

SBM COLLEGE OF ENGINEERING&TECHNOLOGY

TEAM ID:PNT2022TMID49012

CRUDE OIL PRICE PREDICTION

TEAM MEMBERS :

921619106022-HARI KRISHNAN M

921619106028-KARTHIK G

921619106038-MOHAMED YASIN M

921619106016-DHINAKARAN P

INDEX

- 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**
 - 7.1 Feature 1
 - 7.2 Feature 2
- 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 overview

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.

1.2. Purpose

Crude oil price fluctuations have a far reaching impact on global economies and thus price forecasting can assist in minimising the risks associated with volatility in oil prices. Price forecasts are very important to various stakeholders: governments, public and private enterprises, policymakers, and investors. According to economic theory, the price of crude oil should be easily predictable from the equilibrium between demand and supply, wherein demand forecasts are usually made from GDP, exchange rates and domestic prices, and supply is predicted from past production data and reserve data. Predicting demand for oil is usually straightforward, however supply is heavily affected by political activity such as cartelisation by OPEC to regulate prices, technological advances leading to the extraction of higher amounts of oil, and wars and other conflicts which can affect supply unpredictably.

2. LITERATURE SURVEY

2.1. Existing problem

Numerous studies have used traditional and statistical econometric models to forecast crude oil prices. These methods are usually able to handle only linear time series data. However, crude oil market is the most volatile commodities market. Therefore, forecasting oil price via nonlinear models is the appropriate choice. ANN is the most popular nonlinear AI model used to predict crude oil price earlier. Therefore, this approach was used. Finally, we presented the existing literature on forecasting crude oil price using ANNs models. As conclusions drawn from these studies, neural network approach has shown a strong predictive ability, in this field of research. So we have used RNN and LSTM instead of the traditional ANN method.

2.2. References

- Abdullah, S. N. and Zeng, X. (2010) ""Machine learning approach for crude oil price prediction with Artificial Neural Networks-Quantitative (ANN-Q) model"" Proceedings of the International Joint Conference on Neural Networks (IJCNN'2010), 1-8.
- Agnolucci, P. (2009) ""Volatility in crude oil futures: A comparison of the predictive ability of GARCH and implied volatility models"" Energy Economics 31, 316-321.
- Alizadeh, A. and Mafinezhad, K (2010)""Monthly Brent Oil Price Forecasting Using Artificial Neural Networks and A Crisis Index"" Proceedings of the International Conference On Electronics And Information Engineering (ICEIE'2010), 2, 465-468.
- Aloui, C. Hamdi, M., Mensi, W. and Nguyen, D. Y. (2012) ""Further evidence on the timevarying efficiency of crude oil markets"" Energy Studies Review 19(2), 38-51. Amano A. (1987) ""A Small Forecasting Model of the World Oil Market"" Journal of Policy Modeling 9(4), 615-635.
- Amin-Naseri, M. R. and Gharacheh, E. A. (2007) ""A hybrid artificial intelligence approach to monthly forecasting of crude oil price time series"" The Proceedings of the 10th International Conference on Engineering Applications of Neural Networks (CEANN'2007), 160-167.
- Barone-Adesi, G., Bourgoin, F. and Giannopoulos, K. (1998) ""Don't Look Back"" Risk 11, 100-104. Beidas-Strom, S. and Pescatori, A. (2014) ""Oil price volatility and the role of speculation"" IMF working paper (WP/14/218).
- Bernabe, A., Martina, E., Alvarez-Ramirez, J. and Ibarra-Valdez, C. (2004) ""A multi-model approach for describing crude oil price dynamics"" Physica A: Statistical Mechanics and its Applications 338(3), 567-584.
- Blanchard, O. J. and Gali, J. (2007) ""The macroeconomic effects of oil shocks: Why are the 2000s so different from 1970s?"" NBER working paper number 13368, 1-77.
- Cheong, C.W. (2009) ""Modelling and forecasting crude oil markets using ARCH-type models"" Energy Policy 37, 2346-2355.
- Dees, S., Karadeloglou, P., Kaufmann, R. K. and Sanchez, M. (2007) ""Modelling the world oil market: assessment of a quarterly econometric model"" Energy Policy 35, 178-191.
- Fattouh, B. (2012) ""Speculation and oil price formation"" Review of Environment, Energy and Economics (Re3), 1-5.
- Ghaffari, A. and Zare, S. (2009) ""A novel algorithm for prediction of crude oil price variation based on soft computing"" Energy Economics 31, 531-536.

2.3. Problem statement definition

1. Supply

Supply and demand has to do with how much oil is available.

Supply has historically been determined by countries that are part of [OPEC](#). But now, the United States is playing a bigger role in supply thanks to booming production from American shale fields. So if major oil-producing countries are pumping out a lot of crude, the supply will be high.

Just look at what happened in 2014.

"Saudi Arabia made the decision that they were not going to cut back production, they were going to continue to produce at record high levels," said Tamar Essner, senior energy director at Nasdaq IR Solutions.

"At the same time, you had very robust output from the United States, and from other producers around the world."

Oil prices fell sharply as producers pumped more than the world could consume.

OPEC was largely blamed for the free fall in oil prices because it refused to cut down its production. But OPEC said U.S. shale drillers were to blame for pumping too much, and should cut their production first.

In 1973, Arab members of OPEC put an embargo against the United States as a retaliatory measure for U.S. support of Israel during the Yom Kippur War. After the embargo, the oil supply in the U.S. was so scarce and the demand was so high, it drove the price of crude to the point that gas stations began rationing gasoline.

2. Demand

Demand on the other hand is determined by how much need there is for oil at a given time. That need is often for things like heat, electricity and transportation. The more economic growth a region sees, the more demand there will be for oil.

"Economies around the world have picked up since the financial crisis, and growth has gotten stronger so people have been using more energy," Essner said.

And then there's the question of how the market will react to renewable energy.

"A lot of this will be impacted by public policy, but at the end of the day renewable can only displace hydrocarbons if it's economically feasible," Essner said.

"Right now, renewables are still more expensive than hydrocarbons, so consumers aren't going to voluntarily make the switch."

3. Geopolitics

Since supply is determined by the big oil-producing countries, tension with one of those nations can cause major problems. So if there's war or conflict in an oil-producing region, crude inventories could seem threatened, and that could ultimately alter the price of oil.

"Geopolitics has traditionally been a factor in the oil price," Essner said.

"Particularly when situations in the Middle East or other oil-rich regions of the world would flare up and there would be conflict, you would generally speaking see a little bit of an uptick in the price of oil as a result, just by virtue of the risk of supply being disrupted, or of means of transportation being disrupted, such as a canal or pipeline or workers going on protest, things like that."

Just think back to the Gulf War of 1991. Oil production fell, which caused prices to rise.

And in 2003, oil prices soared after the U.S. invaded Iraq. That Middle Eastern nation produces a lot of oil, and with instability in the region, people weren't immediately sure what would happen to the supply.

"That's what makes the oil markets so fascinating, is that it's really a very interesting interplay of financial markets, the economy, and those are two very different things, the currency market, geopolitics and the environment," Essner said.

The energy industry is sure to evolve, and experts are watching to see what role oil will play in the future. But for now, the oil markets remain a powerful force in the world of economics, geopolitics and your commuting budget.

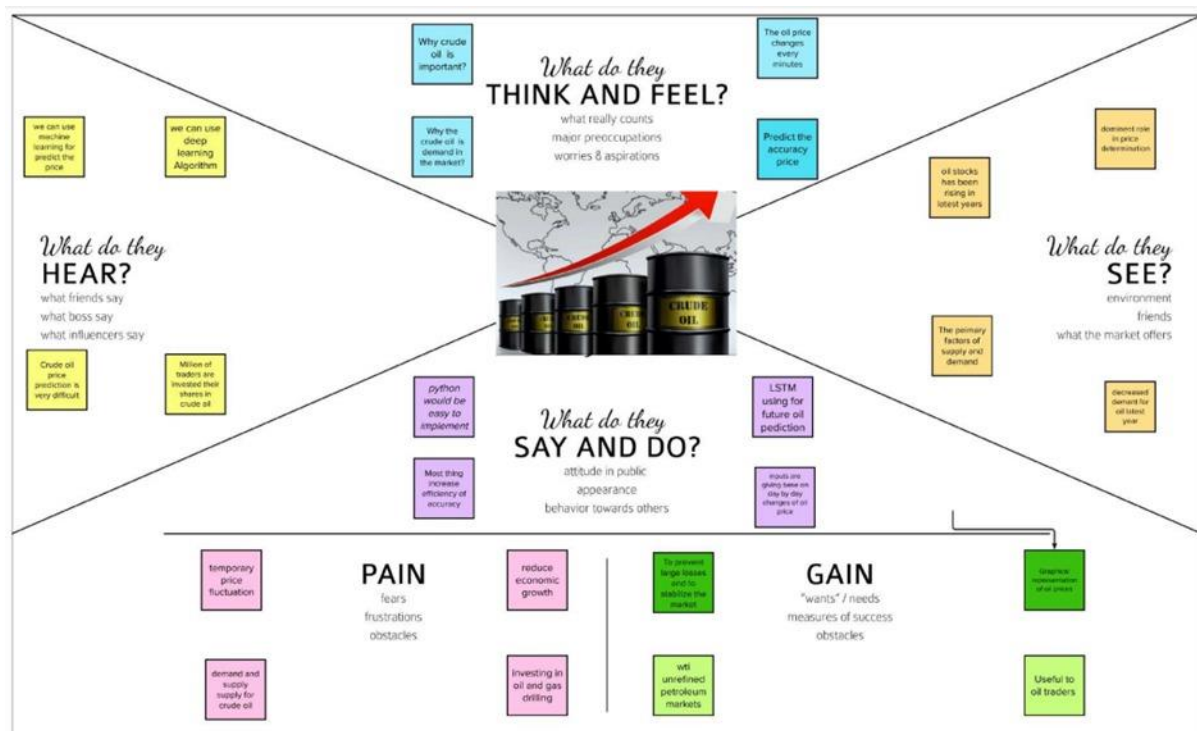
3. IDEATION AND PROPOSED SOLUTION

3.1. Empathy Map Canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes.

It is a useful tool to help teams better understand their users.

Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.



