CRUDE OIL PRICE PREDICTION
USING LSTM

A PROJECT REPORT

Submitted by

TEAM ID: PNT2022TMID35878

MADHAVAN P T S 2019504543

HARISH VEERARAGHAVAN 2019504527

SANTHOSH S 2019504578

YUVAN RAJU M 2019504608

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

MADRAS INSTITUTE OF TECHNOLOGY

CHROMPET

ANNA UNIVERSITY: CHENNAI 600 025

NOVEMBER 2022

1.Introduction

1.1 Project Overview

As financial institutions begin to embrace artificial intelligence, machine learning is increasingly utilized to help make trading decisions. Although there is an abundance of crude oil data for machine learning models to train on, a high noise to signal ratio and the multitude of factors that affect crude oil prices are among the several reasons that predicting the market difficult. At the same time, these models don't need to reach high levels of accuracy because even 60% accuracy can deliver solid returns. One method for predicting stock prices is using a long short-term memory neural network (LSTM) for times series forecasting.

1.2 Purpose

Crude oil Price Prediction using machine learning helps you discover the future value of oil and other financial assets traded on an exchange. The entire idea of predicting stock prices is to gain significant profits. Predicting how the crude oil market will perform is a hard task to do. There are other factors involved in the prediction, such as physical and psychological factors, rational and irrational behaviour, and so on. All these factors combine to make share prices dynamic and volatile. This makes it very difficult to predict oil prices with high accuracy.

2.Literature Survey

2.1 Existing Problem

As the most important strategic resource around the globe, crude oil is the key commodity for the world's economy. Therefore, forecasting crude oil price has always been a very challenging task which drew the interest of researchers, practitioners and institutions. The price of oil is determined by its supply and demand, it is also strongly related to irregular and unforeseen events caused by weather, wars, revolutions, etc. Many other factors like Gross Domestic Production growth, stock levels inventories, foreign exchange rates, world population, political aspects and investor's expectations can significantly affect the price of oil. Furthermore, the time to ship crude oil from one country to another can affect directly their price because oil prices vary in different regions of the worldwide. All these factors can explain the nonlinear evolution and chaotic behaviour of crude oil prices and therefore the high volatility of crude oil market. The oil price fluctuations have a direct effect on the nation's economy therefore, it is of vital importance to predict oil price.

2.2 References

Manel Hamdi, Chaker Aloui (2014) "Forecasting Crude Oil Price Using Artificial Neural Networks: A Literature Survey" Economic Bulletin

Hasim Sak, Andrew Senior, Francoise Beaufays "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling" Google, USA

V Kranthi Sai Reddy (2018) "Stock Market Prediction Using Machine Learning" International Research Journal of Engineering and Technology (IRJET)

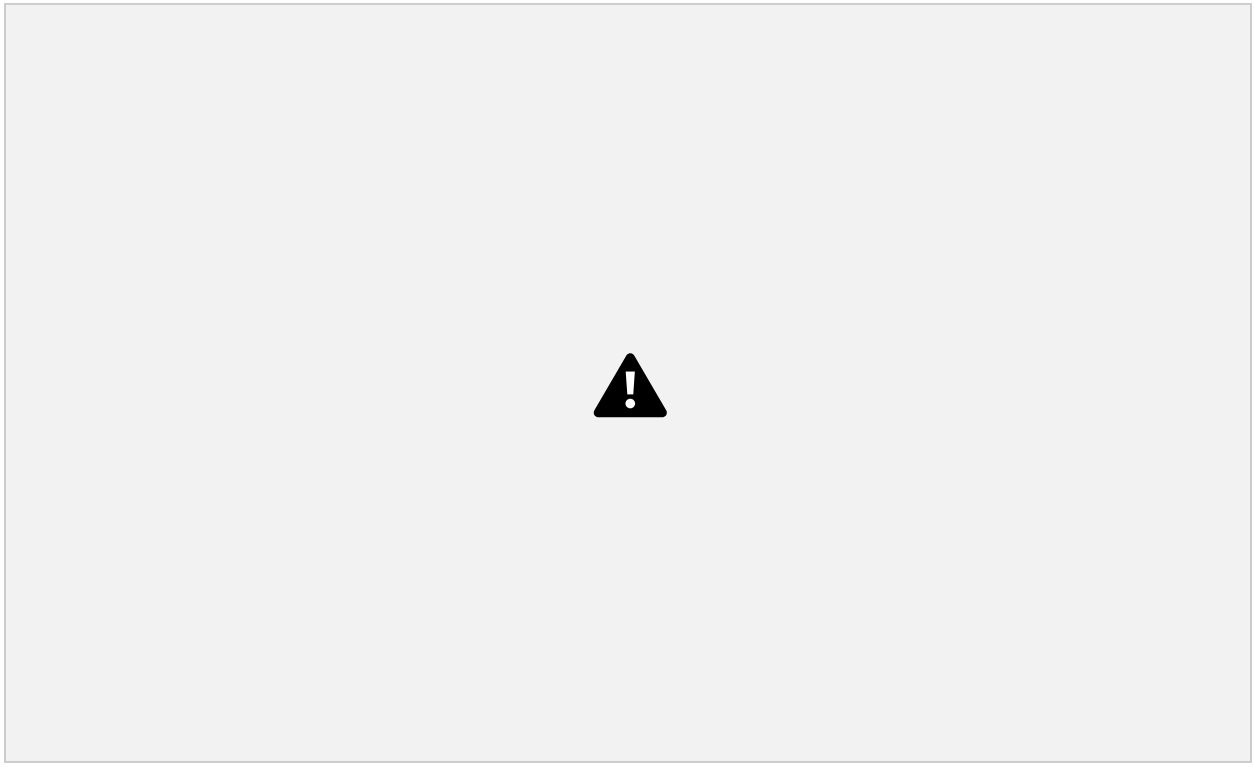
2.3 Problem Statement Definition

The crude oil market is known for being volatile, dynamic, and nonlinear. Accurate crude oil price prediction is extremely challenging because of multiple (macro and micro) factors, such as politics, global economic conditions, unexpected events, a company's financial performance, and so on.

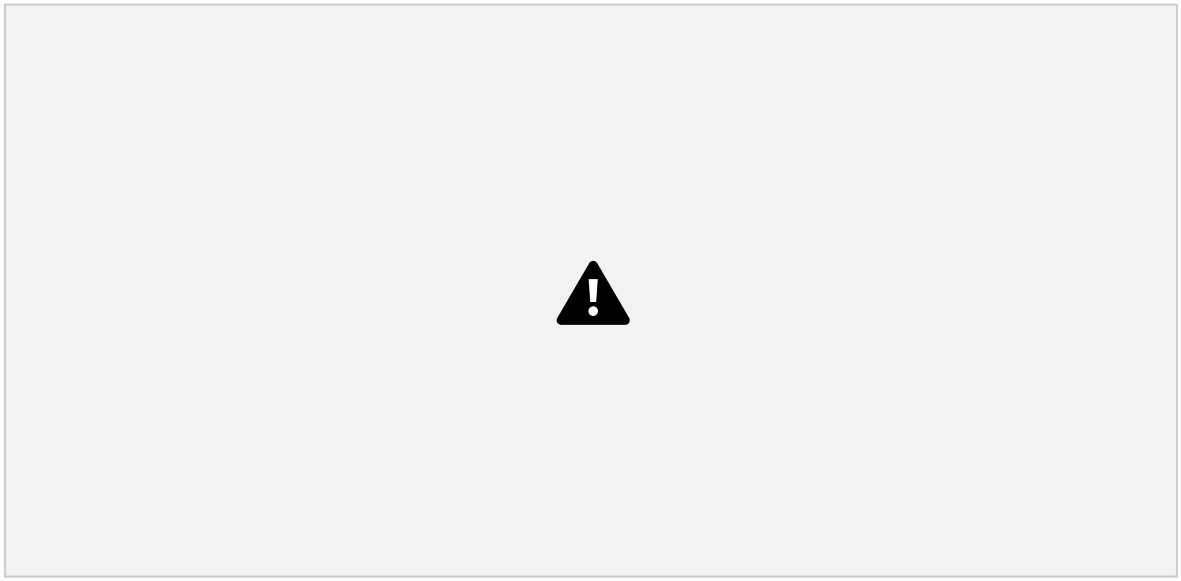
But, all of this also means that there's a lot of data to find patterns in. So, financial analysts, researchers, and data scientists keep exploring analytics techniques to detect crude oil market trends. This gave rise to the concept of algorithmic trends, which uses automated, pre-programmed trading strategies to execute orders.

