

CRUDE OIL PREDICTION

SUBMITTED BY,

ABISHEK

SAJI

SAM

ABISHEK SHAJI

SL.NO	CONTENTS	PAGE NO
1.	INTRODUCTION	1
1.1	Project Overview	1
1.2	Purpose	3
2.	LITERATURE SURVEY	4
2.1	Existing problem	4
2.2	Problem Statement Definition	4
3.	IDEATION & PROPOSED SOLUTION	7
3.1	Empathy Map Canvas	7
3.2	Ideation & Brainstorming	8
3.3	Proposed Solution	10
3.4	Problem Solution fit	11
4.	REQUIREMENT ANALYSIS	12
4.1	Functional requirement	12
4.2	Non-Functional requirements	
5.	PROJECT DESIGN	14
5.1	Data Flow Diagrams	14
5.2	Solution & Technical Architecture	18
6.	PROJECT PLANNING & SCHEDULING	20
6.1	Sprint Planning & Estimation	20
6.2	Sprint Delivery Schedule	21
6.3	Reports from JIRA	
7.	CODING	23
8.	TESTING	36
8.1	Test Cases	36
8.2	User Acceptance Testing	38
9.	RESULTS	39
9.1	Performance Metrics	39
10.	ADVANTAGES & DISADVANTAGES	40
11.	CONCLUSION	41
12.	FUTURE SCOPE	42
13.	APPENDIX	43

CHAPTER 1

INTRODUCTION

Crude oil is a natural liquid fossil fuel found in geological formations beneath the earth's surface. It has mostly been extracted by oil drilling, which comes after the studies of structural geology, sedimentary basin analysis, and reservoir characterization (Guerriero et al., 2012). Crude oil is one of the most important energy resources on earth. So far, it remains the world's leading fuel, with nearly one-third of global energy consumption.

Crude oil prices are determined by many factors and have a big impact on the global environment and economy. Although crude oil prices were firm in early 2014, they fell sharply from mid 2014. In January 2016, the U.S. refiner acquisition cost for crude oil imports, as a proxy for world oil price, is only \$28.81 per barrel on average, and the West Texas Intermediate (WTI) crude oil spot price, as the benchmark oil price in North America, is only \$31.68 per barrel on average (EIA, 2016). The prices have dropped by more than seventy percent since June 2014.

The world's environment is affected by the oil price falling. With the drop of oil prices, the fuel bills are lowered. As a result, consumers are very likely to use more oil and thus increase the carbon emission. In addition, there is less incentive to develop renewable and clean energy resources. On the other hand, sustained low oil prices could lead to a drop in global oil and gas exploration and exploitation activities.

Fluctuating oil prices also play an important role in the global economy (Husain et al., 2015). The fall in oil prices would result in a modest boost to global economic activity, although the owners of oil sectors suffer income losses. Recent research from the World Bank shows that for every 30% decline of oil prices, the global GDP (Gross Domestic Product) would be increased by 0.5%. At the same time, the drop of oil prices would reduce the cost of living, and hence the inflation rate would fall.

There is no doubt that crude oil price forecasts are very useful to industries, governments as well as individuals. Thus, forecasting crude oil prices has been the

subject of research by both academia and industry. Many methods and approaches have been developed for predicting oil prices. However, due to the high volatility of oil prices (Regnier, 2007), it remains one of the most challenging forecasting problems.

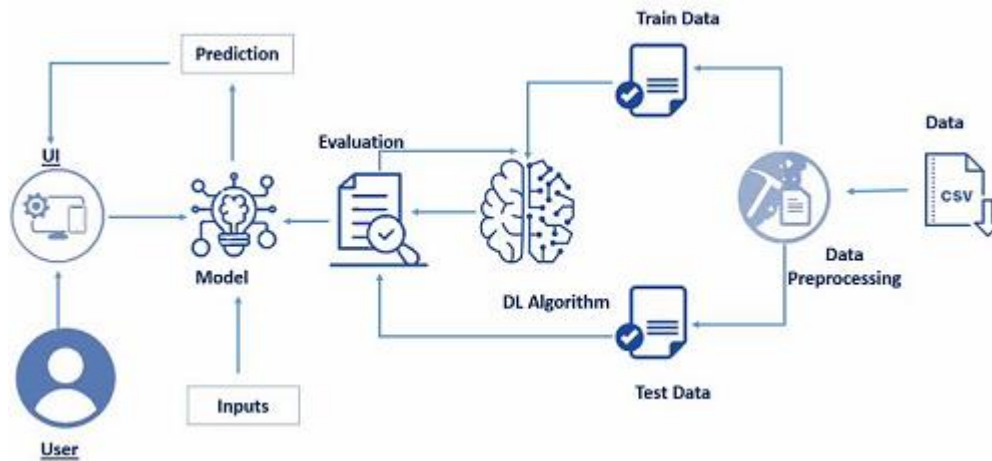
In recent years, machine learning techniques have been used in many applications in geosciences (Alavi et al., 2016). Machine learning provides powerful computational tools and algorithms that can learn from and make predictions on data. In this paper, we propose a novel approach for crude oil price prediction based on a new machine learning paradigm called stream learning (Gama et al., 2009; Bifet et al., 2010). 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. We compare our streaming learning model with three other popular oil price prediction models for predicting two types of oil prices (the U.S. refiner acquisition cost for crude oil imports and the WTI crude oil spot price). 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.

1.1 **Project Overview**

Oil demand is inelastic, therefore the rise in price is good news for producers because they will see an increase in their revenue. Oil importers, however, will experience increased costs of purchasing oil. Because oil is the largest traded commodity, the effects are quite significant. A rising oil price can even shift economic/political power from oil importers to oil exporters. The crude oil price movements are subject to diverse influencing factors.

This Guided Project mainly focuses on applying Neural Networks to predict the Crude Oil Price. This decision helps us to buy crude oil at the proper time. Time series analysis is the best option for this kind of prediction because we are using the Previous history of crude oil prices to predict future crude oil. So we would be implementing RNN

(Recurrent Neural Network) with LSTM(Long Short Term Memory) to achieve the task.



1.2 Purpose

By the end of this project you will be able to:

1. Know fundamental concepts and techniques of time series forecasting and LSTM
2. Gain a broad understanding of time series data.
3. Know how to split the data for time series forecasting.
4. Know how to build a web application using the Flask framework.

CHAPTER 2

LITERATURE SURVEY.

Introduction As the most important strategic resource around the globe, crude oil is the “key” commodity for the world’s economy. Therefore, forecasting crude oil price has always been considered as a very challenging task which drew the interest of researchers, practitioners and institutions. The price of oil is essentially determined by its supply and demand (Hagen, 1994; Stevens, 1995), it is also strongly related to irregular and unforeseen events caused by weather, wars, embargoes and revolutions (Aloui et al. 2012). Moreover, the important role of financial activity, and especially that of the speculation, in oil price formation has recently become such a strongly debated issue (Kilian and Murphy, 2011; Fattouh, 2012 ; Kilian et al., 2013 ; Beidas-Strom and Pescatori, 2014). Many other factors like Gross Domestic Production growth, stock levels inventories, foreign exchange rates, world population, political aspects and investors’ expectations can significantly affect the price of oil (Bernabe et al., 2004; Ghouri, 2006; Nelson et al., 1994; Yousefi and Wirjanto, 2004). Furthermore, the time to ship crude oil from one country to another can affect directly their price because oil prices vary in different regions of the worldwide (Wang et al., 2005a). All these factors can explain the nonlinear evolution and chaotic behavior of crude oil prices and therefore the high volatility of crude oil market (Plourde and Watkins, 1994; Yang et al., 2002). The oil price fluctuations have a direct effect on the nation’s economy (Hamilton, 1996, 2010; Blanchard and Gali, 2007;

Kilian, 2008); therefore, it is of vital importance to predict oil price. The present paper is organized as follows. Section 2 outlines the numerous studies which used traditional and statistical econometric models to forecast crude oil price. Then, a detailed description of artificial neural network model was introduced. In addition, we present the existing literature on crude oil price forecasting using this model. Finally, we conclude in section 3.