3.2. Ideation and Brainstorming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Step-1: Team Gathering, Collaboration and Select the Problem Statement



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

- 🕒 10 minutes to prepare
- 🕒 1 hour to collaborate
- 👥 2-8 people recommended



Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

🕒 10 minutes

- A** Team gathering
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.
- B** Set the goal
Think about the problem you'll be focusing on solving in the brainstorming session.
- C** Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) →



Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

🕒 5 minutes

PROBLEM

How crude oil price can be predicted? what are the ways to predict and what are the impacts?



Key rules of brainstorming

To run a smooth and productive session

- | | |
|-------------------|----------------------------|
| 🗣️ Stay in topic. | 💡 Encourage wild ideas. |
| 👂 Defer judgment. | 👂 Listen to others. |
| 🗣️ Go for volume. | 👁️ If possible, be visual. |

Step-2: Brainstorm, Idea Listing and Grouping

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

🕒 10 minutes

Sree Kayathri

can be predicted using the raw data

artificial intelligence can be very effective in prediction

sampleings and records are useful for prediction

Priyadharshini

deep learning can be used for analysis

history of prices can be used for references

the integration of AI and machine learning will be useful

Rajasimman

Python can be used to represent the ideas

Visual representation can be done to get a better idea about the data

Clear information should be gathered to implement the model

Niranjana

Price prediction can be done by an analyzing the data from different time period

The overall results can be recorded and can be used for the prediction

Professionals can be really helpful in gaining knowledge about the crude oil prices

3

Group Ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

🕒 20 minutes

Basic level

can be predicted using the raw data

artificial intelligence can be very effective in prediction

sampleings and records are useful for prediction

Advanced level

deep learning can be used for analysis

history of prices can be used for references

the integration of AI and machine learning will be useful

Python can be used to represent the ideas

The overall results can be recorded and can be used for the prediction

Professionals can be really helpful in gaining knowledge about the crude oil prices

Price prediction can be done by go analyzing the data from different time period

Visual representation can be done to get a better idea about the data

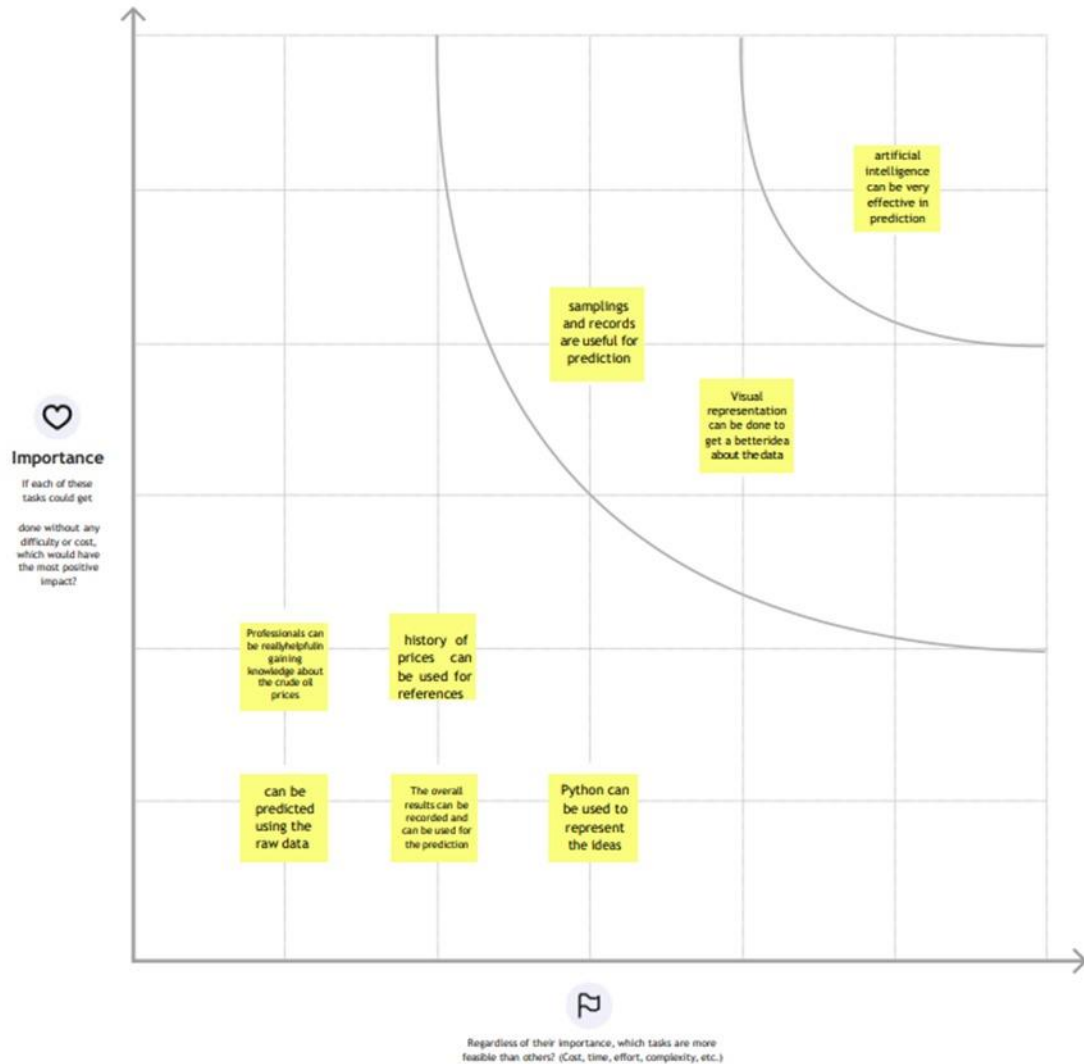
Step-3: Idea Prioritization

4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes



3.3. Proposed Solution:

S.No.	Parameter	Description
	Problem Statement (Problem to be solved)	As with the erratic changes in supply and demand and also the influence of geopolitics, it is very hard to predict the value of crude oil prices in the global market.

	Idea / Solution description	We are going to collect the dataset of the past oil prices with time so that by feeding those to the model and training it and compiling it and when it's achieved the optimal state we can implement it in the web application.
	Novelty / Uniqueness	It may be a traditional idea but the implementation of periodic training will have a better effect on it.
	Social Impact / Customer Satisfaction	By using the web app customer can gain knowledge of the crude oil price and get benefits financially.
	Business Model (Revenue Model)	It will be used by every individual at ease so that they can have an idea of the crude price so, that the use of the crude will be stable in the market
	Scalability of the Solution	The idea we proposed it take the input in the periodic and adjust and train through these so, that it will adapt to very different situations.

3.4. Problem Solution Fit

Problem-Solution fit canvas 2.0

Purpose / Vision

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) Who is your customer? Crude Oil Based Industries and companies for Business purposes	6. CUSTOMER CONSTRAINTS What constraints prevent your customers from taking action or limit their choices of solutions? The risks and problems are the obstacles for the customers which limits them from proceeding further in the process.	5. AVAILABLE SOLUTIONS Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? The frustrations about the results can be avoided by providing a proper timeline and proper planning will be helpful in finishing it in time with the expected output.	Explore AS, differentiate
	2. JOBS-TO-BE-DONE / PROBLEMS Which jobs-to-be-done (or problems) do you address for your customers? The difficulty in predicting the Crude Oil Price more accurately is one of the major problems The information to be collected for providing the desired results may be a problem	9. PROBLEM ROOT CAUSE What is the real reason that this problem exists? What is the back story behind the need to do this job? It can both Man-made error or machine error which can sometimes go wrong. This can cause a problem in proving an accurate or desired result. This is the main root cause of this issue.	7. BEHAVIOUR What does your customer do to address the problem and get the job done? The problems faced by the customer can be reported in a form of a detailed document so that it can be properly addressed by the team and it can rectify.	
Focus on J&P, tap into BE, understand RC	3. TRIGGERS What triggers customers to act? The business ideas trigger customers for the crude oil price prediction for the benefits	10. YOUR SOLUTION If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality. If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour. To address this issue, it needs proper attention in carrying out this process for predicting the crude oil price. Both computer-aided prediction and human calculations should be carried out very carefully.	8. CHANNELS of BEHAVIOUR 8.1 ONLINE What kind of actions do customers take online? Extract online channels from #7 Discount seekers Wandering customers Loyal customers 8.2 OFFLINE What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development. Reliable customers Trustful customers	Extract online & offline CH of BE
	4. EMOTIONS: BEFORE / AFTER How do customers feel when they face a problem or a job and afterwards? If the results are not up to the expected point, it makes them feel frustrated.			

Problem-Solution fit canvas is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 license
 Created by Daria Nepriakhina / Amaltama.com