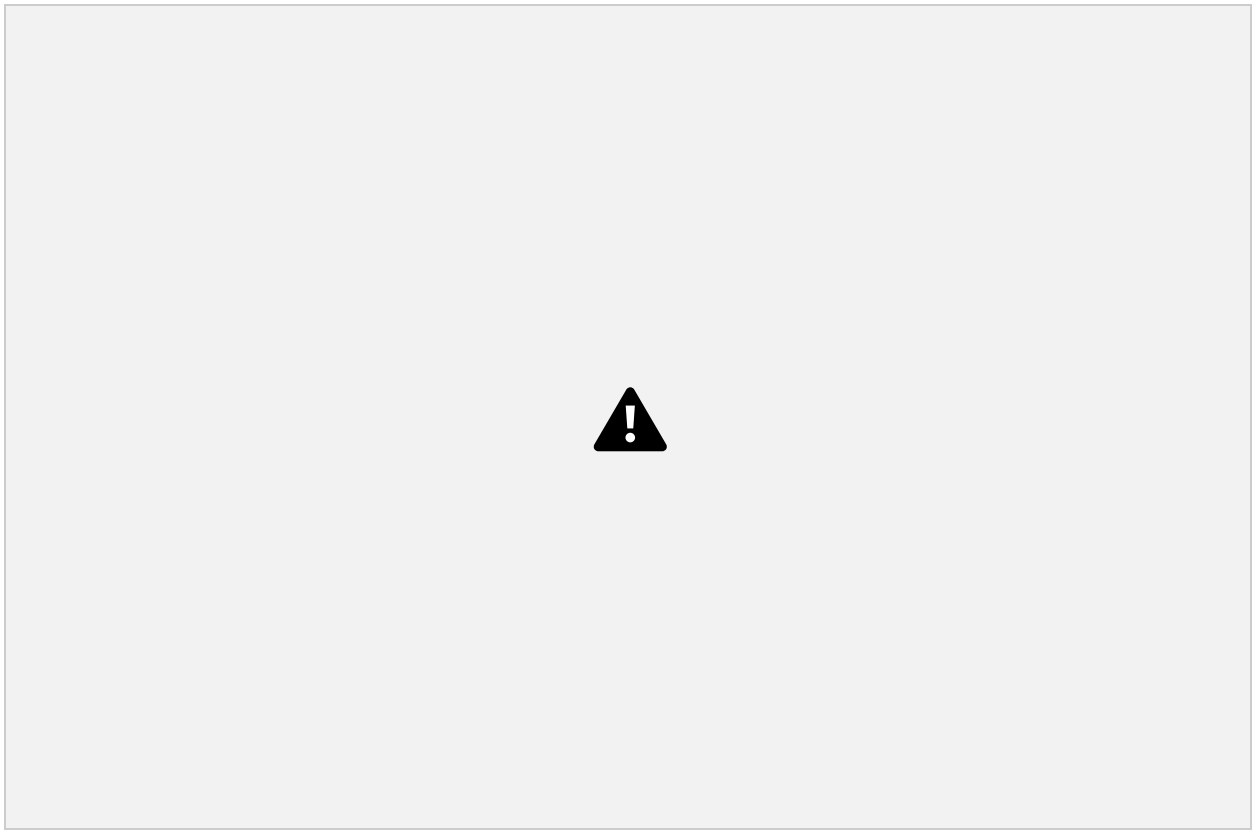
3.Ideation and Proposed Solution

3.1 Empathy Map Canvas



3.2 Ideation and Brainstorming



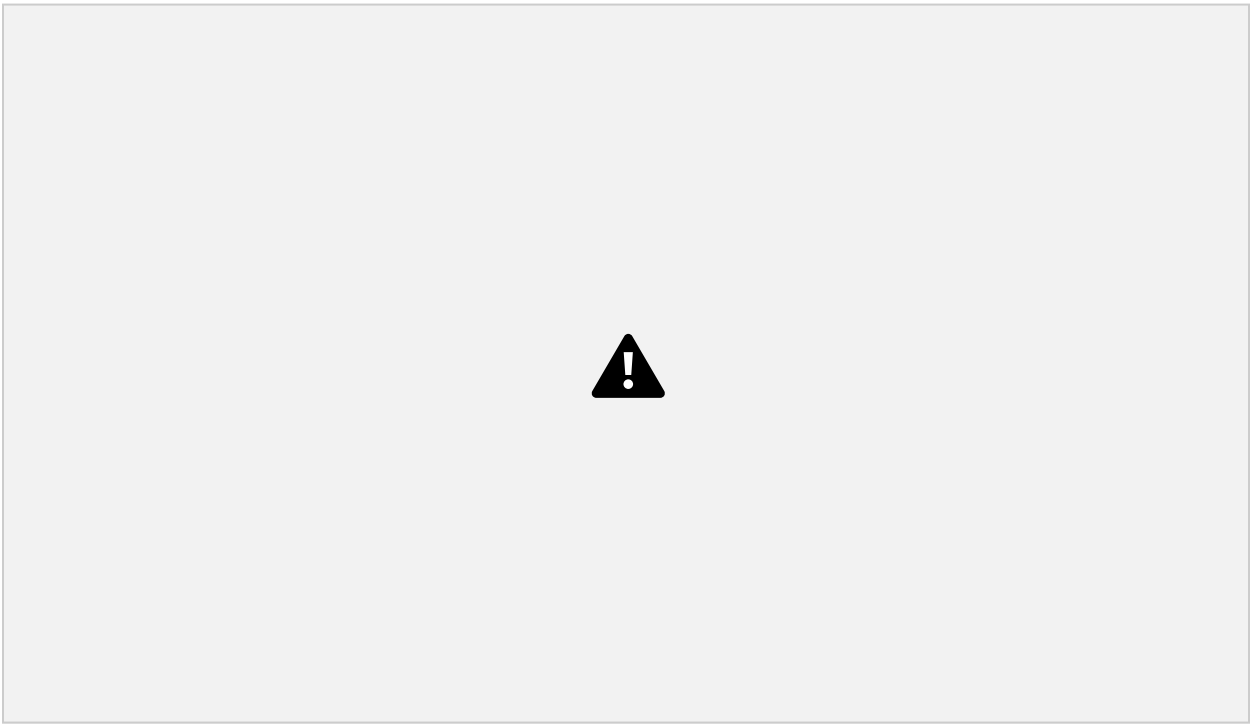


3.3 Proposed Solution

Crude oil is the world's leading fuel, and its prices have a big impact on the global environment and its forecasts are very useful to governments, industry is individuals. The continuous usage of statistical and econometric techniques including AI for crude oil price prediction might demonstrate demotions to the prediction performance.