2. Crude Oil Price Prediction: A Literature Review

2.1 Application of Traditional and Statistical Econometric Models

Among many and different forecasting models that have been developed to predict the "black gold" price, the traditional statistical and econometric methods are the first ones to be applied by academic researchers. The first research about forecasting oil market is proposed by Amano (1987). The author used a small-scale econometric model for oil market prediction. Huntington (1994) utilized a sophisticated econometric model for predicting oil price in the 1980s. In another work, Gulen (1998) applied cointegration analysis to predict the WTI crude oil price. Barone-adesi et al. (1998) suggested a semi-parametric approach based on the filtered historical simulation technique to forecast oil price. Based on the GARCH properties of the oil price volatility, Morana (2001) employed a semi-parametric approach investigated by Barone-adesi et al. (1998) to short-term forecast of Brent crude oil price. In another work, Tang and Hammoudeh (2002) utilized a nonlinear regression to predict OPEC basket price. Using OECD petroleum inventory levels and relative stock inventories, Ye et al. (2002, 2005) adopted a simple linear regression model for short-term monthly prediction of WTI

crude oil spot price. In a related study, Ye et al. (2006) included nonlinear variables such as low- and high- inventory variables to the linear forecasting model suggested by Ye et al. (2002, 2005) to predict short-run WTI crude oil prices. Using OECD stocks, non-OECD demand and OPEC supply, Zamani (2004) applied an econometrics forecasting methodology to short term quarterly WTI crude oil spot price. Lanza et al. (2005) investigated crude oil and product prices by utilizing the error correction models. Sadorsky (2006) applied multiple univariate and multivariate statistical models such as GARCH, TGARCH, AR, and BIGARCH to daily forecast of volatility in petroleum futures price returns. Slightly more recent, Dees et al. (2007) developed a linear model of the world oil market to predict oil demand, supply, and prices focusing mainly on OPEC behavior. Murat and Tokat (2009) investigated the relationship between futures and spot crude oil prices and therefore tested the ability of futures prices to forecast spot price movements using random walk model. Cheong (2009) adopted ARCH models to forecast crude oil markets. On the other hand, more recent studies have applied GARCH as well as different models of the GARCH family to predict oil price. For example, Narayan and Narayan (2007) and Agnolucci (2009) used GARCH model to forecast spot and futures crude oil prices. In a related research, Mohammadi and Su (2010) compared the forecasting results of various GARCH-types models in order to predict the crude oil price. Kang et al. (2009) proposed CGARCH, FIGARCH and IGARCH models to forecast volatility

of crude oil markets. For the same purpose, Wei et al. (2010) extended the study of Kang et al. (2009) by applying linear and nonlinear GARCH-class models.

2.2 Existing problem

Artificial Neural Network also the connectionist systems are computing systems that are based on and are theoretically alike, but not exactly identical to, biological neural networks of a human body. An ANN performs its task by taking in examples and requires no programming with task-specific rules [7]. The purpose of a neural network is to construct or design an output pattern when given an input pattern. An artificial neural network (ANN) has an architecture which is parallelly-distributed with large number of nodes (neurons) and connections.

We use the Back-propagation learning algorithm and the error signal is cultivated through the network in the backward direction by changing and managing weights of the network to maximize the performance of the network[8]. The procedure is done until the network is able to provide desired responses

In the suggested model, there is only one dependent variable, the closing price of crude oil which has been considered, since it's a time series, we have followed the model for general time series forecasting in conducting the experiments.

2.3 Problem Statement Definition

To complete this project you should have the following software and packages.

Navigator :

Anaconda Navigator is a free and open-source distribution of the Python and R

programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS. Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupiter notebook and spyder.

To download Anaconda refer to this [link](#)

To build Deep learning models you must require the following packages

Tensor flow: TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers can easily build and deploy ML-powered applications.

Keras: Keras leverages various optimization techniques to make high-level neural network API easier and more performant. It supports the following features:

- Consistent, simple, and extensible API.
- Minimal structure - easy to achieve the result without any frills.
- It supports multiple platforms and backends.
- It is a user-friendly framework that runs on both CPU and GPU.
- Highly scalability of computation.

Flask: Web framework used for building Web applications

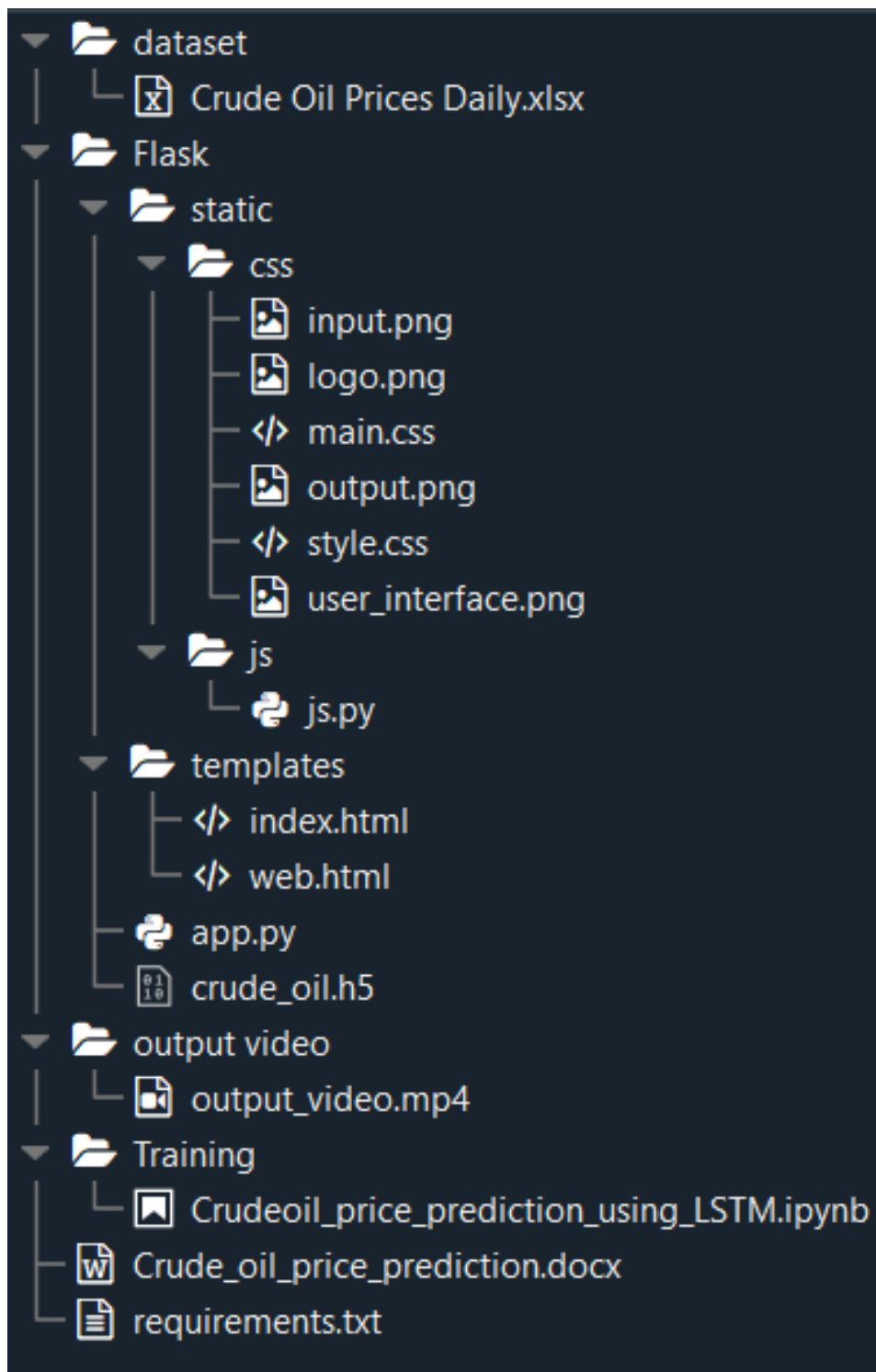
If you are using anaconda navigator, follow the below steps to download the required packages:

