1. INTRODUCTION

Crude oil is a naturally occurring liquid petroleum product composed of

hydrocarbon deposits and other organic materials formed from the remains of animals and plants that lived millions of years ago. These organisms were covered by layers of sand, silt, and rock, subject to heat and pressure, and eventually turned into a type of fossil fuel that is refined into usable products including gasoline, diesel, liquefied petroleum gases, and feedstock for the petrochemical industry. Crude oil is a mixture of comparatively volatile liquid hydrocarbons (compounds composed mainly of hydrogen and carbon), though it also contains some nitrogen, sulfur, and oxygen. Those elements form a large variety of complex molecular structures, some of which cannot be readily identified. Regardless of variations, however, almost all crude oil ranges from 82 to 87 percent carbon by weight and 12 to 15 percent hydrogen by weight.

Crude oils are customarily characterized by the type of hydrocarbon compound that is most prevalent in them: paraffins, naphthenes, and aromatics. Paraffins are the most common hydrocarbons in crude oil; certain liquid paraffins are the major constituents of gasoline (petrol) and are therefore highly valued. Naphthenes are an important part of all liquid refinery products, but they also form some of the heavy asphaltlike residues of refinery processes. Aromatics generally constitute only a small percentage of most crudes. The most common aromatic in crude oil is benzene, a popular building block in the petrochemical industry.

2. LITERATURE SURVEY:

Volatility in the Oil and Energy Markets

Fig. 1: Crude Oil Prices 1900-2014. [8] (Courtesy of British Petroleum)

Oil and energy markets, essential as they are to a functioning economy, also prove to be extremely volatile. In fact, prices for crude oil, refined petroleum, and natural gas are more

volatile than prices of 95% of other commodities. [1] This has largely been the case since the 1970s, when the 1973 oil crisis sent a deep shock to the price of oil, as can be observed in Fig. 1. It is important to note that volatility encompasses what may be seen as positive developments, such as decreases in price, and negative developments, such as price increases caused by shortages. While volatility may occasionally have positive aspects, uncertainty surrounding price fluctuations means it is something that is most businesses and governments aim to avoid. Since 1986, global no

Government Solutions

Government controls on oil and energy prices typically come in the form of price freezes or ceilings for increasing prices rather than price floors, given that price decreases are typically politically popular to consumers. In 2007 and 2008, before and during the Global Financial Crisis, prices and volatility increased substantially, causing many governments to put in place freezes as crude prices first surpassed $100 per barrel and approached $140 per barrel. [3]

Another government tactic for controlling price increases and volatility are price subsidies and tax reductions. These tactics have been substantial in recent years, with many countries spending billions per year on subsidies or tax reductions. Indonesia, for example, committed $6.7 billion to energy subsidies in May 2018. [4] Governments may also target specific industries with subsidies and tax reductions to reduce costs. These industries often include agriculture, public and goods transport, and fisheries, to avoid consumers receiving substantial price shocks to essential purchases.

Other countries employ price stabilization funds, which function by setting domestic prices higher than international prices when prices are low and using the proceeds to lower prices when prices rise more sharply than expected. In practice, the fact that oil prices are consistently rising limits the effectiveness of these funds and their ability to smooth prices. Moreover, these controls can be politically unpopular by forcing consumers to pay more than the international prices, even when prices are low.

A final strategy that some countries employ is the build up of strategic oil reserves. The United States, for example, had 665.1 million barrels of oil in its Strategic Petroleum Reserve in February 2018. [5] These reserves make it possible for these countries to supply the market in the case of sharp price spikes that typically result from physical disruptions to supply, such as hurricanes and other natural disasters. When used correctly, strategic reserves can be incredibly valuable, but critics have chided politicians for attempting to use the reserves for political purposes, as some speculate Bill Clinton did in 2000 to combat rising oil prices. [3]

While some of these controls may be politically popular by keeping prices lower and more stable, they may also create challenges, including supply shortages and underinvestment in the energy sector, in addition to oil companies suffering losses when they sell their refined products at lower prices than the crude they purchased. Ultimately, it may be the case that a shift away from oil to other energy sources is the most certain way to avoid the high level of volatility.

