CRUDE OIL PRICE PREDICTION PROJECT REPORT

TEAM ID : PNT2022TMID43580

TEAM MEMBERS :

HARIPRASAD P :-720919104015

SUBIN T :-720919104037

AFZAL A :-720919104002

NANJUNDESWARAN :-720919104021

INDEX

1.INTRODUCTION

1.1 PROJECT OVERVIEW

Crude oil is the world's leading fuel, and its prices have a big impact on the global environment, economy as well as oil exploration and exploitation activities. Oil price forecasts are very useful to industries, governments and individuals. Although many methods have been developed for predicting oil prices, it remains one of the most challenging forecasting problems due to the high volatility of oil prices. In this paper, we propose a novel approach for crude oil price prediction based on a new machine learning paradigm called stream learning. The main advantage of our stream learning approach is that the prediction model can capture the changing pattern of oil prices since the model is continuously updated whenever new oil price data are available, with very small constant overhead. To evaluate the forecasting ability of our streaming learning model, we compare it with three other popular oil price prediction models. The experiment results show that our stream learning model achieves the highest accuracy in terms of both mean squared prediction error and directional accuracy ratio over a variety of forecast time horizons.

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.

2.LITERATURE SURVAY

2.1 Existing problem

The prediction of the ende oil rates based on the previous datasets on the data and prices as the feature _list are inputs and and target list are predicted values. The implementation was on the Linear Regression Model which is feasible to some extend for the prediction of the crude oil prices. The implementation is on predicting the crude oil prices for the days using Linear Regression Python Machine Algorithm and plotting the graph based on the prediction.

2.2 Reference

2.3 Problem statement definition

There are five main problems identified based on investigations made on previous research. Firstly, data used in the previous predictions are majority employed from WTI or Brent crude oil price without taking into consideration other inputs that are involved together in the market. The crude oil price market volatiles from the contributions made by other factors surround it and neglecting these factors will demote the capability of a prediction tool. A good prediction is the one that can comprehends and correlates between factors, sparks information on the trend and finally, predict it accurately.

Secondly, there are scarce numbers of research that implement the verification and validation technique on the main factors involving in the fluctuation. Besides the global crude oil price, other popular factors that being used in previous research are demand and supply. Although, demand and supply of oil plays vital role to the market volatility, the use of these observations only is not enough to comprehensively render the information offered by the trend.

There are also other factors that contributed to the trend and gave impact to the price. Therefore, by embracing appropriate key factors and later correlate them will help to achieve a thorough and comprehensive prediction for the market.

Thirdly, time-series data are mainly used for prediction.

Nevertheless, data pre-processing and data representation process are made absent in some of the previous research.

These two processes are important to cleanse and reduce errors and noises in data set and uniform it. Later, these will help to organise the process of prediction, make it more systematic and finally, generates more stable result. Without these processes, the prediction tool will be less reliable.

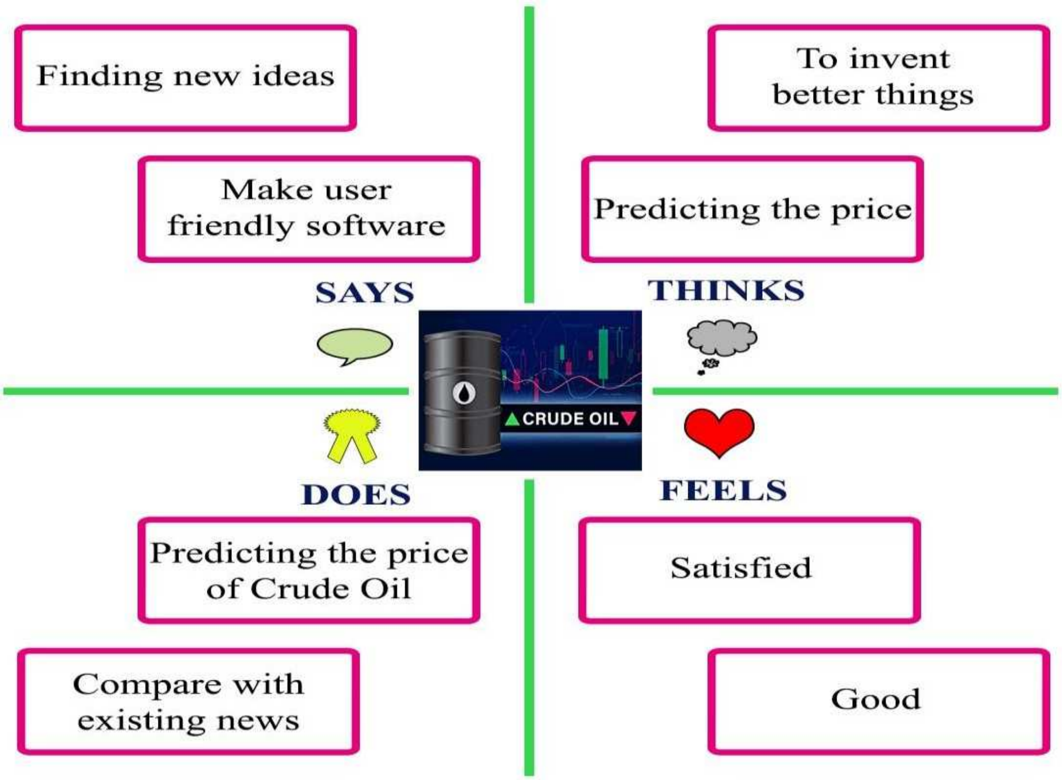
Fourthly, the crude oil price movement was the popular topic studied previously and not the crude oil price itself.

Predicting the movement of the price only is not sufficient to characterise the market where else, crisp prediction will offer far more persona. A prediction on the movement together with the price itself will tender more usable, discrete and practical implementation to the real world problem.

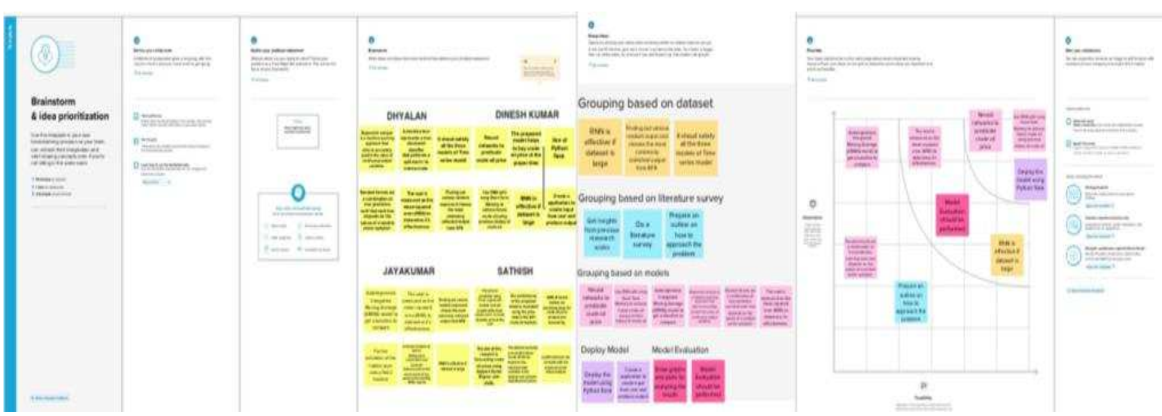
Sincerely, the practicability of the previous study is still dubious as the crude oil market itself is chaotic. Still, there are opportunities for improvement in the future as the advancement of our world technology is rapid.

3.IDEATION AND PROPOSED SOLUTION

3.1 Empathy map and canvas



3.2 Ideation & Brainstrom



3.3 Proposed solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The price of crude oil, the most essential fuel in the world, has a big impact on the environment globally, thus forecasts are very helpful for governments, businesses, and people. Continuous use of statistical and econometric methods, including AI, may have a negative impact on prediction accuracy.
2.	Idea / Solution description	In order to predict future crude oil using historical data on crude oil, RNN is utilised with long short-term memory. The effectiveness of the cost is calculated using the mean squared error. Using the pricing information in the WTO crude oil materials, the proposed model's performance is assessed.
3.	Novelty / Uniqueness	● Crude oil price variations have a significant impact on the world's economies, thus price forecasting can help reduce the risks brought on by this volatility. ● For a variety of stakeholders, including governments, public and private businesses, legislators, and investors, price projections are crucial.
4.	Social Impact / Customer Satisfaction	● It is employed to forecast future pricing and consume oil in accordance with such prices. ● This price directly affects a number of goods and products, and its changes have an impact on the stock markets. ● In addition to economic factors, significant events can have an impact on oil prices.
5.	Business Model (Revenue Model)	● When deciding whether to purchase or sell crude oil, it can be useful to decision-makers who may be businesses, individual investors, or both. ● One of the most profitable commodities for traders to trade is crude oil. ● To anticipate the price of crude oil, RNN and LSTM models are employed as the benchmark model.
6.	Scalability of the Solution	● The dimensions of the data are reduced using the PCA, MDS, and LLE methods. ● RNN and LSTM model accuracy should be increased.

