into n class problem and more scope to classifier.	-

3. IDEATION AND PROPOSED SOLUTION- 3.1 EMPATHY MAP AND CANVAS



3.2 IDEATION AND BRAINSROMING



3.2 PROPOSED SOLUTION

S. No	Parameter	Description
1.	Problem Statement(Problem to be solved)	To predict the price of the Crude Oil with more accuracy.
2.	Idea/ Solution description	We are going to make a new website where the users especially investors can get the accurate price of the crude oil. So, they can analyze and make a decision before investing into a share market.
3.	Novelty/ Uniqueness	The investors can get an advice from experts through a Chatbot.
4.	Social Impact/ Customer Satisfaction	Crude oil price fluctuations have far reaching impact on global economies. Thus, it assists in minimizing the volatility in oil prices.
5.	Business Model(Revenue Model)	We can minimize money using Supply and Demand.
6.	Scalability of the solution	New features can be added based on customer feedback. S0, we can provide user- friendly environment. This will attract the customers to use the web app.

3.4 PROBLEM SOLUTION FIT

Customer: Investors are our primary customers.

JOBS TO BE DONE: Crude Oil Price Prediction is to be done through AI by collecting sufficient data.

TRIGGERS: By comparing the results from various platforms, they prefer to use our platform which yields high accuracy that comes through LSTM Model.

EMOTIONS BEFORE AND AFTER:

Before: Change in price is unpredictable.

After: Capturing the changing patterns of the prices.

AVAILABLE SOLUTONS: Market trends have to be planned so that the model (ANN) will perform effectively.

CUSTOMER CONSTRAINTS: Due to evaporative nature of oil, it becomes very challenging to achieve accuracy.

BEHAVIOUR: Investors can use it not only to initiate rates but also an effective tool to judge various strategies (like demand and supply) relating to investments.

CHANNELS OF BEHAVIOUR:

Before: Financial markets are unpredictable

After: ANN can easily predict the financial markets.

SOLUTION:

Our solution is to design effective model to predict crude oil price and achieve high accuracy through given data. Making it easy for the investors to take decision.

4 REQUIREMENT ANALYSIS:

4.1 FUNCTIONAL REQUIREMENTS

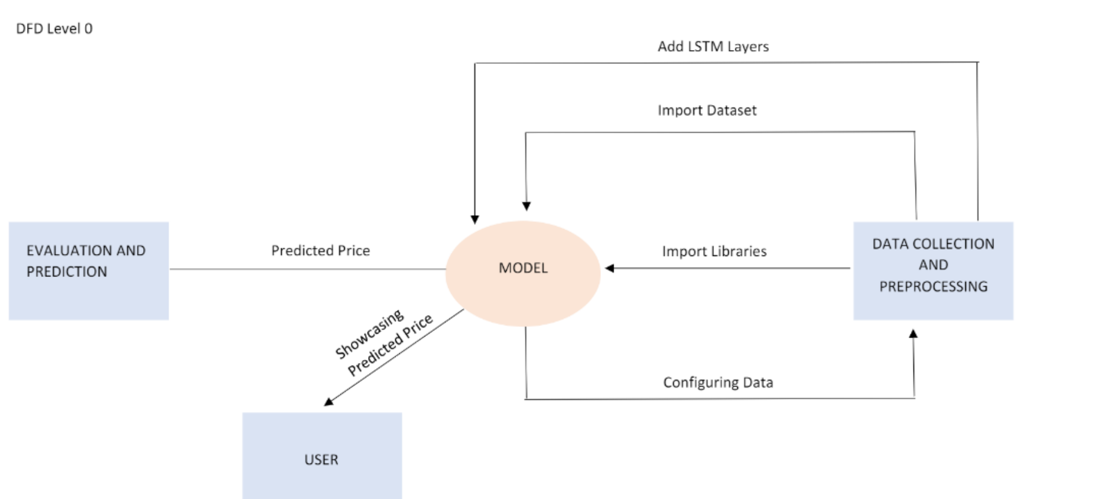
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Data Description	Consider the sampling period(number of days or months or years)
FR-2	Training set	Take more number of observations
FR-3	Test set	Evaluate the accuracy of prediction
FR-4	Model	Model is preferred for the sensitive analysis of the given data
FR-5	Model construction	LSTM Model has three layers which is input,hidden and output layer have to be constructed
FR-6	Experiment	The constructed layer compare the difference between the predicted value and the real value

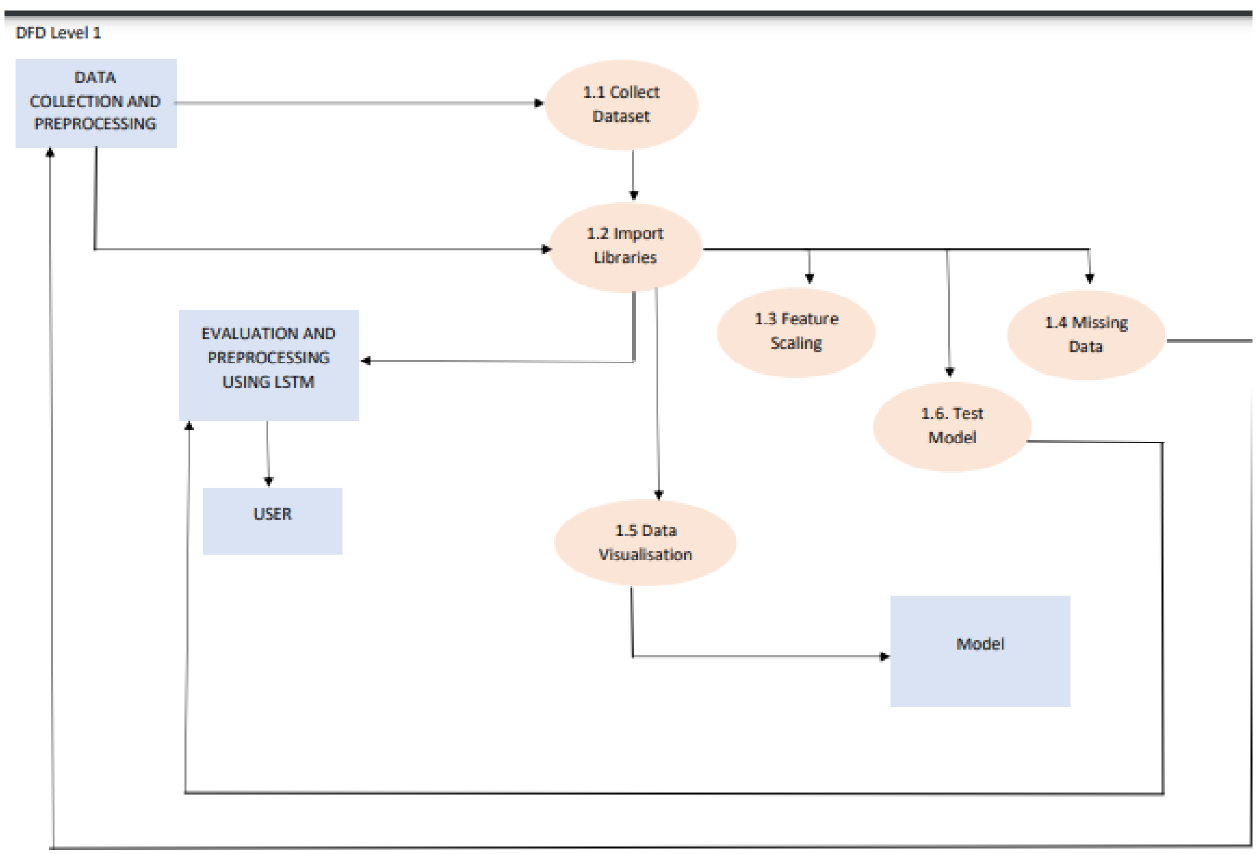
4.2 NON -FUNCTIONAL REQUIREMENTS

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Investors can easily plan their investments regarding crude oil
NFR-2	**Security**	Dataset of future economic status ,supply and demand are protected
NFR-3	**Reliability**	ANN model can access many data without any failure and result more accuracy
NFR-4	**Performance**	Effectively capture the changing pattern of prices
NFR-5	**Availability**	Dataset always provide sufficient data to forecast oil price
NFR-6	**Scalability**	Feature of web app provide easy access to the investors