- open anaconda prompt as administrator
- Type “pip install tensorflow” (make sure you are working on python 64 bit)
- Type “pip install flask”.

- Type "pip install keras

The above steps allow you to install Keras and TensorFlow in the anaconda environment.

a Project folder which contains files as shown below



- We are building a Flask Application which needs HTML pages stored in the templates folder and a python script app.py for server-side scripting

- `app.py` - contains the actual python code that will import the app and start the development server.
- `Crudeoil_price_prediction_using_LSTM.ipynb` - This is where you define models for your application.
- `Crude_oil.h5` - This is our model weights file that will be generated when the model is trained.
- `static` - contains static files i.e. CSS, Javascript, images.
- `templates` - This is where you store your HTML templates i.e. `index.html`, `lweb.html`
- `requirements.txt` - This is where you store your package dependencies.

CHAPTER 3

IDEATION & PROPOSED SOLUTION

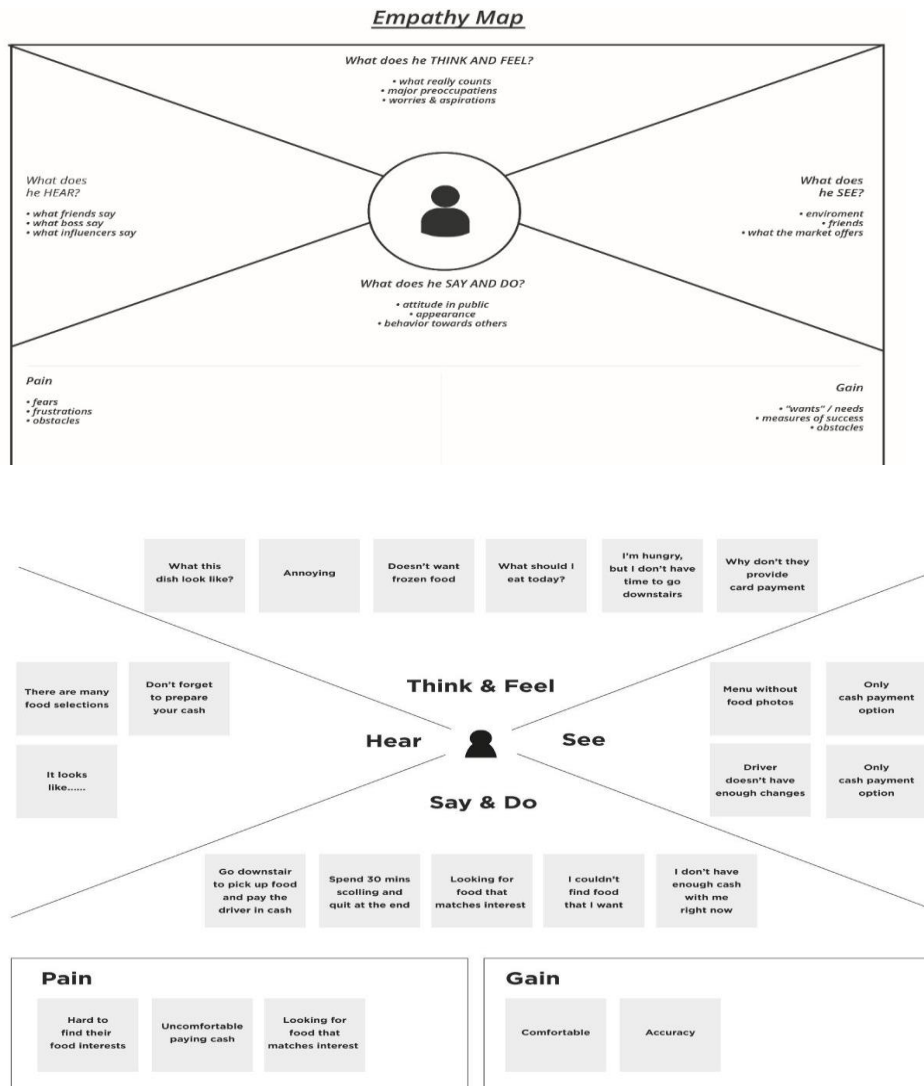
3.1 Empathy Map Canvas

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

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

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

Example:




3.2 Ideation & Brainstorming

Brainstorm & Idea Prioritization Template:

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

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

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



Brainstorm & idea prioritization

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

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

➔

Before you collaborate

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

🕒 10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) ➔

1


Define your problem statement

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

🕒 5 minutes

PROBLEM

How might we [your problem statement]?



Key rules of brainstorming

To run an smooth and productive session

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

Step-2: Brainstorm, Idea Listing and Grouping

2

Brainstorm

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

🕒 10 minutes

TIP
You can select a sticky note and hit the pencil (switch to editing) icon to start editing!

Amar	Yuktash	Person 3	Person 4
<div>idea</div>			

Person 5	Person 6	Person 7	Person 8

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

🕒 20 minutes

Person 4

TIP
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

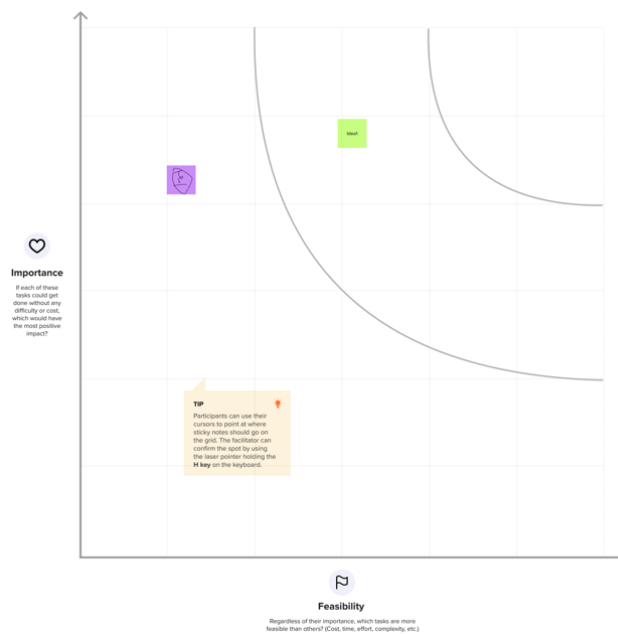
Step-3: Idea Prioritization

4

Prioritize

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

🕒 20 minutes



2.3 Proposed Solution

Project team shall fill the following information in proposed solution template.