3.4 Problem solution fit

Project Title: Crude oil price prediction

Project Design Phase-I - Solution Fit Template

Team ID: PNT2022TMID43580

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?

Oil accounts are the third of the world's energy consumption That is the greatest share for all category of government

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions?

Due to strong chain effects owned by this crude oil market, Bias in the factors involved will have exclusive impact to the price

5. AVAILABLE SOLUTIONS AS

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?

There are innumerable ways and approacheswhich are being used and have been used for predicting the prices of crude oil

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Economic growth is one of the biggest factors affecting petroleum product— and therefore crude oil demand Grow ing economies increase demand for en ergy in general and especially for tran sportion.

9. PROBLEM ROOT CAUSE RC

There is only one dependent variable, the closing priceof crude oil which has been consider its a time series

7. BEHAVIOUR BE

The correct Information Should be given by the Individual.

Focus on J&P, tap into BE, understand RC

3. TRIGGERS TR

What triggers customers to act?

By seeing our friends and colleagues benefited by this web.

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?

Traipsing for job will be reduced.

10. YOUR SOLUTION SL

A contemporary and innovative method of predicting crude oil prices using the artificial neural network.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE

Customer has used this web in any time any where

8.2 OFFLINE

Non-working days are not predicting the price of crude oil.

4.REQUIREMENT ANALYSIS

4.1 Functional Requirements

- User Application : The price of crude oil can be downloaded directly from the Google Play Store application by users.
- User Products Available : The Application's User There are numerous products in the Crude Oil Price App, and users can instantly update the energy and oil prices.
- User Additional Features : The user can view oil price charts and the most recent news. Major Energy Quotes User View The user may use various color schemes.
- User Exceptions : The User Can Exchange Rates and Currency Converter.

4.2 Non-Functional Requirements

- Usability : Give the accurate price prediction of crude oil.
- Security : The data of user will be secure and encrypted.
- Reliability : Give the best and accurate price and knowledge to the customer.
- Availability : The application will be available on all the platform.
- Scalability : The scalability will be approximately 95%.

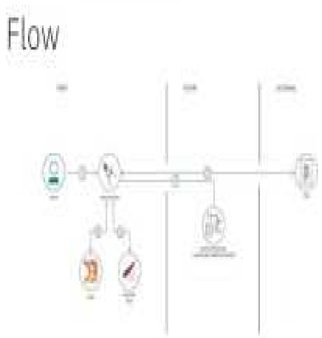
5.PROJECT DESIGN

5.1 Data Flow Diagrams

Data Flow Diagrams:

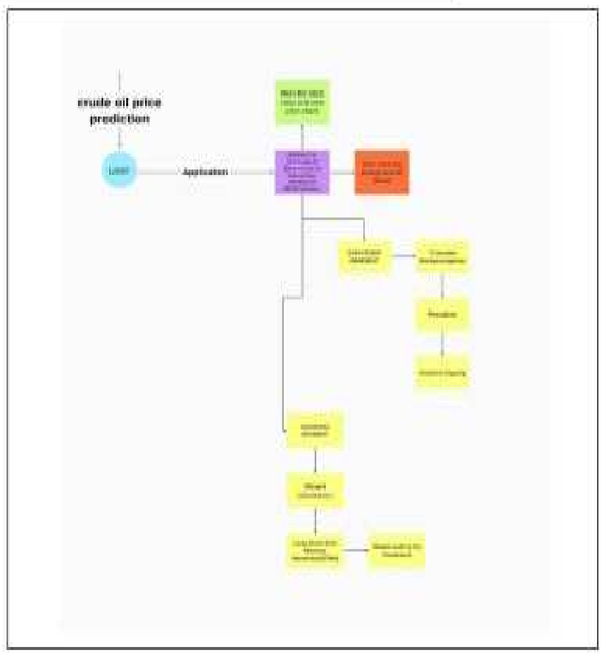
The classic visual representation of how information moves through a system is a data flow diagram (DFD). A tidy and understandable DFD can graphically represent the appropriate quantity of the system demand. It displays how information enters and exits the system, what modifies the data, and where information is kept.

Example: (Simplified)



1. User configures credentials for the Watson Natural Language Understanding service and starts the app.
2. User selects data file to process and load.
3. Apache Tika extracts text from the data file.
4. Extracted text is passed to Watson NLU for enrichment.
5. Enriched data is visualized in the UI using the D3.js library.

Example: DFD Level 0 (Industry Standard)



5.2 Solution & Technical Architecture

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions.

Its goals are to:

- Find the best tech solution to solve existing problems in the organizations – LSTM technique.
- Describe the structure, characteristics, behavior, and other aspects of the software is move to LSTM for predicting more accurate Crude Oil Prices.
- Define features, development phases, and solution requirements of the project.
- Provide specifications according to which the solution is defined, managed, and delivered.

Solution Architecture Diagram:

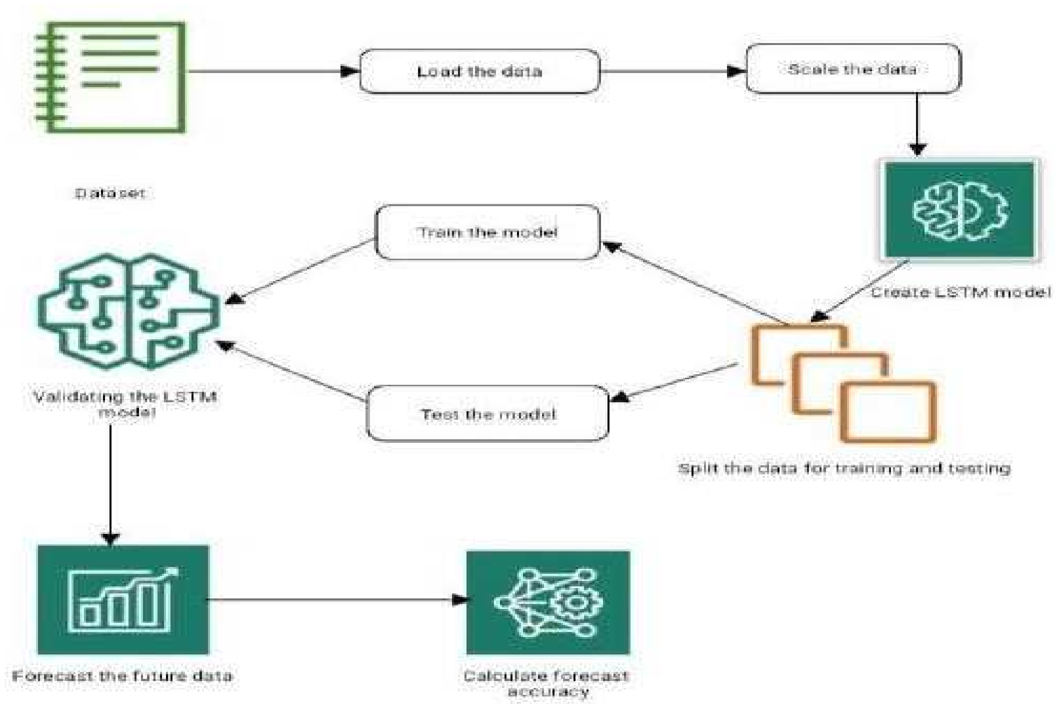
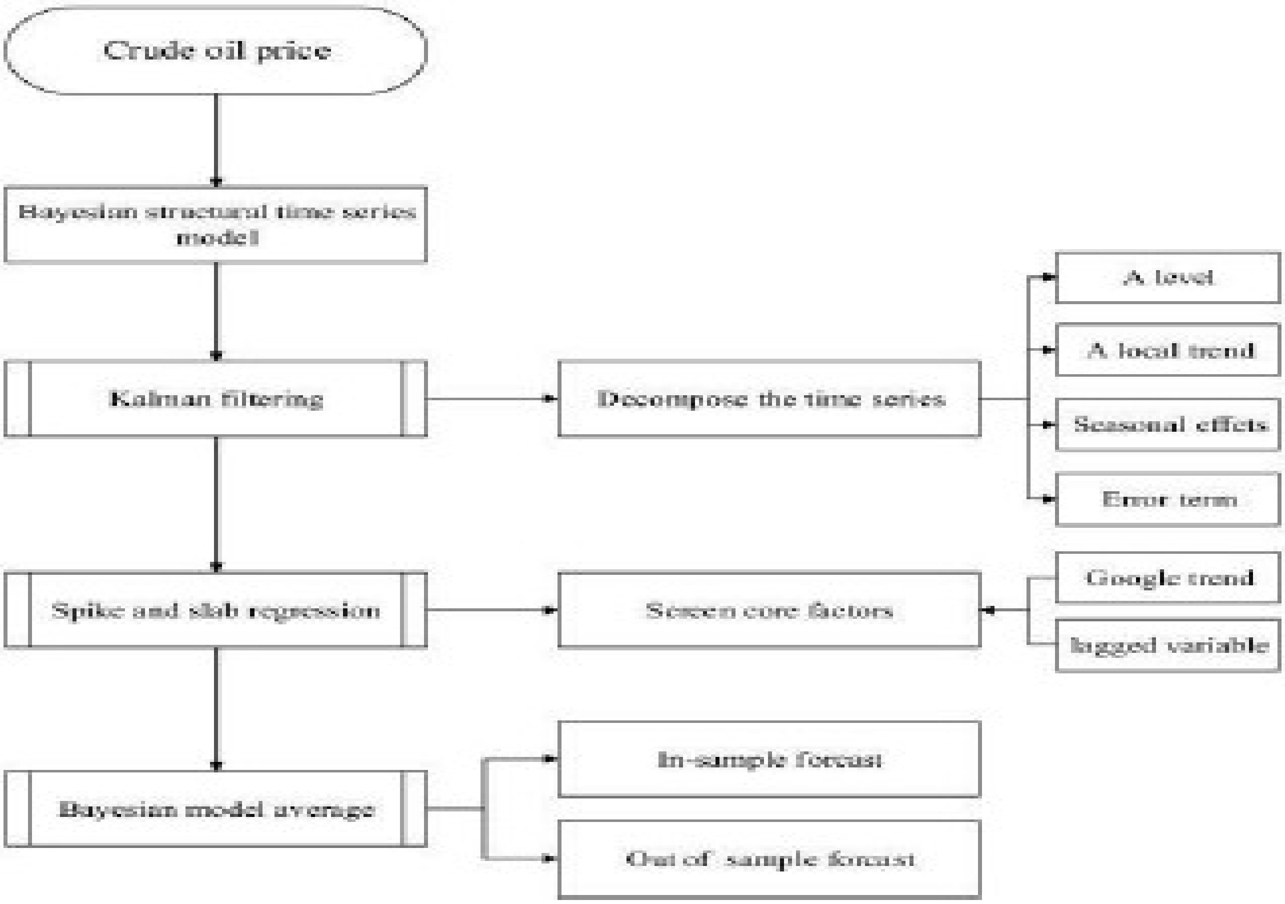