5.PROJECT DESIGN

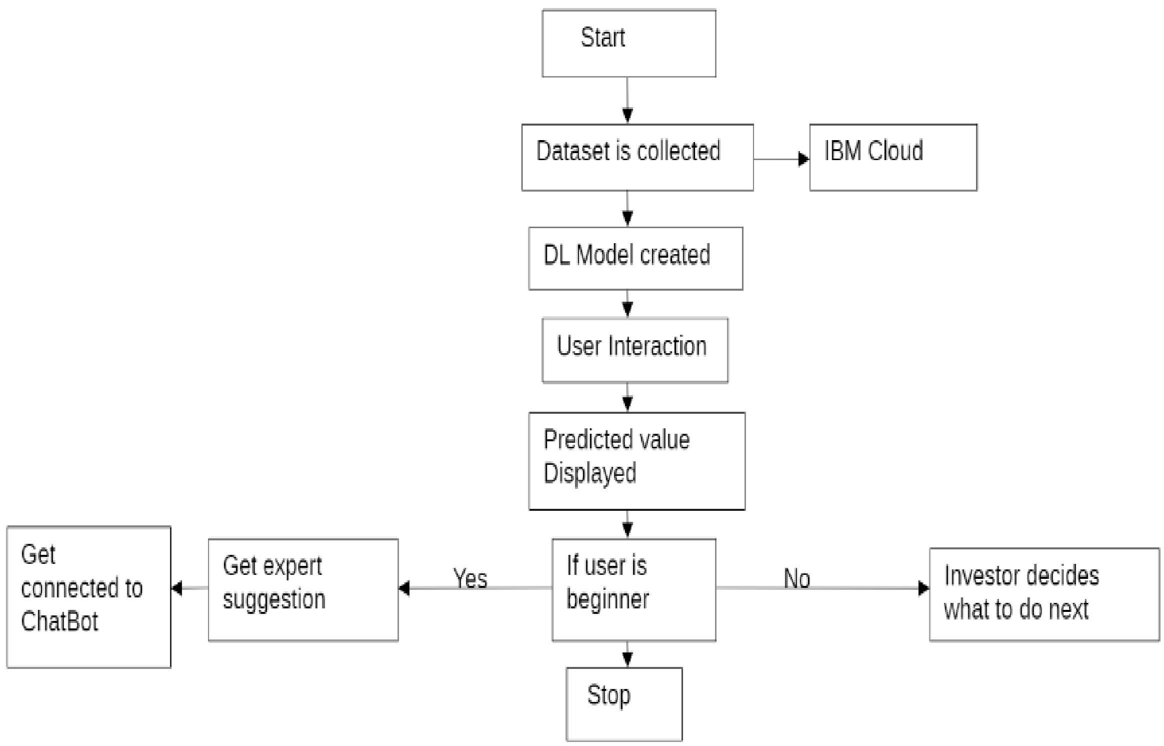
5.1 DATAFLOW DIAGRAMS





SOLUTION AND TECHNICAL ARCHITECTURE

This is the architectural diagram for crude oil price prediction,



COMPONENTS AND TECHNOLOGIES

S.NO	Component	Description	Technology
1.	User Interface	How investor interacts with application e.g. Web UI, ChatBot	HTML, CSS, JavaScript / Angular Js / Python Flask etc
2.	Application logic-1	DL Model for price prediction	LSTM Model
3.	Dataset collect	For predicting the prize of the crude oil need a dataset	Free resources for data collection
4.	User interact input	After train the model we give the input to the model	Trained model by RNN
5.	Data storage	The predicted values are used for display	IBM Cloud
6.	Display output	Prize list for display to the investor	Python Flask
7.	Expert suggestion	If the new investor visit the site give ideas	ChatBot
8.	ChatBot	Investor can interact and get knowledge for market shares	Watson AI

APPLICATION AND CHARACTERISTICS

S.NO	Characteristics	Description	Technology
1.	Open source Data	Data for price predicting technology	Many open source websites
2.	Availability	The investor can get our service at anytime	Reverse proxy
3.	Performance	The user can get the data so quick even the users increases there won't be any loss of performance	

5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through web application	I can register & access the dashboard with Facebook Login	Low	Sprint-2
	Login	USN-4	As a user, I can log into the application by entering email & password		High	Sprint-1
	Accuracy	USN-4	As a user, I can verify the predicted output by matching with accuracy.	I can access my accuracy details by entering my email and password.	Medium	Sprint-1
Customer (Web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through web application	I can register & access the dashboard with Facebook Login	Low	Sprint-2
	Login	USN-4	As a user, I can log into the application by entering email & password		High	Sprint-1
	Accuracy	USN-4	As a user, I can verify the predicted output by matching with accuracy.	I can access my accuracy details by entering my email and password.	Medium	Sprint-1
Administrator	Login	Nil	As a administrator, I can login using email id and password.	I can edit the application to match the evaluation and prediction process.	High	Sprint-1

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

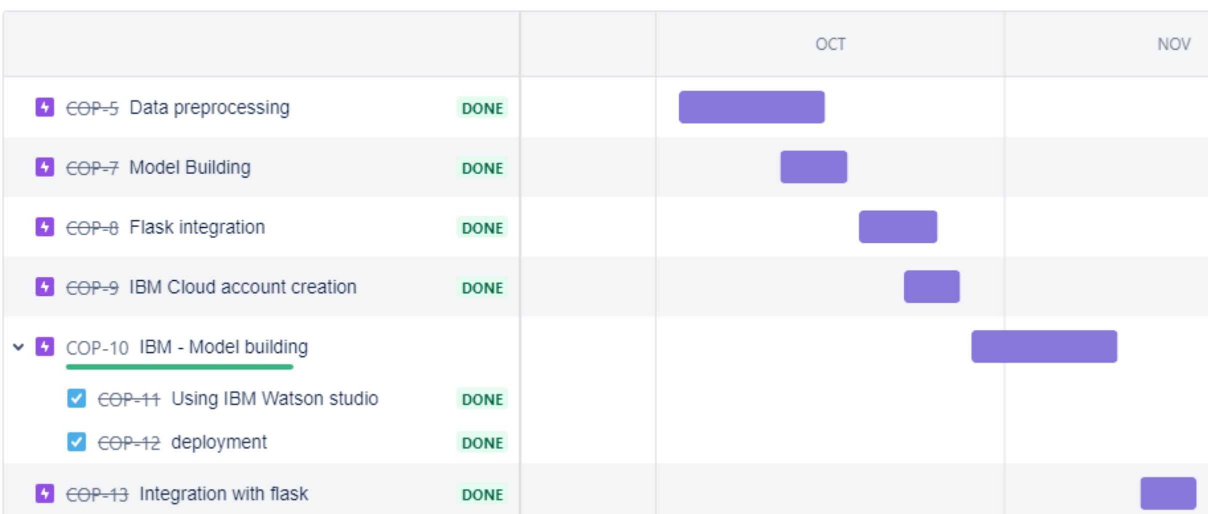
Use the below template to create product backlog and sprint schedule:

Sprint	Functional Requirement (Epic)	User Story Number	User Story/Task	Story Points	Priority	Team Members
Sprint 1	Data Collection	USN -1	As a user, we used to get the details of crude oil from the past years.	10	High	SINDHUJA S
Sprint 1	Data Pre-processing	USN-2	As a user, we used to get clarity about the data we have collected like if any data is missing, or if any addition of data is needed etc.	10	High	HEMA R
Sprint 2	Model Building	USN-3	As a user, we have to build the model where more than 60% of the data is used for training and more than 30% for testing is used to get the output with efficient prediction.	15	High	SAHANA R
Sprint 3	Integration with Flask	USN-4	As a user, we need more interaction with our web application that we have designed and our application must be user-friendly, we need to integrate Python with flask.	20	High	VISHALINI DEVI R
Sprint 4	Application Building on IBM Cloud	USN-5	As a user, we have to take care of scalability and storage, so we're building on IBM Cloud.	20	High	SAHANA R