Business Solutions

While governments have powerful pricing tools at their disposals when it comes to controlling prices for short periods of time, individual businesses have more limited options, which most often involve price smoothing. Most in need of these tools are companies with large fuel expenses, such as airlines, shipping lines, and transportation companies. The need for such products was made clear in 2008, when Northwest CEO, Doug Steenland, testified that U.S. airline carriers were on track to spend "$61.2 billion on jet fuel [that] year, $20 billion more than in 2007, and [were] projected to incur losses totaling close to $10 billion". [3] Companies most often turn to hedging oil prices through forwards, futures, and options, which allow them to lock in either a certain cost or a range of costs in order to avoid massive fluctuations. [6] While there is the possibility of losing out on savings if fuel prices drops, financial markets far prefer to avoid the uncertainty and can gain greatly by hedging. In fact, unusually aggressive hedging practices by Southwest Airlines allowed the company to save more $455 million in 2004, $892 million in 2005, $675 million in 2006 and $439 million in the first nine months of 2007, during a period of increasing prices. [7] Providers of futures contracts often include fuel management companies, oil companies, financial institutions, and utilities providers with teams specializing in the industry.

2.2 References

[1] E. Regnier, "Oil and Energy Price Volatility," Energy Econ. **29**, 405 (2007).

[2] J. E. Rentschler, "Oil Price Volatility, Economic Growth and the Hedging Role of Renewable Energy," World Bank, Policy Research Working Paper 6603, September 2013.

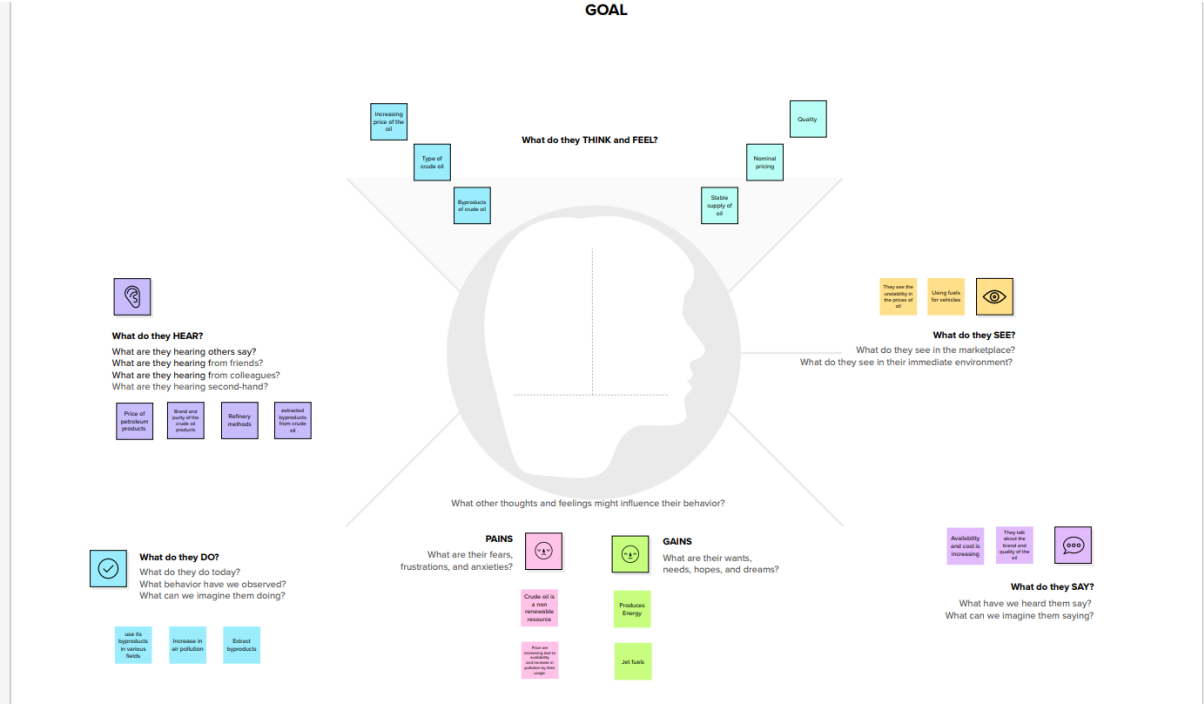
[3] R. McNally, *Crude Volatility: the History and the Future of Boom-Bust Oil Prices* (Columbia University Press, 2017).