RNN is used with Long Short Term Memory to achieve future crude oil using previous history of crude oil. The cost is measured as the mean squared error to determine its effectiveness. The performance of the proposed model is evaluated using the price data in the WTO crude oil materials

3.4 Problem Solution Fit



4.Requirement Analysis

4.1 Functional Requirement

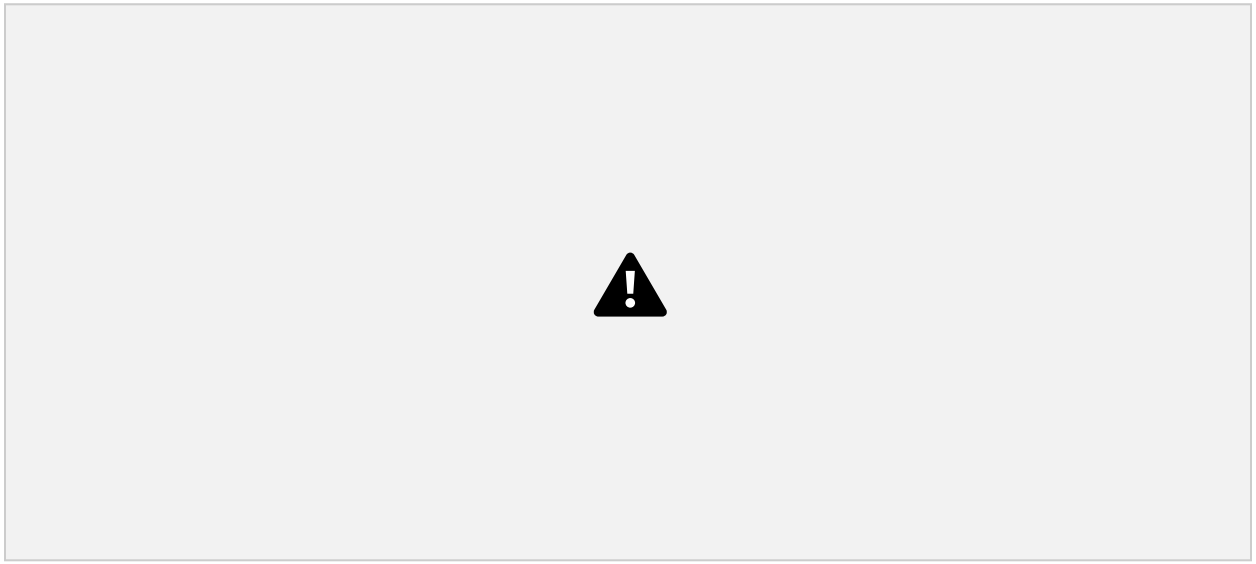
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Mobile number or through email
FR-2	login	User can login through registered email ID/ Mobile number

4.2 Non-Functional Requirement

NFR No.	Non-Functional Requirement	Description
NFR-1	Usability	UI is designed to be user-friendly. We use charts and visualisations which gives a clear understanding of price activity.
NFR-2	Security	We follow certain security protocols like using user credentials.
NFR-3	Reliability	The Data which represented in web app is extremely accurate in predicting the right data
NFR-4	Performance	The performance in this project is determined through how accurately you can predict the price of the crude oil every time we use the prediction model.
NFR-5	Availability	The web app will be made available to all operating systems.
NFR-6	Scalability	According to user base the project scalability is done.

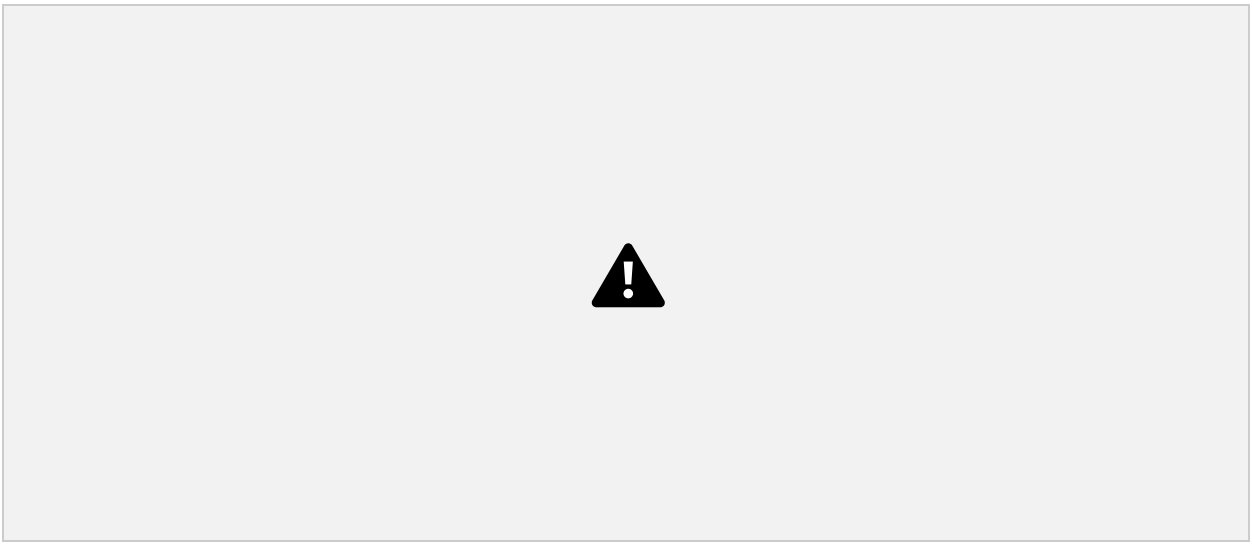
5.Project Design

5.1 Data Flow Diagrams



5.2 Solution and Technical Architecture

RNN is used with Long Short Term Memory to achieve future crude oil using previous history of crude oil. The cost is measured as the mean squared error to determine its effectiveness. The performance of the proposed model is evaluated using the price data in the WTO crude oil materials



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	User 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	User can register for the application through Gmail.	I can register through already logged in Gmail account.	Medium	Sprint-1
	Login	USN-3	User 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	USN-4	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	Already created Gmail can be used for Login.	As the web user, I can login simply by using Gmail account.	Medium	Sprint-2
Administrator	News		Admin will give the recent news of Oil Prices.	Provide the recent oil prices.	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 and Scheduling

6.1 Sprint Planning and Estimation

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/web site by entering my email, password, and confirming my password.	5	High	Madhavan P T S
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application.	2	High	Harish Veeraraghavan
Sprint-1		USN-3	As a user, I can register for the application through Gmail.	5	Medium	Santhosh S

Sprint-1	Login	USN-4	As a user, I can log into the application by entering email & password.	3	High	Yuvan Raju M
Sprint-2	Input necessary details	USN-5	As a user I can give the necessary input details to predict the price of the crude oil.	6	High	Madhavan P T S
Sprint-2	Data Pre-processing	USN-6	Raw data is transformed into suitable format for the prediction of the price of crude oil.	5	High	Santhosh S
Sprint-3	Prediction of Oil Price	USN-7	As a user I can predict the price of crude oil using the application or through the website.	6	High	Harish Veeraraghavan
Sprint-3		USN-8	As a user I can get accurate predictions of the crude oil price.	4	Medium	Yuvan Raju M
Sprint-4	Feedback	USN-9	As a user I can give the review of the	4	High	Madhavan P T S

			application/web site.			