6.2. Sprint Delivery Schedule

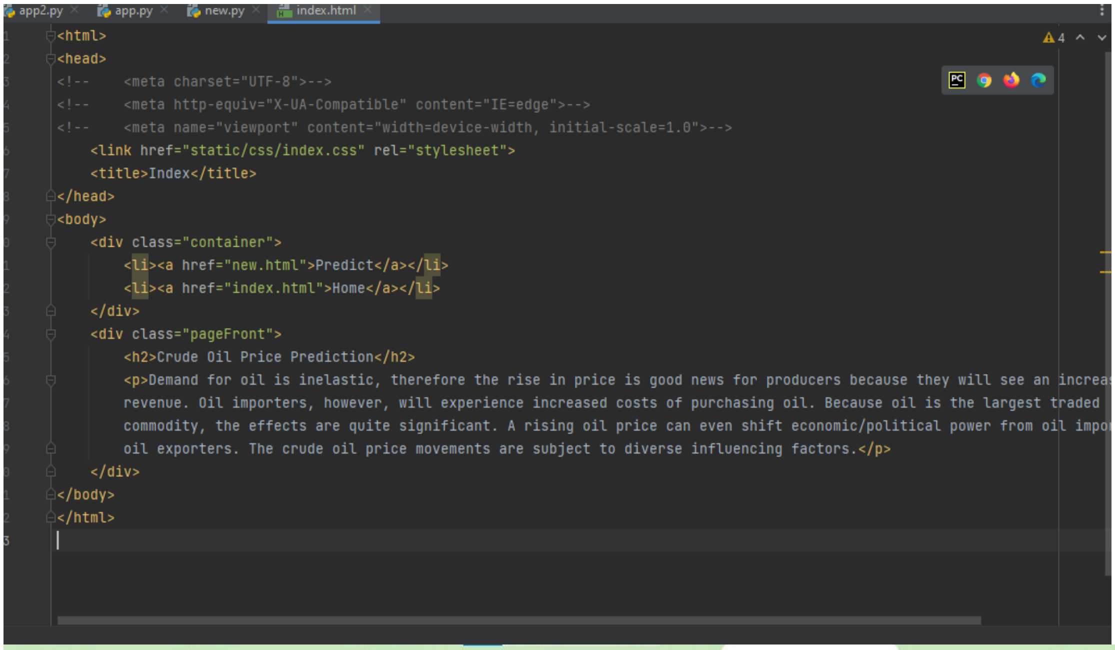
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date	Story Points Completed	Sprint Release Date(Actual)
Sprint 1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint 2	20	6 Days	31 Oct 2022	05 Nov 2022	20	5 Nov 2022
Sprint 3	20	6 Days	07 Nov 2022	12 Nov 2022	20	17 Nov 2022
Sprint 4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 Reports from JIRA