Figure 1: Architecture and data flow of the Crude oil Price Prediction

Technical Architecture Diagram :


Crude oil price
Bayesian structural time series model
Kalman filtering
Decompose the time series
A level
A local trend
Seasonal effets
Error term
Spike and slab regression
Screen core factors
Google trend
lagged variable
Bayesian model average
In-sample forcast
Out of sample forcast

5.3 User Stories

User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Application	USN-1	You can download the crude oil price by opening the Google Play Store app directly as a user.	I can access own decisions.	High	Sprint-1
	Available Products	USN-2	Users of the application may instantly update the energy and oil prices while using it because there are so many different products in the crude oil price app.	I can receive the data once click then confirm	High	Sprint-1
	Additional Features	USN-3	Users can read the most recent news and see oil price charts. Major Energy Quotes User View The user may use many colour schemes.	I can view then read the price prediction.	High	Sprint-2
	Expectations	USN-4	User Can Convert Currency And Exchange Rates	I can expect	Medium	Sprint-1
	Login	USN-5	Log in as a user without using an email address, username, or password.		High	Sprint-1
Customer (Web user)			As a user I can view the crude oil price	I can view the price directly	High	Sprint-1
Customer Care Executive			As a user I executive the given price history	I can accept the terms	medium	Sprint-1
Administrator			As a manager, it anticipates the results.	Show the result	High	Sprint-1

6.PROJECT PLANNING & SCHEDULING

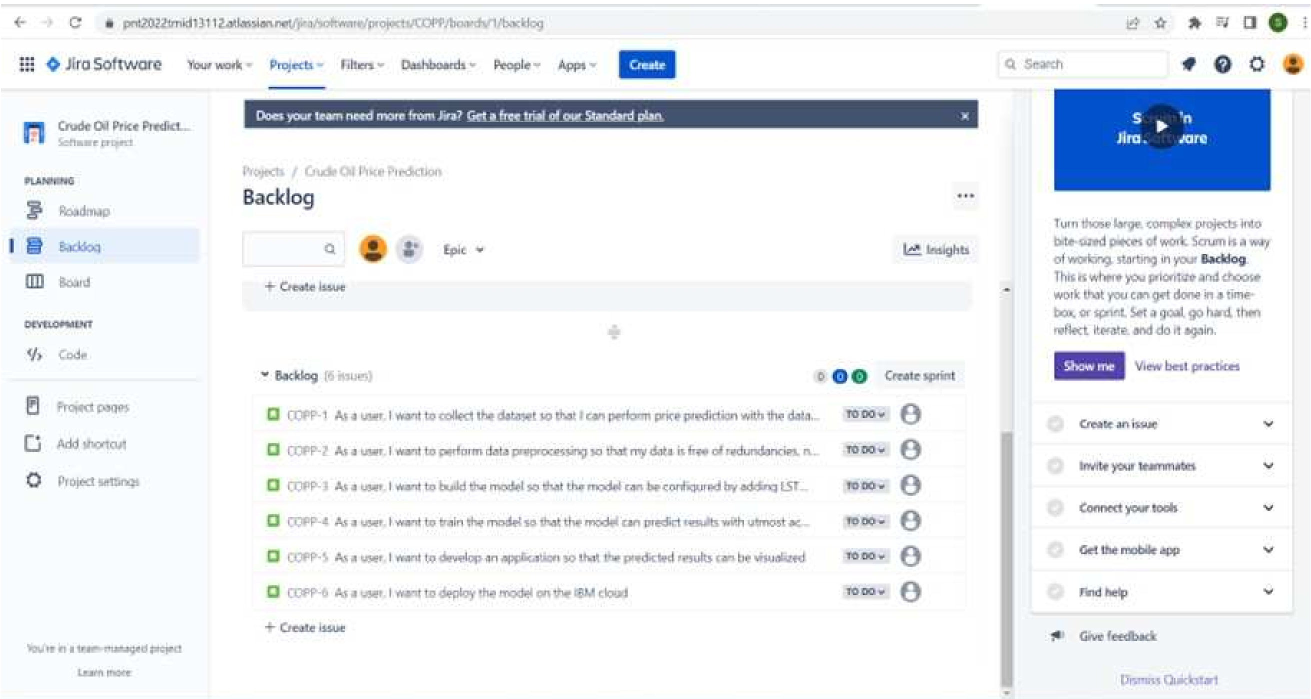
6.1 Sprint Planning & Estimation

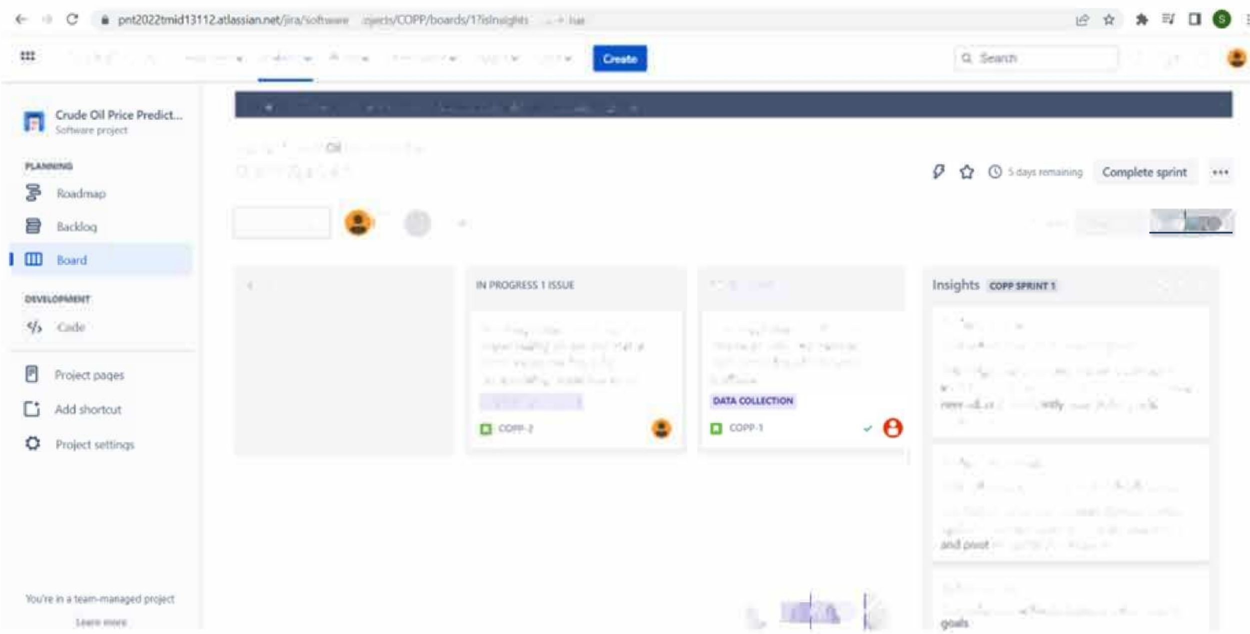
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-2	Registration	USN-3	As a user, I can register for the application by entering my email, password, and confirming my password.	10	High	SUBIN.T
Sprint-2		USN-4	As a user, I will receive confirmation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