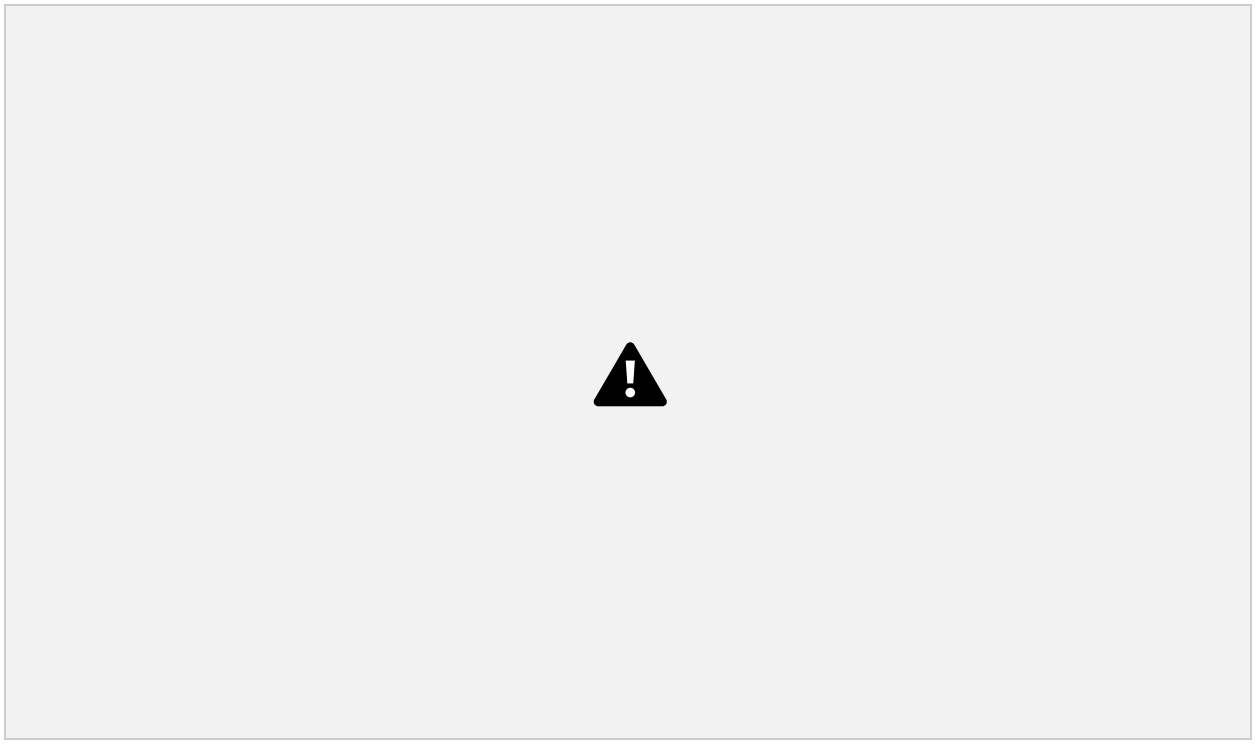
6.2 Sprint Delivery Plan

Sprint	Total Story Points	Duration	Average Velocity	Sprint Start Date	Sprint End Date
Sprint-1	15	6 Days	15/6=2.5	24 Oct 2022	29 Oct 2022
Sprint-2	11	6 Days	11/6=1.83	31 Oct 2022	05 Nov 2022
Sprint-3	10	6 Days	10/6=1.66	07 Nov 2022	12 Nov 2022
Sprint-4	4	6 Days	4/6=0.33	14 Nov 2022	19 Nov 2022

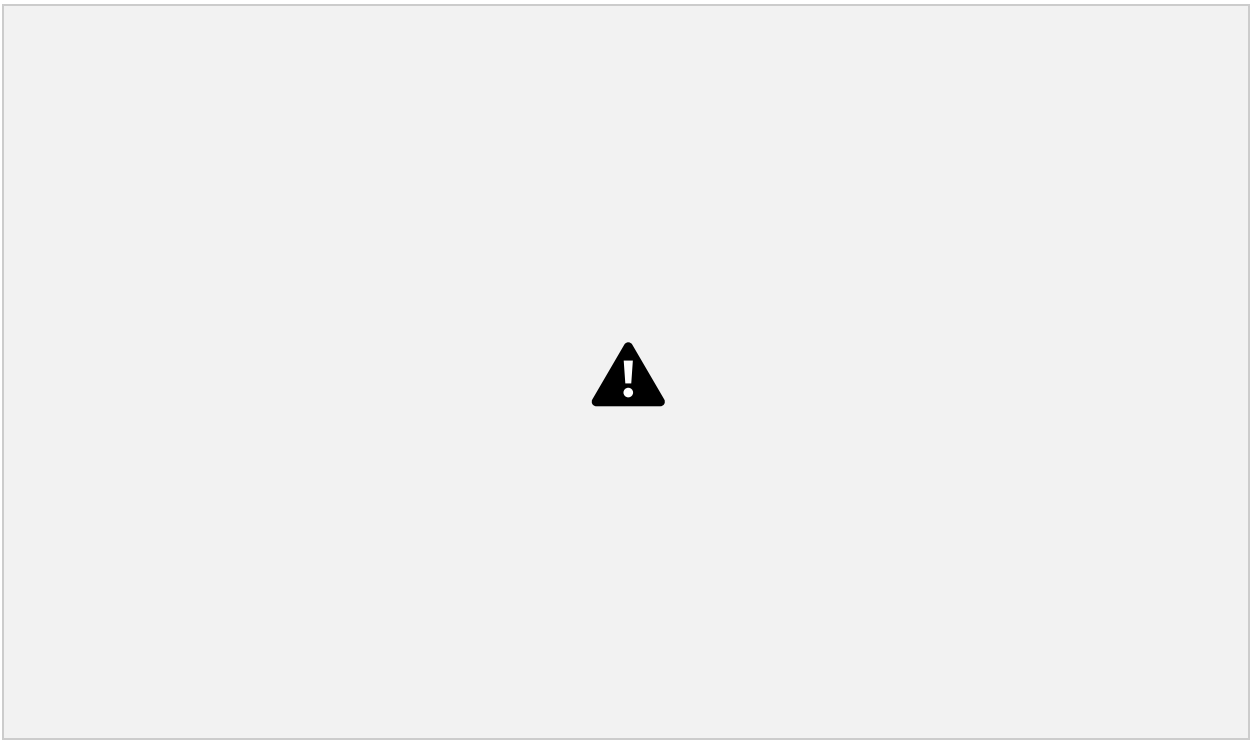
6.3 reports from jira

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let’s calculate the team’s average velocity (AV) per iteration unit (story points per day)



Burndown Chart:
A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



7.CODING AND SOLUTIONING

7.1 Interactive UI

The area where interactions between people and machines take place is known as a user interface (UI) in the subject of industrial design known as human-computer interaction. This interaction's purpose is to enable efficient machine operation and control from the human end, while the machine also feeds information back to the operators to support their decision-making. The general objective of user interface design is to provide an interface that makes it simple, effective, and pleasurable (user[1]friendly) to operate a machine in a way that yields the desired outcome (i.e., maximum usability). This typically means that the machine reduces undesirable outputs to the user while simultaneously requiring the operator to input as little as possible to produce the desired output. We have included a user interface in our project to make it easier for users to forecast the price of crude oil in the future. Users simply need to visit the website to access the interface and can click a button to forecast the price. Once the button has been clicked, the user will be taken to another website

where they can enter the price of crude oil for 10 days. In that case, the user should click Predict. The user can then view the price of crude oil after ten days.

7.2 Cloud Integration

The on-demand availability of computer system resources, in particular data storage (cloud storage) and processing power, without direct active supervision by the user, is known as cloud computing. Functions in large clouds are frequently dispersed over several sites, each of which is a data centre. Cloud computing often uses a "pay as you go" model, which can help reduce capital expenses but may also result in unanticipated running expenses for users. Cloud computing depends on resource 43 sharing to accomplish coherence. Our project is cloud-integrated, allowing it to run anywhere and be accessible at any time. Anytime the user desires, they will be able to forecast the price of crude oil. Through the IBM Cloud, this is accomplished. On the IBM Watson Studio, which makes use of the Watson Machine Learning Platform, we developed and trained the model. We generated a deployment space and ran the code using the API key to deploy the model. The Flask app, which is used to link to the backend and frontend, was then finally integrated.