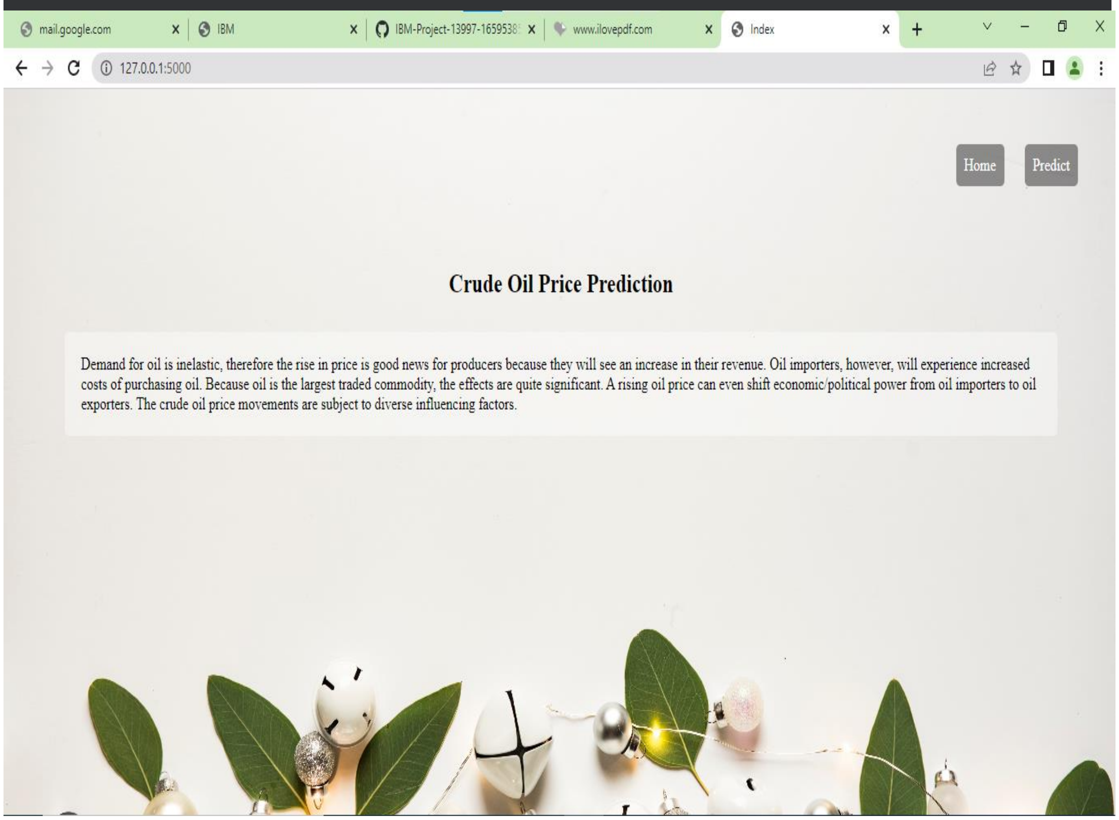


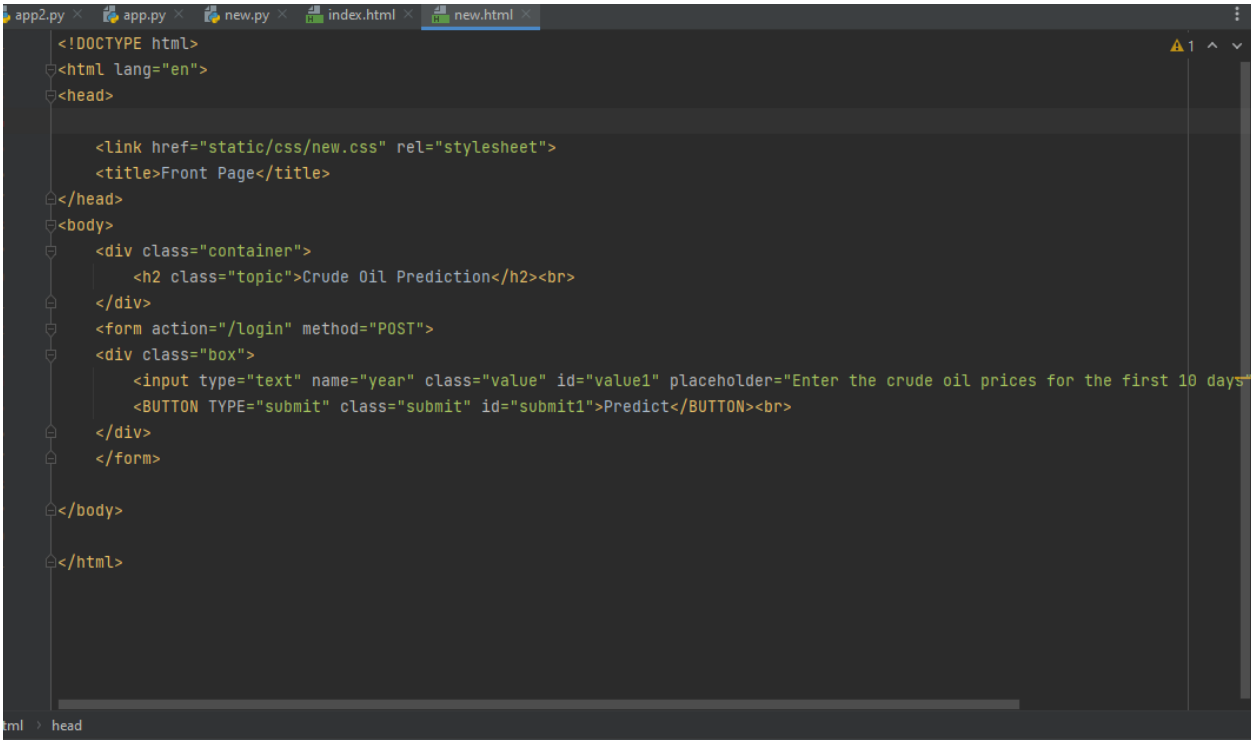
7. CODING & SOLUTIONING

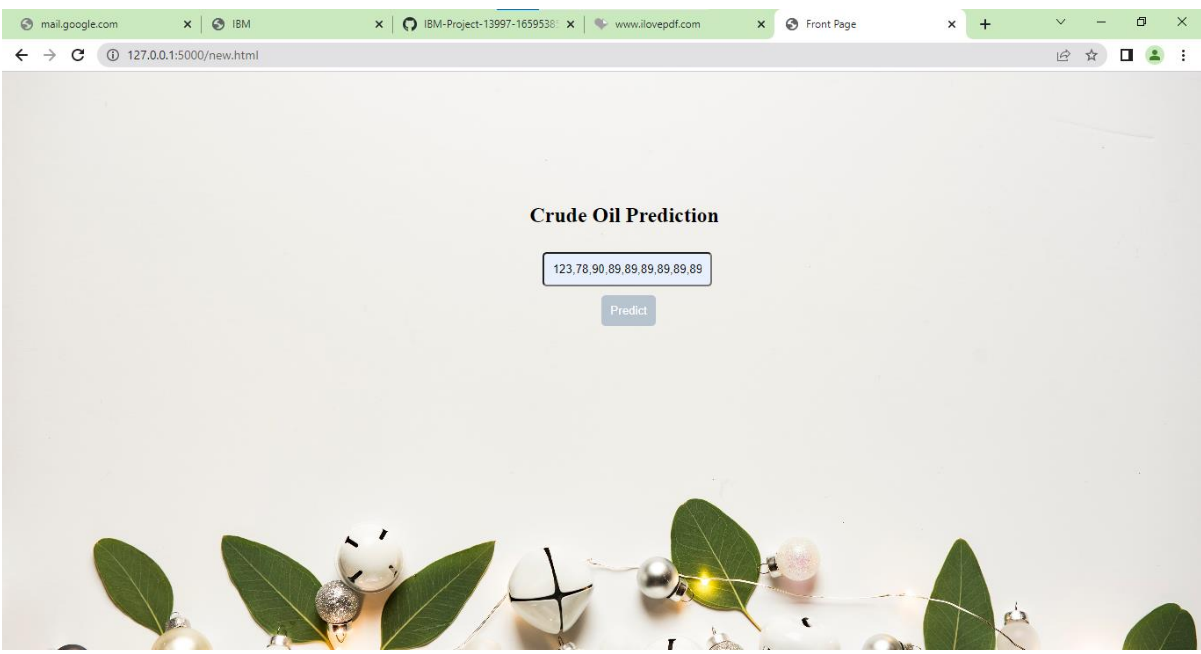
7.1 Feature 1




```
<html>
<head>
<!--    <meta charset="UTF-8">-->
<!--    <meta http-equiv="X-UA-Compatible" content="IE=edge">-->
<!--    <meta name="viewport" content="width=device-width, initial-scale=1.0">-->
    <link href="static/css/index.css" rel="stylesheet">
    <title>Index</title>
</head>
<body>
    <div class="container">
        <li><a href="new.html">Predict</a></li>
        <li><a href="index.html">Home</a></li>
    </div>
    <div class="pageFront">
        <h2>Crude Oil Price Prediction</h2>
        <p>Demand for oil is inelastic, therefore the rise in price is good news for producers because they will see an increa
        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 impo
        oil exporters. The crude oil price movements are subject to diverse influencing factors.</p>
    </div>
</body>
</html>
```

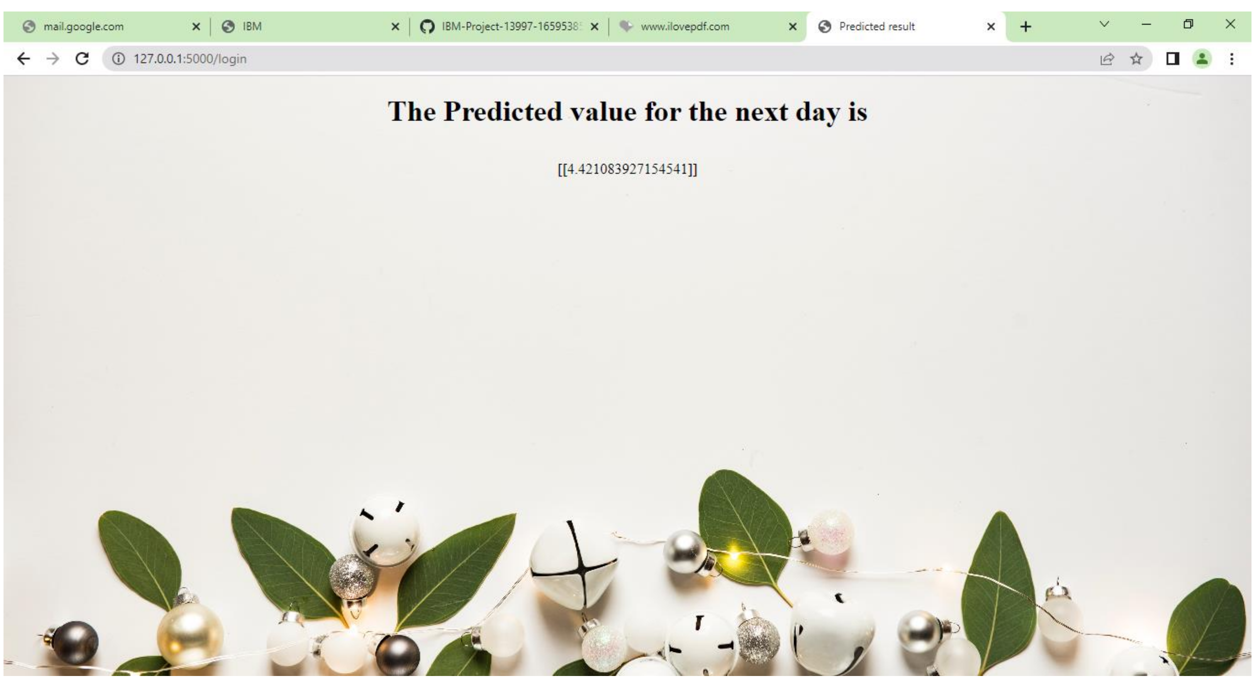



app2.py
app.py
new.py
index.html
new.html
<!DOCTYPE html>
<html lang="en">
<head>
<link href="static/css/new.css" rel="stylesheet">
<title>Front Page</title>
</head>
<body>
<div class="container">
<h2 class="topic">Crude Oil Prediction</h2>
</div>
<form action="/login" method="POST">
<div class="box">
<input type="text" name="year" class="value" id="value1" placeholder="Enter the crude oil prices for the first 10 days"
<BUTTON TYPE="submit" class="submit" id="submit1">Predict</BUTTON>
</div>
</form>
</body>
</html>
tml
head


mail.google.com
IBM
IBM-Project-13997-165953
www.ilovepdf.com
Front Page
127.0.0.1:5000/new.html
Crude Oil Prediction
123,78,90,89,89,89,89,89,89
Predict