3.3 Problem Solution fit

Solution Fit Template

1. CUSTOMER SEGMENT(S) Who is your customer? i.e. working parents of 0-5 y.o. kids	6. CUSTOMER CONSTRAINTS What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.	5. AVAILABLE SOLUTIONS Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking
2. JOBS-TO-BE-DONE / PROBLEMS Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.	9. PROBLEM ROOT CAUSE What is the real reason that this problem exists? What is the back story behind the need to do this job? i.e. customers have to do it because of the change in regulations.	7. BEHAVIOUR What does your customer do to address the problem and get the job done? i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 Functional requirements

Extension plugin should provide a warning pop-up when they visit a website that is phished; therefore it should strictly follow the following:

- a. Extension plugin ability to present the pop-up to the users screen should be quick enough to the point, users will be aware before entering any confidential or sensitive details into a phishing website.
- b. Extension plugin should not need the facilities and services from an 3rd party service or APIs, due the reason that those services will always the potential to leak users browsing data and pattern when it gets compromised by hackers
- c. Extension plugin will have the capability to also detect latest and new phishing websites

4.2Non-Functional requirements

Graphical User Interface design Interface developed should be done with the understanding that it must meet the simplicity of what users would like to see when they need an extension for detecting things, and also it needs to adhere to non IT literate users as well. It must also provide the exact information on what the user wants like identifying a phishing website quickly without needing to click on many options. The process of identifying phishing website should be taken directly from the web-page user wants to view through their URL and the result from it should be easily understood by the users.

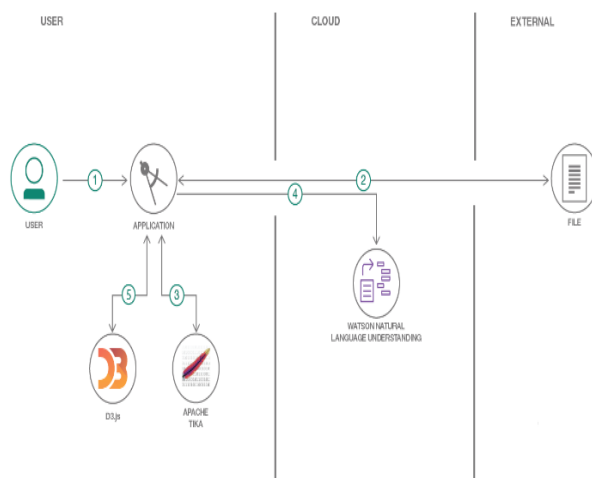
CHAPTER 5

PROJECT DESIGN

5.1 Data Flow Diagrams

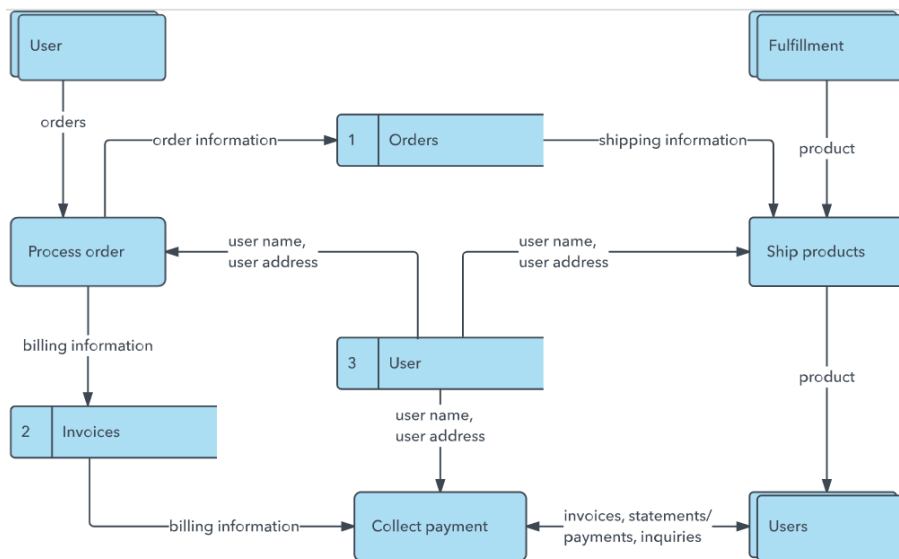
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

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.

Example: DFD Level 0 (Industry Standard)



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

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 business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.

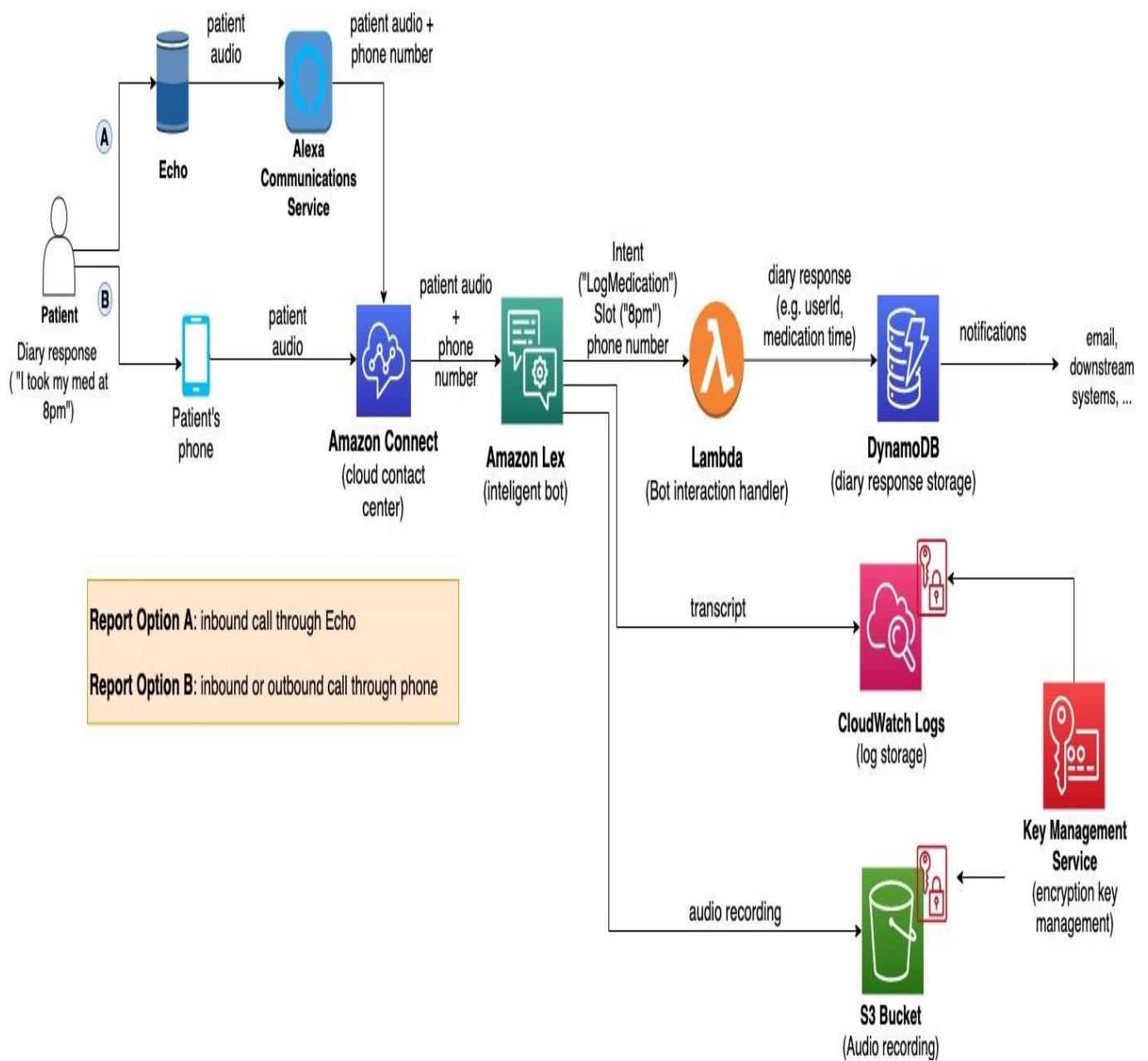


Figure 1: Architecture and data flow of the voice patient diary sample application