8.TESTING

8.1 Test Cases

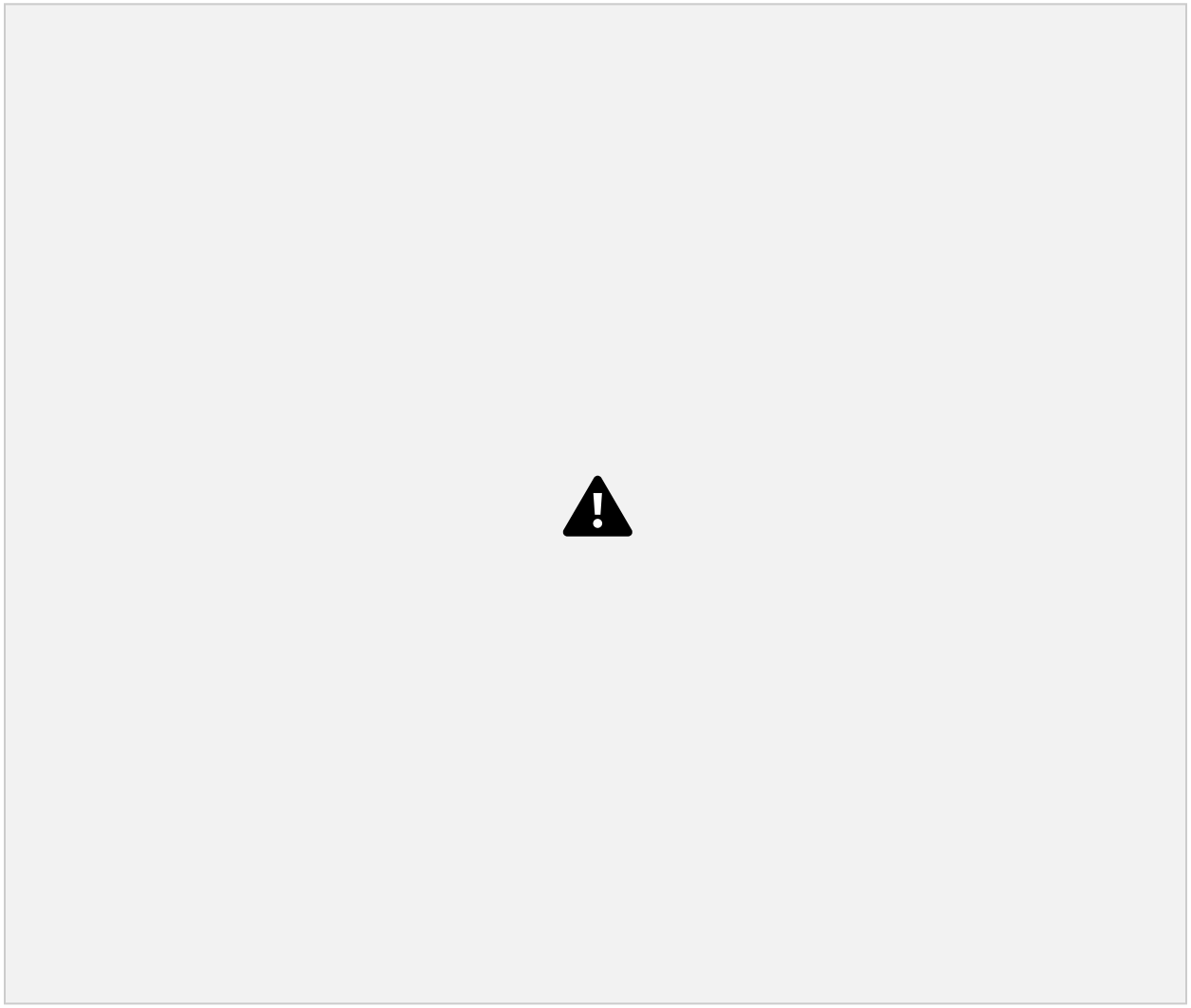
The following test scenarios were tested successfully.

Test Scenarios

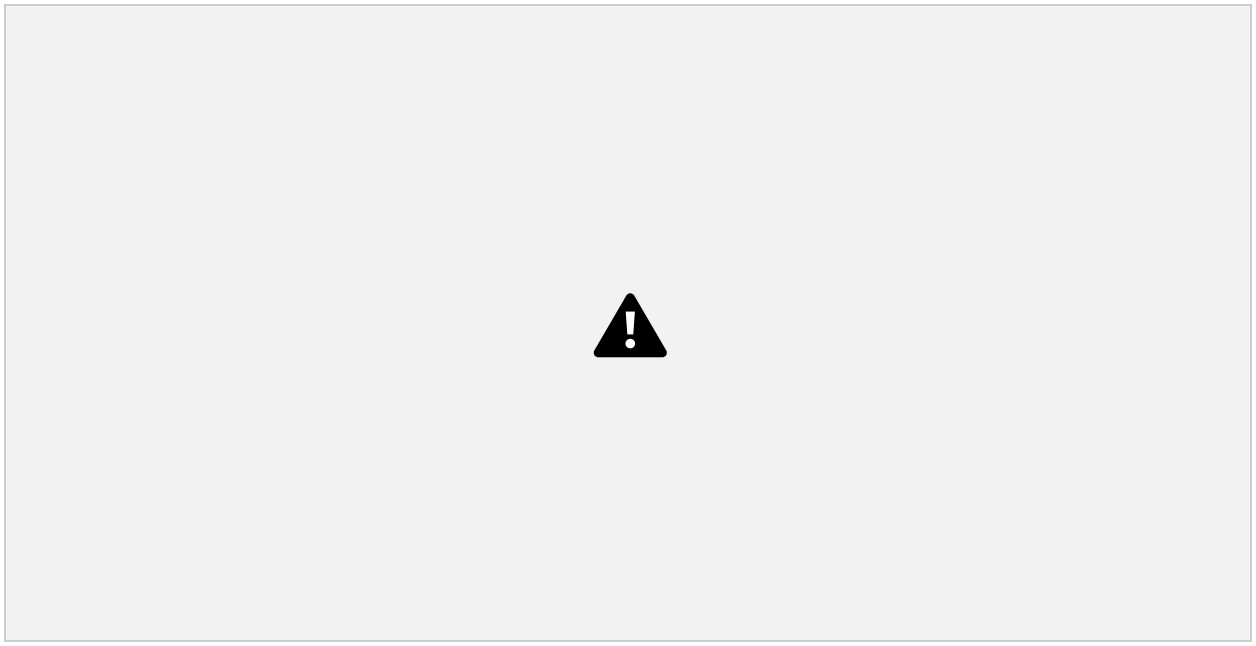
1 Verify the UI elements in the home page