7.2 Feature 2

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Predicted result</title>
    <link href="static/css/result.css" rel="stylesheet">
</head>
<body class="result">
    <h1>The Predicted value for the next day is</h1><br>  
    <p>{{result}}</p>
</body>
</html>
```



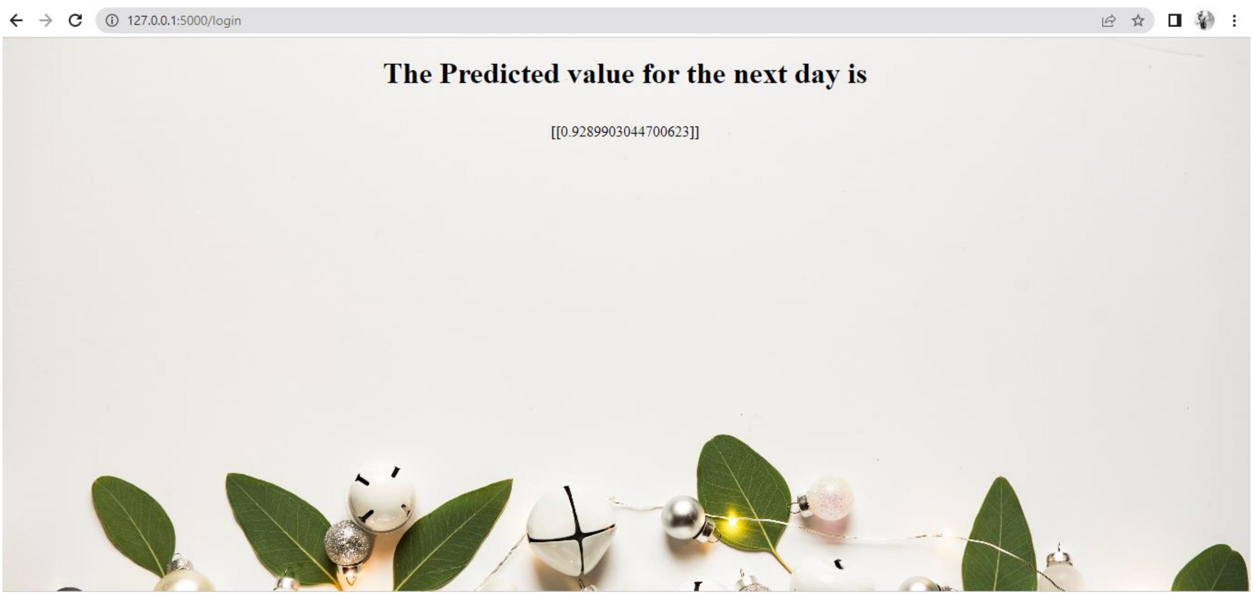
8. Testing

8.1 Test Cases

Test Case 1:

Input 1: 0.5677, 0.8765, 0.5678, 0.3456, 0.8765, 0.3456, 0.2456, 0.9876, 0.1673, 0.9876

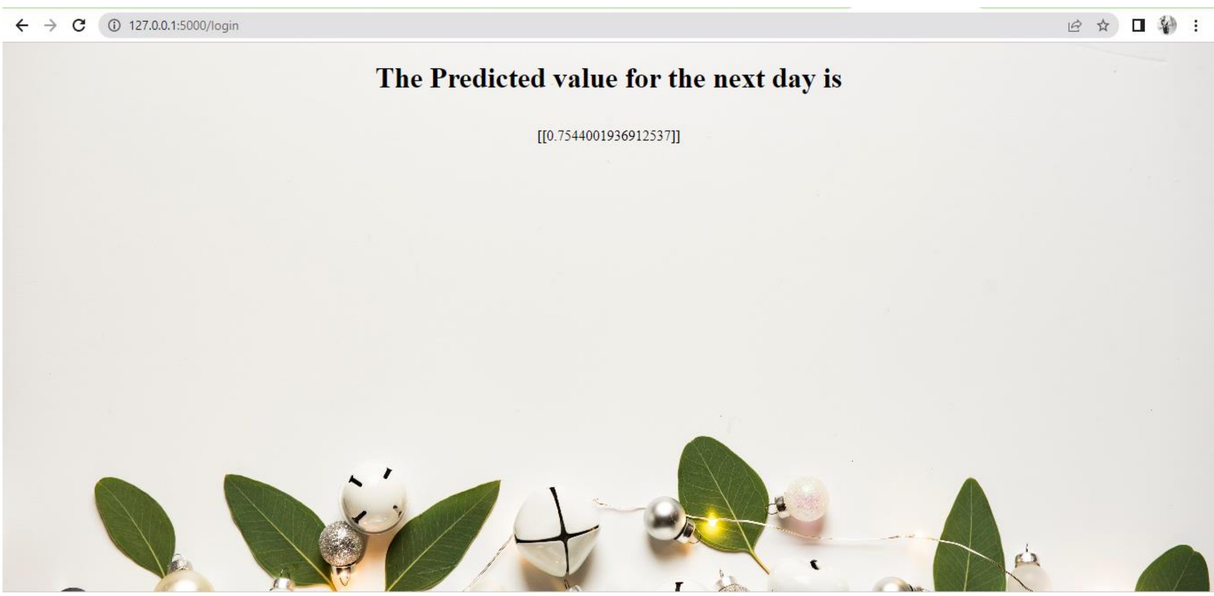
Output: 0.9289903044700623



Test Case 2:

Input 2: 0.2314, 0.9876, 0.5643, 0.6566, 0.888, 0.4567, 0.3211, 0.9876, 0.5645, 0.7654

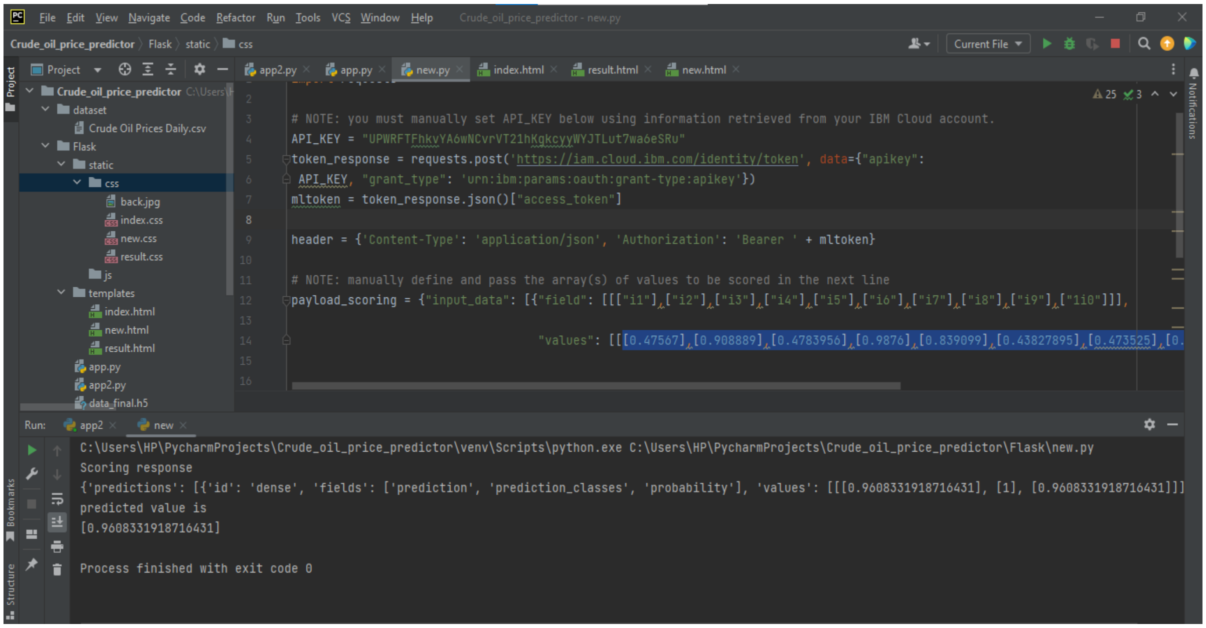
Output: 0.7544001936912537



Test case 3(Pycharm):

Input 3: 0.47567, 0.908889, 0.4783956 , 0.9876, 0.839099 , 0.43827895 , 0.473525 , 0.750590 , 0.567745 , 0.98766

Output: 0.9608331918716431



8.2 User Acceptance Testing

For the end users, our web application will have a HOME page to make the client understand WHAT OUR WEB APP WILL DO FOR THEM.

When users understand it and goes for the next page PREDICT, the end users are asked to enter the price for the past ten days and asked to submit it in order to get the 11^{th} day (the next day) price of the crude oil.

The result (Predicted price) will be displayed on the next page (RESULT PAGE).

Our web app have no complexity, the end users are coming to our web page in order to get the next day price of the crude oil and web app will serve the best for them.

(DON'T HAVE TO BEAT AROUND THE BUSH)

9. RESULT

9.1 Performance Metrics:

We use different standard performance metrics in the oil price prediction literature for comparing different oil price prediction models. The first metric is Mean Squared Prediction Error (MSPE). MSPE of a prediction model measures the average of the squares of the prediction errors. The prediction error is the difference between the true value and the predicted value. Let $y_1, y_2, \ldots, y_n$ be the true oil prices and $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$ be the predicted oil prices under an oil price prediction model, and then the MSPE of that model is:

$$MSPE=\frac{1}{n}\sum_{i=1}^{n} (\hat{y}_i-y_i)^2$$

For comparison purposes, we use the no-change model as the baseline model and express the MSPE of another model as a ratio relative to the MSPE of the no-change model. If the MSPE ratio of a model is less than 1, then the model is more accurate than the no-change model in terms of MSPE.

10. ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

- No complexity (web app works to the point)
- Faster response (less latency)
- Can be scaled easily
- Simple user interface
- Light weight

DISADVANTAGES:

- Flask can handle only smaller application.
- The user need to enter past 10 days crude oil price values, it may cause discomfort to the end users even though they can get the result immediately.
- Maintenance cost because of Flask.

CONCLUSION:

The web app will provide the best service to the end users (mainly INVESTORS) who are expert in investing and just looking for the crude oil price for their benefit as the web app will not let them wait and gives response in no time. Our Web app shows that our model achieves the highest accuracy in terms of both mean squared prediction error and directional accuracy ratio over a variety of forecast time horizons. The model used in our web app is LSTM (Long Short Term Memory). LSTM leads to more successful runs and learns much faster in compare to other algorithms like RTRN, BPTT, and RCC etc. LSTM also solves complex. The complexity to update each weight is reduced to O (1) with LSTM which is an added advantage. The LSTM cell adds long term memory in an even more performant way because it allows even more parameters to be learnt. This makes it the most powerful RNN (Recurrent Neural Network) to do forecasting, especially when you have a longer term trend in your data. IBM Watson provides tool to work collaborative and make work easier with data as well as in training the model. IBM Watson enables one-click deployment with Machine- Learning.

Immediate response:

The end users don't need to wait for longer seconds. The end users will get what they are looking for in just a second. There is no latency.

Scalability:

As our web app is deployed on IBM cloud, the code can be managed and deployed easily. If the numbers of users are monotonically increased, our web application can be scaled in no time.

12. FUTURE SCOPE

The current market situation, amid the Covid-19 outbreak, is not expected to last for the long- term; however, its long term effect on the markets may be felt for a few years to come, especially with the threat of a recession also coming from this outbreak. This could ripple out and affect the long term oil price forecast, and will need to be taken into consideration. There's also a delta variant spreading that is causing a return of lockdowns in some regions which could once again harm oil prices.

Oil price predictions long term are still vitally important to the oil investing market as the commodity, although quite volatile, is one that is often traded over longer periods of time. Oil is also a commodity that is still in high demand, and is finite, so it is expected to grow in demand over the long terms. Additionally, the prediction of oil in the long term is something that is important to different groups in the industry.

Compared with the ANN and ARIMA model, the average prediction accuracy of the LSTM model was 66.67% (33.33%) and 439587 (673.8) times higher, respectively. So, we can conclude that the LSTM model can improve the forecasting accuracy for both kinds of prices in the short term.

The number of investors in crude oil is keep on increasing as it is still in high demand. When youngsters are interested in investing but when they are in beginning stage surely our web app provide a greater service to make them understand the pattern everyday by giving them the prediction.

As price of crude oil is a complex changing pattern and new price comes every time, it is important to update our model. As we use LSTM model the complexity is very reduced to O(1) the model doesn’t take more time to retrain the model with new data, helping us to easily serve for the customers with better accuracy.

12. APPENDIX

SOURCE CODE

Data pre - processing:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

dataset1 = pd.read_csv(r"C:\Users\HP\Downloads\Crude Oil Prices Daily.csv")
dataset1.head()
```