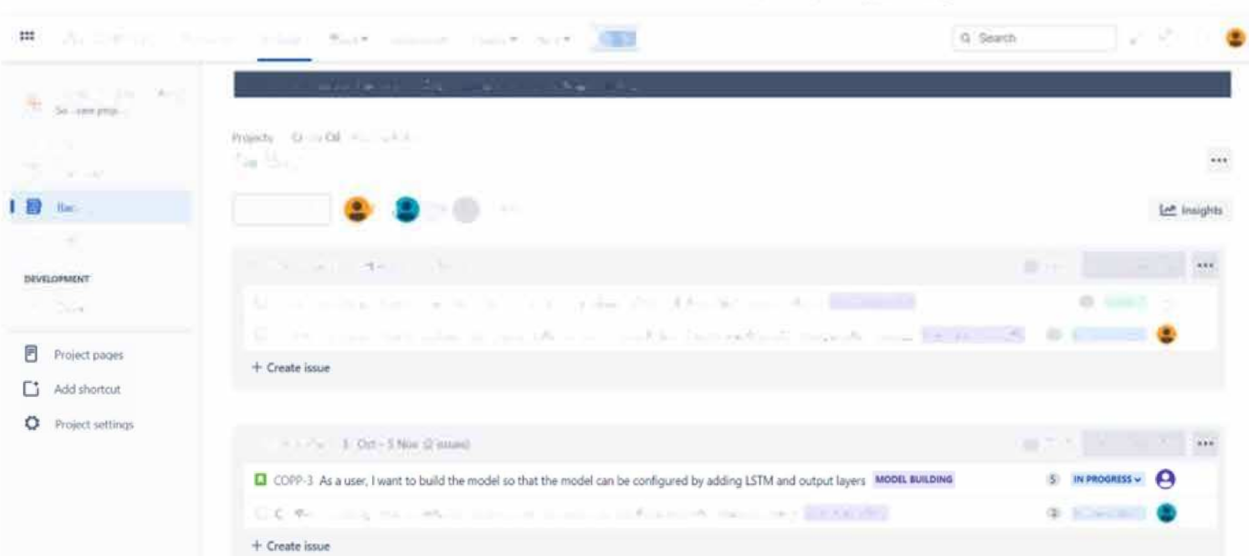
6.2 Sprint Delivery Schedule

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		
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022		
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022		

6.3 Reports From JIRA



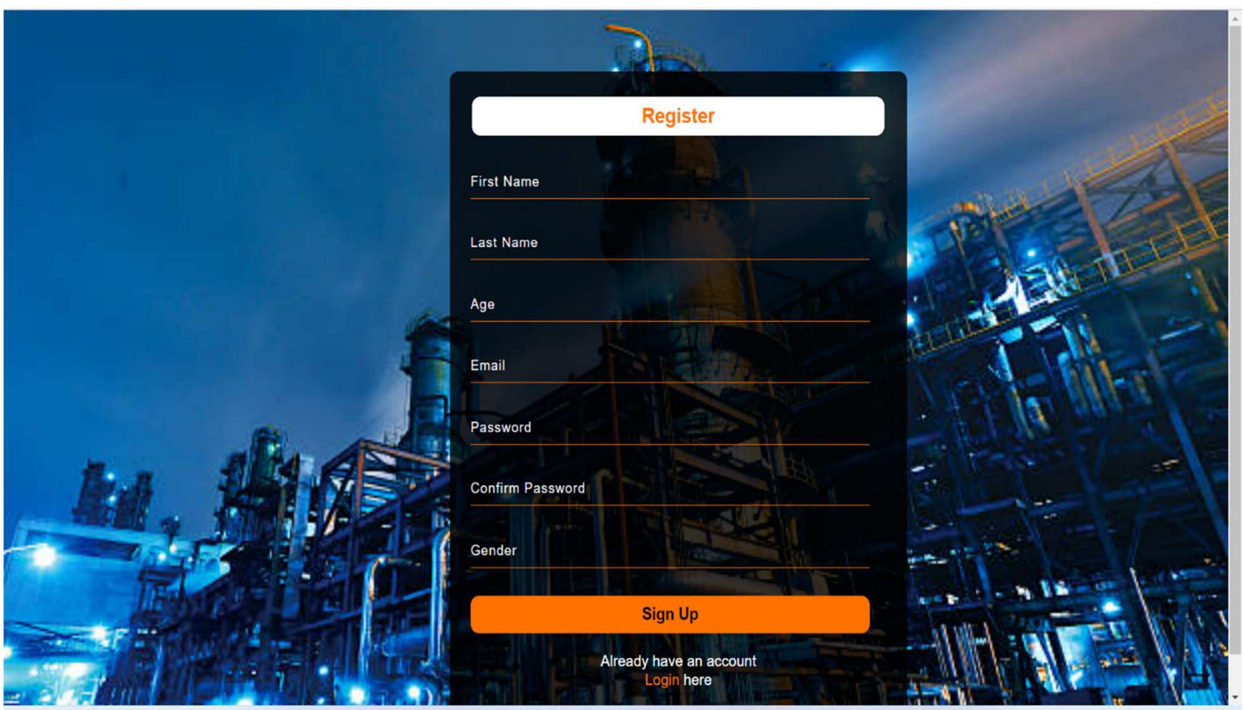

pnt2022tmid13112.atlassian.net/jira/software
Create
Search
Crude Oil Price Predict...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
5 days remaining
Complete sprint
IN PROGRESS 1 ISSUE
COPP-2
DATA COLLECTION
COPP-1
Insights
COPP SPRINT 1

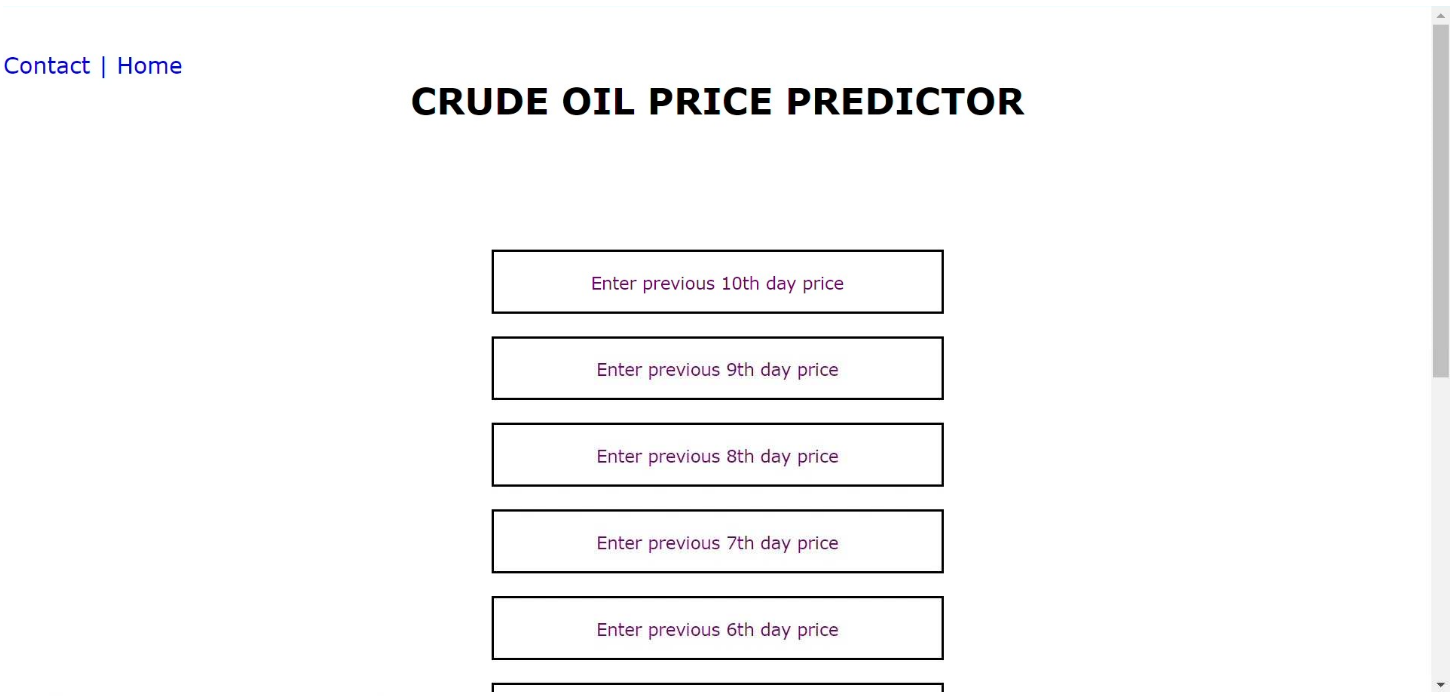

Search
DEVELOPMENT
Project pages
Add shortcut
Project settings
Insights
Create issue
COPP-3 As a user, I want to build the model so that the model can be configured by adding LSTM and output layers
MODEL BUILDING
IN PROGRESS
Create issue