2 Verify whether the user can navigate to the prediction page

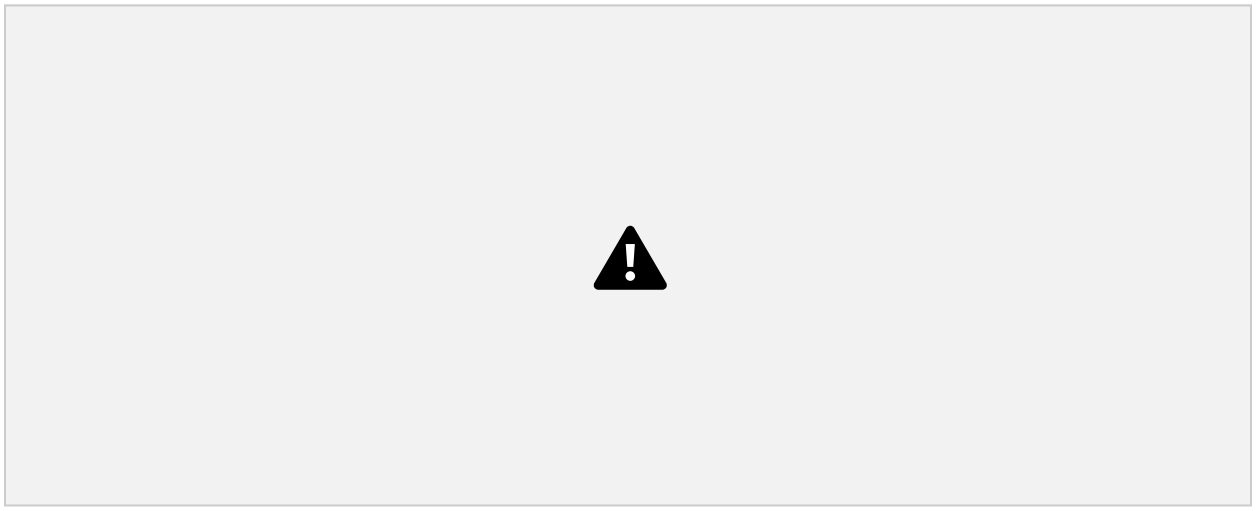
3 Verify the UI elements in the prediction page
4 Verify user is able to enter value in the text box.
5 Verify user is able to enter numbers in the text box
6 Verify model can handle with no inputs
7 Verify model can handle multiple input
8 Verify model can handle unsupported input
9 Verify model can predict the output
10 Verify the predicted results are displayed
11 Verify user can enter the value after prediction



8.2 User Acceptance Testing:
Defect Analysis



Test Case Analysis:



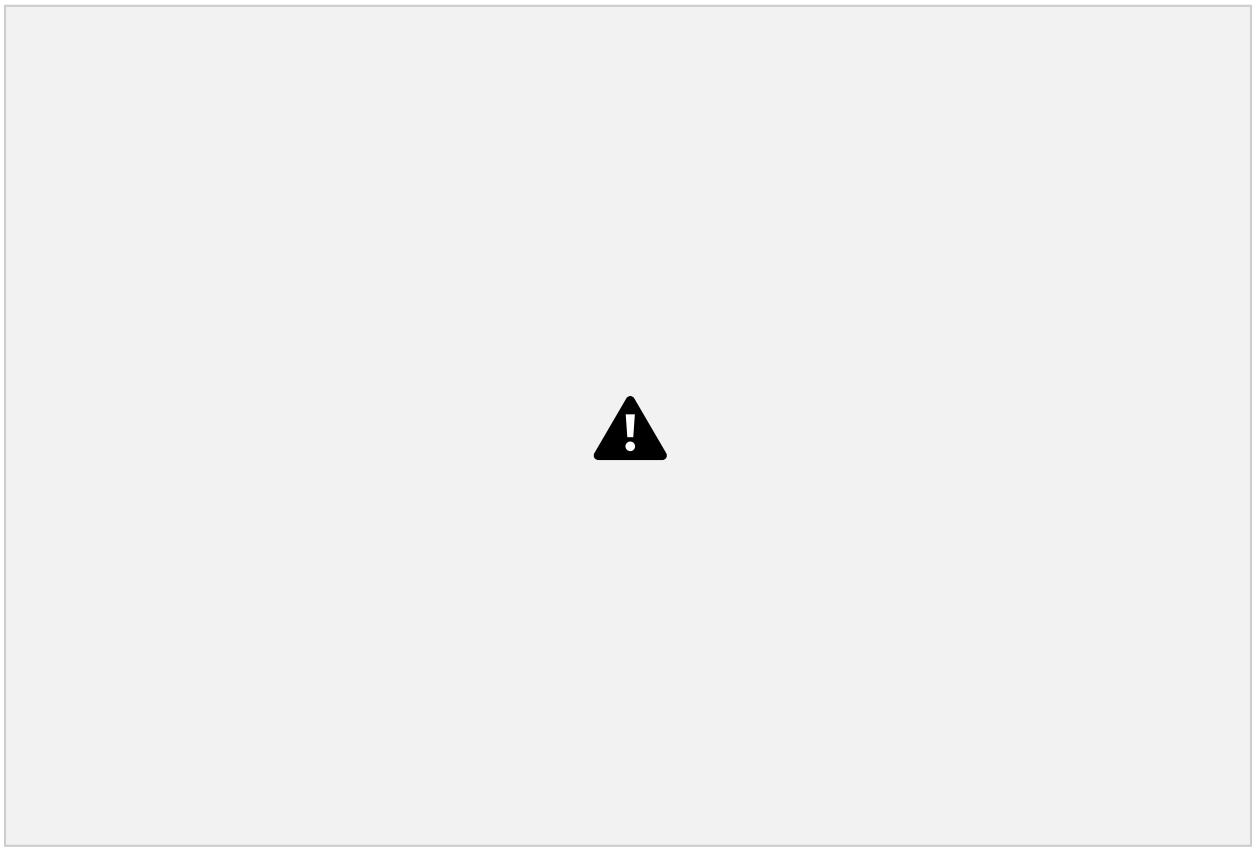
9.RESULTS

9.1 Performance Metrics:

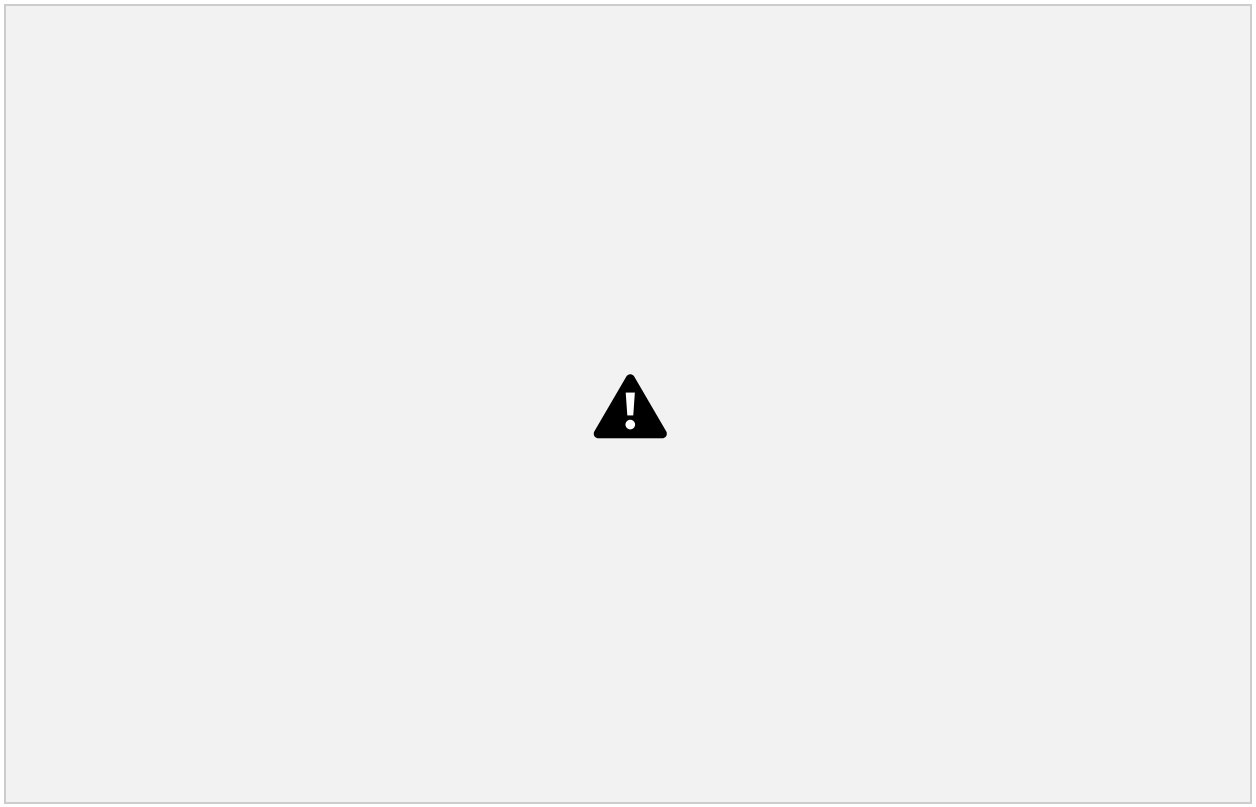
We attempted to forecast the output of the crude oil by entering various input variables in order to assess the accuracy and performance of this project. These are the input values.