2.3 Problem Statement Definition:

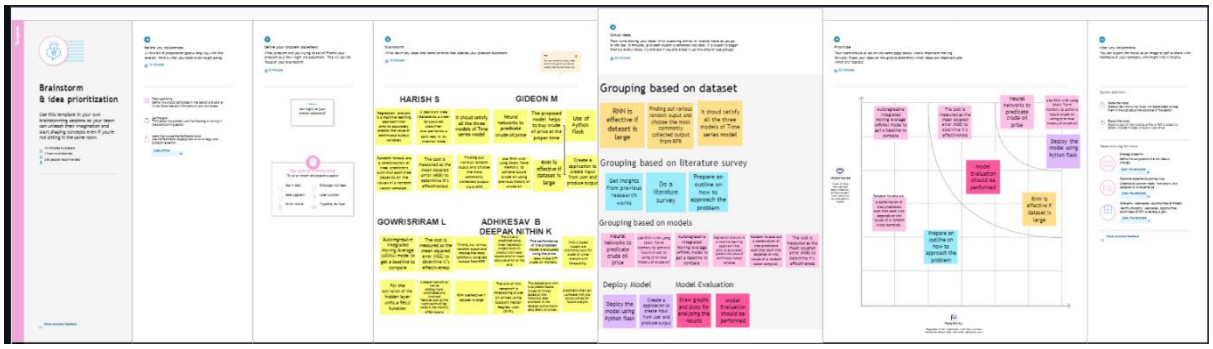
Oil is one of the world’s most important commodities, and it is a commodity that not only fuels the globe, but it is also one that has a massive, and very important market. The oil market is one that can be very attractive to investors because, despite the push towards cleaner fuel, the reliance on this commodity is still very high.

3. IDEATION & PROPOSED SOLUTION:

3.1 Empathy Map Canvas:



3.2 Ideation & Brainstorming:



3.3 Proposed Solution:

Oil and energy markets, essential as they are to a functioning economy, also prove to be extremely volatile. In fact, prices for crude oil, refined petroleum, and natural gas are more volatile than prices of 95% of other commodities. [1] This has largely been the case since the 1970s, when the 1973 oil crisis sent a deep shock to the price of oil, as can be observed in Fig. 1. It is important to note that volatility encompasses what may be seen as positive developments, such as decreases in price, and negative developments, such as price increases caused by shortages. While volatility may occasionally have positive aspects, uncertainty surrounding price fluctuations means it is something that is most businesses and governments

aim to avoid. Since 1986, global nominal oil prices have risen from $8.7 to $145.7, with a standard deviation of $25.7. [2] To avoid the instability, there are diverse measures in place that various institutions employ.

3.4 Problem Solution fit :

There is no systematic approach to receive a rapid answer from an oil predictor. A week of waiting is required. The proposed solution should enable consumers to contact with the oil predictor and receive payments both online and offline.

4. REQUIREMENT ANALYSIS:

4.1 Functional requirement:

Time series modelling techniques

Several methods are proposed in the literature to build time series models. They include autoregressive integrated moving average (ARIMA), generalised auto regressive conditional heteroscedastic (GARCH), Holt-Winters, autoregressive neural networks, and support vector regression.2 Various hybrid models are also suggested such ascombination of ARIMA and neural networks with support vector regression, genetic algorithms and wavelets.3-7 Discussion of various methodologies applied for crude oil price modelling can be found in review articles available in the literature.8,7 We have used ARIMA and autoregressive neural networks for modelling oil prices, as these techniques cover both linear and non-linear types of modelling. A short description of these methods is given below.

