**INTRODUCTION**

**1.1 Project Overview**

Crude oil is among the main assets in this day and age, it is the central fuel and its expense straightforwardly affects the worldwide environment, our economy and oil investigation, abuse and different exercises. Expectation of oil costs has turned into a need of great importance, it is a shelter to numerous huge and little ventures, people, and the public authority.

As a vital info figure in modern creation, the value unpredictability of raw petroleum frequently achieves monetary unpredictability, so estimating raw petroleum cost has forever been an urgent issue in financial matters. In our review, we built a LSTM (short for Long Momentary Memory brain organisation) model to lead this determining in view of information

**1.2 Purpose**

The major goal of this project is to employ neural networks to forecast the price of crude oil. This choice enables us to purchase crude oil at the appropriate time. The best solution for this type of prediction is time series analysis because we are using past data on crude oil prices to forecast future prices. Therefore, to complete the objective, by designing an RNN (Recurrent Neural Network) with an LSTM (Long Short Term Memory).

**LITERATURE SURVEY**

| **S.no** | **Author** | **Title** | **Objective** |
| --- | --- | --- | --- |
| **1** | Nidhi Moitra et al. (2020) | Crude Oil Price Prediction Using Lstm [1] | In this paper, Recurrent neural networks that are LSTM-based are used to predict the price of crude oil. The most effective and powerful models for processing time-series-based sequential data are recurrent neural networks (RNNs). In addition to prediction, LSTM variations can be utilised for tasks including polyphonic modelling, speech recognition, and handwriting recognition. |
| 2 | Varun Gupta et al. (2018) | Crude Oil Price Prediction Using LSTM Networks [2] | In this study, For the objective of predicting the price of crude oil, LSTM-based recurrent neural networks have been utilised. One of the most effective RNN architectures is LSTM. The hidden layer of the network's LSTM introduces the memory cell, which makes it well-suited to grasp the changing structure of data with a high capacity for prediction. |
| 3 | Zhenda Hu et al. (2021) | Crude oil price prediction using CEEMDAN and LSTMattention with news sentiment index | This paper combines Complete Ensemble Empirical Mode Decomposition with Adaptive Noise (CEEMDAN), Long Short-Term Memory (LSTM) with attention mechanism and addition, following the well known “decomposition and ensemble” framework to study the crude oil prices |
| 4 | Kexian Zhang et al. (2022) | Forecasting crude oil price using LSTM neural networks | An ANN (short for Artificial Neural Network) model and a typical ARIMA (short for Autoregressive Integrated Moving Average) model are taken as the comparable models. The results show that the LSTM model has strong generalisation ability, with stable applicability in forecasting crude oil prices with different timescales. |
| 5 | Shaolong Sun et al. (2021) | Analysis and forecasting of crude oil price based on the variable selection-LSTM integrated model | This paper assesses and selects core influence factors with the elastic-net regularised generalised linear Model (GLMNET), spike-slab lasso method, and Bayesian model average (BMA) and the new machine learning method long short-term Memory Network (LSTM) is developed for crude oil price forecasting. |
| 6 | Norshakirah Aziz et al. (2020) | Predictive analytics for crude oil price using rnn-lstm neural network | This study demonstrated the use of RNN-LSTM networks for predicting the crude oil price based on historical data alongside other technical analysis indicators. This study aims to certify the capability of a prediction model built based on the RNN-LSTM network to predict the future price of crude oil. |
| 7 | Rayan H. Assaad et al. (2021) | Predicting the Price of Crude Oil and its Fluctuations Using LSTM, and Convolutional Neural Networks | Deep neural networks, long short term memory (LSTM) neural networks, and a combination of convolutional and LSTM neural networks are being used here. The findings suggest that LSTM networks are the best architectures to predict the crude oil price. The outcomes of this paper could potentially help in making the oil price prediction mechanism more traceable. |
| 8 | Kaijian He et al. (2017) | Forecasting Crude Oil Prices: a Deep Learning based Model | In this paper, we use the deep learning model to capture the unknown complex nonlinear characteristics of the crude oil price movement. We further propose a new hybrid crude oil price forecasting model based on the deep learning model |
| 9 | Rajesh Prasad et al. (2020) | CPPCNDL: Crude oil price prediction using complex network and deep learning algorithms | This paper proposed a hybrid model for crude oil price prediction that uses the complex network analysis and long short-term memory (LSTM) of the deep learning algorithms. The complex network analysis tool called the visibility graph is used to map the dataset on a network and K-core centrality was employed to extract the non-linearity features of crude oil and reconstruct the dataset |
| 10 | Lin Yao et al. (2021) | Prediction of Oil Price Using LSTM | In this paper, we selected the LSTM algorithm to do the oil price’s prediction, to reach good results. RMSE and MAE are selected to represent the prediction’s precision. In this paper, we use a two-layer LSTM network, and the Dense layer is used for the output layer |