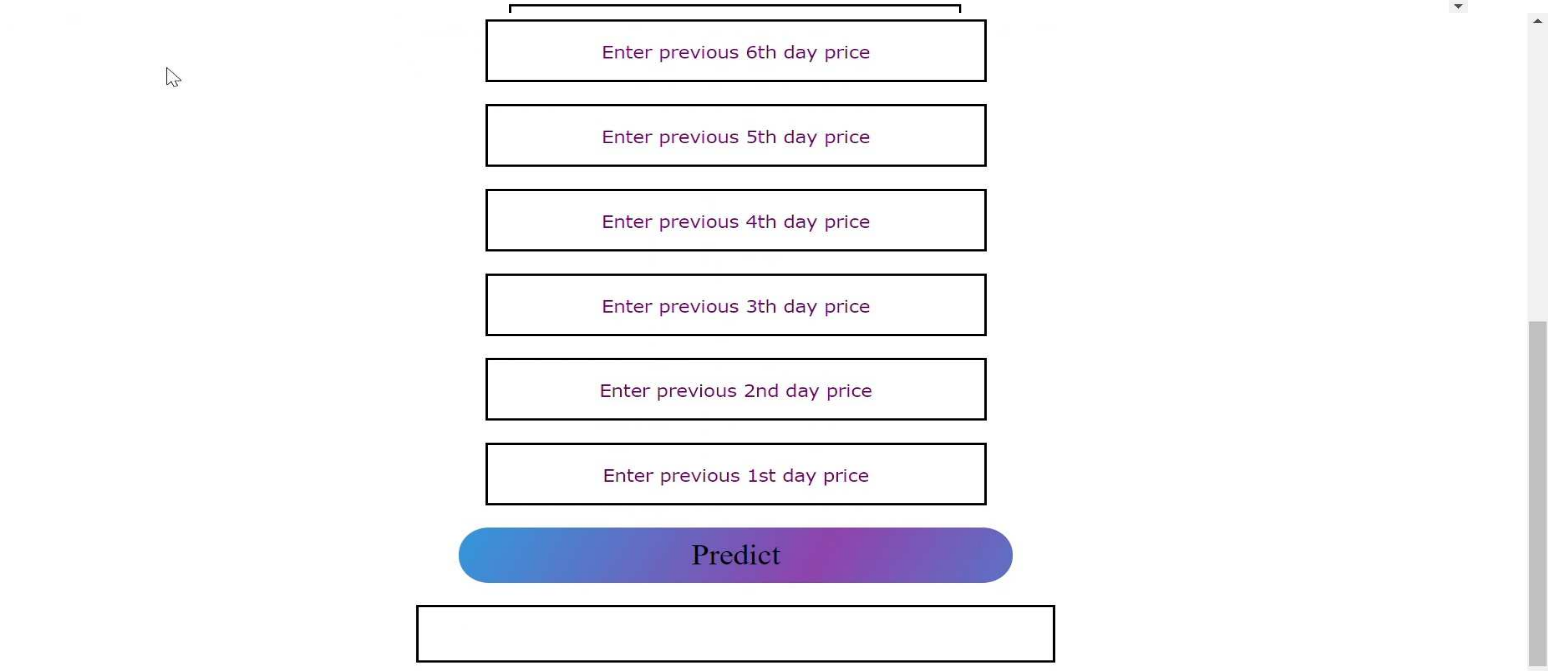
7.CODING AND SOLUTIONING (explain the features add in the project along with code)

7.1 Feature 1

- The crude oil prediction website provides two options
  - Home
  - predict
- The home allows the user to have an insight on the importance of crude oil priceprediction
- The predict allows the user to give the 10 days input and arrive at the predictionresults




Contact | Home
CRUDE OIL PRICE PREDICTOR
Enter previous 10th day price
Enter previous 9th day price
Enter previous 8th day price
Enter previous 7th day price
Enter previous 6th day price


Enter previous 6th day price
Enter previous 5th day price
Enter previous 4th day price
Enter previous 3th day price
Enter previous 2nd day price
Enter previous 1st day price
Predict

Code:

```
<!DOCTYPE html>

<html>
    <head>
        <title>Registration Form</title>
        <link rel="stylesheet"
        href="register.css" type="text/css">
    </head>

    <body>
        <div class="main">

            <div class="form">
                <h2>Register</h2>

                <input type="text" name="fname" placeholder="First Name">

                <input type="text" name="lname" placeholder="Last Name">

                <input type="number" name="age" placeholder="Age">

                <input type="email" name="email" placeholder="Email">

                <input type="password" name="pwd" placeholder="Password">

                <input type="password" name="cpwd" placeholder="Confirm Password">

                <input type="text" name="gender" placeholder="Gender">

                <button class="btnn">
                    <a href="#">Sign Up</a>
                </button>

                <p class="link">Already have an account<br>

                <a href="homepage.html">Login </a> here</a></p>
            </div>
        </div>
    </body>
</html>
```

8. TESTING

8.1 Test case

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 priceprediction project at the time of the release to user acceptance testing

Defect Analysis:

The report shows the number of resolved and closed bugs at each severity level andhow 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

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

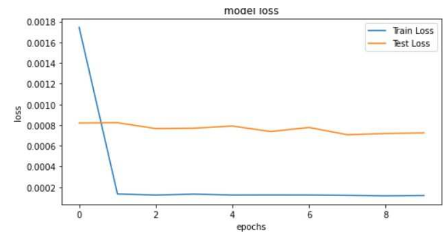
Test case analysis
This report shows the number of test cases that have passed, failed, and untested

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

9.RESULT

9.1 Performance Metrics

S.No	Parameters	Values	Screenshot
1.	Model Summary		Model: "sequential_1" Layer (type) Output Shape Param # === lstm_3 (LSTM) (None, 10, 50) 10400 lstm_4 (LSTM) (None, 10, 50) 20200 lstm_5 (LSTM) (None, 50) 20200 dense_1 (Dense) (None, 1) 51 === Total params: 50,851 Trainable params: 50,851 Non-trainable params: 0

2.	Accuracy		Train Mean Absolute Error: 1.0278217422906264 Train Root Mean Squared Error: 1.4285248639934083 Test Mean Absolute Error: 2.780526920817909 Test Root Mean Squared Error: 3.6348234466523737 

10.ADVANTAGES & DISADVANTAGES

ADVANTAGES

- Prediction of crude oil price can help the importers to choose the right time to buyas 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

DISADVANATGES

- 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 dataor 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.

APPENDIX

SOURCE CODE

MODEL:

#DATA PREPROCESSING

#IMPORTING THE LIBRARIES

```
{
 "nbformat": 4,
 "nbformat_minor": 0,
 "metadata": {
  "colab": {
   "provenance": []
  },
  "kernelspec": {
   "name": "python3",
   "display_name": "Python 3"
```

```
    },
    "language_info": {
      "name": "python"
    }
  },
  "cells": [
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "P39WdS_RH_ol"
      },
      "outputs": [],
      "source": [
        "import pandas as pd\n",
        "import numpy as np\n",
        "import matplotlib.pyplot as plt"
      ]
    }
  ]
}
```

#HANDLING MISSING VALUES

```
{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
```

"provenance": []
},
"kernelspec": {
"name": "python3",
"display_name": "Python 3"
},
"language_info": {
"name": "python"
}
},
"cells": [
{
"cell_type": "code",
"source": [
"import pandas as pd\n",
"import numpy as np\n",
"import matplotlib.pyplot as plt"
],
"metadata": {
"id": "GiRQ27X4JRcH"
},
"execution_count": 1,
"outputs": []
},
{
"cell_type": "code",
"source": [
"data=pd.read_excel(\"/content/Crude Oil Prices Daily.xlsx\")"

],
"metadata": {
"id": "dbaHUfMiJW8I"
},
"execution_count": 2,
"outputs": []
},
{
"cell_type": "code",
"source": [
"data.isnull().any()"
],
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "RAUyZB0-Jp0t",
"outputId": "9f1ec869-c35f-4764-8989-0d5d82646e18"
},
"execution_count": 3,
"outputs": [
{
"output_type": "execute_result",
"data": {
"text/plain": [
"Date   False\n",
"Closing Value   True\n",
"dtype: bool"

```
      ]
     },
     "metadata": {},
     "execution_count": 3
    }
   ]
  },
  {
   "cell_type": "code",
   "source": [
    "data.isnull().sum()"
   ],
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "yorF39lCJsyV",
    "outputId": "4001984b-54cc-4b54-f73b-926ff03ac00f"
   },
   "execution_count": 4,
   "outputs": [
    {
     "output_type": "execute_result",
     "data": {
      "text/plain": [
       "Date             0\n",
       "Closing Value    7\n",
       "dtype: int64"
```

```
        ]
      },
      "metadata": {},
      "execution_count": 4
    }
  ]
},
{
  "cell_type": "code",
  "source": [
    "data.dropna(axis=0,inplace=True)"
  ],
  "metadata": {
    "id": "g3FWuWVPJwms"
  },
  "execution_count": 5,
  "outputs": []
},
{
  "cell_type": "code",
  "source": [
    "data.isnull().sum()"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "vIHbyOs4JzIf",
```