ARIMA

ARIMA is the most widely used and well known technique for time series analysis, developed by Box and Jenkins. In an ARIMA model, future values are predicted as a linear combination of previous oil prices and the associated errors. This model consists of three parts: the AR (autoregressive) component is a linear combination of past observations; MA (moving average) is a linear combination of lagged error terms; and I (integrated) replaces the original series with differenced series. An ARIMA model is represented in the form of Equation 3:

$$\Delta Dyt = c+\varphi 1\Delta Dyt\text{-}1 + \ldots +\varphi p\Delta Dyt\text{-}p + \varepsilon t + \theta 1\varepsilon t\text{-}1 + \ldots + \theta q\ \varepsilon t\text{-}q \qquad (3)$$

Where, yt is value of variable at next time step t, φ and θ are AR and MA coefficients respectively, ε is lagged error terms and ΔDyt represents Dth differenced time series. To build an ARIMA model, we need to specify three parameters: p, d and q; p represents the number of lag observations in an AR model; d is the degree of differencing; and q is the order of the moving average model. Before building a time series model, we need to make time series stationary. A time series is said to be stationary if mean, variance and covariance are constant in time. There is a formal statistical test known as the Augmented Dickey-Fuller (ADF) test for testing the stationarity of time series. Most of the time series data is usually not stationary, however it can be made stationary by data transformation techniques such as differencing and logarithm. Parameters p and q in

the ARIMA model are determined from autocorrelation plots of time series. There are a set of rules which can guide in the interpretation of correlation plots and identify possible ARIMA models for further testing. For the model diagnosis, correlation plots of residuals and the Ljung box test are used. We have used a built-in ARIMA model in R to model the crude oil prices.

Autoregressive neural network

An autoregressive neural network (ANN) is a non-linear model in which future prices are expressed as a non-linear function of lagged prices in the series, in contrast to linear modelling in ARIMA. Additionally, neural network based models have the ability to learn and capture patterns in data sets without the need to specify the exact model form. Multilayer perceptron (MLP) is the most widely used ANN in forecasting problems. Typically, the model is composed of input layer, hidden layer and output layer. The connecting nodes in these layers are called neurons. Input to the neurons is mapped using transfer functions and the weighted average of output from all the nodes is sent to next layer. There are various parameters that need to be specified for an ANN model: number of hidden layers, number of neurons in each layer, type of transfer function, and number of lags. The selection of appropriate network parameters is crucial to the fitting and forecast accuracy of an ANN model. We have used the nnetar function in R to build a neural network model.

4.2 Non-Functional requirements:

Oil futures prices: Central banks and the International Monetary Fund (IMF) mainly use oil futures contract prices as their gauge. Traders in crude oil futures set prices by two factors: supply and demand and market sentiment. However, futures prices can be a poor predictor, because they tend to add too much variance to the current price of oil.

2. **Regression-based structural models**: Statistical computer programming calculates the probabilities of certain behaviors on the price of oil. For instance, mathematicians may consider forces such as events in OPEC member nations, inventory levels, production costs, or consumption levels. Regression-based models have strong predictive power, but their creators may fail to include one or more factors, or unexpected variables may step in to cause these regression-based models to fail.

Time-series analysis: Some economists use time-series models, such as exponential smoothing models and autoregressive models, which include the categories of ARIMA and the ARCH/GARCH, to correct for the limitations of oil futures prices. These models analyze the history of oil at various points in time to extract meaningful statistics and predict future values based on previously observed values. Time-series analysis sometimes errs, but usually produces more accurate results when economists apply it to shorter time spans.