**IDEATION & PROPOSED SOLUTION**

**3.1 Empathy Map Canvas**

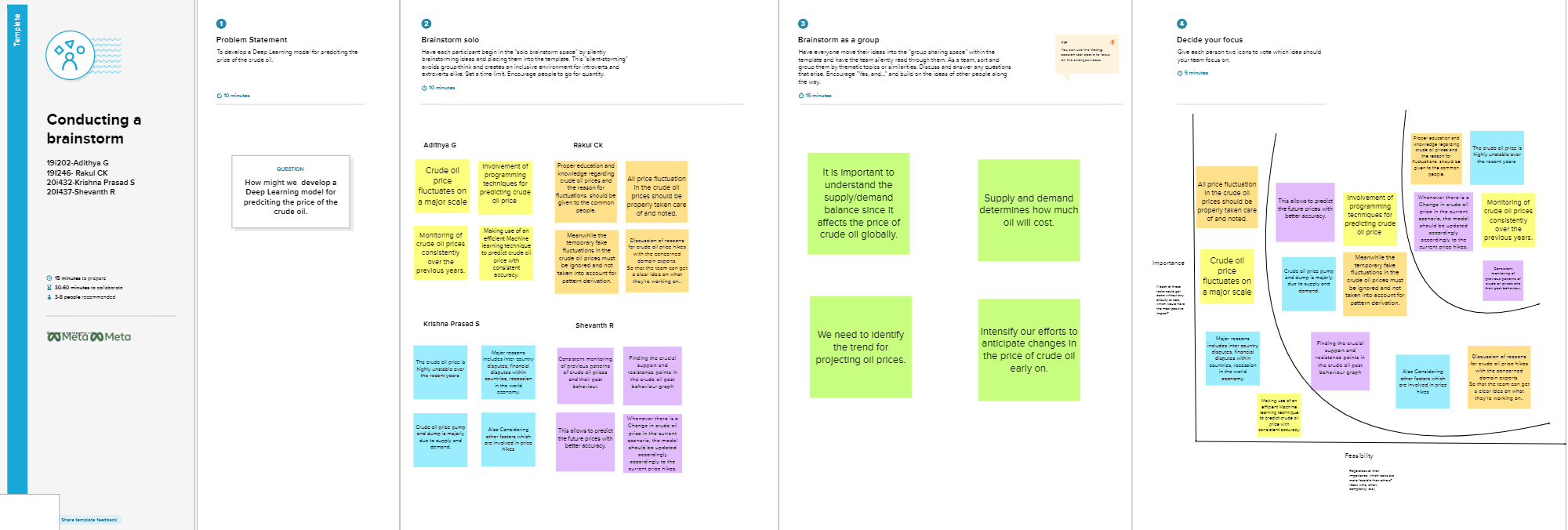
An empathy map is a widely-used visualisation tool within the field of UX and HCI practice. In relation to empathetic design, the primary purpose of an empathy map is to bridge the understanding of the end user.

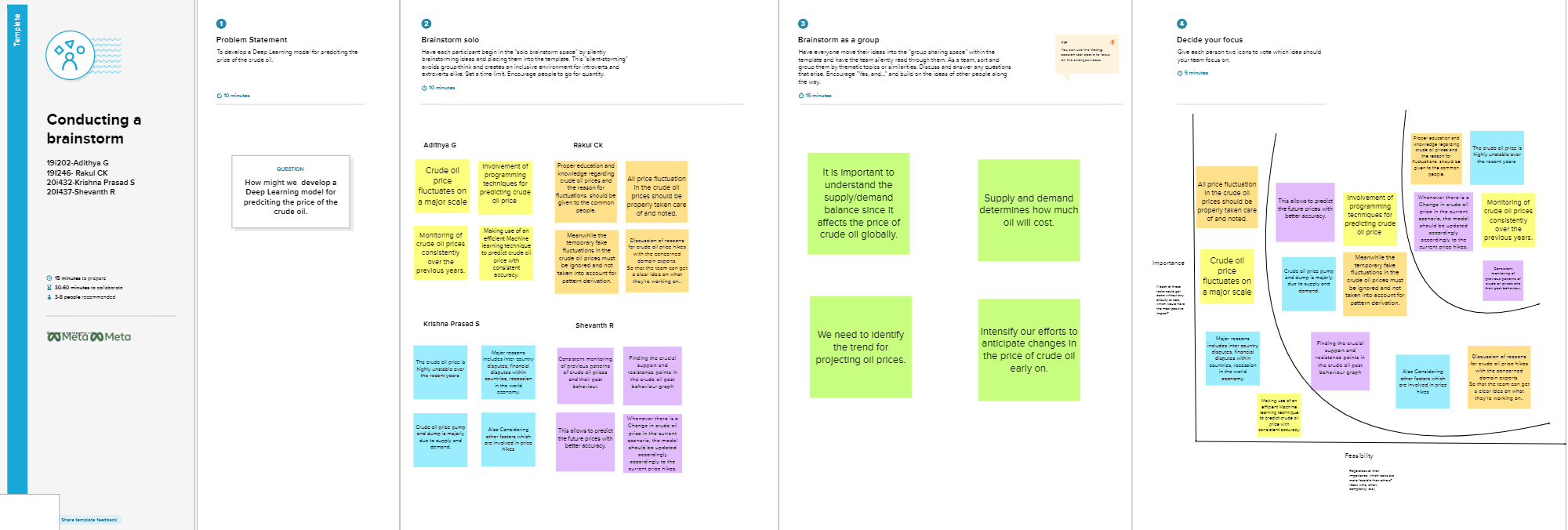


**3.2 Ideation and Brainstorming**

Ideation is the creative process of generating, developing, and communicating new ideas, where an idea is understood as a basic element of thought that can be either visual, concrete, or abstract.