"outputId": "47f79e73-6c7b-4d04-a0bd-09ae39d18bcf"
},
"execution_count": 6,
"outputs": [
{
"output_type": "execute_result",
"data": {
"text/plain": [
"Date   0\n",
"Closing Value   0\n",
"dtype: int64"
]
},
"metadata": {},
"execution_count": 6
}
]
},
{
"cell_type": "code",
"source": [
"data_oil=data.reset_index()['Closing Value']\n",
"data_oil"
],
"metadata": {
"id": "IoxUNwrvJ2b6",
"outputId": "2011717f-466c-4af2-973c-3980b6229a4d",
"colab": {

```
    "base_uri": "https://localhost:8080/"
   }
  },
  "execution_count": 7,
  "outputs": [
   {
    "output_type": "execute_result",
    "data": {
     "text/plain": [
      "0       25.56\n",
      "1       26.00\n",
      "2       26.53\n",
      "3       25.85\n",
      "4       25.87\n",
      "        ...  \n",
      "8211    73.89\n",
      "8212    74.19\n",
      "8213    73.05\n",
      "8214    73.78\n",
      "8215    73.93\n",
      "Name: Closing Value, Length: 8216, dtype: float64"
     ]
    },
    "metadata": {},
    "execution_count": 7
   }
  ]
 }
```

 ]
}

#FEATURE SCALING

{
 "nbformat": 4,
 "nbformat_minor": 0,
 "metadata": {
  "colab": {
   "provenance": []
  },
  "kernelspec": {
   "name": "python3",
   "display_name": "Python 3"
  },
  "language_info": {
   "name": "python"
  }
 },
 "cells": [
  {
   "cell_type": "code",
   "source": [
    "import pandas as pd\n",
    "import numpy as np\n",
    "import matplotlib.pyplot as plt"
   ],
   "metadata": {

"id": "GiRQ27X4JRcH"
},
"execution_count": 1,
"outputs": []
},
{
"cell_type": "code",
"source": [
"data=pd.read_excel(\"/content/Crude Oil Prices Daily.xlsx\")"
],
"metadata": {
"id": "dbaHUfMiJW8I"
},
"execution_count": 2,
"outputs": []
},
{
"cell_type": "code",
"source": [
"data.isnull().any()"
],
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "RAUyZB0-Jp0t",
"outputId": "9f1ec869-c35f-4764-8989-0d5d82646e18"
},

```
  "execution_count": 3,
  "outputs": [
   {
    "output_type": "execute_result",
    "data": {
     "text/plain": [
      "Date             False\n",
      "Closing Value     True\n",
      "dtype: bool"
     ]
    },
    "metadata": {},
    "execution_count": 3
   }
  ]
 },
 {
  "cell_type": "code",
  "source": [
   "data.isnull().sum()"
  ],
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/"
   },
   "id": "yorF39lCJsyV",
   "outputId": "4001984b-54cc-4b54-f73b-926ff03ac00f"
  },
```

"execution_count": 4,
"outputs": [
{
"output_type": "execute_result",
"data": {
"text/plain": [
"Date   0\n",
"Closing Value   7\n",
"dtype: int64"
]
},
"metadata": {},
"execution_count": 4
}
]
},
{
"cell_type": "code",
"source": [
"data.dropna(axis=0,inplace=True)"
],
"metadata": {
"id": "g3FWuWVPJwms"
},
"execution_count": 5,
"outputs": []
},
{

"cell_type": "code",
"source": [
"data.isnull().sum()"
],
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "vIHbyOs4JzIf",
"outputId": "47f79e73-6c7b-4d04-a0bd-09ae39d18bcf"
},
"execution_count": 6,
"outputs": [
{
"output_type": "execute_result",
"data": {
"text/plain": [
"Date   0\n",
"Closing Value   0\n",
"dtype: int64"
]
},
"metadata": {},
"execution_count": 6
}
]
},
{

"cell_type": "code",
"source": [
 "data_oil=data.reset_index()['Closing Value']\n",
 "data_oil"
],
"metadata": {
 "colab": {
  "base_uri": "https://localhost:8080/"
 },
 "id": "IoxUNwrvJ2b6",
 "outputId": "2011717f-466c-4af2-973c-3980b6229a4d"
},
"execution_count": 7,
"outputs": [
 {
  "output_type": "execute_result",
  "data": {
   "text/plain": [
    "0   25.56\n",
    "1   26.00\n",
    "2   26.53\n",
    "3   25.85\n",
    "4   25.87\n",
    "   ... \n",
    "8211   73.89\n",
    "8212   74.19\n",
    "8213   73.05\n",
    "8214   73.78\n",

"8215   73.93\n",
"Name: Closing Value, Length: 8216, dtype: float64"
]
},
"metadata": {},
"execution_count": 7
}
]
},
{
"cell_type": "code",
"source": [
"from sklearn.preprocessing import MinMaxScaler\n",
"scaler=MinMaxScaler(feature_range=(0,1))\n",
"data_oil=scaler.fit_transform(np.array(data_oil).reshape(-1,1))"
],
"metadata": {
"id": "5m-DUFI9J_WN"
},
"execution_count": 8,
"outputs": []
},
{
"cell_type": "code",
"source": [
"data_oil"
],
"metadata": {

"id": "fyMwOo1jKLqL",
"outputId": "6e2bdf01-77aa-4e93-f117-6ab963e21c0a",
"colab": {
"base_uri": "https://localhost:8080/"
}
},
"execution_count": 9,
"outputs": [
{
"output_type": "execute_result",
"data": {
"text/plain": [
"array([[0.11335703],\n",
"   [0.11661484],\n",
"   [0.12053902],\n",
"   ...,\n",
"   [0.46497853],\n",
"   [0.47038353],\n",
"   [0.47149415]])"
]
},
"metadata": {},
"execution_count": 9
}
]
}
]
}

DATA VISUALIZATION

{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "provenance": []
    },
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3"
    },
    "language_info": {
      "name": "python"
    }
  },
  "cells": [
    {
      "cell_type": "code",
      "source": [
        "import pandas as pd\n",
        "import numpy as np\n",
        "import matplotlib.pyplot as plt"
      ],
      "metadata": {
        "id": "GiRQ27X4JRcH"
      },

"execution_count": 1,
"outputs": []
},
{
"cell_type": "code",
"source": [
"data=pd.read_excel(\"/content/Crude Oil Prices Daily.xlsx\")"
],
"metadata": {
"id": "dbaHUfMiJW8I"
},
"execution_count": 2,
"outputs": []
},
{
"cell_type": "code",
"source": [
"data.isnull().any()"
],
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "RAUyZB0-Jp0t",
"outputId": "9f1ec869-c35f-4764-8989-0d5d82646e18"
},
"execution_count": 3,
"outputs": [

```
    {
     "output_type": "execute_result",
     "data": {
      "text/plain": [
       "Date             False\n",
       "Closing Value    True\n",
       "dtype: bool"
      ]
     },
     "metadata": {},
     "execution_count": 3
    }
   ]
  },
  {
   "cell_type": "code",
   "source": [
    "data.isnull().sum()"
   ],
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "yorF39lCJsyV",
    "outputId": "4001984b-54cc-4b54-f73b-926ff03ac00f"
   },
   "execution_count": 4,
   "outputs": [
```

```
    {
      "output_type": "execute_result",
      "data": {
        "text/plain": [
          "Date             0\n",
          "Closing Value    7\n",
          "dtype: int64"
        ]
      },
      "metadata": {},
      "execution_count": 4
    }
  ]
},
{
  "cell_type": "code",
  "source": [
    "data.dropna(axis=0,inplace=True)"
  ],
  "metadata": {
    "id": "g3FWuWVPJwms"
  },
  "execution_count": 5,
  "outputs": []
},
{
  "cell_type": "code",
  "source": [
```

"data.isnull().sum()"
],
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "vIHbyOs4JzIf",
"outputId": "47f79e73-6c7b-4d04-a0bd-09ae39d18bcf"
},
"execution_count": 6,
"outputs": [
{
"output_type": "execute_result",
"data": {
"text/plain": [
"Date   0\n",
"Closing Value   0\n",
"dtype: int64"
]
},
"metadata": {},
"execution_count": 6
}
]
},
{
"cell_type": "code",
"source": [