CHAPTER 6

PROJECT PLANNING & SCHEDULING

The goal of DevOps is to cut down the project timelines, increase the productivity, and manage rapid development-deployment cycles without impacting business and quality. It requires efficient sprint management. The objective of this paper is to develop different sprint level project management tools for quick project level Go/No-Go decision making (using real-time projects data and machine learning), sprint estimation technique (gamified-consensus based), statistical understanding of overall project management maturity, project sentiment & perception. An attempt is made to devise a model to calibrate the perception or the tone of a project culture using sentiment analysis.

6.1 Sprint Planning & Estimation

In recent approaches in software engineering to develop software there is Agile Software Development methodology (ASD) which is the most powerful framework nowadays. Agile Project Management offers a method can characterize by its agility[1,2]. It also enables organizations to increase the benefits of an agile methodology without introducing unnecessary risks [3]. within agile there are many methodologies as SCRUM or Extreme Programming. SCRUM is a lightweight process framework for agile development, and the most widely-used one.

- A “process framework” is a particular set of practices that must be followed in order for a process to be consistent with the framework. (For example, the Scrum process framework requires the use of development cycles called Sprints, the XP framework requires pair programming, and so forth.)
- “Lightweight” means that the overhead of the process is kept as small as possible, to maximize the amount of productive time available for getting useful work done. Scrum divides the development process of software to

Sprints, and Sprint is repeatable work cycle in Scrum

- Dates don't account for the non-project related work that inevitably creeps into our days: emails, meetings, and interviews that a team member may be involved in.
- Dates have an emotional attachment to them. Relative estimation removes the emotional attachment.
- Each team will estimate work on a slightly different scale, which means their velocity (measured in points) will naturally be different. This, in turn, makes it impossible to play politics using velocity as a weapon.
- Once you agree on the relative effort of each story point value, you can assign points quickly without much debate.

6.2 Sprint Delivery Schedule

For a long-lived big product such as a ship-control system or printer, the Product Backlog contains thousands of items. During initial Product Backlog , the Product Owner is responsible for prioritizing many of these items in some way to try to improve ROI 'Return On Investment' or to improve delivery of 'value.' That suggests a measure of 'value' for each item. A deeper discussion of value is deferred until the following topics. Yet, with relative value points (RVPs) as a lightweight proxy for 'value,' use planning poker to experiment with relative value points (RVPs) and their estimation[8]. Successful Planning Poker Shared responsibilities are important on a software project in order to accomplish a task; however the requirements should be declared by the product owner and be located in product backlog

6.3 Reports from JIRA

The process of portfolio management and reporting would not be complete without automatically emailing Jira reports to managers.

"The automation counterpart of Better PDF Exporter, Better PDF Automation for Jira is used for automatic Jira report emailing.", Frantisek explains.

As part of the Jira project reporting workflow, some reports have to be sent to different managers, board members and other stakeholders automatically.

Better PDF Automation is a *free* app that helps Roland's team distribute all the reports. This way all reports are prepared and automatically emailed on time. Basically, preparation for most of the meetings are done automatically.

Frantisek and his team deciphered the original requirement and delivered a solution that made the customer happy. With Better PDF Exporter for Jira, Frantisek showed the customer that Jira reports can be flexible, portable and useful for managers.

CHAPTER 7

CODING

```
#!/usr/bin/env python
# coding: utf-8

# <a href="https://colab.research.google.com/github/Rudresh99/Crude-Oil-Prices-Brent---Europe/blob/master/Lstm_Crude_Oil_Price_Prediction.ipynb" target="_parent"></a>

# In[1]:

from google.colab import drive
drive.mount('/content/drive')

# In[2]:

from io import StringIO
import requests
import json
import pandas as pd

# @hidden_cell
# This function accesses a file in your Object Storage. The definition contains your credentials.
# You might want to remove those credentials before you share your notebook.
def
get_object_storage_file_with_credentials_d3bd5b94a9334de59a55a7fed2bedea(container, filename):
    """This functions returns a StringIO object containing the file content from Bluemix Object Storage."""

    url1 = ''.join(['https://identity.open.softlayer.com',
'/v3/auth/tokens'])
    data = {'auth': {'identity': {'methods': ['password'],
'password': {'user': {'name':
'member_adcb54bd899a7e39e31582bccad1577f68f1992f','domain': {'id':
'4619da2fa8524beda11c89d2d1969c5b'}},
'password': 'P*/m8,!#7s6H9poz'}}}}}
    headers1 = {'Content-Type': 'application/json'}
    resp1 = requests.post(url=url1, data=json.dumps(data), headers=headers1)
    resp1_body = resp1.json()
    for e1 in resp1_body['token']['catalog']:
```

```

        if(e1['type']=='object-store'):
            for e2 in e1['endpoints']:
                if(e2['interface']=='public'and
e2['region']=='dallas'):
                    url2 = ''.join([e2['url'],'/', container, '/',
filename])
                    s_subject_token = resp1.headers['x-subject-token']
                    headers2 = {'X-Auth-Token': s_subject_token, 'accept':
'application/json'}
                    resp2 = requests.get(url=url2, headers=headers2)
                    return StringIO(resp2.text)

df_data_1 =
pd.read_csv(get_object_storage_file_with_credentials_d3bd5b94a9334de59a55a7f
ed2bedeaa('CrudeAi', 'DCOILBRETEU.csv'))
df_data_1.head()

# Data Source
# -----
# U.S. Energy Information Administration, Crude Oil Prices: Brent - Europe
[DCOILBRETEU], retrieved from FRED, Federal Reserve Bank of St. Louis;
https://fred.stlouisfed.org/series/DCOILBRETEU, January 10, 2018.
#
# ![alt text](https://user-images.githubusercontent.com/52448964/85650354-
06f7c300-b6c3-11ea-947b-51bd32c88f8f.png)

# In[3]:

df_data_1 = df_data_1[df_data_1.DCOILBRETEU != "."]
print(df_data_1.shape)

# In[4]:

import matplotlib.pyplot as plt
df_data_1_plot = df_data_1.iloc[:,1:2].values.astype(float)
# Visualising the Data
plt.plot(df_data_1_plot, color = 'red', label = 'Crude Oil Prices')
plt.title('Crude Oil Prices Historical Data')
plt.xlabel('Time (Days)')
plt.ylabel('Crude Oil Prices')
plt.legend()
plt.show()