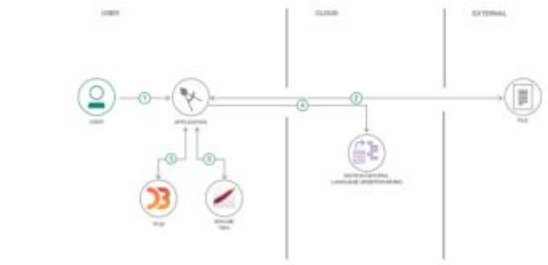
5. PROJECT DESIGN :

5.1 Data Flow Diagrams:

Data Flow Diagrams:

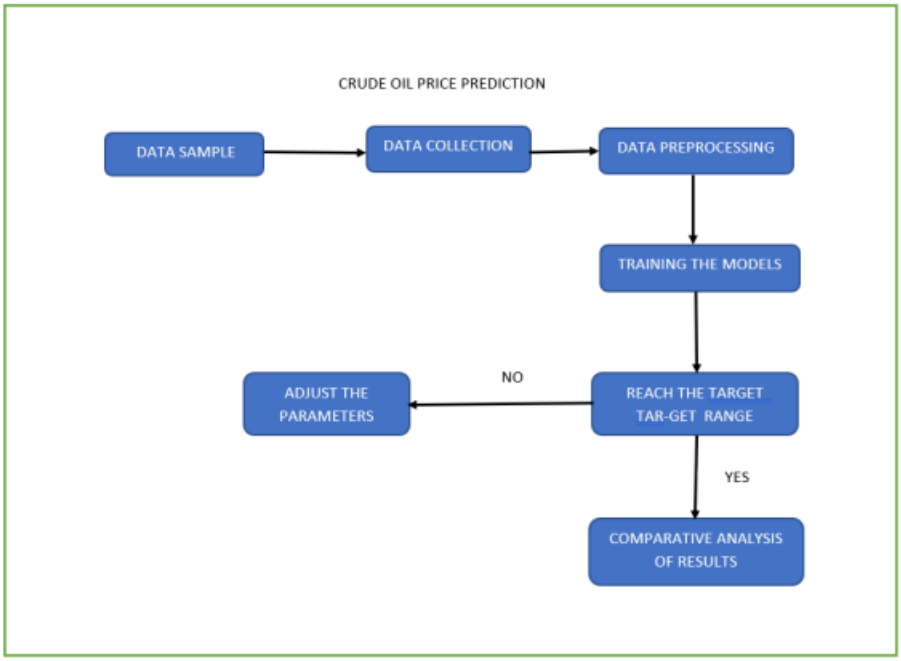
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

Flow



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.

5.2 Solution & Technical Architecture:



6. PROJECT PLANNING & SCHEDULING:

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

6.2 Sprint Delivery Schedule

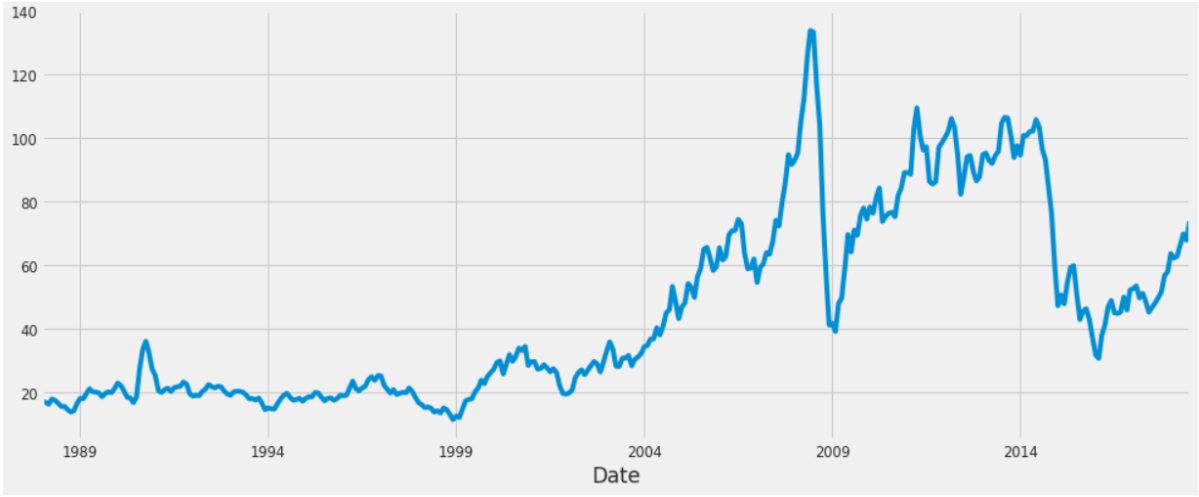
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022		
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022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