[0.44172960165852215, 0.48111950244335855, 0.49726047682511476, 0.4679401747371539, 0.4729749740855915, 0.47119798608026064, 0.47341922108692425, 0.4649785280616022, 0.4703835332444839, 0.471494150747815 87]

The anticipated outcome after providing the input values is 0.46976325.



It can be seen that the graph was drawn using the provided data and a projection for the next 10 days. There was a little discrepancy between the output and the real pricing.



The developed system shows a clear prediction of the future prices which has very less deviations from the true prices by using LSTM in tensorflow and keras in python. There is always a thin line between the overfitting of the model and its best performance. This project helps a lot to learn about the developed model and the algorithm and using this model as a base, a much more complicated model can be easily developed. The facet of more prediction algorithms for crude oil can concoct with the help of this system.

This system concludes that the machine learning model LSTM (Long Short[1]Term Method) predicts the future price of crude oil by bordering the actual price of the crude oil price.

10.ADVANTAGES AND DISADVANTAGES

Advantages

- High Accuracy
- Removes the investment bias
- Develop the habit of complete analysis

● Minimise our losses

● Allows smart way of making money

High Accuracy:

The model which we predicted had a high accuracy of above 90 per cent in all aspects. The other advantages of predicting the price of crude oil are discussed below.

Removes the investment bias:

The Indian stock market offers a variety of chances for traders and investors, but it is also helpful to be aware of the market environment before taking a position in a particular stock. Take the weather prediction as an example to help you comprehend this; being aware of the weather forecast for the coming week enables you to make appropriate plans. The situation with stock market investments is comparable. Let's look at a few of the major benefits connected with stock market prediction now to help you grasp.

Develop the habit of complete analysis:

Investors don't always conduct a thorough research of the stock before learning how to anticipate the stock market and putting what they have learned into practise. They only start to establish the habit of comprehensive analysis before making any investing decisions after they learn how to apply formulae and procedures to forecast stock market movements. Once or initially, making a successful stock market prediction gives investors the confidence to form the habit of conducting a thorough analysis each time. Here, "complete analysis" refers to both the fundamental and the technical analysis of the stocks because the combination of these two forecasting methods results in predictions that are more precise.

Minimise our losses:

Another benefit of stock market prediction is that it significantly reduces your losses or restricts them. Investors sometimes make the error of not doing their studies thoroughly before learning how to anticipate, which results in them frequently employing the incorrect prediction strategies. As

a result, many put their money into the stocks based solely on intuition or merely wild estimates in the hopes that the prices will rise, and they will profit. They lose most of the time because it doesn't happen. They can reduce their losses by correctly implementing and using the appropriate forecast strategies. The converse of this is also true, and given the information provided, you can make wise selections.

Allows smart way of making money:

Making steadily increasing profits through the use of your trading expertise and knowledge is the smart method to make money. The most desired and ideal approach to make money in the stock market is to become a day trader and make money every day, unless of course a person has long-term aspirations. But in order to do that, you must be aware of the various difficulties and difficulties that come with intraday trading, as well as how to deal with them. That can only occur when you understand how to forecast the stock market using a variety of tools and tactics and how to maximise intraday trading, enabling yourself to consistently make money.

Disadvantages

● Forecasts are never 100% accurate

● It can be time-consuming and resource-intensive

Forecasts are never 100% accurate:

Let’s face it: it’s hard to predict the future. Even if you have a great process in place and forecasting experts on your payroll, your forecasts will never be spot on. Some products and markets simply have a high level of volatility. And in general, there is just an endless number of factors that influence demand.

It can be time-consuming and resource-intensive:

Forecasting involves a lot of data gathering, data organizing, and coordination. Companies typically employ a team of demand planners who are responsible for coming up with the forecast. But in order to do this well, demand planners need substantial input from the sales and marketing teams. In addition, it’s not uncommon for processes to be manual and

labour-intensive, thus taking up a lot of time. Fortunately, if you have the right technology in place, this is much less of an issue.

11.CONCLUSION

In today's world and in such a dynamic atmosphere where everyone wants to know what will happen in the future, artificial intelligence and deep learning are the foundation for upgrading technology. The path to future prediction has been established by several facilities. It previously hard to predict the prices of cryptocurrencies since they change randomly, but machine learning has made it feasible.

By integrating LSTM in TensorFlow and keras in Python, the constructed model demonstrates a clear prediction of the future prices with very little variance from the genuine prices. Between the model being overfitted and performing at its optimum, there is always a fine line. With a few minor adjustments, the model may be applied to different time series data.

With the knowledge gained from this research, a far more complex model may be created with relative ease utilising the generated model and algorithm as a foundation. With the aid of this model, more prediction algorithms for bitcoin may be developed.

This project comes to the conclusion that the LSTM (Long Short-Term Method) machine learning algorithm predicts the future price of crude oil by edging the current price of the oil with high accuracy

12.FUTURE SCOPE

The Long Short-Term Method (LSTM) machine learning algorithm is shown to have a high degree of accuracy in predicting the future price of crude oil by edging the current price of the oil.

In the future, it will be possible to estimate crude oil prices by taking into account additional variables that influence the price, such as tweets, national news, natural disasters, the cost of forecasting, conflict, demand, and floods. By doing this, the model's precision and accuracy would both be enhanced.

The dataset will be obtained from Kaggle, a sizable platform that is frequently used for data mining and doing analysis. The model would similarly be created using these elements. If this is carried out, the accuracy of forecasting the price of crude oil will exceed 98 percent.

13 APPENDIX

Source Code

```
#Import the libraries that will be needed in the program.
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import os
import seaborn as sns

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='aTlCyZRbJNCY4qazYOUGZRu3TtwwY-a-OERmvl8USkui',
    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 = 'crudeoil-donotdelete-pr-6gegwhwphc398u'
object_key = 'Crude Oil Prices Daily.xlsx'

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

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

```
#Check whether any null values are there or not if it is present then the following can be done,
#Imputing data using the Imputation method in sklearn.
#Filling NaN values with mean, median, and mode using fillna() method.
#Delete the records
data.tail(10)
data.isnull().any()
data.isnull().sum()
data.dropna(axis=0,inplace=True)
data.isnull().sum()

data['Closing Value'].plot(kind='bar')
plt.title('Variation of oil price over years')

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

data_oil

#Feature scaling is a method used to normalize the range of independent variables or features
of data.
# The next step is to scale the crude oil prices between (0, 1) to avoid intensive computation.
# Common methods include Standardization and Normalization.
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler(feature_range=(0,1))
data_oil=scaler.fit_transform(np.array(data_oil).reshape(-1,1))

data_oil

#Data visualization is where a given data set is presented in a graphical format.
# It helps the detection of patterns, trends, and correlations that might go undetected in
text-based data.
# Visualize our data using the Matplotlib and seaborn library.
plt.plot(data_oil)
plt.figure(figsize=(15000000,5))

plt.rcParams("figure.figsize")=(20,6)

#When you are working on a model and you want to train it, you have a dataset.
# But after training, we have to test the model on some test dataset.
```