"data_oil=data.reset_index()['Closing Value']\n",
"data_oil"
],
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "IoxUNwrvJ2b6",
"outputId": "2011717f-466c-4af2-973c-3980b6229a4d"
},
"execution_count": 7,
"outputs": [
{
"output_type": "execute_result",
"data": {
"text/plain": [
"0   25.56\n",
"1   26.00\n",
"2   26.53\n",
"3   25.85\n",
"4   25.87\n",
"   ... \n",
"8211   73.89\n",
"8212   74.19\n",
"8213   73.05\n",
"8214   73.78\n",
"8215   73.93\n",
"Name: Closing Value, Length: 8216, dtype: float64"

```
        ]
      },
      "metadata": {},
      "execution_count": 7
    }
  ]
},
{
  "cell_type": "code",
  "source": [
    "from sklearn.preprocessing import MinMaxScaler\n",
    "scaler=MinMaxScaler(feature_range=(0,1))\n",
    "data_oil=scaler.fit_transform(np.array(data_oil).reshape(-1,1))"
  ],
  "metadata": {
    "id": "5m-DUFI9J_WN"
  },
  "execution_count": 8,
  "outputs": []
},
{
  "cell_type": "code",
  "source": [
    "data_oil"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
```

```
    },
    "id": "fyMwOo1jKLqL",
    "outputId": "6e2bdf01-77aa-4e93-f117-6ab963e21c0a"
   },
   "execution_count": 9,
   "outputs": [
    {
     "output_type": "execute_result",
     "data": {
      "text/plain": [
       "array([[0.11335703],\n",
       "       [0.11661484],\n",
       "       [0.12053902],\n",
       "       ...,\n",
       "       [0.46497853],\n",
       "       [0.47038353],\n",
       "       [0.47149415]])"
      ]
     },
     "metadata": {},
     "execution_count": 9
    }
   ]
  },
  {
   "cell_type": "code",
   "source": [
    "plt.plot(data_oil)"
```

],
"metadata": {
  "id": "GdNJartuKUfk",
  "outputId": "0640d4b9-abe5-4e76-a910-871af84bc19c",
  "colab": {
    "base_uri": "https://localhost:8080/",
    "height": 282
  }
},
"execution_count": 10,
"outputs": [
  {
    "output_type": "execute_result",
    "data": {
      "text/plain": [
        "[<matplotlib.lines.Line2D at 0x7f5a0c7b5850>]"
      ]
    },
    "metadata": {},
    "execution_count": 10
  },
  {
    "output_type": "display_data",
    "data": {
      "text/plain": [
        "<Figure size 432x288 with 1 Axes>"
      ],
      "image/png":

#CREATING A DATA SET WITH SLIDING WINDOWS


```
{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "provenance": []
    },
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3"
    },
    "language_info": {
      "name": "python"
    }
  },
  "cells": [
    {
      "cell_type": "code",
      "source": [
        "import pandas as pd\n",
        "import numpy as np\n",
        "import matplotlib.pyplot as plt"
      ],
      "metadata": {
        "id": "GiRQ27X4JRcH"
      },
```

"execution_count": 1,
"outputs": []
},
{
"cell_type": "code",
"source": [
"data=pd.read_excel(\"/content/Crude Oil Prices Daily.xlsx\")"
],
"metadata": {
"id": "dbaHUfMiJW8I"
},
"execution_count": 2,
"outputs": []
},
{
"cell_type": "code",
"source": [
"data.isnull().any()"
],
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "RAUyZB0-Jp0t",
"outputId": "9f1ec869-c35f-4764-8989-0d5d82646e18"
},
"execution_count": 3,
"outputs": [

{
"output_type": "execute_result",
"data": {
"text/plain": [
"Date False\n",
"Closing Value True\n",
"dtype: bool"
]
},
"metadata": {},
"execution_count": 3
}
]
},
{
"cell_type": "code",
"source": [
"data.isnull().sum()"
],
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "yorF39lCJsyV",
"outputId": "4001984b-54cc-4b54-f73b-926ff03ac00f"
},
"execution_count": 4,
"outputs": [

```
    {
     "output_type": "execute_result",
     "data": {
      "text/plain": [
       "Date             0\n",
       "Closing Value    7\n",
       "dtype: int64"
      ]
     },
     "metadata": {},
     "execution_count": 4
    }
   ]
  },
  {
   "cell_type": "code",
   "source": [
    "data.dropna(axis=0,inplace=True)"
   ],
   "metadata": {
    "id": "g3FWuWVPJwms"
   },
   "execution_count": 5,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
```

```
    "data.isnull().sum()"
   ],
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "vIHbyOs4JzIf",
    "outputId": "47f79e73-6c7b-4d04-a0bd-09ae39d18bcf"
   },
   "execution_count": 6,
   "outputs": [
    {
     "output_type": "execute_result",
     "data": {
      "text/plain": [
       "Date             0\n",
       "Closing Value    0\n",
       "dtype: int64"
      ]
     },
     "metadata": {},
     "execution_count": 6
    }
   ]
  },
  {
   "cell_type": "code",
   "source": [
```

"data_oil=data.reset_index()['Closing Value']\n",
"data_oil"
],
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "IoxUNwrvJ2b6",
"outputId": "2011717f-466c-4af2-973c-3980b6229a4d"
},
"execution_count": 7,
"outputs": [
{
"output_type": "execute_result",
"data": {
"text/plain": [
"0   25.56\n",
"1   26.00\n",
"2   26.53\n",
"3   25.85\n",
"4   25.87\n",
"   ... \n",
"8211   73.89\n",
"8212   74.19\n",
"8213   73.05\n",
"8214   73.78\n",
"8215   73.93\n",
"Name: Closing Value, Length: 8216, dtype: float64"

```
        ]
      },
      "metadata": {},
      "execution_count": 7
    }
  ]
},
{
  "cell_type": "code",
  "source": [
    "from sklearn.preprocessing import MinMaxScaler\n",
    "scaler=MinMaxScaler(feature_range=(0,1))\n",
    "data_oil=scaler.fit_transform(np.array(data_oil).reshape(-1,1))"
  ],
  "metadata": {
    "id": "5m-DUFI9J_WN"
  },
  "execution_count": 8,
  "outputs": []
},
{
  "cell_type": "code",
  "source": [
    "data_oil"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
```

```
    },
    "id": "fyMwOo1jKLqL",
    "outputId": "6e2bdf01-77aa-4e93-f117-6ab963e21c0a"
   },
   "execution_count": 9,
   "outputs": [
    {
     "output_type": "execute_result",
     "data": {
      "text/plain": [
       "array([[0.11335703],\n",
       "       [0.11661484],\n",
       "       [0.12053902],\n",
       "       ...,\n",
       "       [0.46497853],\n",
       "       [0.47038353],\n",
       "       [0.47149415]])"
      ]
     },
     "metadata": {},
     "execution_count": 9
    }
   ]
  },
  {
   "cell_type": "code",
   "source": [
    "plt.plot(data_oil)"
```

],
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/",
"height": 282
},
"id": "GdNJartuKUfk",
"outputId": "0640d4b9-abe5-4e76-a910-871af84bc19c"
},
"execution_count": 10,
"outputs": [
{
"output_type": "execute_result",
"data": {
"text/plain": [
"[<matplotlib.lines.Line2D at 0x7f5a0c7b5850>]"
]
},
"metadata": {},
"execution_count": 10
},
{
"output_type": "display_data",
"data": {
"text/plain": [
"<Figure size 432x288 with 1 Axes>"
],
"image/png":

    },
    "metadata": {
     "needs_background": "light"
    }
   }
  ]
 },
 {
  "cell_type": "code",
  "source": [
   "training_size=int(len(data_oil)*0.65)\n",
   "test_size=len(data_oil)-training_size\n",
   "train_data,test_data=data_oil[0:training_size,:],data_oil[training_size:len(data_oil),:1]"
  ],
  "metadata": {
   "id": "S9THgFu3KaKH"
  },
  "execution_count": 11,
  "outputs": []
 },
 {
  "cell_type": "code",
  "source": [
   "training_size,test_size"
  ],
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/"

},
"id": "mbvhl2HCKmSG",
"outputId": "5142d0f7-21d1-46bb-e69e-bd71c69ef729"
},
"execution_count": 12,
"outputs": [
{
"output_type": "execute_result",
"data": {
"text/plain": [
"(5340, 2876)"
]
},
"metadata": {},
"execution_count": 12
}
]
},
{
"cell_type": "code",
"source": [
"train_data.shape"
],
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "M14MiYVzKrJg",

"outputId": "b23a9806-135a-424d-f65b-6029116ad975"
},
"execution_count": 13,
"outputs": [
{
"output_type": "execute_result",
"data": {
"text/plain": [
"(5340, 1)"
]
},
"metadata": {},
"execution_count": 13
}
]
},
{
"cell_type": "code",
"source": [
"def create_dataset(dataset,time_step=1):\n",
" dataX,dataY=[],[]\n",
" for i in range(len(dataset)-time_step-1):\n",
"   a=dataset[i:(i+time_step),0]\n",
"   dataX.append(a)\n",
"   dataY.append(dataset[i+time_step,0])\n",
" return np.array(dataX),np.array(dataY)"
],
"metadata": {