7. CODING & SOLUTIONING:

7.1 Feature 1
7.2 Feature 2

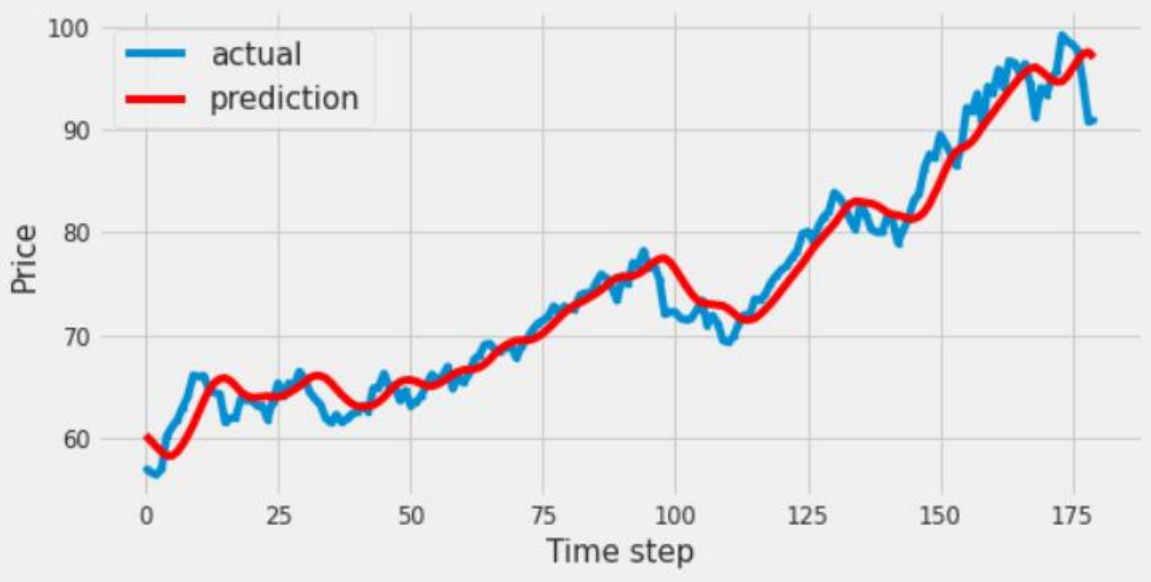
8. TESTING:

8.1 Test Cases:



9. RESULTS

9.1 Performance Metrics



10. ADVANTAGES & DISADVANTAGES:

e linear and nonparametric models to predict one month, three months, six months, one year, eighteen months, and two years ahead crude oil price in out-of-sample background. Mainly, our forecast depends on three predictor variables, the change in crude oil inventories, its previous prices, and product spread. By employing mean-squared prediction error (MSPE) and stochastic dominance (SD) tests, we find that the prediction result of our nonparametric models is significantly better than the random walk model, while the corresponding linear models' performance is better than the random walk model only for longer horizon forecasts (one to two years). In general, for the evaluation period from January 1995 to March 2021, the conclusion is that our model applying nonparametric estimation always outperforms all other models in different horizon forecasting. And for the nonparametric model including all three predictors, we document MSPE reduction as high as 62.0% and the directional accuracy ratio as high as 78.1% compared to the random walk model at the two years horizon.