```

```

# Stateful vs. Stateless LSTM
# -----
#
# 1. Stateless: LSTM updates parameters on batch 1 and then
initiates cell states (meaning - memory, usually with zeros) for batch 2
# 2. Stateful: it uses batch 1 last output cell states as initial states
for batch 2.
#
# https://3qeqr26caki16dnhd19sv6by6v-wpengine.netdna-ssl.com/wp-content/uploads/2017/03/Box-and-Whisker-Plot-of-Test-RMSE-of-Stateful-vs-Stateless-LSTM-Results.png
#
# When to use which?
# -----
#
# - When sequences in batches are related to each other (e.g. prices of one
commodity), we should better use stateful mode
# - Else, when one sequence represents a complete sentence, we should go
with stateless mode
#
#
#
# Batch-size: which batch-size to choose?
# -----
#
# Very important decision!
#
# Imagine, you must learn to recognize a bird...
# You are presented images of different birds.
#
# What would you prefer:
# 1. To see the one image at a time, make your notes about special bird
quilities (set your weights) and then see another bird and so on
# 2. OR may be you would better learn if you see - let's say 5 - bird
images at ones. May be then you can faster notice the bird's intrinsic
properties?
#
# I'd say - the second method is more efficient for humans. We need more
examples of an entity, that we have to distinguish.
#
# So the machines! Therefore we select a batch size of 64.
# Later in programming assignment we will see how the batch size impacts the
prediction accuracy.
# In[ ]:

```

```

#import packages
import numpy as np
import pandas as pd
from keras.preprocessing import sequence
from keras.models import load_model

# In[ ]:

# defining the batch size and number of epochs
batch_size = 64
epochs = 120
timesteps = 30

# Batch-size and trainings-set size
# -----
#
# With **stateful LSTMs** the trainings-set size must be divisible without
remainder by the batch-size (modulo = 0)

# In[ ]:

length = len(df_data_1)
print(length)
length *= 1 - 0.1
print(length)

# In[5]:

7004.7%64.0

# In[6]:

6976.0%64.0

# In[7]:

```

```
def get_train_length(dataset, batch_size, test_percent):
    # subtract test_percent to be excluded from training, reserved for
    testset
    length = len(dataset)
    length *= 1 - test_percent
    train_length_values = []
    for x in range(int(length) - 100, int(length)):
        modulo = x % batch_size
        if (modulo == 0):
            train_length_values.append(x)
            print(x)
    return (max(train_length_values))
```

In[8]:

```
length = get_train_length(df_data_1, batch_size, 0.1)
print(length)
```

In[9]:

```
#Adding timesteps * 2
upper_train = length + timesteps*2
df_data_1_train = df_data_1[0:upper_train]
training_set = df_data_1_train.iloc[:,1:2].values
training_set.shape
```

In[10]:

```
# Feature Scaling
#scale between 0 and 1. the weights are esier to find.
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(np.float64(training_set))
training_set_scaled.shape
```

In[11]:

```
X_train = []
y_train = []
```

```

# Creating a data structure with n timesteps

print(length + timesteps)
for i in range(timesteps, length + timesteps):
    X_train.append(training_set_scaled[i-timesteps:i,0])
    y_train.append(training_set_scaled[i:i+timesteps,0])

print(len(X_train))
print(len(y_train))
#create X_train matrix
#30 items per array (timestep)
print(X_train[0:2])
print(np.array(X_train).shape)
#create Y_train matrix
#30 items per array (timestep)
print(y_train[0:2])
print(np.array(y_train).shape)

# In[12]:

# Reshaping
X_train, y_train = np.array(X_train), np.array(y_train)
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
y_train = np.reshape(y_train, (y_train.shape[0], y_train.shape[1], 1))
print(X_train.shape)
print(y_train.shape)

# In[13]:

# Building the LSTM
# Importing the Keras libraries and packages

from keras.layers import Dense
from keras.layers import Input, LSTM
from keras.models import Model
import h5py

# In[14]:

# Initialising the LSTM Model with MAE Loss-Function

```

```

# Using Functional API

inputs_1_mae = Input(batch_shape=(batch_size,timesteps,1))
#each layer is the input of the next layer
lstm_1_mae = LSTM(10, stateful=True, return_sequences=True)(inputs_1_mae)
lstm_2_mae = LSTM(10, stateful=True, return_sequences=True)(lstm_1_mae)

output_1_mae = Dense(units = 1)(lstm_2_mae)

regressor_mae = Model(inputs=inputs_1_mae, outputs = output_1_mae)

#adam is fast starting off and then gets slower and more precise
#mae -> mean absolute error Loss function
regressor_mae.compile(optimizer='adam', loss = 'mae')
regressor_mae.summary()

# In[ ]:

from keras import backend as K

K.set_session(K.tf.Session(config=K.tf.ConfigProto(intra_op_parallelism_threads=1, inter_op_parallelism_threads=1)))

# Citation from Redwood Center for Theoretical Neuroscience (Berkeley University)
# -----
# http://redwood.berkeley.edu
#
# http://redwood.berkeley.edu/vs265/Brian-Cheung-LSTMS.pdf
#
# Overall LSTM Structure
# -----
# ![alt
text](https://www.researchgate.net/profile/Oezal_Yildirim/publication/324056729/figure/fig3/AS:619912796073986@1524810136091/Basic-structure-of-the-BLSTM-network-The-LSTM-nets-at-the-bottom-indicate-the-forward.png)
#
#
#
# LSTM Node Anatomy
# -----
# ![alt text](https://devopedia.org/images/article/217/1055.1569518278.jpg)

# One of the best articles:

```

```

# -----
# http://colah.github.io/posts/2015-08-Understanding-LSTMs/

# How LSTM Param Number is computed?
# -----
#
# 1. To decide how to handle the memory each LSTM Cell has 3
Gates:
#     - input (what to let in),
#     - forget (what to forget) and
#     - output (what to write to the output)
# 2. LSTM Cell State is its memory
# 3. LSTM Hidden State is equivalent to the Cell output:
#     - lstm_hidden_state_size (number of neurons = memory cells) =
lstm_outputs_size
# 4. Parameters:
#     - weights for the inputs (lstm_inputs_size)
#     - weights for the outputs (lstm_outputs_size)
#     - bias variable
# 5. Result from previous point - for all 3 Gates and for Cell State ( =
4)
#
# \begin{equation}
# \textbf{PARAMETERS} = 4 \times \textbf{LSTM outputs}
size \times (\textbf{weights LSTM inputs size} + \textbf{weights LSTM}
outputs size) + 1 \textbf{ bias variable}
# \end{equation}
#
#
#

# In[ ]:

# 1st LSTM Layer
parameters = 4 * 10 * (1 + 10 + 1)
print(parameters)

# In[ ]:

parameters = 4 * 10 * (10 + 10 + 1)
print(parameters)

# In[ ]:

```



```

#Statefull
for i in range(epochs):
    print("Epoch: " + str(i))
    #run through all data but the cell, hidden state are used for the next
    batch.
    regressor_mae.fit(X_train, y_train, shuffle=False, epochs = 1,
batch_size = batch_size)
    #resets only the states but the weights, cell and hidden are kept.
    regressor_mae.reset_states()

#Stateless
#between the batches the cell and hidden states are lost.
#regressor_mae.fit(X_train, y_train, shuffle=False, epochs = epochs,
batch_size = batch_size)

# In[ ]:

#save model
import h5py
regressor_mae.save(filepath="my_model_with_mae_30_ts.h5")

# In[ ]:

#Load model
import h5py
regressor_mae = load_model(filepath="my_model_with_mae_30_ts.h5")

# In[ ]:

def get_test_length(dataset, batch_size):

    test_length_values = []
    for x in range(len(dataset) - 200, len(dataset) - timesteps*2):
        modulo=(x-upper_train)%batch_size
        if (modulo == 0):
            test_length_values.append(x)
            print(x)
    return (max(test_length_values))