Brainstorming is a group creativity technique by which efforts are made to find a conclusion for a specific problem by gathering a list of ideas spontaneously contributed by its members

****

****

**3.3 Proposed Solution**

We'll be using LSTM. The LSTM introduces the memory cell, a unit of Computation that replaces traditional artificial neurons in the hidden layer of the network. With these memory cells, networks are able to effectively associate memories and input remotely in time, hence suited to grasp the structure of data dynamically over time with high prediction capacity.

There are plenty of algorithms based on machine learning but they don't provide proper results. We'll try to predict crude oil prices using Long Short-Term Memory (LSTM) based recurrent neural networks which will be more suitable for this problem. Price forecasts are very important to various stakeholders: governments, public and private enterprises, policymakers, and Investors.

It is used to forecast future pricing and to optimise oil cost. This price directly affects a number of goods and products, and its changes have an impact on the stock markets. When deciding whether to buy or sell crude oil, it can assist decision-makers from businesses, private investors, or individuals.

**3.4 Problem Solution Fit**

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realised for it actually solves the customer's problem.

* **CUSTOMER SEGMENT(S)**

You must analyse historical prices and forecast/predict future prices using the data. Investors, sellers and government.

* **JOBS-TO-BE-DONE / PROBLEMS**

User friendly application. Website that is able to predict crude oil price.

* **TRIGGERS TR**

Profit for customers

* **EMOTIONS: BEFORE / AFTER**

Unpredictable, fear, joy, happiness

* **AVAILABLE SOLUTIONS**

There are plenty of algorithms based on machine learning but they don't provide proper results. We'll try to predict crude oil prices using Long Short-Term Memory (LSTM) based recurrent neural networks which will be more suitable for this problem.

Price forecasts are very important to various stakeholders: governments, public and private enterprises, policymakers, and Investors.

* **CUSTOMER CONSTRAINTS**

The customer need to enter correct value to get results

* **BEHAVIOUR**

The information provided must be true

* **CHANNELS of BEHAVIOUR**

8.1 ONLINE - All the current affairs,

8.2 OFFLINE - Analysing the previous record risk management

* **PROBLEM ROOT CAUSE**

Varying patterns of crude oil

* **YOUR SOLUTION8**

In this approach we'll be using LSTM. The LSTM introduces the memory cell, a unit of Computation that replaces traditional artificial neurons in the hidden layer of the network.

With these memory cells, networks are able to effectively associate memories and input remote in time, hence suit to grasp the structure of data dynamically over time with high prediction capacity

**REQUIREMENT ANALYSIS**

**4.1 Functional requirements**

| User Registration | Registration through Form Registration through Gmail Registration through LinkedIN |
| --- | --- |
| User Confirmation | Confirmation via Email  Confirmation via OTP |
| Login | User can login through registered email ID or Mobile number |
| Price Charts | The price of oil is displayed using a line graph. |
| Buying | The user can buy the oil at the particular time point. |
| Logout | The user can logout of their login . |

**4.2 Non Functional Requirements**

| Usability | The user interface is simple. We display the data in charts to provide a clear comprehension of price activity. |
| --- | --- |
| Security | We adhere to specific security procedures, such as the usage of user credentials and OTP verification. |
| Reliability | The data shown in the web app is extremely accurate and predicts the correct facts |
| Performance | The performance of the app is determined by the precise price points. |
| Availability | It can be accessed using all types of devices include pcs, and smart phones with different types of OS. |
| Scalability | The app is easily scalable with the help of cloud |