11. CONCLUSION:

Forecasting crude oil prices is a very challenging problem due to the high volatility of oil prices. In this paper, we developed a new oil price prediction approach using ideas and tools from stream learning, a machine learning paradigm for analysis and inference of continuous flow of non-stationary data. Our stream learning model will be updated whenever new oil price data are available, so the model continuously evolves over time, and can capture the changing pattern of oil prices. In addition, updating the model requires only a small constant time per new data example, as opposed to re-training the model using the entire training data set. The experiment results show that our stream learning model outperformed three other popular oil price prediction models over a variety of forecast time horizons.

12. FUTURE SCOPE

rude oil is the mixture of petroleum liquids and gases that is extracted from the ground by oil wells. It is an important source of fuel and is used in the production of several products. Given the important role price of the crude oil plays, it becomes extremely important for managers to predict future oil price while making operational decisions such as: when to purchase material, how much to produce and what modes of transportation to use. The goal of this paper is to develop a forecasting model to predict the oil prices that aid management to reduce operational costs, increase profit and enhance competitive advantage. We first analyze the primary theories related to the forecast of oil price followed by the reviews of two main streams of forecast theory, which are Target Capacity Utilization Rule (TCU) and Exhaustible Resources Theory. We implement a Target Capacity Utilization Rule recursive simulation model and test it on the historical data from 1987 through 2017 to predict crude oil prices for 1991 through 2017. We tried several variations of the base model and the best method produced MAD, MSE, MAPE and MPE of 12.676, 280.92, 0.2597, 0.028, respectively. We further estimated the forecasts of the oil prices at a monthly level based on our yearly forecast of oil prices from our best method. The calculated MAD, MSE, MAPE and MPE values are 5.66, 82.1163, 0.1246 and 0.038, respectively, which shows our model is promising again at a monthly level.