AMALTAMA

4. REQUIREMENT ANALYSIS

4.1.Functional requirement

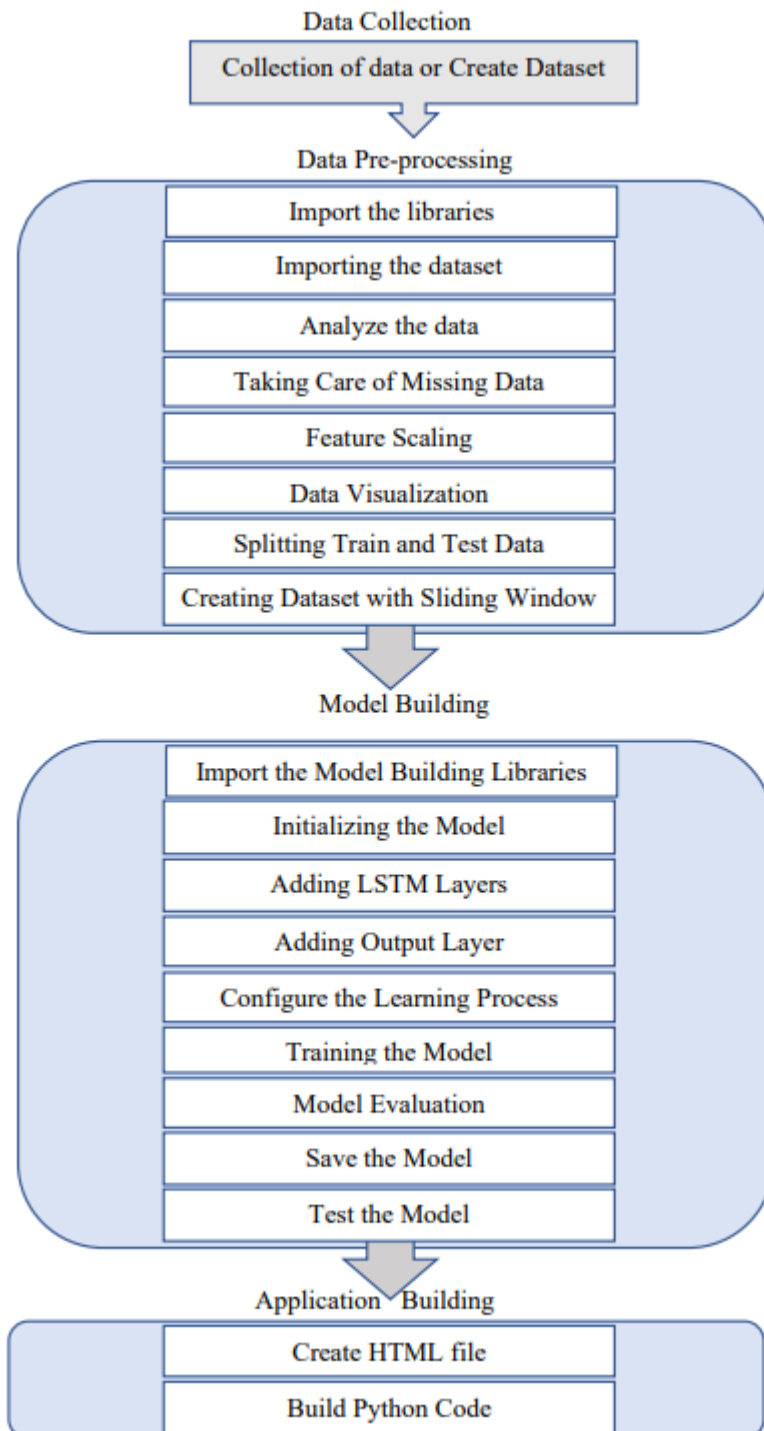
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Graph	Showing graph by obtaining the data from the dataset
FR-4	Support	Providing answers for the queries asked by users.
FR-5	News	Information of the oil prices will be updated by admin
FR-6	Notification	Notification will be sent for the users price alert
Fr-7	Database	Information of the User will be stored

4.2. Non-Functional Requirements

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	It can use by wide variety of client as it is very simple to learn and not complex to proceed.
NFR-2	Security	We are using login for the user and the information will be hashed so that it will be very secure to use.
NFR-3	Reliability	It will be reliable that it can update with very time period so that the accuracy will be good.
NFR-4	Performance	It will be perform fast and secure even at the lower bandwidth.
NFR-5	Availability	Prediction will be available for every user but only for premium user news,database and price alert will be alert.
NFR-6	Scalability	It is scalable that we are going to use data in kb so that the quite amount of storage is satisfied.

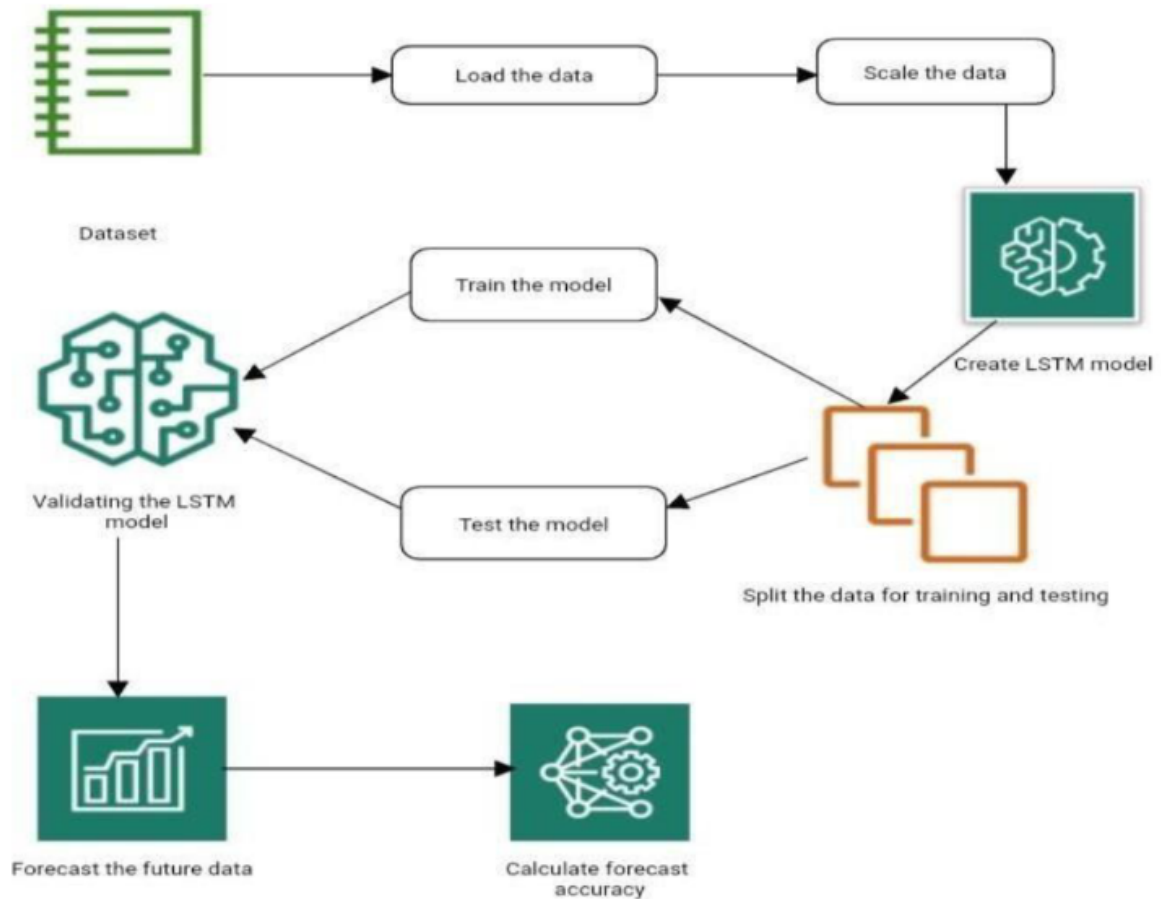
5. Project Design

5.1. Project Flow Diagrams

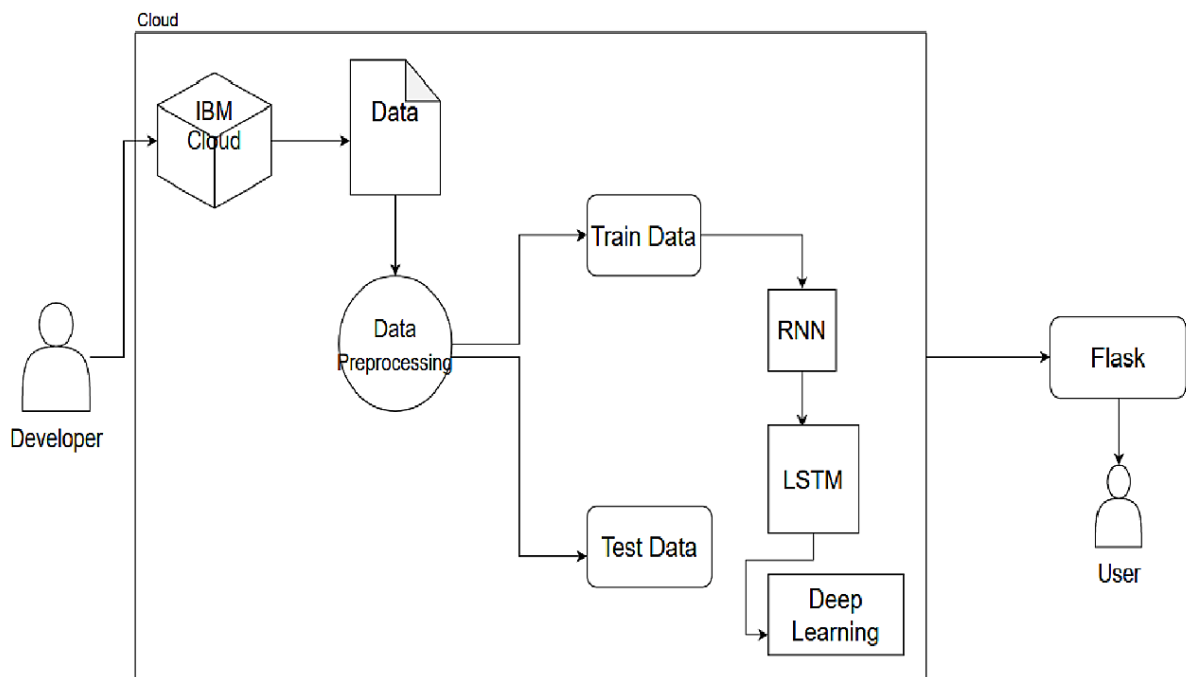


5.2. Solution & Technical Architecture

Solution Architecture



Technical Architecture



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/ Displays Line graph / Bar graph.	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 Facebook	I can register & access the my Account	Low	Sprint-2
		USN-4	As a user,I can register for the application through Gmail	I can register through already logged in gmail account.	Medium	Sprint-1
	Login	USN-5	As a user,I can log into the application by entering email & password	After registration, I can log in by only email & password.	High	Sprint-1
	Line\Bar graph		After entering the inputs,the model will display predictions in Line\Bar Graph Format.	I can get the expected prediction in various formats.	High	Sprint-3
Customer (Web user)	Login	USN-1	As the web user,I can login simply by using Gmail or Facebook account.	Already created gmail can be used for Login.	Medium	Sprint-2

Customer Care Executive	Support		The Customer care service will provide solutions for any FAQ and also provide ChatBot.	I can solve the problems arised by Support.	Low	Sprint-3
Administrator	News		Admin will give the recent news of Oil Prices.	Provide the recent oil prices.	High	Sprint-4
	Notification		Admin will notify when the oil prices changes.	Notification by Gmail.	High	Sprint-4
	Access Control		Admin can control the access of users.	Access permission for Users.	High	Sprint-4
	Database		Admin can store the details of users.	Stores User details.	High	Sprint-4

6. PROJECT PLANNING & SCHEDULING

6.1. Sprint Planning & Estimation

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date(Planned)	Story Points Completed (as on Planned End Date)	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	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	18 Nov 2022

6.2. Sprint Delivery Schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	10	High	santhosh
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	10	High	santhosh
Sprint-1	Login	USN-3	As a user, I can log into the application by entering email & password.	15	High	surya
Sprint-2	Input Necessary Details	USN-4	As a user, I can give Input Details to Predict Likelihood of crude oil	15	High	surya
Sprint-2	Data Pre-processing	USN-5	Transform raw data into suitable format for prediction.	15	High	karthik
Sprint-3	Prediction of Crude Oil Price	USN-6	As a user, I can predict Crude oil using machine learning model.	20	High	surya narayanan
Sprint-3		USN-7	As a user, I can get accurate prediction of crude oil	5	Medium	surya narayanan
Sprint-4	Review	USN-8	As a user, I can give feedback of the application.	20	High	karthik

6.3. Reports from JIRA

← → C pnt2022mid13112.atlassian.net/jira/software/projects/COPP/boards/1/backlog

Jira Software Your work Projects Filters Dashboards People Apps Create Search

Crude Oil Price Predict...
Software project

PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings

You're in a team-managed project
Learn more

Does your team need more from Jira? Get a free trial of our Standard plan.