"id": "vrH4kcM5K9S8"
},
"execution_count": 14,
"outputs": []
},
{
"cell_type": "code",
"source": [
"time_step=10\n",
"x_train,y_train=create_dataset(train_data,time_step)\n",
"x_test,y_test=create_dataset(test_data,time_step)"
],
"metadata": {
"id": "dv2OIjslLA3v"
},
"execution_count": 15,
"outputs": []
},
{
"cell_type": "code",
"source": [
"print(x_train.shape),print(y_train.shape)"
],
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "ApfZGkVhLD2p",

```
    "outputId": "979643ec-00c5-4f15-93f0-8529e83335a0"
   },
   "execution_count": 16,
   "outputs": [
    {
     "output_type": "stream",
     "name": "stdout",
     "text": [
      "(5329, 10)\n",
      "(5329,)\n"
     ]
    },
    {
     "output_type": "execute_result",
     "data": {
      "text/plain": [
       "(None, None)"
      ]
     },
     "metadata": {},
     "execution_count": 16
    }
   ]
  },
  {
   "cell_type": "code",
   "source": [
    "print(x_test.shape),print(y_test.shape)"
```

],
"metadata": {
  "colab": {
    "base_uri": "https://localhost:8080/"
  },
  "id": "iLQiXr21LE35",
  "outputId": "3dca6c25-95ed-4b85-bb1a-d2e218179c5d"
},
"execution_count": 17,
"outputs": [
  {
    "output_type": "stream",
    "name": "stdout",
    "text": [
      "(2865, 10)\n",
      "(2865,)\n"
    ]
  },
  {
    "output_type": "execute_result",
    "data": {
      "text/plain": [
        "(None, None)"
      ]
    },
    "metadata": {},
    "execution_count": 17
  }

]
},
{
"cell_type": "code",
"source": [
"x_train"
],
"metadata": {
"id": "w95tBQ4gLJWj",
"outputId": "f1567ef1-fb46-4909-857b-c3c34c6f1dd1",
"colab": {
"base_uri": "https://localhost:8080/"
}
},
"execution_count": 18,
"outputs": [
{
"output_type": "execute_result",
"data": {
"text/plain": [
"array([[0.11335703, 0.11661484, 0.12053902, ..., 0.10980305, 0.1089886 ,\n",
"   0.11054346],\n",
"   [0.11661484, 0.12053902, 0.11550422, ..., 0.1089886 , 0.11054346,\n",
"   0.10165852],\n",
"   [0.12053902, 0.11550422, 0.1156523 , ..., 0.11054346, 0.10165852,\n",
"   0.09906708],\n",
"   ...,\n",
"   [0.36731823, 0.35176958, 0.36080261, ..., 0.36391234, 0.37042796,\n",

"   0.37042796],\n",
"   [0.35176958, 0.36080261, 0.35354657, ..., 0.37042796, 0.37042796,\n",
"   0.37879461],\n",
"   [0.36080261, 0.35354657, 0.35295424, ..., 0.37042796, 0.37879461,\n",
"   0.37916482]])"
]
},
"metadata": {},
"execution_count": 18
}
]
},
{
"cell_type": "code",
"source": [
"x_train=x_train.reshape(x_train.shape[0],x_train.shape[1],1)\n",
"x_test=x_test.reshape(x_test.shape[0],x_test.shape[1],1)"
],
"metadata": {
"id": "59VQDX1LLL3D"
},
"execution_count": 19,
"outputs": []
}
]
}

MODEL BUILDING

 # Importing the model building libraries

```
from tensorflow.keras.models import
Sequentialfrom tensorflow.keras.layers
import Dense
from tensorflow.keras.layers import
LSTM# Initializing the model
model=Sequential()
```

 # Adding LSTM Layers

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

Adding Output Layers

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

 # Configure The Learning Process

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

 # Train The Model

```
model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=10,batch_size=6
4,verbose=1)
```

 # Model Evaluation

```
train_predict=model.predict(X_train)
test_predict=model.predict(X_test)
train_predict=scalar.inverse_transform(train_predict
)test_predict=scalar.inverse_transform(test_predict)
import math
from sklearn.metrics import mean_squared_error

math.sqrt(mean_squared_error(y_train,train_predict))
```

Save The Model

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

Test The Model

```
look_back= 10
trainPredictPlot =
np.empty_like(data_oil)
trainPredictPlot[:, :]= np.nan
trainPredictPlot[look_back:len(train_predict)+look_back, :]=
train_predicttestPredictPlot =np.empty_like(data_oil)
testPredictPlot[:, :]= np.nan
testPredictPlot[len(train_predict)+(look_back*2)+1:len(data_oil)-1, :]=
test_predictplt.plot(scalar.inverse_transform(data_oil))
plt.plot(trainPredictPl
ot)
plt.plot(testPredictPlo
t) plt.show()
len(test_data)
x_input=test_data[2866:].reshape(1,-
1)x_input.shape
temp_input=list(x_input)
temp_input=temp_input[0].toli
st()temp_input
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+

    1else:

        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)
day_new=np.arange(1,11)
day_pred=np.arange(11,2
1)len(data_oil)
plt.plot(day_new,scalar.inverse_transform(data_oil[8206:]
))plt.title("Review of prediction")
plt.plot(day_pred,scalar.inverse_transform(lst_output))
plt.show()
df3=data_oil.tolist()
```

```
df3.extend(lst_outpu
t)
plt.title("Past data nad next 10 days output
prediction")plt.plot(df3[8100:])
df3=scalar.inverse_transform(df3).tolist()
plt.title("Past data nad next 10 days output prediction after reversing the scaled values")
```

Index.html:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Crudeoil price prediction</title>
        <!--<link rel="stylesheet" href="in.css">
        <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;
700&display=swap"rel="stylesheet">
        <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">-->
      <style
         >ul {
  list-style-type:
  none;margin: 0;
  padding: 0;
  overflow:
  hidden;
  border: 1px solid
  #e7e7e7; background-
  color: #057514;
}

li {
  float: left;
```

```
}

li a {
  display: inline-
  block; color: rgb(78,
  3, 3); text-align:
  center; padding:
  14px 16px; text-
  decoration: none;
  background-color:rgb(18, 116, 5) ;
}
li a:hover{
```

```
   border:1px solid;
   background-color: lightseagreen;
}
     </style>
   </head>
   <body>
      <nav class="navbar navbar-inverse">
         <div class="container-fluid">
            <ul>
              <li class="parts"><a href="#">Home</a></li>
              <li class="parts"><a href="predict.html">predict</a></li>
          </ul>
         </div>
       </nav>

      <h1 >Crudeoil price prediction</h1>

      <style>
         body {
           background-image:
           url('static/css/image.jpeg');background-
           repeat: no-repeat;
           background-attachment:
         fixed;background-size: 100%
         100%;

         }
         </style>

       <h3 style="font-family:system-ui;">
          Demand for oil is inelastic, therefore the rise in
          price is good news for producers because they will see an
          increase in their revenue. Oil importers, however, will
          experience
          increased costs of purchasing oil. Because oil is the largest traded
          commodity,the effects are quite significant. A rising oil price can even
          shift
economic/political
         power from oil importers to oil exporters. The crude oil rice movements are
```

```
subject to
          diverse influencing factors</h3>
       </body>
       </html>
```

Predict.html:

```
<html>
   <head>
      <link rel="stylesheet" href="static/css/style.css">
      <style>
         body {
           background-image:
           url('static/css/image3.jpg');background-
           repeat: no-repeat;
           background-attachment:
         fixed;background-size: 100%
         100%;

         }
         </style>

   </head>
   <script>
      document.getElementByID("demo").innerHTML =
document.getElementById("ten");
   </script>
<body>
<form action="/method" method="POST" enctype = "multipart/form-data">
<div class="container">
   <!--<div class="brand-logo"></div>-->
   <div class="brand-title">predict the oil price</div>
   <div class="inputs">
     <label>Enter Price</label>
     <input type="text" placeholder="Enter ten days price" id="ten" name="val"/>
     <button
```

```
        type="submit">Predict</button><br><br>
        The next day price is :  {{prediction}}
    </div>
  </div>
</form>
</body>
</html>
```

App.py:

```
from flask import Flask, render_template, request,
redirectimport 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 install ibm_watson_machine_learning

from ibm_watson_machine_learning import
APIClientwml_credentials = {
        "url" : "https://us-south.ml.cloud.ibm.com",
        "apikey" : "cRkqykhsnLO1Ogs_xoYjgLkNTtTS1Qxyi0Mn1GSlQ1P5"
}
client = APIClient(wml_credentials)

#for creating a 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 & Project Demo Link

GitHub : https://github.com/IBM-EPBL/IBM-Project-28401-1660111580

PROJECT DEMO LINK : https://youtu.be/dFCIT-6Z28M