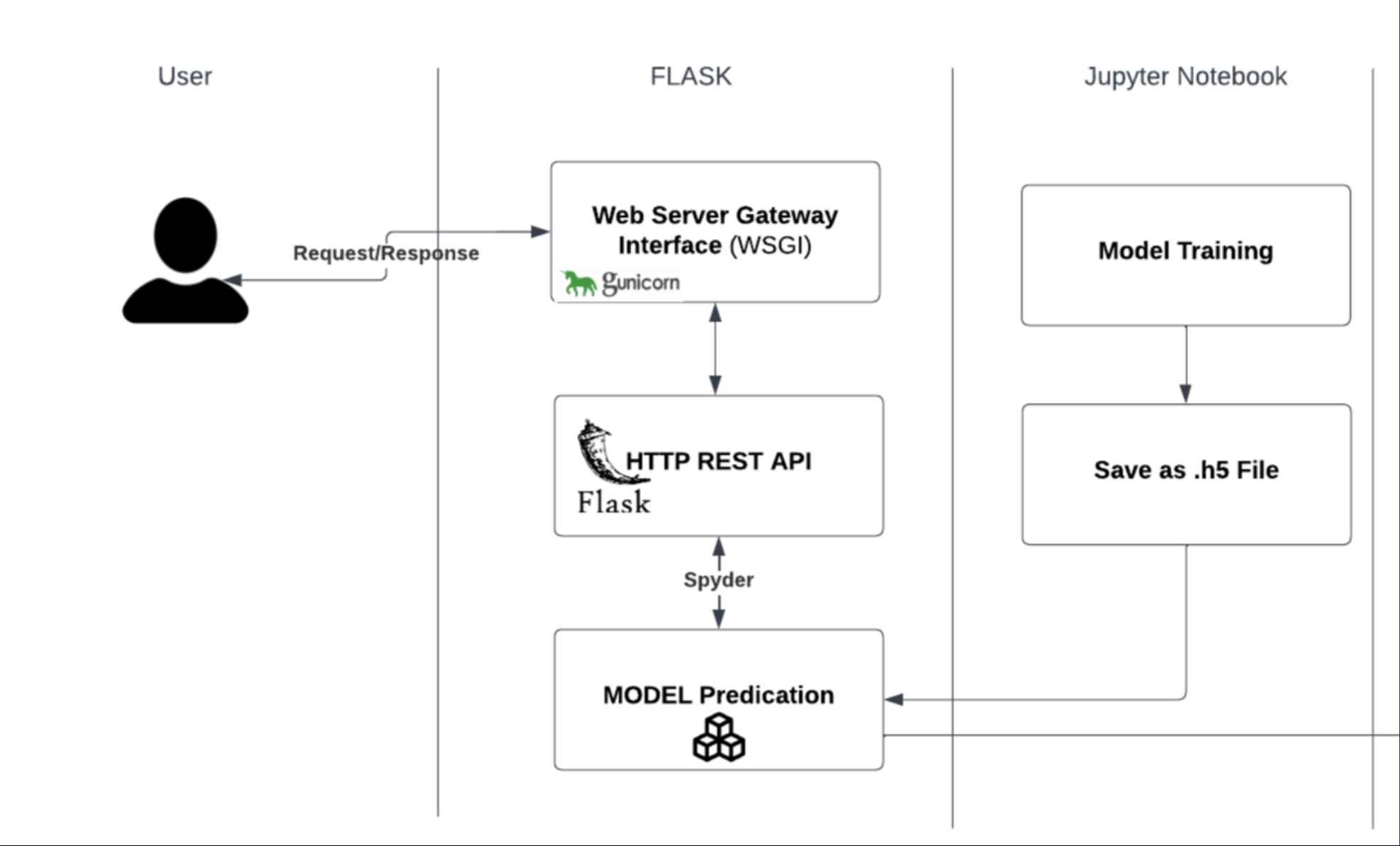
**PROJECT DESIGN**

**5.1 Data Flow Diagrams**

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

**5.2 Solution & Technical Architecture**

Technical Architecture (TA) is a form of IT architecture that is used to design computer systems. It involves the development of a technical blueprint with regard to the arrangement, interaction, and interdependence of all elements so that system-relevant requirements are met.

****

**5.3 User Stories**

A user story is an informal, general explanation of a software feature written from the perspective of the end user or customer. The purpose of a user story is to articulate how a piece of work will deliver a particular value back to the customer.

| **User Type** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Acceptance criteria** | **Priority** | **Release** |
| --- | --- | --- | --- | --- | --- | --- |
| Customer(Web App) | Registration | USN-1 | As a user, I can register for the application by entering my email, password, and confirming my password | I can access my account / dashboard | High | Sprint-1 |
|  | Login | USN-2 | As a user, I can log into the application by entering email & password |  | High | Sprint-1 |
|  | View | USN-3 | As a user, they can see the crude oil price history. | I can access the Data available | High | Sprint-1 |
|  | Input | USN-4 | By Entering the year(date) the model Will analyse and display the output | The analysed result will Be displayed in the screen | High | Sprint-1 |
| Administrator | Model | USN-4 | Update/improve results of the current mode | Access the prev data | High | Sprint-2 |