	Date	Closing Value
0	1/2/1986	25.56
1	1/3/1986	26.00
2	1/6/1986	26.53
3	1/7/1986	25.85
4	1/8/1986	25.87

In [9]:

```
dataset1.isnull().any()
```

Out[9]:

```
Date              False
Closing Value      True
dtype: bool
```

In [10]:

```
dataset1.isnull().sum()
```

Out[10]:

```
Date             0
Closing Value    7
dtype: int64
```

In [11]:

```
dataset1.dropna(axis=0,inplace=True)
```

In [12]:

```
data_final=dataset1.reset_index()['Closing Value']
```

In [13]:

```
data_final
```

```
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 [15]:

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

In [16]:

```
plt.plot(data_final)
```

```
Training the model:
```

```
training_size=int(len(data_final)*0.65)
test_size=len(data_final)-training_size
train_data,test_data=data_final[0:training_size,:],data_final[training_size:l
en(data_final),:1]
```

In [18]:

```
training_size,test_size
```

Out[18]:

```
(5340, 2876)
```

In [19]:

```
train_data.shape
```

Out[19]:

```
(5340, 1)
```

In [20]:

```
test_data.shape
```

Out[20]:

```
(2876, 1)
```

In [21]:

```
def create_dataset(dataset,timestep=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 [22]:

```
time_step=10
x_train, y_train= create_dataset(train_data,time_step)
x_test, y_test = create_dataset(test_data, time_step)


print(x_test.shape),print(y_test.shape)
```

```
(2865, 10)
(2865,)
```

Out[23]:

```
(None, None)
```

In [24]:

```
print(x_train.shape),print(y_train.shape)
```

```
(5329, 10)
(5329,)
```

Out[24]:

```
(None, None)
```

In [25]:

```
x_train
```

Out[25]:

```
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 [1]:

```
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)


#importing libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```python
from tensorflow.keras.layers import LSTM
```

In [ ]:

```python
conda install tensorflow
```

In [29]:

```python
model = Sequential()
```

In [30]:

```python
model.add(LSTM(50,return_sequences=True,input_shape=(10,1)))
model.add(LSTM(50,return_sequences=True))
model.add(LSTM(50))
```

In [31]:

```python
model.add(Dense(1))
```

In [32]:

```python
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
_________________________________________________________________
```

In [33]:

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

In [35]:

```python
model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=50,batch_siz
e=64,verbose=1)
```

```
Epoch 1/50
84/84 [==============================] - 6s 69ms/step - loss: 8.2759e-05 -
val_loss: 4.0892e-04
Epoch 2/50
84/84 [==============================] - 4s 47ms/step - loss: 7.5317e-05 -
val_loss: 4.8240e-04
Epoch 3/50
84/84 [==============================] - 4s 49ms/step - loss: 7.9198e-05 -
val_loss: 0.0010
Epoch 4/50
84/84 [==============================] - 6s 67ms/step - loss: 8.3020e-05 -
val_loss: 3.3202e-04
```

```
Epoch 5/50
84/84 [==============================] - 4s 48ms/step - loss: 6.4561e-05 -
val_loss: 3.6391e-04
Epoch 6/50
84/84 [==============================] - 4s 47ms/step - loss: 6.7649e-05 -
val_loss: 3.4241e-04
Epoch 7/50
84/84 [==============================] - 5s 65ms/step - loss: 6.1172e-05 -
val_loss: 2.9344e-04
Epoch 8/50
84/84 [==============================] - 6s 68ms/step - loss: 6.5263e-05 -
val_loss: 5.5413e-04
Epoch 9/50
84/84 [==============================] - 4s 48ms/step - loss: 5.9849e-05 -
val_loss: 3.9287e-04
Epoch 10/50
84/84 [==============================] - 4s 51ms/step - loss: 5.5862e-05 -
val_loss: 2.5168e-04
Epoch 11/50
84/84 [==============================] - 6s 74ms/step - loss: 5.9357e-05 -
val_loss: 3.5915e-04
Etc…
```

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

In [38]:

```
len(test_data)
```

Out[38]:

```
2876
```

In [39]:

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

Out[39]:

```
(1, 10)
```

In [41]:

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

In [42]:

```
temp_input
```

Out[42]:

```
[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 input {}".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 day input [0.4811195  0.49726048 0.46794017 0.47297497 0.47119799
0.47341922
 0.46497853 0.47038353 0.47149415 0.47458544]
0 day input [[0.4780081]]
1 day input [0.49726048 0.46794017 0.47297497 0.47119799 0.47341922
0.46497853
 0.47038353 0.47149415 0.47458544 0.47800809]
1 day input [[0.48115236]]
2 day input [0.46794017 0.47297497 0.47119799 0.47341922 0.46497853
0.47038353
 0.47149415 0.47458544 0.47800809 0.48115236]
```

Etc…

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

In [47]:

```
len(data_final)
```

Out[47]:

```
8216
```

In [48]:

```
plt.plot(day_new,scaler.inverse_transform(data_final[8206:]))
plt.plot(day_pred,scaler.inverse_transform(lst_output))
```

```
df3=data_final.tolist()
df3.extend(lst_output)
plt.plot(df3[8100:])
```

Flask integration:

```
#
app.py
file
        import numpy as np
        from flask import Flask, render_template, request
        from tensorflow.keras.models import load_model
        import os
        app = Flask(__name__)
        model = load_model('data_final.h5', )
        @app.route('/')
        def home():
            return render_template("index.html")
        @app.route('/index.html')
        def home1():
            return render_template("index.html")
        @app.route('/new.html')
        def home2():
            return render_template("new.html")
        @app.route('/login',methods=['POST','GET'])
        def login():
         if request.method == 'POST':
            x_input=str(request.form['year'])
            x_input=x_input.split(',')
            print(x_input)
            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("result.html",result=str(lst_output))
        if __name__=='__main__':
          app.run(debug=True, port=5000)
```

HTML FILE :

INDEX.HTML

```
<html>
         <head>
         <!--    <meta charset="UTF-8">-->
         <!--    <meta http-equiv="X-UA-Compatible" content="IE=edge">-->
         <!--    <meta name="viewport" content="width=device-width, initial-scale=1.0">-->
             <link href="static/css/index.css" rel="stylesheet">
             <title>Index</title>
         </head>
         <body>
             <div class="container">
                 <li><a href="new.html">Predict</a></li>
                 <li><a href="index.html">Home</a></li>
             </div>
```

```
    <div class="pageFront">
        <h2>Crude Oil Price Prediction</h2>
        <p>Demand for oil 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.</p>
    </div>
</body>
</html>
```

INDEX.CSS

```
*{
        margin: 0;
        padding: 0;
    }
    body{
        background-image: url(back.jpg);
        background-color: #ffffff;
        height: 500px;
        background-position: center;
        background-repeat: no-repeat;
        background-size: cover;
    }
    .container{
        padding: 20px;
        margin: 20px;
        display: flex;
        gap: 5px;
        flex-direction: row-reverse;
    }
    .container a{
        text-decoration: none;
        color: #fff;

    }
    .container li{
```

```
            list-style-type: none;
            margin: 10px;
            background-color: #36363689;
            padding: 10px;
            border: 0px solid;
            color: #fff;
            border-radius: 5px;
        }
        .pageFront{
            margin: 20px;
            padding: 25px;
            color:
        }
        .pageFront h2{
            text-align: center;
            color:
        }
        .pageFront p{

            margin: 30px;
            background-color: rgba(255, 255, 255, 0.4);
            padding: 20px;
            border: 0px solid;
            color: rgba(0, 0, 0, 1);
            border-radius: 5px;
        }
```

NEW.HTML

```
<!DOCTYPE
html>
            <html lang="en">
            <head>
                <link href="static/css/new.css" rel="stylesheet">
                <title>Front Page</title>
            </head>
            <body>
                <div class="container">
                    <h2 class="topic">Crude Oil Prediction</h2><br>
                </div>
                <form action="/login" method="POST">
```

```
        <div class="box">
            <input type="text" name="year" class="value" id="value1"
    placeholder="Enter the crude oil prices for the first 10 days"
    multiple></input><br>
            <BUTTON TYPE="submit" class="submit" id="submit1">Predict</BUTTON><br>
        </div>
        </form>
    </body>
    </html>
```

INDEX.CSS

```
*{
            margin: 0;
            padding: 0;
        }
        body{
            background-image: url(back.jpg);
            background-color: #fbfbfb;
            background-position: center;
            background-repeat: no-repeat;
            background-size: cover;
            height: 500px;
        }
        .container{
            text-align: center;
            margin-top: 150px;
            margin-left: 50px;
        }
        h4{
            display: flex;
            align-items: center;
            justify-content: center;
            width: 30%;
            text-align: center;
            margin: 10px;
            background-color: #93919189;
            padding: 15px;
            border: 0px solid;
            color: #fff;
            border-radius: 10px;
        }
        .box{
```

```
        display: flex-column;
        align-items: center;
        justify-content: center;
        margin-left: 45%;
    }
    #submit1{
       padding: 10px;
       text-align: center;
       color: #fff;
       background-color: #B7C4CF;
       border:none;
       border-radius: 5px;
       margin-top: 10px;
       margin-left: 9%;
    }
    #value1{
       padding: 10px;
       text-align: center;
       color: #fff;
       background-color: #EAEAEA;
       border-radius: 5px;
       margin-top: 10px;
    }
```

RESULT.HTML

```
<!DOCTYPE
html>
            <html lang="en">
            <head>
                <meta charset="UTF-8">
                <title>Predicted result</title>
                <link href="static/css/result.css" rel="stylesheet">
            </head>
            <body class="result">
                <h1>The Predicted value for the next day is</h1><br>  
                <p>{{result}}</p>
            </body>
            </html>
```

RESULT.CSS

```
*{
          margin:0;
          padding:o;
        }
        body{
          background-image: url(back.jpg);
            background-color: #fbfbfb;
            background-position: center;
            background-repeat: no-repeat;
            background-size: cover;
            height: 500px;
        }
        .result{
          display: flex-column;
          padding:20px;
          text-align: center;
        }
```

RUN THE APP IN LOCAL BROWSER:

python app.py

TRAIN THE MODEL IN IBM CLOUD:

```
import pandas as pd
```

In [2]:

```
import numpy as np
```

In [3]:

```
import matplotlib.pyplot as plt
```

In [4]:

```
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='1jSTCAAD90bmuA0lBsbjxwfJKm880sHlfRc5PC_lT-M0',
    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 = 'crudeoilpricepredictor-donotdelete-pr-joybhciaj8ce4g'
object_key = 'Crude Oil Prices Daily.csv'

body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__,
body )

dataset1 = pd.read_csv(body)
dataset1.head()
```

	Date	Closing Value
0	1/2/1986	25.56
1	1/3/1986	26.00
2	1/6/1986	26.53
3	1/7/1986	25.85
4	1/8/1986	25.87

In [5]:

```
dataset1.isnull().any()
```

Out[5]:

```
Date             False
Closing Value     True
dtype: bool
```

In [8]:

```
dataset1.isnull().sum()
```

Out[8]:

```
Date             0
Closing Value    0
dtype: int64
```

In [7]:

```
dataset1.dropna(axis=0,inplace=True)
```

In [9]:

```
data_final=dataset1.reset_index()['Closing Value']
```

In [10]:

```
data_final
```

Out[10]:

```
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 [11]:

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

In [12]:

```
plt.plot(data_final)
```

```
training_size=int(len(data_final)*0.65)
test_size=len(data_final)-training_size
train_data,test_data=data_final[0:training_size,:],data_final[training_size:l
en(data_final),:1]
```

In [14]:

```
training_size,test_size
```

Out[14]:

```
(5340, 2876)
```

In [15]:

```
train_data.shape
```

Out[15]:

```
(5340, 1)
```

In [16]:

```
test_data.shape
```

Out[16]:

```
(2876, 1)
```

In [17]:

```
def create_dataset(dataset,timestep=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, y_test = create_dataset(test_data, time_step)
```

In [19]:

```
print(x_test.shape),print(y_test.shape)
```

```
(2865, 10)
(2865,)
```

Out[19]:

```
(None, None)
```

In [20]:

```
print(x_train.shape),print(y_train.shape)
```

```
(5329, 10)
(5329,)
```

Out[20]:

```
(None, None)
```

In [21]:

```
x_train
```

Out[21]:

```
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]])
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 [23]:

```
#importing libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
```

In [ ]:

```
conda install tensorflow
```

In [24]:

```
model = Sequential()
```

In [25]:

```
model.add(LSTM(50,return_sequences=True,input_shape=(10,1)))
model.add(LSTM(50,return_sequences=True))
model.add(LSTM(50))
```

In [26]:

```
model.add(Dense(1))
```

In [27]:

```
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
_________________________________________________________________
```

In [28]:

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

In [57]:

```
model1=model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=50,ba
tch_size=64,verbose=1)
```