Projects / Crude Oil Price Prediction
Backlog

+ Create issue

+ Backlog (6 issues) 0 0 0 Create sprint

- COPP-1 As a user, I want to collect the dataset so that I can perform price prediction with the data... TO DO
- COPP-2 As a user, I want to perform data preprocessing so that my data is free of redundancies, n... TO DO
- COPP-3 As a user, I want to build the model so that the model can be configured by adding LST... TO DO
- COPP-4 As a user, I want to train the model so that the model can predict results with utmost ac... TO DO
- COPP-5 As a user, I want to develop an application so that the predicted results can be visualized TO DO
- COPP-6 As a user, I want to deploy the model on the IBM cloud TO DO

+ Create issue

Scrum in Jira Software

Turn those large, complex projects into bite-sized pieces of work. Scrum is a way of working, starting in your **Backlog**. This is where you prioritize and choose work that you can get done in a time-box, or sprint. Set a goal, go hard, then reflect, iterate, and do it again.

Show me View best practices

- Create an issue
- Invite your teammates
- Connect your tools
- Get the mobile app
- Find help

Give feedback

Dismiss Quickstart

← → C pnt2022mid13112.atlassian.net/jira/software/projects/COPP/boards/1?insightsOpen=true

Jira Software Your work Projects Filters Dashboards People Apps Create Search

Crude Oil Price Predict...
Software project

PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings

You're in a team-managed project
Learn more

Does your team need more from Jira? Get a free trial of our Standard plan.

Projects / Crude Oil Price Prediction
COPP Sprint 1 5 days remaining Complete sprint

+ Create issue

GROUP BY None Insights

TO DO

IN PROGRESS 1 ISSUE

As a user, I want to perform data preprocessing so that my data is free of redundancies, noisy value, missing values and so on

DATA PREPROCESSING

COPP-2

DONE 1 ISSUE ✓

As a user, I want to collect the dataset so that I can perform price prediction with the data available

DATA COLLECTION

COPP-1

Insights COPP SPRINT 1

Sprint progress
Add estimates to track your progress
This insight helps you stay across the status of work in your sprint, so your team can adjust when needed, and confidently meet sprint goals. [Learn more](#)

Sprint burndown
Add estimates to manage and maintain scope
This insight helps you compare planned worked against completed work, so you can track scope and pivot as needed. [Learn more](#)

Epic progress
Link epics and estimate issues to drive your big goals

Quickstart

Jira Software Your work Projects Filters Dashboards People Apps Create

Crude Oil Price Predict... Software project

PLANNING Roadmap Backlog Board

DEVELOPMENT Code Project pages Add shortcut Project settings

Does your team need more from Jira? Get a free trial of our Standard plan.

Projects / Crude Oil Price Prediction

Backlog

Q Search Epic Insights

▼ COPP Sprint 1 24 Oct – 29 Oct (2 issues) Complete sprint

- COPP-1 As a user, I want to collect the dataset so that I can perform price prediction with the data available DATA COLLECTION 3 DONE
- COPP-2 As a user, I want to perform data preprocessing so that my data is free of redundancies, noisy value, missing values... DATA PREPROCESSING 5 IN PROGRESS

+ Create issue

▼ COPP Sprint 2 31 Oct – 5 Nov (2 issues) Complete sprint

- COPP-3 As a user, I want to build the model so that the model can be configured by adding LSTM and output layers MODEL BUILDING 5 IN PROGRESS
- COPP-4 As a user, I want to train the model so that the model can predict results with utmost accuracy MODEL TRAINING 3 IN PROGRESS

+ Create issue

Jira Software Your work Projects Filters Dashboards People Apps Create

Crude Oil Price Predict... Software project

PLANNING Roadmap Backlog Board

DEVELOPMENT Code Project pages Add shortcut Project settings

Does your team need more from Jira? Get a free trial of our Standard plan.

Projects / Crude Oil Price Prediction

Backlog

Q Search Epic Insights

▼ COPP Sprint 3 23 Oct – 12 Nov (1 issue) Complete sprint

- COPP-5 As a user, I want to develop an application so that the predicted results can be visualized APPLICATION DEVELOPMENT 8 IN PROGRESS

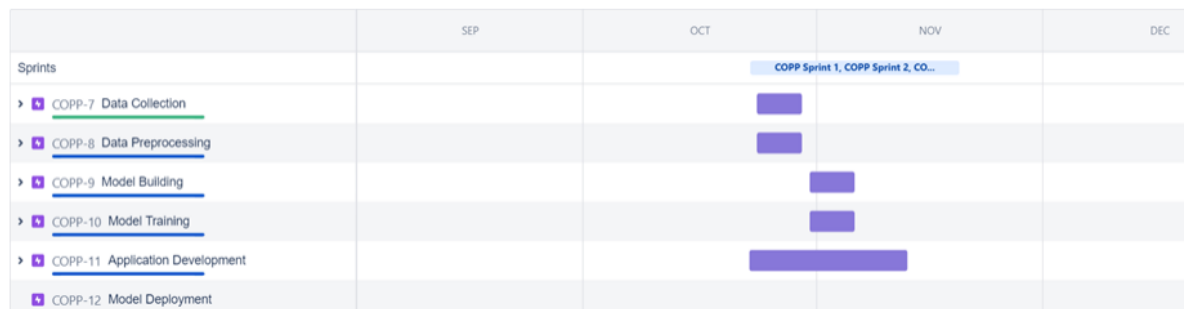
+ Create issue

▼ COPP Sprint 4 23 Oct – 19 Nov (1 issue) Complete sprint

- COPP-6 As a user, I want to deploy the model on the IBM cloud TO DO

+ Create issue

▼ Backlog (0 issues) Create sprint



7. CODING & SOLUTIONING

7.1. Feature 1

```
import numpy as np
```

In [1]:

```
import pandas as pd
```

```
import seaborn as sns
```

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

```
from google.colab import files
```

In [2]:

```
uploaded = files.upload()
```

In [3]:

```
import io
ds = pd.read_excel(io.BytesIO(uploaded['Crude Oil Prices Daily.xlsx']))
ds.head()
ds[:10]
```

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
5	1986-01-09	26.03
6	1986-01-10	25.65
7	1986-01-13	25.08
8	1986-01-14	24.97
9	1986-01-15	25.18

```
ds.isnull().sum()
```

In [4]:

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

Out[4]:

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

In [5]:

```
ds.isnull().sum()
```

In [6]:

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

Out[6]:

```
data=ds.reset_index()['Closing Value']
data
```

In [7]:

```
0      25.56
1      26.00
2      26.53
3      25.85
4      25.87
...
8211   73.89
```

Out[7]:

```
8212    74.19
8213    73.05
8214    73.78
8215    73.93
Name: Closing Value, Length: 8216, dtype: float64
```

In [8]:

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

In [9]:

```
data
```

Out[9]:

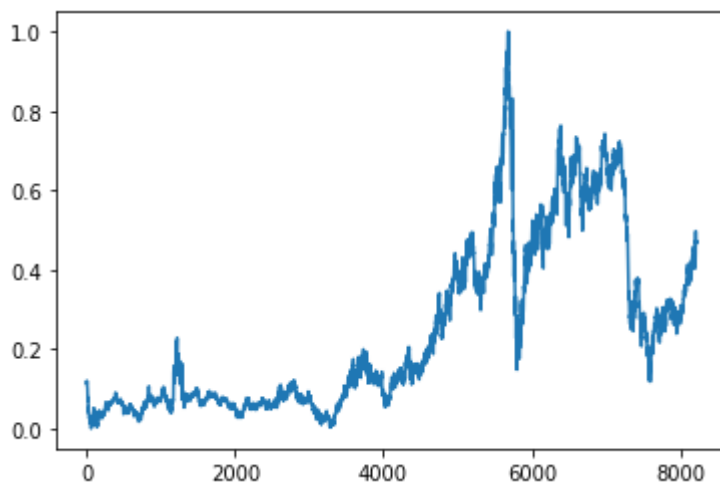
```
array([[0.11335703],
       [0.11661484],
       [0.12053902],
       ...,
       [0.46497853],
       [0.47038353],
       [0.47149415]])
```

In [10]:

```
plt.plot(data)
```

Out[10]:

```
[<matplotlib.lines.Line2D at 0x7f9e733ad2d0>]
```



In [11]:

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

In [12]:

```
training_size,test_size
```

Out[12]:

```
(5340, 2876)
```

In [13]:

```
train_data.shape
```

Out[13]:

```
(5340, 1)
```

In [14]:

```
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)
```

In [15]:

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

In [16]:

```
print(x_train.shape),print(y_train.shape)
(5329, 10)
(5329,)
```

Out[16]:

```
(None, None)
```

In [17]:

```
print(x_test.shape),print(y_test.shape)
(2865, 10)
(2865,)
```

Out[17]:

```
(None, None)
```

In [18]:

```
x_train
```

Out[18]:

```
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 [19]:

```
x_test
```

Out[19]:

```
array([[0.38005331, 0.36872501, 0.37324152, ..., 0.3537687 ,
0.35465719,
      0.3499926 ],
      [0.36872501, 0.37324152, 0.38205242, ..., 0.35465719, 0.3499926
,
      0.3465867 ],
      [0.37324152, 0.38205242, 0.38042352, ..., 0.3499926 , 0.3465867
,
      0.34355101],
      ...,
      [0.40604176, 0.41218718, 0.41041019, ..., 0.46794017,
0.47297497,
      0.47119799],
      [0.41218718, 0.41041019, 0.43513994, ..., 0.47297497,
0.47119799,
```

```

        0.47341922],
        [0.41041019, 0.43513994, 0.4417296 , ..., 0.47119799,
0.47341922,
        0.46497853]])

```

In [20]:

```

x_train1=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 [21]:

```

x_train1

```

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]])])
```

In [22]:

```
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Dense  
from tensorflow.keras.layers import LSTM  
INITIALIZING THE MODEL
```

In [23]:

```
model=Sequential()  
ADDING LSTM AND OUTPUT LAYERS
```

In [24]:

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

In [25]:

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

In [26]:

```
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

CONFIGURING THE LEARNING PROCESS

In [27]:

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

In [28]:

```
model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=3,batch_size=64,verbose=1)  
Epoch 1/3  
84/84 [=====] - 9s 39ms/step - loss: 0.0017 -  
val_loss: 8.1129e-04  
Epoch 2/3  
84/84 [=====] - 2s 24ms/step - loss: 1.2676e-  
04 - val_loss: 7.8078e-04  
Epoch 3/3  
84/84 [=====] - 2s 23ms/step - loss: 1.2624e-  
04 - val_loss: 7.7794e-04
```

Out[28]:

```
<keras.callbacks.History at 0x7f9e150bd650>  
MODEL EVALUATION
```

In [29]:

```

##Transformback to original form
train_predict=scaler.inverse_transform(train_data)
test_predict=scaler.inverse_transform(test_data)
### Calculate RMSE performance metrics
import math
from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(train_data,train_predict))

```

Out[29]:

29.347830443269938

MODEL SAVING

In [30]:

```

from tensorflow.keras.models import load_model

```

In [31]:

```

model.save("crude_oil.hs")
WARNING:absl:Found untraced functions such as lstm_cell_layer_call_fn,
lstm_cell_layer_call_and_return_conditional_losses,
lstm_cell_1_layer_call_fn,
lstm_cell_1_layer_call_and_return_conditional_losses,
lstm_cell_2_layer_call_fn while saving (showing 5 of 6). These
functions will not be directly callable after loading.

```

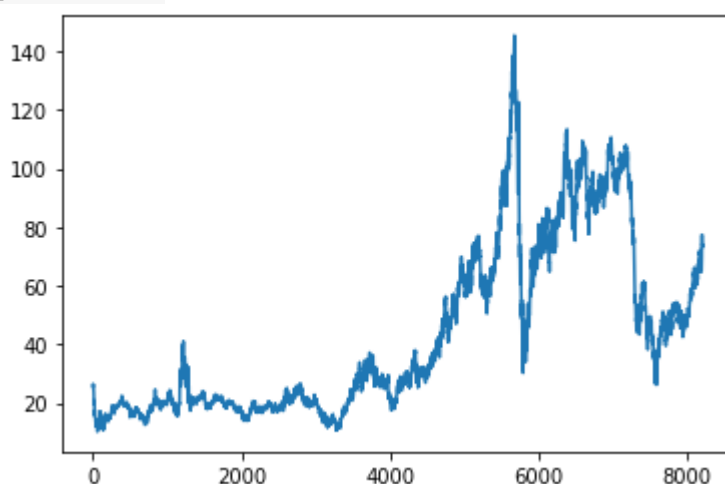
MODEL TESTING

In [32]:

```

### Plotting
look_back=10
trainpredictPlot = np.empty_like(data)
trainpredictPlot[:, :] = np.nan
trainpredictPlot[look_back:len(train_predict)+look_back, :] =
train_predict
# shift test predictions for plotting
testPredictplot = np.empty_like(data)
testPredictplot[:, :] = np.nan
testPredictplot[look_back:len(test_predict)+look_back, :] =
test_predict
# plot baseline and predictions
plt.plot(scaler.inverse_transform(data))
plt.show()

```



In [33]:

```

len(test_data)

```

Out[33]:

2876

In [34]:


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

Out[34]:

```
(1, 10)
```

In [35]:

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

In [36]:

```
temp_input
```

Out[36]:

```
[0.44172960165852215,
0.48111950244335855,
0.49726047682511476,
0.4679401747371539,
0.4729749740855915,
0.47119798608026064,
0.47341922108692425,
0.4649785280616022,
0.4703835332444839,
0.47149415074781587]
```

In [37]:

```
lst_output=[]
n_steps=10
i=0
while(i<10):
    if(len(temp_input)>10):
        #print(temp_input)
        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)) #print(x_input)
        yhat = model.predict(x_input, verbose=0)
        print("{} day output {}".format(i,yhat))
        temp_input.extend(yhat[0].tolist())
        temp_input=temp_input[1:] #print(temp_input)
        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
[0.4805713]
11
1 day input [0.4811195  0.49726048 0.46794017 0.47297497 0.47119799
0.47341922
0.46497853 0.47038353 0.47149415 0.4805713 ]
1 day output [[0.4844224]]
2 day input [0.49726048 0.46794017 0.47297497 0.47119799 0.47341922
0.46497853
0.47038353 0.47149415 0.4805713  0.48442239]
2 day output [[0.4833879]]
3 day input [0.46794017 0.47297497 0.47119799 0.47341922 0.46497853
0.47038353
0.47149415 0.4805713  0.48442239 0.48338789]
```

```

3 day output [[0.48069027]]
4 day input [0.47297497 0.47119799 0.47341922 0.46497853 0.47038353
0.47149415
0.4805713 0.48442239 0.48338789 0.48069027]
4 day output [[0.4820817]]
5 day input [0.47119799 0.47341922 0.46497853 0.47038353 0.47149415
0.4805713
0.48442239 0.48338789 0.48069027 0.48208171]
5 day output [[0.48304394]]
6 day input [0.47341922 0.46497853 0.47038353 0.47149415
0.4805713 0.48442239
0.48338789 0.48069027 0.48208171 0.48304394]
6 day output [[0.48441863]]
7 day input [0.46497853 0.47038353 0.47149415 0.4805713 0.48442239
0.48338789
0.48069027 0.48208171 0.48304394 0.48441863]
7 day output [[0.48566842]]
8 day input [0.47038353 0.47149415 0.4805713 0.48442239 0.48338789
0.48069027
0.48208171 0.48304394 0.48441863 0.48566842]
8 day output [[0.48811078]]
9 day input [0.47149415 0.4805713 0.48442239 0.48338789 0.48069027
0.48208171
0.48304394 0.48441863 0.48566842 0.48811078]
9 day output [[0.48995987]]

```

In [38]:

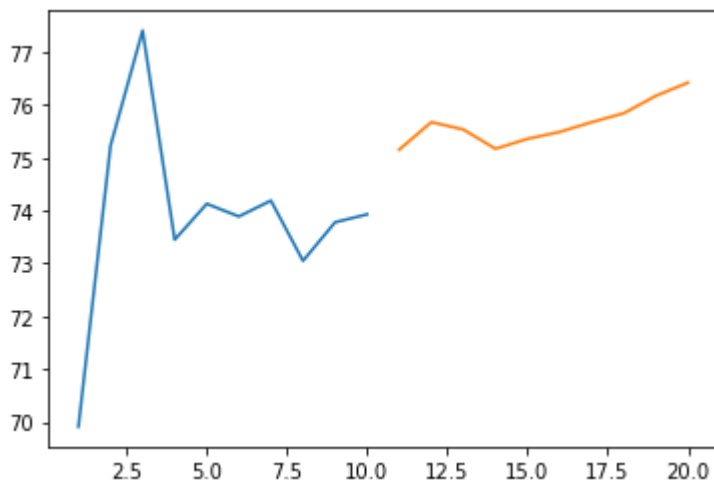
```

day_new=np.arange(1,11)
day_pred=np.arange(11,21)
len(data)
plt.plot(day_new, scaler.inverse_transform(data[8206:]))
plt.plot(day_pred, scaler.inverse_transform(lst_output))

```

Out[38]:

[<matplotlib.lines.Line2D at 0x7f9e151ef6d0>]



In [39]:

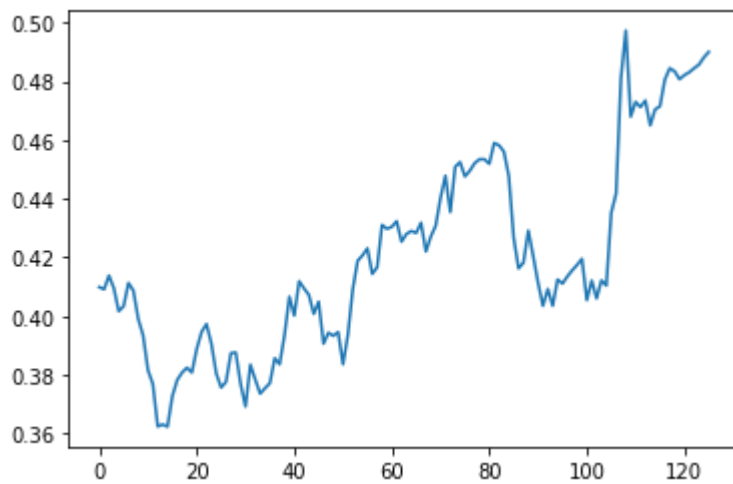
```

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

```

Out[39]:

[<matplotlib.lines.Line2D at 0x7f9e10cc3d10>]

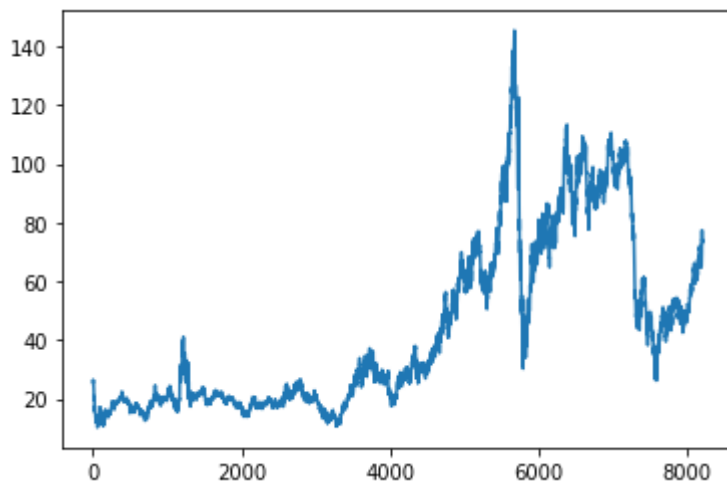


In [40]:

```
df3=scaler.inverse_transform(df3).tolist()
plt.plot(scaler.inverse_transform(data))
```