**PROJECT PLANNING & SCHEDULING**

**6.1 Sprint Planning & Estimation**

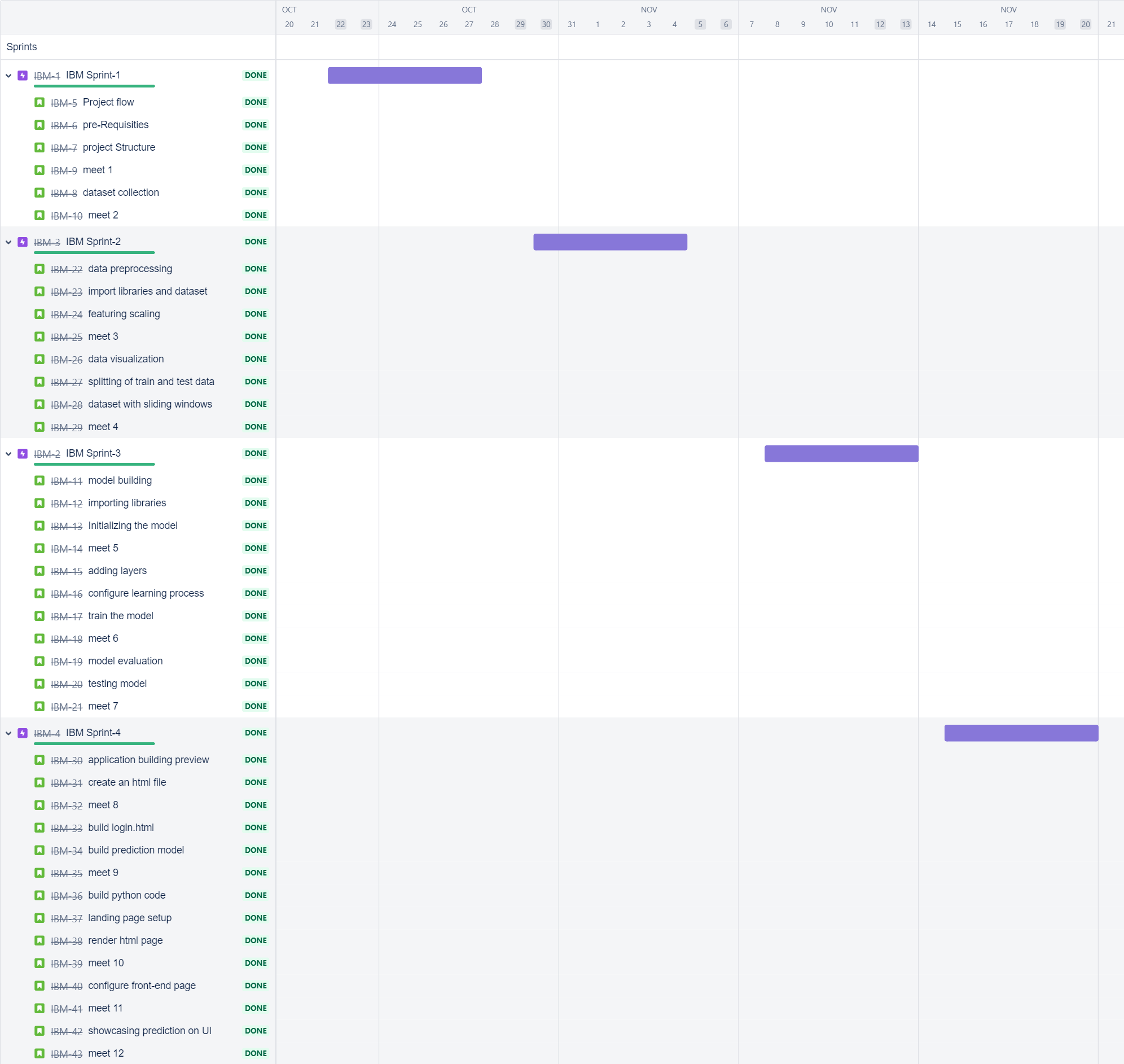
Sprint planning is an event in scrum that kicks off the sprint. The purpose of sprint planning is to define what can be delivered in the sprint and how that work will be achieved. Sprint planning is done in collaboration with the whole scrum team.