13. APPENDIX:

Source Code:

```
import numpy as np
import pandas as pd
import datetime
import matplotlib.pyplot as plt
import warnings
import itertools
import statsmodels.api as sm
import seaborn as sns
import math
from pylab import rcParams
from sklearn.preprocessing import MinMaxScaler

#Jupyter Notebook Specific
sns.set_context("paper", font_scale=1.3)
sns.set_style('white')
warnings.filterwarnings("ignore")
plt.style.use('fivethirtyeight')
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='jxEP1bzVWo0XmoQGhCkfcFTMUvezzzWxvUmiPIDP9Y5F',
    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 = 'crudeoilpriceprediction-donotdelete-pr-ekltddoz8wv161'
object_key = 'Crude Oil Prices Daily.xlsx'

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

df_data_0 = pd.read_excel(body.read())
df_data_0['Date']=pd.to_datetime(df_data_0['Date'])
df_data_0.isnull().any()
df_data_0.dropna(axis=0,inplace=True)
data_oil = df_data_0.reset_index()['Closing Value']
len(data_oil)
plt_1 = plt.figure(figsize=(15, 6))
time = pd.to_datetime(df_data_0['Date'])
data = list(df_data_0['Closing Value'])
copdata = pd.Series(data, time)
plt.plot(copdata)
#observe for any seasonal patterns
ax = df_data_0.plot(x='Date', y='Closing Value', figsize=(12,6))
xcoords = ['1988-01-01','1989-01-01','1990-01-01','1991-01-01','1992-01-
01','1993-01-01','1994-01-01',
           '1995-01-01','1996-01-01','1997-01-01','1998-01-01','1999-01-
01','2000-01-01','2001-01-01',
           '2002-01-01','2003-01-01','2004-01-01','2005-01-01','2006-01-
01','2007-01-01','2008-01-01',
           '2009-01-01','2010-01-01','2011-01-01','2012-01-01','2013-01-
01','2014-01-01','2015-01-01',
           '2016-01-01','2017-01-01', '2018-01-01', '2019-01-01', '2020-01-
01','2021-01-01']
for xc in xcoords:
    plt.axvline(x=xc, color='black', linestyle='--')
#Decompose the plot
df1 = df_data_0
df1 = df1.sort_values('Date')
df1 = df1.groupby('Date')['Closing Value'].sum().reset_index()
df1.set_index('Date', inplace=True)
df1=df1.loc[datetime.date(year=1988,month=1,day=1):]

q = df1['Closing Value'].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()
sc = MinMaxScaler(feature_range = (0, 1))
data_oil = sc.fit_transform(np.array(data_oil).reshape(-1,1))
len(data_oil)
train_size = int(len(data_oil) * 0.65)
test_size = len(data_oil) - train_size
train, test = data_oil[0:train_size, :], data_oil[train_size:len(data_oil),
:]
len(test)
def create_dataset(_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)


time_step = 10
X_train , Y_train = create_dataset(train,time_step)
X_test , Y_test = create_dataset(test,time_step)

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)
X_train.shape
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
regressor = Sequential()
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))

regressor.add(Dense(units = 1))

regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')
# early_stopping = EarlyStopping(monitor='val_loss',patience=20)
history =regressor.fit(X_train, Y_train, epochs = 20, batch_size =
64,validation_data=(X_test, Y_test),verbose=1)
# history =regressor.fit(X_train, Y_train, epochs = 100, batch_size =
64,validation_data=(X_test, Y_test)
train_predict = regressor.predict(X_train)
test_predict = regressor.predict(X_test)
```

```
# invert predictions
train_predict = sc.inverse_transform(train_predict)
Y_train = sc.inverse_transform([Y_train])
test_predict = sc.inverse_transform(test_predict)
Y_test = sc.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();
aa=[x for x in range(180)]
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();
#test the model
look_back = 10
trainPredictPlot = np.empty_like(data_oil)
trainPredictPlot[:,:] = np.nan
trainPredictPlot[look_back:len(train_predict)+look_back,:] = train_predict

testPredictPlot = np.empty_like(data_oil)
testPredictPlot[:,:] = np.nan
testPredictPlot[len(train_predict)+(look_back*2)+1:len(data_oil)-1,:] =
test_predict

plt.plot(sc.inverse_transform(data_oil))
plt.plot(trainPredictPlot)
plt.plot(testPredictPlot)
plt.show()
len(test)
x_input = test[2866:].reshape(1,-1)
x_input.shape
temp_input = list(x_input[0])
temp_list = temp_input[0].tolist()
temp_input
lst_output = []
n_steps = 10
i = 0
while(i<10):
```

```
  if(len(temp_input) > 10):
    x_input = np.array(temp_input[1:])
    print("{} day input {}".format(i,x_input),end="\n")
    x_input = x_input.reshape(1,-1)
    x_input = x_input.reshape((1,n_steps,1))

    yhat = regressor.predict(x_input,verbose=0)
    print("{} day output {}".format(i,yhat),end="\n")
    temp_input.extend(yhat[0].tolist())
    temp_input = temp_input[1:]
    print("-----------------------",end="\n")
    lst_output.extend(yhat.tolist())
    i = i+1
  else:
    x_input = x_input.reshape((1,n_steps,1))
    yhat = regressor.predict(x_input,verbose=0)
    print("{} day output {}".format(i,yhat),end="\n")
    temp_input.extend(yhat[0].tolist())
    lst_output.extend(yhat.tolist())
    i = i+1
day_new = np.arange(1,11)
day_pred = np.arange(11,21)
len(data_oil)

print(day_new)
print(day_pred)
plt.plot(day_new,sc.inverse_transform(data_oil[8206:]))
plt.plot(day_pred,sc.inverse_transform(lst_output))
```

Out[39]:

```
[]
```

GITHUB link

https://github.com/IBM-EPBL/IBM-Project-15304-1659596911

DEMO LINK:

https://youtu.be/IZvD4tttYsA