Out[40]:

[<matplotlib.lines.Line2D at 0x7f9e10f89e10>]



7.2 Feature 2

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Home page</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>

  <div class="main">
    <div class="navbar">
      <div class="icon">
        <h2 class="logo">CRUDE OIL</h2>
      </div>
```

```

<div class="menu">
  <ul>
    <li><a href="#">HOME</a></li>
    <li><a href="#">ABOUT</a></li>
    <li><a href="#">SERVICE</a></li>
    <li><a href="#">CONTACT</a></li>
  </ul>
</div>

<div class="search">
  <input class="srch" type="search" name="" placeholder="Type To text">
  <a href="#"> <button class="btn">Search</button></a>
</div>

</div>
<div class="content">
  <h1>Crude Oil<br><span>Price Prediction</span><br></h1>
  <p class="par"> Crude oil means a mixture of hydrocarbons that exists in
liquid phase in<br>
natural underground reservoirs and remains liquid <br>at atmospheric
pressure
after passing through <br>surface separating facilities.</p>

  <button class="cn"><a href="register.html">JOIN US</a></button>

  <div class="form">
    <h2>Login Here</h2>
    <input type="email" name="email" placeholder="Enter Email Here">
    <input type="password" name="" placeholder="Enter Password Here">
    <button class="btnn"><a href="#">Login</a></button>

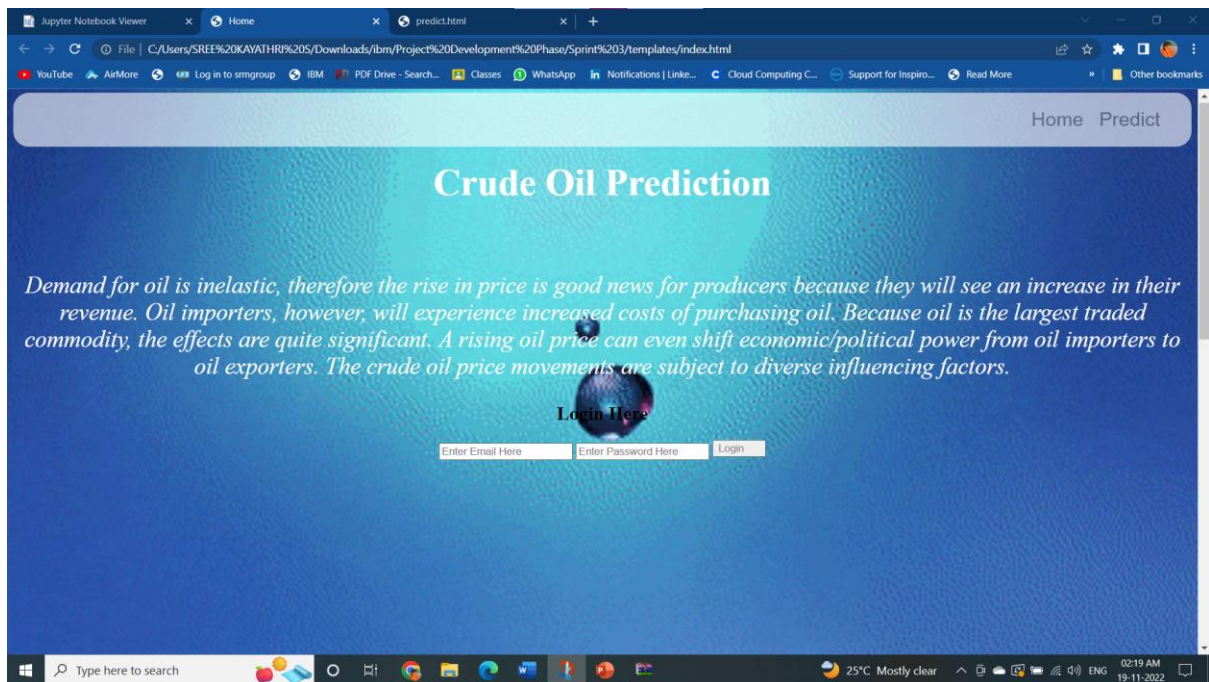
    <p class="link">Don't have an account<br>
    <a href="#">Sign up </a> here</a></p>
    <p class="liw">Log in with</p>

    <div class="icons">
      <a href="#"><ion-icon name="logo-facebook"></ion-icon></a>
      <a href="#"><ion-icon name="logo-google"></ion-icon></a>
    </div>

  </div>
</div>
</div>
</div>
<script src="https://unpkg.com/ionicons@5.4.0/dist/ionicons.js"></script>
</body>

```

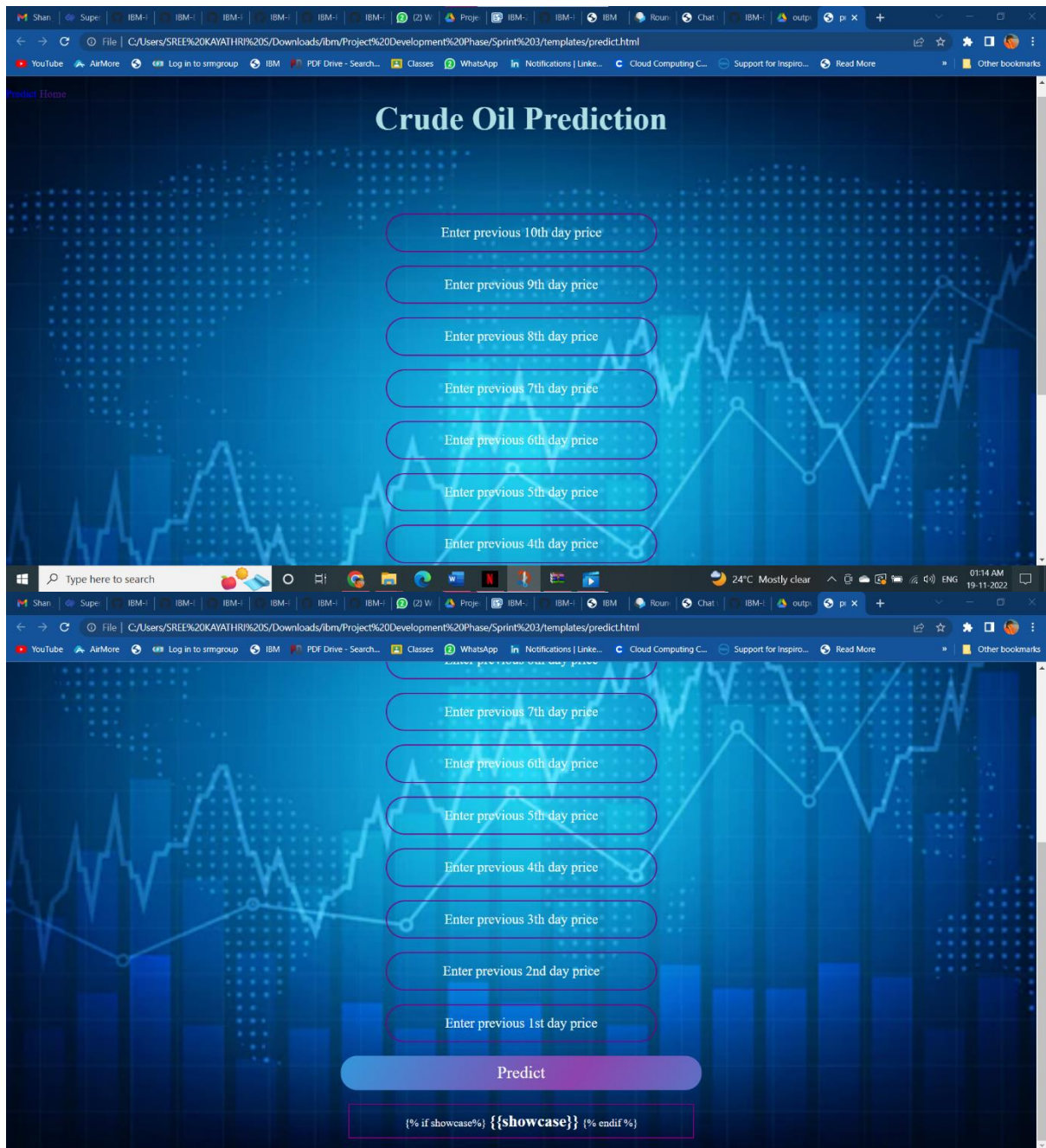
</html>



predict.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Registration Form</title>
    <link rel="stylesheet"
      href="register.css" type="text/css">
  </head>
  <body>
    <div class="main">
      <div class="register">
        <h2>Register Here</h2>
        <form id="register" method="post">
          <label>First Name : </label>
          <br>
          <input type="text" name="fname"
            id="name" placeholder="Enter Your First Name">
          <br><br>
          <label>Last Name : </label>
          <br>
          <input type="text" name="lname"
            id="name" placeholder="Enter Your last Name">
          <br><br>
          <label>Your Age : </label>
          <br>
          <input type="number" name="age">
```

[illegible]



8. TESTING

8.1 Test Cases

Test case analysis This report shows the number of test cases that have passed, failed, and untested.