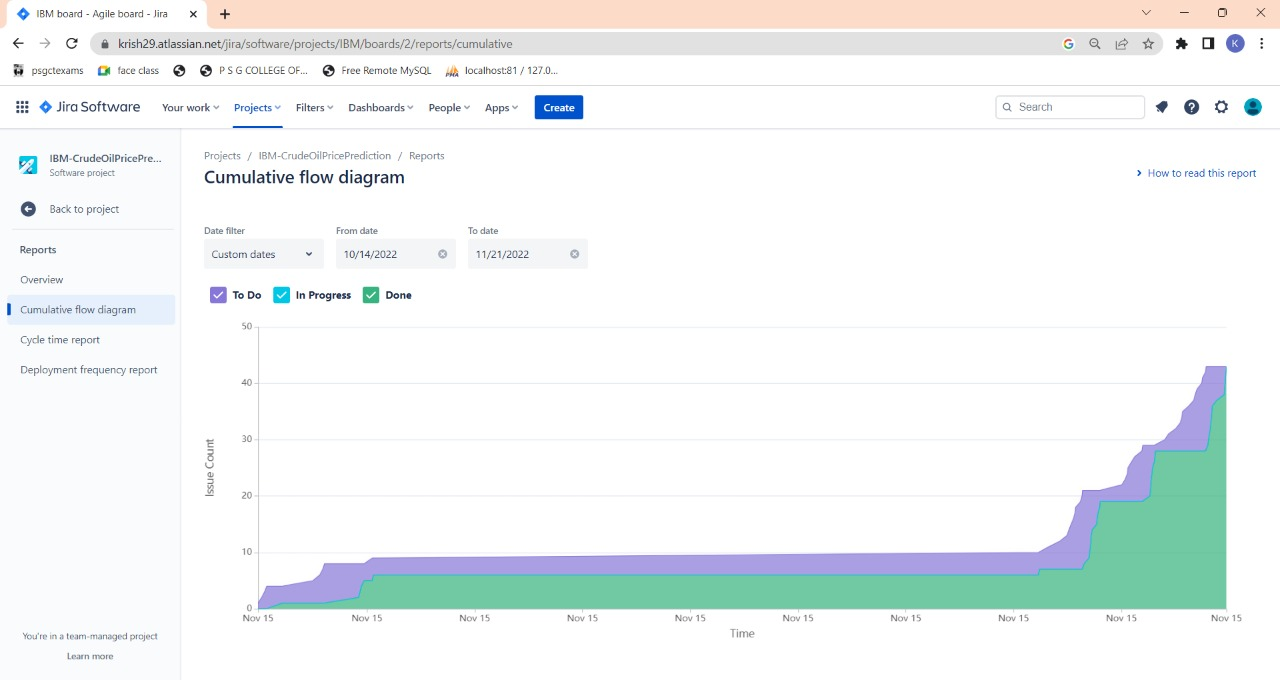
**6.2 Sprint Delivery Schedule**

| **Sprint** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Story Points** | **Priority** | **Team Members** |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-1 | Data Collection | USN-1 | Collecting the Dataset | 10 | High | G.Adithya C.K .Rakul S.Krishna Prasath R.Shevannth |
| Sprint-1 |  | USN-2 | Data Pre-processing | 7 | Medium | G.Adithya C.K .Rakul S.Krishna Prasath R.Shevannth |
| Sprint-2 | Model Building | USN-3 | Import the required libraries, add the necessary layers and compile the model | 10 | High | G.Adithya C.K .Rakul S.Krishna Prasath R.Shevannth |
| Sprint-2 |  | USN-4 | Training the data classification model using RNN and other systems. | 7 | Medium | G.Adithya C.K .Rakul S.Krishna Prasath R.Shevannth |
| Sprint-3 | Training and Testing | USN-5 | Training the model and testing the model’s performance | 10 | High | G.Adithya C.K .Rakul S.Krishna Prasath R.Shevannth |
| Sprint-4 |  | USN-6 | Build the system and deploy the model in IBM cloud | 7 | Medium | G.Adithya C.K .Rakul S.Krishna Prasath R.Shevannth |

**6.3 Reports from JIRA**

It is a product for software developers, project managers and other software development teams.

****

****

**CODING & SOLUTIONING**

**7.1 Feature 1**

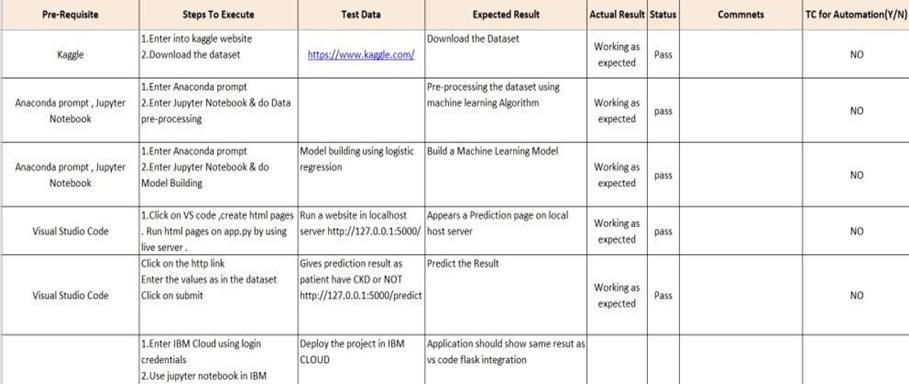
A LSTM Price forecasting machine learning model (Add in GITHUB)

**7.2 Feature 2**

A User Interface for forecasting based on the past 10 days price (Add in Github)