```

```

# In[ ]:

test_length = get_test_length(df_data_1, batch_size)
print(test_length)
upper_test = test_length + timesteps*2
testset_length = test_length - upper_train
print(testset_length)

# In[ ]:

print(upper_train, upper_test, len(df_data_1))

# In[ ]:

# construct test set

#subsetting
df_data_1_test = df_data_1[upper_train:upper_test]
test_set = df_data_1_test.iloc[:,1:2].values

#scaling
scaled_real_bcg_values_test = sc.fit_transform(np.float64(test_set))

#creating input data
X_test = []
for i in range(timesteps, testset_length + timesteps):
    X_test.append(scaled_real_bcg_values_test[i-timesteps:i, 0])
X_test = np.array(X_test)

#reshaping
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

# In[ ]:

X_test.shape

# In[ ]:

```

```

#prediction
predicted_bcg_values_test_mae = regressor_mae.predict(X_test,
batch_size=batch_size)
regressor_mae.reset_states()

print(predicted_bcg_values_test_mae.shape)

#reshaping
predicted_bcg_values_test_mae = np.reshape(predicted_bcg_values_test_mae,
                                             (predicted_bcg_values_test_mae.shape[
0],
                                             predicted_bcg_values_test_mae.shape[
1]))

print(predicted_bcg_values_test_mae.shape)
#inverse transform
predicted_bcg_values_test_mae =
sc.inverse_transform(predicted_bcg_values_test_mae)

#creating y_test data
y_test = []
for j in range(0, testset_length - timesteps):
    y_test = np.append(y_test, predicted_bcg_values_test_mae[j, timesteps-
1])

# reshaping
y_test = np.reshape(y_test, (y_test.shape[0], 1))

print(y_test.shape)

# In[ ]:

# Visualising the results
plt.plot(test_set[timesteps:len(y_test)].astype(float), color = 'red', label
= 'Real Crude Oil Prices')
plt.plot(y_test[0:len(y_test) - timesteps].astype(float), color = 'blue',
label = 'Predicted Crude Oil Prices')
plt.title('Crude Oil Prices Prediction - MAE')
plt.xlabel('Time')
plt.ylabel('Crude Oil Prices')
plt.legend()
plt.show()

```

```
# In[ ]:
```

```
#MSE (mean squared error)
import math
from sklearn.metrics import mean_squared_error
rmse = math.sqrt(mean_squared_error(test_set[timesteps:len(y_test)],
y_test[0:len(y_test) - timesteps]))
print(rmse)
```

```
# In[ ]:
```

```
#MAE (mean absolute error)
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(test_set[timesteps:len(y_test)],
y_test[0:len(y_test) - timesteps])
print(mae)
```

```
# In[ ]:
```

```
# In[ ]:
```

```
# Initialising the LSTM Model with MSE Loss Function
```

```
inputs_1_mse = Input(batch_shape=(batch_size,timesteps,1))
lstm_1_mse = LSTM(10, stateful=True, return_sequences=True)(inputs_1_mse)
lstm_2_mse = LSTM(10, stateful=True, return_sequences=True)(lstm_1_mse)
```

```
output_1_mse = Dense(units = 1)(lstm_2_mse)
```

```
regressor_mse = Model(inputs=inputs_1_mse, outputs = output_1_mse)
```

```
#mse -> mean squared error as loss function
regressor_mse.compile(optimizer='adam', loss = 'mse')
regressor_mse.summary()
```

```
# In[ ]:
```

```

from keras import backend as K

K.set_session(K.tf.Session(config=K.tf.ConfigProto(intra_op_parallelism_threads=1, inter_op_parallelism_threads=1)))

# In[ ]:

epochs = 120
for i in range(epochs):
    print("Epoch: " + str(i))
    regressor_mse.fit(X_train, y_train, shuffle=False, epochs = 1,
batch_size = batch_size)
    regressor_mse.reset_states()

# In[ ]:

import h5py
regressor_mse.save(filepath="my_model_with_mse_30_ts.h5")

# In[ ]:

regressor_mse = load_model(filepath="my_model_with_mse_30_ts.h5")

# In[ ]:

predicted_bcg_values_test_mse = regressor_mse.predict(X_test,
batch_size=batch_size)
regressor_mse.reset_states()

predicted_bcg_values_test_mse = np.reshape(predicted_bcg_values_test_mse,
                                             (predicted_bcg_values_test_mse.shape[
0],
                                             predicted_bcg_values_test_mse.shape[
1]))
predicted_bcg_values_test_mse =
sc.inverse_transform(predicted_bcg_values_test_mse)

pred_mse = []

```

```

for j in range(0, testset_length - timesteps):
    pred_mse = np.append(pred_mse, predicted_bcg_values_test_mse[j,
timesteps-1])

pred_mse = np.reshape(pred_mse, (pred_mse.shape[0], 1))

# In[ ]:

# Visualising the results
plt.plot(test_set[timesteps:len(pred_mse)].astype(float), color = 'red',
label = 'Real Crude Oil Prices')
plt.plot(pred_mse[0:len(pred_mse) - timesteps], color = 'green', label =
'Predicted Crude Oil Prices with MSE')
plt.title('Crude Oil Prices Prediction - MSE')
plt.xlabel('Time')
plt.ylabel('Crude Oil Prices')
plt.legend()
plt.show()

# In[ ]:

from sklearn.metrics import mean_squared_error
rmse = math.sqrt(mean_squared_error(test_set[timesteps:len(pred_mse)],
pred_mse[0:len(pred_mse) - timesteps]))
print(rmse)

# In[ ]:

mean = np.mean(np.float64(test_set[timesteps:len(pred_mse)]))
print(mean)

# In[ ]:

rmse/mean * 100

# In[ ]:

from sklearn.metrics import mean_absolute_error

```

```
mae = mean_absolute_error(test_set[timesteps:len(pred_mse)],  
pred_mse[0:len(pred_mse) - timesteps])  
print(mae)
```

```
# In[ ]:
```

```
mae/mean * 100
```