Section	Total Cases	Not Tested	Fail	Pass
ML Model	4	0	0	4
Flask Application	4	0	0	4

IBM cloud	4	0	0	4
Exception Reporting	2	0	0	2
Final Report output	4	0	0	4

8.2 User Acceptance Testing

The purpose is to briefly explain the test coverage and open issues of the crude oil price prediction project at the time of the release to user acceptance testing

Defect Analysis:

Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't fix	0	0	0	1	1
Totals	8	0	2	2	12

Test case analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
ML Model	4	0	0	4
Flask Application	4	0	0	4
IBM Cloud	4	0	0	4
Exception Reporting	2	0	0	2
Final Report Output	4	0	0	4

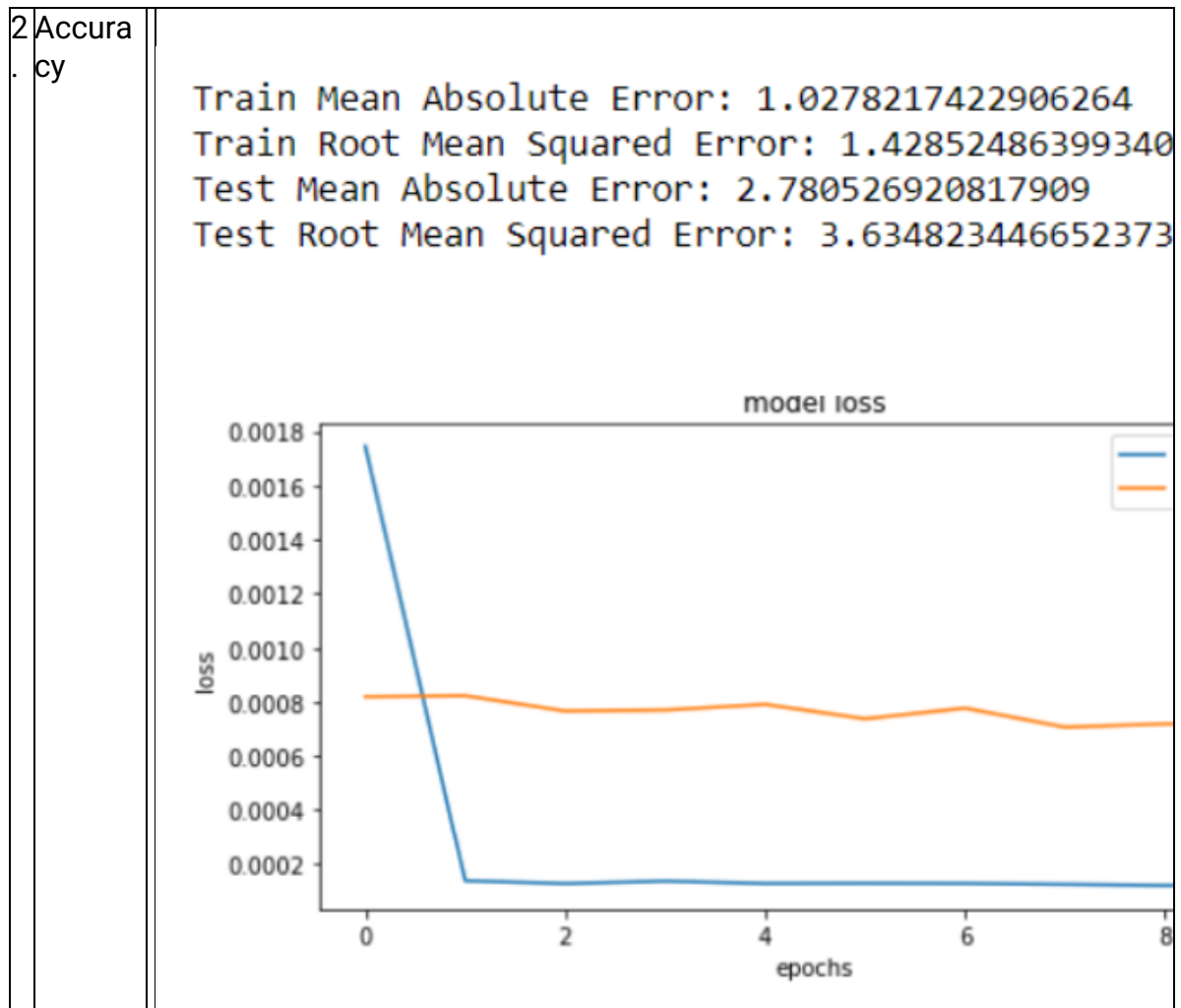
The report shows the number of resolved and closed bugs at each severity level and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	3	0	0	0	3
Duplicate	1	0	1	0	2
External	0	0	0	0	0
Fixed	4	0	1	1	6

9. RESULTS

9.1 Performance Metrics

S.No	Parameters	Values	Screenshot																				
1.	Model Summary		<div>Model: "sequential_1"</div> <table><thead><tr><th>Layer (type)</th><th>Output Shape</th></tr></thead><tbody><tr><td colspan="2">=====</td></tr><tr><td>lstm_3 (LSTM)</td><td>(None, 10, 50)</td></tr><tr><td>lstm_4 (LSTM)</td><td>(None, 10, 50)</td></tr><tr><td>lstm_5 (LSTM)</td><td>(None, 50)</td></tr><tr><td>dense_1 (Dense)</td><td>(None, 1)</td></tr><tr><td colspan="2">=====</td></tr><tr><td colspan="2">Total params: 50,851</td></tr><tr><td colspan="2">Trainable params: 50,851</td></tr><tr><td colspan="2">Non-trainable params: 0</td></tr></tbody></table>	Layer (type)	Output Shape	=====		lstm_3 (LSTM)	(None, 10, 50)	lstm_4 (LSTM)	(None, 10, 50)	lstm_5 (LSTM)	(None, 50)	dense_1 (Dense)	(None, 1)	=====		Total params: 50,851		Trainable params: 50,851		Non-trainable params: 0	
Layer (type)	Output Shape																						
=====																							
lstm_3 (LSTM)	(None, 10, 50)																						
lstm_4 (LSTM)	(None, 10, 50)																						
lstm_5 (LSTM)	(None, 50)																						
dense_1 (Dense)	(None, 1)																						
=====																							
Total params: 50,851																							
Trainable params: 50,851																							
Non-trainable params: 0																							



10. ADVANTAGES & DISADVANTAGES

Advantages:

- Prediction of crude oil price can help the importers to choose the right time to buy as they wait for the prices to fall down
- Prediction of crude oil prices can help the exporters to increase the demand
- It can even help in shifting the political powers
- can assist in minimizing the risks associated with volatility in oil prices

Disadvantages

- The prediction results may lack accuracy
- Volatility in prices may be misleading

11. CONCLUSION

LSTM network is better than other traditional neural networks for forecasting prices as it aims in using a back propagation model. Traditional neural networks such as CNN on the other hand predicts the next outgoing but doesn't necessarily save the previous data or connection which is based on feed-forwarding, in the sense the

previous data is not necessary to predict the future data. LSTM focuses on storing the previous data and prediction which is rather encouraging and more approximate. The outcomes derived are relatively encouraging. The results show that large lookups do not necessarily improve the accuracy of the predictions of crude oil prices. Hence it can be concluded, the model with a single LSTM model is definitely the most accurate.

12. FUTURE SCOPE

The project's future potential is enormous. The project can be implemented with the real-time functionalities that are necessary. Because it is quite versatile in terms of expansion, the project can be upgraded in the near future as and when the need arises. The complete prediction value can be increased in a much better, accurate, and error-free manner with the proposed approach. The project can be enhanced with real time data.

13. APPENDIX

Source Code

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

In []:

```
from google.colab import files
uploaded = files.upload()
```

In []:

```
import io
ds = pd.read_excel(io.BytesIO(uploaded['Crude Oil Prices Daily.xlsx']))
ds.head()
ds[:10]
```

Out[]:

	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
5	1986-01-09	26.03
6	1986-01-10	25.65

7	1986-01-13	25.08
8	1986-01-14	24.97
9	1986-01-15	25.18

```
ds.isnull().sum()
```

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

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

```
ds.isnull().sum()
```

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

```
data=ds.reset_index()['Closing Value']
data
```

```
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
```

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

```
data
```

```
array([[0.11335703],
       [0.11661484],
       [0.12053902],
       ...,
       [0.46497853],
       [0.47038353],
       [0.47149415]])
```

```
plt.plot(data)
```

```
[<matplotlib.lines.Line2D at 0x7f9e733ad2d0>]
```

In []:

Out[]:

In []:

In []:

Out[]:

In []:

Out[]:

In []:

In []:

Out[]:

In []:

Out[]:

In []:

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

In []:

```
training_size,test_size
```

Out[]:

```
(5340, 2876)
```

In []:

```
train_data.shape
```

Out[]:

```
(5340, 1)
```

In []:

```
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)
```

In []:

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

In []:

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

```
(5329, 10)
```

```
(5329,)
```

Out[]:

```
(None, None)
```

In []:

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

```
(2865, 10)
```

```
(2865,)
```

Out[]:

```
(None, None)
```

In []:

```
x_train
```

Out[]:

```
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 []:

```
x_test
```

Out[]:

```
array([[0.38005331, 0.36872501, 0.37324152, ..., 0.3537687 ,
        0.35465719,
        0.3499926 ],
       [0.36872501, 0.37324152, 0.38205242, ..., 0.35465719, 0.3499926
        ,
        0.3465867 ],
       [0.37324152, 0.38205242, 0.38042352, ..., 0.3499926 , 0.3465867
        ,
        0.34355101],
       ...,
       [0.40604176, 0.41218718, 0.41041019, ..., 0.46794017,
        0.47297497,
        0.47119799],
       [0.41218718, 0.41041019, 0.43513994, ..., 0.47297497,
        0.47119799,
        0.47341922],
       [0.41041019, 0.43513994, 0.4417296 , ..., 0.47119799,
        0.47341922,
        0.46497853]])
```

In []:

```
x_train1=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 []:

```
x_train1
```

Out[]:

```
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 []:

```

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
INITIALIZING THE MODEL

```

In []:

```

model=Sequential()
ADDING LSTM AND OUTPUT LAYERS

```

In []:

```

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

```

In []:

```

model.add(Dense(1))

```

In []:

```

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		

CONFIGURING THE LEARNING PROCESS

In []:

```

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

```

```
In [ ]:
model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=3,batch_size=64,verbose=1)
Epoch 1/3
84/84 [=====] - 9s 39ms/step - loss: 0.0017 - val_loss: 8.1129e-04
Epoch 2/3
84/84 [=====] - 2s 24ms/step - loss: 1.2676e-04 - val_loss: 7.8078e-04
Epoch 3/3
84/84 [=====] - 2s 23ms/step - loss: 1.2624e-04 - val_loss: 7.7794e-04
```

```
<keras.callbacks.History at 0x7f9e150bd650>
MODEL EVALUATION
```

```
In [ ]:
##Transformback to original form
train_predict=scaler.inverse_transform(train_data)
test_predict=scaler.inverse_transform(test_data)
### Calculate RMSE performance metrics
import math
from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(train_data,train_predict))
```

```
29.347830443269938
MODEL SAVING
```