**TESTING**

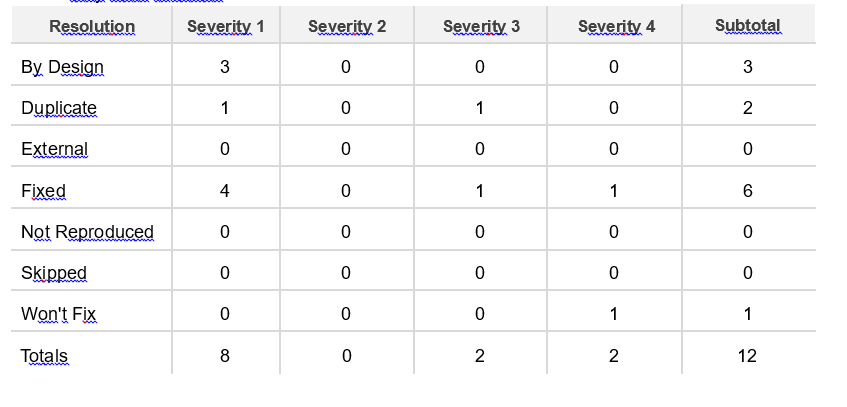
**8.1 Test Cases**

****

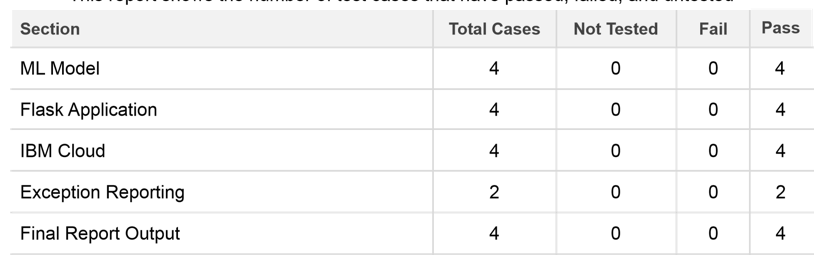
**8.2 User Acceptance Testing**

**1. Defect Analysis**

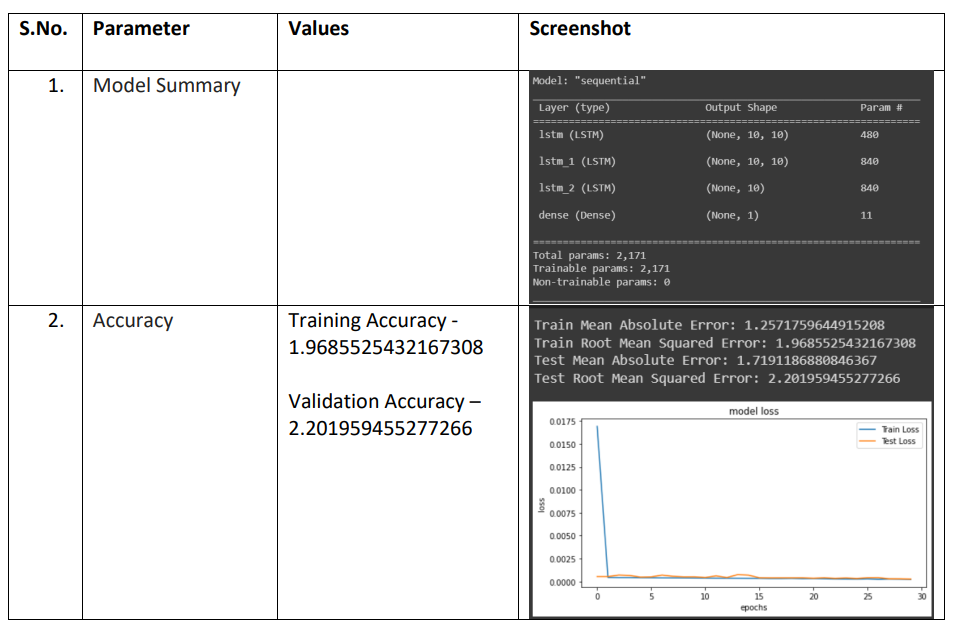
**This report shows the number of resolved or closed bugs at each severity level, and how**

****

**2. Test Case Analysis**

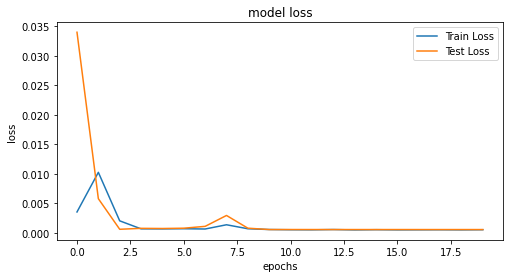
****

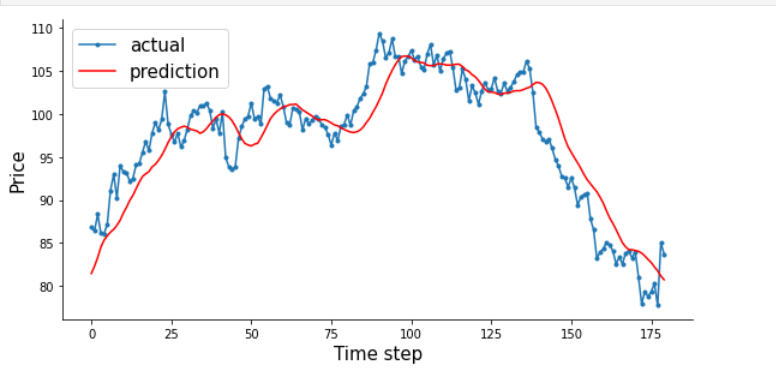
**8.3 Performance Testing**

****

**RESULTS**

**9.1 Performance Metrics**

****

****

**ADVANTAGES**

* continuously captures the unstable pattern of the crude oil price
* We’ll gain valuable insight
* forecasting can assist in minimising the risks associated with volatility in oil prices.

**DISADVANTAGES**

* The system developed is never 100% accurate.
* It can be time-consuming and resource-intensive

**CONCLUSION**

Therefore the DL model was deployed as a web app and the user interface is handy for stakeholders who do not have much knowledge in programming. The predicted value is displayed in the user interface.

**FUTURE SCOPE**

To have a better grasp of the pricing value, transform the machine learning model from univariate to multivariate. This can significantly improve prediction accuracy.

**APPENDIX**

**Source Code**

**Data Collection**

**import** numpy **as** np

**import** pandas **as** pd

**import** datetime

**from** pylab **import** rcParams

**import** matplotlib.pyplot **as** plt

**import** warnings

**import** itertools

**from** sklearn.preprocessing **import** MinMaxScaler

**import** statsmodels.api **as** sm

df**=**pd**.**read\_csv("Crude Oil Prices Daily.csv")

df**.**head()

df['Date'] **=** pd**.**to\_datetime(df['Date'])

**Data Preprocessing**

df = df.sort\_values('Date')

df = df.groupby('Date')['Price'].sum().reset\_index()

df.set\_index('Date', inplace=True)

def DfInfo(df\_initial):

tab\_info = pd.DataFrame(df\_initial.dtypes).T.rename(index={0: 'column type'})

tab\_info=tab\_info.append(pd.DataFrame(df\_initial.isnull().sum()).T.rename(index={0: 'null values (nb)'}))

tab\_info=tab\_info.append(pd.DataFrame(df\_initial.isnull().sum()/ df\_initial.shape[0] \* 100).T. rename(index={0: 'null values (%)'}))

return tab\_info

DfInfo(df)

df.index

# **Data Visualization**

df1 = df

df1 = df1.sort\_values('Date')

df1 = df1.groupby('Date')['Price'].sum().reset\_index()

df1.set\_index('Date', inplace=True)

df1=df1.loc[datetime.date(year=1988,month=1,day=1):]

q = df1['Price'].resample('MS').mean()

q.plot(figsize=(15, 6))

plt.show()

rcParams['figure.figsize'] = 18, 8

decomposition = sm.tsa.seasonal\_decompose(q, model='additive')

fig = decomposition.plot()

plt.show()

plt\_1 = plt.figure(figsize=(15, 6))

time = pd.to\_datetime(df['Date'])

data = list(df['Price'])

copdata = pd.Series(data, time)

plt.plot(copdata)