```
# For this, you will need a dataset that is different from the training set you used earlier.
# But it might not always be possible to have so much data during the development phase.
# In such cases, the solution is to split the dataset into two sets, one for training and the other
for testing.
training_size=int(len(data_oil)*0.65)
test_size=len(data_oil)-training_size
train_data,test_data=data_oil[0:training_size,:],data_oil[training_size:,:1]
train_data.shape,test_data.shape
#The size of the train and test data after splitting.
training_size,test_size

# convert an array of values into a dataset matrix
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)

# reshape into Xat, t+1, t+2, t+3 and Y=t+4
time_step = 10
x_train,y_train = create_dataset(train_data, time_step)
x_test,y_test = create_dataset(test_data,time_step)

#Shape of training data.
print(x_train.shape),print(y_train.shape)

#Shape of test data.
print (x_test.shape), print (y_test.shape)

#The data of X_train.
x_train

# reshape input to be [samples, time steps, features] which is required for LSTM.
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)
```

```
!pip install ibm_watson_machine_learning

from ibm_watson_machine_learning import APIClient
wml_credentials = {
                "url": "https://us-south.ml.cloud.ibm.com",
                "apikey": "FuOVK4NIKK7mntk0VwDBd3gD0p_Rg9Y5P3gIKXFhviha"
}
client = APIClient(wml_credentials)

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

model_details = client.repository.store_model(model = "body.tgz" , meta_props = {
   client.repository.ModelMetaNames.NAME : "Crude_oil",
   client.repository.ModelMetaNames.TYPE : "tensorflow_rt22.1",
   client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid
})
model_id = client.repository.get_model_id(model_details)

model_id
```

```
#Create the stacked LSTM model.
#Import the model building libraries.
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM

#Initializing the libraries.
models=Sequential()

#Adding LSTM layers.
models.add(LSTM(50, return_sequences=True, input_shape=(10,1)))
models.add(LSTM(50, return_sequences=True))
models.add(LSTM(50))

#Adding output layers.
models.add(Dense(1))

#Information about the model and it's layers.
models.summary()

#Configuring the learning process.
#Metrics are used to evaluate the performance of your model.
models.compile(loss="mean_squared_error",optimizer="adam")

#Train the model.
# RNN weights are updated every 64 stock prices with a batch size of 64.
models.fit(x_train, y_train, validation_data=(x_test,y_test), epochs=50, batch_size=64,
verbose=1)

#Tranform to original from.
train_predict= models.predict(x_train)
train_predict=scaler.inverse_transform(train_predict)
test_predict= models.predict(x_test)
test_predict=scaler.inverse_transform(test_predict)
```

```
#Calculate RMSeperformance metrics.
import math
from sklearn.metrics import mean_squared_error
math.sqrt (mean_squared_error(y_train, train_predict))

#Save the model.
from tensorflow.keras.models import load_model
models.save("crude_oil.h5")

!tar -zcvf body.tgz crude_oil.h5

ls -1

##Plotting
#Shift train predictions for plotting.
look_back=10
trainPredictPlot = np.empty_like(data_oil)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[look_back:len(train_predict)+look_back, :] = train_predict
# shift test predictions for plotting
testPredictPlot = np.empty_like(data_oil)
testPredictPlot[:, :] = np.nan
testPredictPlot[len(train_predict)+(look_back*2)+1:len(data_oil)-1,:] = test_predict
# plot baseline and predictions
plt.plot(scaler.inverse_transform(data_oil))
plt.plot(trainPredictPlot)
plt.plot(testPredictPlot)
plt.show()

#Print length of data.
len(test_data)

#Create the input and reshape it and convert it into list
x_input=test_data[5742: ].reshape (1,-1)
x_input.shape

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

```
temp_input

#For predicting next 10 days crude oil prices we consider n_steps=10.
#We create the input for prediction, index starting from the date 10 days before the first date in the test dataset.
# Then, reshape the inputs to have only 1 column and predict using model_predict predefined function.
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

#Create a visualization plot to easily review the prediction.
day_new=np.arange(1,11)
```

```
day_pred=np.arange (11,21)
len(data_oil)
plt.plot(day_new, scaler.inverse_transform(data_oil[16422:]))
plt.plot(day_pred, scaler.inverse_transform(lst_output))
k=scaler.inverse_transform(lst_output)
k

#Merge the the past data and next 10 days output prediction.
df3=data_oil.tolist()
df3.extend(lst_output)
plt.plot(df3[16400:])

#Reversing the predictions.
df3=scaler.inverse_transform(df3).tolist()

INTEGRATE FLASK WITH SCORING END POINT
import os
from flask import Flask,redirect, url_for, request
from flask import render_template
from flask import request
from flask_sqlalchemy import SQLAlchemy
import sqlalchemy
from sqlalchemy import create_engine
from sqlalchemy import Table,Column,Integer,String,ForeignKey
from sqlalchemy import select
from sqlalchemy.orm import Session
from sqlalchemy.orm import declarative_base
from sqlalchemy.orm import relationship
import tensorflow as tf
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import requests
import numpy as np

current_dir=os.path.abspath(os.path.dirname(__file__))
app=Flask(__name__)
```

```
app.config['SQLALCHEMY_DATABASE_URI']='sqlite:///'+os.path.join(current_dir,'authenticate.s
qlite3')
db=SQLAlchemy()
db.init_app(app)
app.app_context().push()

class Users(db.Model):
    __tablename__='users'
    user_name=db.Column(db.String,primary_key=True,nullable=False)
    password=db.Column(db.String,nullable=False)
@app.route('/index',methods=['GET',"POST"])
def index_page():
    if request.method=="GET":
        return render_template('index_new.html')
    elif request.method=="POST":
        x_input=str(request.form["past"])

        x_input=x_input.split(',')
        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):
                    print("temp_input",temp_input)
                    x_input=np.array(temp_input[1:])
                    print("{} day input {}".format(i,x_input))
                    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
            print(lst_output)
        return render_template('results.html',lst_output=lst_output)
@app.route('/', methods=["GET","POST"])
def hello_world():
    if request.method=="GET":
        return render_template("login crudeoil.html")
    elif request.method=="POST":
        username=request.form["user_name"]
        #print(username)
        delt=Users.query.filter(Users.user_name==username)
        try:
            print(delt[0])
        except:
            print('hi')
            return render_template('error.html')
        password=request.form["password"]
        user=Users.query.all()
        delt1=Users.query.filter(Users.password==password)
        try:
            print(delt1[0])
        except:
            print('hey')
            return render_template('error.html')
        #print(user)
```

```
        return redirect(url_for('index_page'))

        #return render_template('indexcrudeoil.html')
engine=create_engine("sqlite:///./authenticate.sqlite3")
@app.route('/register', methods=["GET","POST"])
def register():
    if request.method=="GET":
        return render_template("login crudeoil2.html")
    elif request.method=="POST":
        #username=request.form["user_name"]
        #password=request.form["password"]
        with Session(engine,autoflush=False) as session:
            session.begin()
            user=Users(user_name=request.form['user_name'],password=request.form['password'])
            session.add(user)
            session.flush()
            session.commit()
        return render_template("login crudeoil2.html")
model=load_model("crude_oil.h5")
print("Loaded model from disk")
if __name__=='__main__':
            app.debug=False
            app.run()
```

GitHub Link:

The "Crude Oil Price Prediction" project has been posted on GitHub.

link: https://github.com/IBM-EPBL/IBM-Project-405-1658300298

Project Demonstration:

The following link will take you to a demonstration of the "Crude Oil Price Prediction" project.

link:
https://drive.google.com/file/d/1VMNsdnoYHFzySXtIk3n6HENr6pZF7Vq_/view?usp=drivesdk