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

```
model.save("crude_oil.hs")
WARNING:absl:Found untraced functions such as lstm_cell_layer_call_fn,
lstm_cell_layer_call_and_return_conditional_losses,
lstm_cell_1_layer_call_fn,
lstm_cell_1_layer_call_and_return_conditional_losses,
lstm_cell_2_layer_call_fn while saving (showing 5 of 6). These
functions will not be directly callable after loading.
```

MODEL TESTING

```
In [ ]:
### Plotting
look_back=10
trainpredictPlot = np.empty_like(data)
trainpredictPlot[:, :]= np.nan
trainpredictPlot[look_back:len(train_predict)+look_back, :] =
train_predict
# shift test predictions for plotting
testPredictplot = np.empty_like(data)
testPredictplot[:, : ] = np.nan
testPredictplot[look_back:len(test_predict)+look_back, :] =
test_predict
# plot baseline and predictions
plt.plot(scaler.inverse_transform(data))
plt.show()
```

```
len(test_data)
```


2876

In []:

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

Out[]:

```
(1, 10)
```

In []:

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

In []:

```
temp_input
```

Out[]:

```
[0.44172960165852215,
0.48111950244335855,
0.49726047682511476,
0.4679401747371539,
0.4729749740855915,
0.47119798608026064,
0.47341922108692425,
0.4649785280616022,
0.4703835332444839,
0.47149415074781587]
```

In []:

```
lst_output=[]
n_steps=10
i=0
while(i<10):
    if(len(temp_input)>10):
        #print(temp_input)
        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)) #print(x_input)
        yhat = model.predict(x_input, verbose=0)
        print("{} day output {}".format(i,yhat))
        temp_input.extend(yhat[0].tolist())
        temp_input=temp_input[1:] #print(temp_input)
        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
[0.4805713]
11
1 day input [0.4811195  0.49726048 0.46794017 0.47297497 0.47119799
0.47341922
0.46497853 0.47038353 0.47149415 0.4805713 ]
1 day output [[0.4844224]]
2 day input [0.49726048 0.46794017 0.47297497 0.47119799 0.47341922
0.46497853
0.47038353 0.47149415 0.4805713  0.48442239]
2 day output [[0.4833879]]
3 day input [0.46794017 0.47297497 0.47119799 0.47341922 0.46497853
```

```

0.47038353
0.47149415 0.4805713 0.48442239 0.48338789]
3 day output [[0.48069027]]
4 day input [0.47297497 0.47119799 0.47341922 0.46497853 0.47038353
0.47149415
0.4805713 0.48442239 0.48338789 0.48069027]
4 day output [[0.4820817]]
5 day input [0.47119799 0.47341922 0.46497853 0.47038353 0.47149415
0.4805713
0.48442239 0.48338789 0.48069027 0.48208171]
5 day output [[0.48304394]]
6 day input [0.47341922 0.46497853 0.47038353 0.47149415
0.4805713 0.48442239
0.48338789 0.48069027 0.48208171 0.48304394]
6 day output [[0.48441863]]
7 day input [0.46497853 0.47038353 0.47149415 0.4805713 0.48442239
0.48338789
0.48069027 0.48208171 0.48304394 0.48441863]
7 day output [[0.48566842]]
8 day input [0.47038353 0.47149415 0.4805713 0.48442239 0.48338789
0.48069027
0.48208171 0.48304394 0.48441863 0.48566842]
8 day output [[0.48811078]]
9 day input [0.47149415 0.4805713 0.48442239 0.48338789 0.48069027
0.48208171
0.48304394 0.48441863 0.48566842 0.48811078]
9 day output [[0.48995987]]

```

In []:

```

day_new=np.arange(1,11)
day_pred=np.arange(11,21)
len(data)
plt.plot(day_new, scaler.inverse_transform(data[8206:]))
plt.plot(day_pred, scaler.inverse_transform(lst_output))

```

Out[]:

```
[<matplotlib.lines.Line2D at 0x7f9e151ef6d0>]
```

In []:

```

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

```

Out[]:

```
[<matplotlib.lines.Line2D at 0x7f9e10cc3d10>]
```

In []:

```

df3=scaler.inverse_transform(df3).tolist()
plt.plot(scaler.inverse_transform(data))

```

Out[]:

```
[<matplotlib.lines.Line2D at 0x7f9e10f89e10>]
```

7.2 Feature 2

index.html

```

<!DOCTYPE html>
<html lang="en">
<head>

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Home page</title>
<link rel="stylesheet" href="style.css">
</head>
<body>

<div class="main">
  <div class="navbar">
    <div class="icon">
      <h2 class="logo">CRUDE OIL</h2>
    </div>

    <div class="menu">
      <ul>
        <li><a href="#">HOME</a></li>
        <li><a href="#">ABOUT</a></li>
        <li><a href="#">SERVICE</a></li>
        <li><a href="#">CONTACT</a></li>
      </ul>
    </div>

    <div class="search">
      <input class="srch" type="search" name="" placeholder="Type To text">
      <a href="#"><button class="btn">Search</button></a>
    </div>

  </div>
  <div class="content">
    <h1>Crude Oil<br><span>Price Prediction</span><br></h1>
    <p class="par"> Crude oil means a mixture of hydrocarbons that exists in
liquid phase in<br>
natural underground reservoirs and remains liquid <br>at atmospheric
pressure
after passing through <br>surface separating facilities.</p>

    <button class="cn"><a href="register.html">JOIN US</a></button>

    <div class="form">
      <h2>Login Here</h2>
      <input type="email" name="email" placeholder="Enter Email Here">
      <input type="password" name="" placeholder="Enter Password Here">
      <button class="bttn"><a href="#">Login</a></button>

      <p class="link">Don't have an account<br>
<a href="#">Sign up </a> here</p>
      <p class="liw">Log in with</p>

      <div class="icons">

```

```

        <a href="#"><ion-icon name="logo-facebook"></ion-icon></a>
        <a href="#"><ion-icon name="logo-google"></ion-icon></a>
    </div>

</div>
</div>
</div>
</div>
<script src="https://unpkg.com/ionicons@5.4.0/dist/ionicons.js"></script>
</body>
</html>

```

predict.html

```

<!DOCTYPE html>
<html>
    <head>
        <title>Registration Form</title>
        <link rel="stylesheet"
            href="register.css" type="text/css">
    </head>
    <body>
        <div class="main">
            <div class="register">
                <h2>Register Here</h2>
                <form id="register" method="post">
                    <label>First Name : </label>
                    <br>
                    <input type="text" name="fname"
                        id="name" placeholder="Enter Your First Name">
                    <br><br>
                    <label>Last Name : </label>
                    <br>
                    <input type="text" name="lname"
                        id="name" placeholder="Enter Your last Name">
                    <br><br>
                    <label>Your Age : </label>
                    <br>
                    <input type="number" name="age"
                        id="name" placeholder="How Old Are You">
                    <br><br>
                    <label>Email : </label>
                    <br>
                    <input type="email" name="email"
                        id="name" placeholder="Enter Your Valid Email">
                    <br><br>
                    <label>Gender : </label>
                    <br>

```

```

        &nbsp; &nbsp; &nbsp; &nbsp;
        <input type="radio" name="gender"
        id="male">
        &nbsp;
        <span id="male">Male</span>
        &nbsp; &nbsp; &nbsp; &nbsp;
        <input type="radio" name="gender"
        id="female">
        &nbsp;
        <span id="female">Female</span>
        <br><br>
        <input type="submit" value="Submit"
        name="submit" id="submit">
    </form>
</div>
</div>
</body>
</html>

```

index.css

```

h1 {
    text-align: center;
    color: floralwhite;
    font-size: 50px;
    font-family: roboto;
}

p {
    font-family: roboto;
    color: ghostwhite;
    margin-right: 30px;
    margin-left: 30px;
    text-align: center;
    font-size: 20px;
    font-weight: bold;
}

body {
    background: url(index.png);
    background-repeat: no-repeat;
    background-size: cover;
}

.button {
    display: inline-block;
    border-radius: 4px;
    background-color: black;
    border: none;
    color: #FFFFFF;
}

```

```

        text-align: center;
        font-size: 20px;
        padding: 12px;
        width: 100px;
        transition: all 0.5s;
        cursor: pointer;
        margin: 5px;
    }

    a {
        font-size: 20px;
        font-family: roboto;
        color: ghostwhite;
        margin-right: 30px;
        margin-left: 30px;
        text-align: center;
        font-size: 20px;
        font-weight: bold;
    }

    table {
        background: slateblue;
        opacity: 0.8;
        margin-left:auto;
        margin-right:auto;
        margin-bottom: 0px;
    }

    th,
    td {
        text-align: left;
        color: black;
        font-size: 30px;
        font-family: roboto;
    }

```

Predict.css

```

body{
    background: url(index.png);
    background-repeat: no-repeat;
    background-size: cover;
}

```

App.py:

```

from flask import Flask, render_template, request, redirect
import numpy as np
# from tensorflow.k

```

```

from keras.saving.save import load_model
app = Flask(__name__, template_folder='template')

```

```

@app.route('/', methods=["GET"]) def index():
    return render_template('index.html')

@app.route('/predict.html', methods=["POST", "GET"])@app.route('/method',
methods=["POST", "GET"])
def method():
    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
    val = lst_output[9]
    return render_template('predict.html', prediction=val)

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

if __name__ == "__main__":
    model =load_model(r'crudeoilprediction.h5') app.run(debug=True)

#cloud deployment code in ml model

!pip installibm_watson_machine_learning

from ibm_watson_machine_learning importAPIClient wml_credentials = (
    "url" : "https://us-south.ml.cloud.ibm.com",
    "apikey" : "cRkqykhsnLO1 Ogs_xoYjgLkNTtTS1QxyioMn1GSIQ1P5" client=
    APIClient(wml_credentials)
    #for creatinga deployment phase
    def guid_from_space_name(client, space_name): space=client.spaces.get_details()
    #print(space)
    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) client.software_specifications.list()
software_spec_uid=
client.software_specifications.get_uid_by_name("tensorflow_rt22.1-py3.9")
software_spec_uid
```

GitHub Link

<https://github.com/IBM-EPBL/IBM-Project-1115-1658375264>

Project Demo

https://youtu.be/vUR_M0IF7qM