```
Epoch 1/50
84/84 [==============================] - 2s 20ms/step - loss: 3.8141e-05 -
val_loss: 2.2677e-04
Epoch 2/50
84/84 [==============================] - 2s 20ms/step - loss: 3.7947e-05 -
val_loss: 6.4153e-04
Epoch 3/50
84/84 [==============================] - 2s 25ms/step - loss: 3.5235e-05 -
val_loss: 3.1454e-04
Epoch 4/50
84/84 [==============================] - 2s 25ms/step - loss: 3.4177e-05 -
val_loss: 2.0435e-04
Epoch 5/50
84/84 [==============================] - 2s 21ms/step - loss: 3.3932e-05 -
val_loss: 2.1217e-04
Epoch 6/50
84/84 [==============================] - 2s 21ms/step - loss: 3.1182e-05 -
val_loss: 2.1916e-04
Epoch 7/50
84/84 [==============================] - 2s 24ms/step - loss: 3.2765e-05 -
val_loss: 1.8484e-04
Epoch 8/50
```

ETC…

```
from tensorflow.keras.models import load_model
```

model.save("data_final.h5")

In [31]:

```
model.save("data_final.h5")
len(test_data)
```

Out[31]:

```
2876
```

In [32]:

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

Out[32]:

```
(1, 10)
```

In [33]:

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

In [34]:

```
temp_input
```

Out[34]:

```
[0.44172960165852215,
 0.48111950244335855,
 0.49726047682511476,
 0.4679401747371539,
 0.4729749740855915,
 0.47119798608026064,
 0.47341922108692425,
 0.4649785280616022,
```

ETC…

```
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 input {}".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
```

```
[0.4733196]
11
1 day input [0.4811195  0.49726048 0.46794017 0.47297497 0.47119799
0.47341922
 0.46497853 0.47038353 0.47149415 0.47331959]
1 day input [[0.47587287]]
2 day input [0.49726048 0.46794017 0.47297497 0.47119799 0.47341922
0.46497853
 0.47038353 0.47149415 0.47331959 0.47587287]
```

ETC…

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

In [37]:

```
len(data_final)
```

Out[37]:

```
8216
```

In [38]:

```
plt.plot(day_new,scaler.inverse_transform(data_final[8206:]))
plt.plot(day_pred,scaler.inverse_transform(lst_output))
```

```
df3=data_final.tolist()
df3.extend(lst_output)
plt.plot(df3[8100:])
from ibm_watson_machine_learning import APIClient
wml_credentials = {
    "url": "https://us-south.ml.cloud.ibm.com",
    "apikey": "UPWRFTFhkvYA6wNCvrVT21hKgkcyyWYJTLut7wa6eSRu"
}
client=APIClient(wml_credentials)
```

In [41]:

```
!pip install ibm_watson_machine_learning
```

```
Requirement already satisfied: ibm_watson_machine_learning in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.0.257)
Requirement already satisfied: importlib-metadata in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (4.8.2)
Requirement already satisfied: ibm-cos-sdk==2.11.* in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (2.11.0)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (0.8.9)
Requirement already satisfied: lomond in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (0.3.3)
Requirement already satisfied: requests in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (2.26.0)
```

```
Requirement already satisfied: pandas<1.5.0,>=0.24.2 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from
ibm_watson_machine_learning) (1.3.4)
```

ETC…

```
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'])
```

In [51]:

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

```
Space UID =84775292-16ec-4944-b70d-9a69920281de
```

In [53]:

```
client.set.default_space(space_uid)
client.software_specifications.list()
```

```
----------------------------- ------------------------------------ ----
NAME                          ASSET_ID                             TYPE
default_py3.6                 0062b8c9-8b7d-44a0-a9b9-46c416adcbd9 base
kernel-spark3.2-scala2.12     020d69ce-7ac1-5e68-ac1a-31189867356a base
pytorch-onnx_1.3-py3.7-edt    069ea134-3346-5748-b513-49120e15d288 base
scikit-learn_0.20-py3.6       09c5a1d0-9c1e-4473-a344-eb7b665ff687 base
```

ETC…

```
software_spec_uid=client.software_specifications.get_uid_by_name("runtime-
22.1-py3.9")
software_spec_uid
```

Out[102]:

```
'12b83a17-24d8-5082-900f-0ab31fbfd3cb'
```

In [99]:

```
model_details =
client.repository.store_model(model='data_final.tgz',meta_props={
    client.repository.ModelMetaNames.NAME:"Crude_oil_price",
    client.repository.ModelMetaNames.TYPE:"tensorflow_rt22.1",
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid }
                                             )
model_id=client.repository.get_model_uid(model_details)
```

```
This method is deprecated, please use get_model_id()
```

In [66]:

```
model_result_path="data_final.h5"
model.save(model_result_path)
```

In [68]:

```
!tar -zcvf data_final.tgz data_final.h5
```

```
data_final.h5
```

In [96]:

```
model_id
```

Out[96]:

```
'f83e2c1b-0695-4350-9647-3b1905af367f'
```

ETC…

PYTHON CODE (new.py):

```
Import
requests
        # NOTE: you must manually set API_KEY below using information retrieved from your
        IBM Cloud account.
        API_KEY = "UPWRFTFhkvYA6wNCvrVT21hKgkcyyWYJTLut7wa6eSRu"
        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}
        # NOTE: manually define and pass the array(s) of values to be scored in the next
        line
        payload_scoring = {"input_data": [{"field":
        [[["i1"],["i2"],["i3"],["i4"],["i5"],["i6"],["i7"],["i8"],["i9"],["1i0"]]],
                                           "values":
        [[[0.47567],[0.908889],[0.4783956],[0.9876],[0.839099],[0.43827895],[0.473525],[0.75
        0590],[0.567745],[0.98766]]]}]}
        response_scoring = requests.post('https://us-
        south.ml.cloud.ibm.com/ml/v4/deployments/f73b8e94-628a-4cd8-8a84-
        a5b527eb1468/predictions?version=2022-11-17', json=payload_scoring,
         headers={'Authorization': 'Bearer ' + mltoken})
        print("Scoring response")
        print(response_scoring.json())
        predictions=response_scoring.json()
        print("predicted value is")
        print(predictions['predictions'][0]['values'][0][0])
```

FLASK INTEGRATION – IBM CLOUD (app2.py)

```
import
requests
           import numpy as np
           from flask import Flask, render_template, request, jsonify
           # NOTE: you must manually set API_KEY below using information retrieved from your
           IBM Cloud account.
           API_KEY = "UPWRFTFhkvYA6wNCvrVT21hKgkcyyWYJTLut7wa6eSRu"
           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('/index.html')
           def home1():
               return render_template("index.html")
           @app.route('/new.html')
           def home2():
               return render_template("new.html")
           @app.route('/login',methods=['POST','GET'])
           def login():
            if request.method == 'POST':
               x=str(request.form['year'])
               x=x.split(',')
               print(x)
               for w in range(0, len(x)):
                   x[w]=float(x[w])
               print(x)
               t=[[ [x[0]], [x[1]], [x[2]], [x[3]], [x[4]], [x[5]], [x[6]], [x[7]], [x[8]],
           [x[9]]]]
               payload_scoring = {
               "input_data": [{"field": [[["i1"], ["i2"], ["i3"], ["i4"], ["i5"], ["i6"],
           ["i7"], ["i8"], ["i9"], ["1i0"]]],
                                    "values":t }]}
               response_scoring = requests.post('https://us-
           south.ml.cloud.ibm.com/ml/v4/deployments/f73b8e94-628a-4cd8-8a84-
           a5b527eb1468/predictions?version=2022-11-17', json=payload_scoring,
```

```
        headers={'Authorization': 'Bearer ' + mltoken})
        print("Scoring response")
        print(response_scoring.json())
        predictions=response_scoring.json()
        print("predicted value is")
        print(predictions['predictions'][0]['values'][0][0])
        pred=predictions['predictions'][0]['values'][0][0]
        return render_template("result.html",result=str(pred))
if __name__=='__main__':
  app.run(debug=True, port=5000)
```

DEMO LINK:

https://drive.google.com/drive/folders/1vRcITCfPjSKsqN-98wwzZf5tNMzSkWQC?usp=share_link

GITHUB LINK:

https://github.com/IBM-EPBL/IBM-Project-18457-1659685499