CHAPTER 8

TESTING

The email could be classified as authentic. In the other hand, the Sender ID has constraints to be adopted due to a defective syntax and a licensed technology. Finally, the DomainKeys solution presents some vulnerabilities since it could be possible to forge a pair of keys to legalize an email message. In order to obtain successful attacks, usually phishers collect electronic addresses and spread messages to the greater possible number of users. Thousands (if not millions) of phishing emails are sent daily. Although some of them will not be received due to filters or invalid address, it is assumed that 0.01% of the users will read and believe in the message. It means that if phishers send one million messages, one hundred users will believe in the message, click on its content and send personal information. After the click the user is directed to a fake website. James [4] exemplifies one of technique used by phishers. The technique is based on making one mirror copy of a commercial bank website using CGI (common gateway interface) programming code to send collected data by an email account. Immediately afterward, the victim is guided to the real website site through an URI (uniform resource indicator) handled in a way that the victim does not realize the differences between the two websites. Although some existing filters prevent users from receiving phishing emails, anyone can attest by their mailboxes that they are faulty. In order to propose novel approaches to filters phishing messages we have testes the ability of several machine learning algorithms to detect phishing emails. This article is organized in 4 sections. Section 2 describes main approaches to detect phishing attacks in related works. Section 3 presents machine learning algorithms used in our project to detect phishing messages. Section 4 presents the results of this work and section 5 points out some conclusions. This article focuses on the approach of phishing detention where an email message configure the vector of initial attack [6]. In this context, most predominant solution is the analysis of the body of the email. The body analysis includes the general structure of the email,

links and semantics. For the analysis of the general structure of the email, it is verified if the email possesses HTML format, contains scripts and fulfilling forms [7, 8, 9]. Links sometimes can reveal a destination and disguise a fraud. The verification of the semantics enriches the detection since they carry a context in its essence. A few works include the analysis of the heading of the email as an additional parameter. In the heading of the email, generally, the examined field is the sender address. All these analysis although present some vulnerabilities. Email servers do not always possess tools for validating the sender, for instance. In some cases, forged emails are so close to real messages that they escape filtering methodologies being delivered to mailboxes. The regular use of emails on HTML facilitates the use of the techniques that are part of the modus operandi of phishers (phishers need to direct the users for a fake website in order to capture their personal data). In order to reach their goals, phishers try to drive users to fake links presented as official links. Such disguise technique is supported by HTML language. Through the resource of creation of links, phishers create an email HTML with URI indicating a legitimate website. However, in its structure of creation of link, the parameter href describes the destination to it. The use of scripts, especially Javascript, is emphasized as an additional verification of the body of an email also to be used as technological resource will occult the real destination of an URI shown in form of link [6,7,8]. The status bar of a email client or internet navigator always presents the real destination of one link when the user points the mouse on. If the email - visually - shows website URI legitimate, the validation of the suggested destination can be made by the bar of status.

8.2 User acceptance Testing Weka is a valuable tool to evaluate classifying and clustering techniques. It implements as well evaluating methods through its Experimenter tool. For this study we have run several algorithms considering all possible parameters settings. Table 1 presents the ordered ten best results from the application of decision trees (Id3 and J48), neural nets (Multilayer Perceptron), Bayesian nets (Bayes Net and

Naïve Bayes)The best global result was achieved by the multilayer perceptron algorithm. The neural net produced an accuracy of 96.5% of the data, producing only 7 false positive results (legitimate messages classified as phishing messages) and 11 false negative results (phishing messages classified as legitimate messages). Although the neural net produces a non inspectable model, it could certainly be applied to an anti-phishing filter. Concerning the decision tree algorithms, the J48 achieved the best result with an accuracy of 94.94%. Even though the ID3 algorithm produced a lower accuracy compared to J48, the number of false negatives has decreased. From the security point of view, it means that ID3 may be better at preventing that phishing messages arrive at users mailboxes. On the other side, we consider that false positive classification cause less damage to the user.

CHAPTER 9

RESULTS

The whole project is based on python, machine learning, and flask. 4.1Data Collection: We collected the data regarding the crude oil and its prices from Kaggle, Google and Github repositories. 4.2Data Preprocessing: The raw data cannot be used directly for training the model. Hence we perform preprocessing on the raw data. First we import the libraries which are frequently used in our project. Here we have used numpy which mainly focuses on operations on arrays, matplotlib for plots and pandas to work on data. After importing the libraries we import our data set. There may be a chance of missing data in the dataset. Missing data may deviate our results. In order to avoid this we use the SimpleImputer class to replace missing data with mean. Based on the data set and dependent variables we replace missing values with mean, median, constant number etc. If the data set is too heavy and there is only 1% of missing data we ignore the rows with missing values. Now we encode the categorical data using OneHotEncoder class. This method transforms the categorical variable into a set of binary variables (also known as dummy variables). It used N-1 features to show N labels. This improves the machine to understand the data. Next we split the data into a testing set and training set. We apply machine learning algorithms on the training set. In our project we used random forest regression, decision tree regression, simple linear regression. We give the train set as input for the models and train them. 4.3User Interface and output: A user interface improves the usage of the model. We use flask and HTML to build the user interface. From the above metrics results, we conclude that Random Forest Regression has less error. Hence we use Random Forest Regression in our project. The predicted value will be in Dollar/BBL units as well as in rupees.

CHAPTER 10

ADVANTAGES & DISADVANTAGES

No	Techniques Used	Advantages	Disadvantages
1	<i>Methods based on Bag-of-Words model</i>	-Build secure connection between user's mail transfer Agent (MTA) and mail user agent (MUA)	-Time consuming - huge number of features -consuming memory
2	<i>Compared multi Classifiers algorithms</i>	-Provide clear idea about the effective level of each classifier on phishing email detection	Non standard classifier
3	<i>hybrid system</i>	-High level of accuracy by take the advantages of many classifiers	-Time consuming because this technique has many layers to make the final result
4	<i>Classifiers Model-Based Features</i>	- High level of accuracy - create new type of features like Markov features	-huge number of features -many algorithm for classification which mean time consuming -higher cost -need large mail server and high memory requirement
5	<i>Clustering of Phishing Email</i>	-Fast in classification process	-Less accuracy because it depend on unsupervised learning , need feed continuously
6	Evolving Connectionist System (ECOS) for phishing email detection	fast ,less consuming memory, high accuracy, Evolving with time, online working	Need feed continuously

CHAPTER 11

CONCLUSION

Machine learning is one of the techniques of Artificial Intelligence which is used for extracting valuable knowledge from large databases[12]. Among all the models used, Random Forest Regression gave us the best results. The error is very less when compared to other regression models. The proposed system can accurately predict the prices of crude oil which helps us to buy crude oil in advance and decrease the expenses spent.

CHAPTER 12

REFERENCES

- 1) Kaufmann, R. K., & Ullman, B. (2009). Oil prices, speculation, and fundamentals: Interpreting causal relations among spot and futures prices. *Energy Economics*, 31(4), 550–558.
- 2) Shobhit Nigam. "Chapter 84 Single Multiplicative Neuron Model in Reinforcement Learning" , Springer Science and Business Media LLC, 2019
- 3) "Harmony Search and Nature Inspired Optimization Algorithms" , Springer Science and BusinessMedia LLC, 2019
- 4) Shuang Gao, Yalin Lei. "A new approach for crude oil price prediction based on stream learning" , Geoscience Frontiers, 2017
- 5) Ramakanta Mohanty. "Software Reliability Prediction Using Group Method of Data Handling" , Lecture Notes in Computer Science, 2009
- 6) Kulkarni, S., Haidar, I., 2009. Forecasting model for crude oil price using artificial neural networks and commodity future prices. *International Journal of Computer Science and Information Security* 2 (1).
- 7) Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*, 2nd edition, Prentice Hall, 842pages
- 8) Onur Dursun. "3 Methodology" , Walter de Gruyter GmbH, 2014
- 9) Lakshmanan, Indhurani, and Subburaj Ramasamy. "An Artificial Neural-Network Approach to Software Reliability Growth Modeling" , *Procedia Computer Science*, 2015.

- 10) Haykin, S. (2009). Neural Networks and Learning Machines, 3rd edition, Pearson, 938 pages
N. Raj Kiran, V. Ravi. "Software reliability prediction by soft computing techniques" , Journal of Systems and Software, 2008
- 11) Lean Yu. "An EMD-Based Neural Network Ensemble Learning Model for World Crude Oil Spot Price Forecasting" , Studies in Fuzziness and Soft Computing, 2008