# **Scaling**

scaler = MinMaxScaler(feature\_range = (0, 1))

df = scaler.fit\_transform(df)

import joblib

joblib.dump(scaler, 'scaler.save')

# **Split Data Into Train/Test**

train\_size = int(len(df) \* 0.70)

test\_size = len(df) - train\_size

train, test = df[0:train\_size, :], df[train\_size:len(df), :]

# **Sliding Window**

def create\_data\_set(\_data\_set, \_look\_back=1):

data\_x, data\_y = [], []

for i in range(len(\_data\_set) - \_look\_back - 1):

a = \_data\_set[i:(i + \_look\_back), 0]

data\_x.append(a)

data\_y.append(\_data\_set[i + \_look\_back, 0])

return np.array(data\_x), np.array(data\_y)

look\_back=10

x\_train,y\_train,x\_test,y\_test = [],[],[],[]

x\_train,y\_train=create\_data\_set(train,look\_back)

x\_train = np.reshape(x\_train, (x\_train.shape[0], x\_train.shape[1], 1))

x\_test,y\_test=create\_data\_set(test,look\_back)

x\_test = np.reshape(x\_test, (x\_test.shape[0], x\_test.shape[1], 1))

print(x\_test.shape[1])

print(x\_train.shape), print(y\_train.shape)

print(x\_test.shape), print(y\_test.shape)

len(test)

# **Importing The Model Building Libraries**

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import LSTM

from keras.layers import Dropout

from sklearn.metrics import mean\_squared\_error

from keras.callbacks import ReduceLROnPlateau, EarlyStopping, ModelCheckpoint

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import mean\_absolute\_error

import seaborn as sns

# **Add Layers**

#Initializing The Model

regressor = Sequential()

#Adding LSTM Layers

regressor.add(LSTM(units = 60, return\_sequences = True, input\_shape = (x\_train.shape[1], 1)))

regressor.add(Dropout(0.1))

regressor.add(LSTM(units = 60, return\_sequences = True))

regressor.add(Dropout(0.1))

regressor.add(LSTM(units = 60))

regressor.add(Dropout(0.1))

#Adding Output Layers

regressor.add(Dense(units = 1))

# **Configure The Learning Process**

regressor.compile(optimizer = 'adam', loss = 'mean\_squared\_error')

reduce\_lr = ReduceLROnPlateau(monitor='val\_loss',patience=5)

# **Train The Model**

history =regressor.fit(x\_train, y\_train, epochs = 20, batch\_size = 64,validation\_data=(x\_test, y\_test), callbacks=[reduce\_lr],shuffle=False)

# **Save Model**

egressor.save('model1.h5')

**Model Evaluation**

train\_predict = regressor.predict(x\_train)

test\_predict = regressor.predict(x\_test)

train\_predict = scaler.inverse\_transform(train\_predict)

y\_train = scaler.inverse\_transform([y\_train])

test\_predict = scaler.inverse\_transform(test\_predict)

y\_test = scaler.inverse\_transform([y\_test])

print('Train Mean Absolute Error:', mean\_absolute\_error(y\_train[0], train\_predict[:,0]))

print('Train Root Mean Squared Error:',np.sqrt(mean\_squared\_error(y\_train[0], train\_predict[:,0])))

print('Test Mean Absolute Error:', mean\_absolute\_error(y\_test[0], test\_predict[:,0]))

print('Test Root Mean Squared Error:',np.sqrt(mean\_squared\_error(y\_test[0], test\_predict[:,0])))

plt.figure(figsize=(8,4))

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val\_loss'], label='Test Loss')

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epochs')

plt.legend(loc='upper right')

plt.show();

# **Test The Model**

plt.figure(figsize=(8,4))

plt.plot(aa, y\_test[0][:180], marker='.', label="actual")

plt.plot(aa, test\_predict[:,0][:180], 'r', label="prediction")

plt.tight\_layout()

sns.despine(top=True)

plt.subplots\_adjust(left=0.07)

plt.ylabel('Price', size=15)

plt.xlabel('Time step', size=15)

plt.legend(fontsize=15)

plt.show();

from keras.models import load\_model

model=load\_model('model1.h5')

model.summary()

#Enter Last 10 Days Price:

string = '2,0,13,86,60,76,30,11,55,66'

string = string.split(',')

x\_input = [eval(i) for i in string]

sc = joblib.load("scaler.save")

x\_input = sc.fit\_transform(np.array(x\_input).reshape(-1,1))

x\_input = x\_input.reshape((1,10,1))

print(x\_input.shape)

X\_input

print('PRICE: ',res[0][0]\*100)

#test the model

look\_back = 10

trainPredictPlot = np.empty\_like(df)

trainPredictPlot[:,:] = np.nan

trainPredictPlot[look\_back:len(train\_predict)+look\_back,:] = train\_predict

testPredictPlot = np.empty\_like(df)

testPredictPlot[:,:] = np.nan

testPredictPlot[len(train\_predict)+(look\_back\*2)+1:len(df)-1,:] = test\_predict

plt.plot(scaler.inverse\_transform(df))

plt.plot(trainPredictPlot)

plt.plot(testPredictPlot)

plt.show()

***GitHub & Project Demo Link***

[***https://github.com/IBM-EPBL/IBM-Project-13872-1659534448***](https://github.com/IBM-EPBL/IBM-Project-13872-1659534448)

***https://youtu.be/OxzeMBq-AEE***