Project Report

Team ID: PNT2022TMID24172

Team Leader: A S Lokesh Reddy

Team Members: Sukumar G

Sarvesh Kumar Y

S Venkata Krishna

Project Title: Efficient Water Quality Analysis and Prediction using Machine Learning

1. INTRODUCTION:

1.1 PROJECT OVERVIEW:

Water quality analysis is a complex topic due to the different factors that influence it. This concept is inextricably linked to the various purposes for which water is used. Different needs necessitate different standards. There is a lot of study being done on water quality prediction. Water quality is normally determined by a set of physical and chemical parameters that are closely related to the water's intended usage. The acceptable and unacceptable values for each variable must then be established. Water that meets the predetermined parameters for a specific application is considered appropriate for that application. If the water does not fulfill these requirements, it must be treated before it may be used. Water quality can be assessed using a variety of physical and chemical properties. As a result, studying the behavior of each individual variable independently is not possible in practice to accurately describe water quality on a spatial or temporal basis.

The ecosystem and human health are directly impacted by the water quality. Water is used for many different things, including drinking, farming, and industrial uses. A crucial indicator of effective water management is the water quality index (WQI). The aim of this work was to classify a dataset of water quality in various locations across India using machine learning techniques including RF, NN, MLR, SVM, and BTM. Features including dissolved oxygen (DO), total coliform (TC), biological oxygen demand (BOD), nitrate, pH, and electric conductivity determine the quality of water (EC). These characteristics are handled in five steps: feature correlation, applied machine learning classification, model's feature significance, and data pre-processing utilizing min-max normalization and missing data management using RF.

1.2 PURPOSE:

The purpose of predicting the water quality analysis using machine learning plays a major role in determining the water quality the people are using. The water is one of the most important factors in the person's life. The main motive of this application is that the people have to use the more quality water. The quality of the water can be determined from the factor like Biochemical oxygen demand, ph, carbon monoxide, dissolved oxygen, and Sodium content of the water. This can be implemented in the day-to-day life of the people.

2.LITERATURE SURVEY:

2.1 EXISTING SYSTEM:

[1] This study was designed to evaluate the quality of drinking water in selected areas of capital of Pakistan. Its adjacent city Rawalpindi. Drinking water samples collected from selected localities of Rawalpindi and Islamabad are analyzed for different water quality parameters such as pH, alkalinity, hardness, total dissolved solids, chloride, bicarbonates, sodium, potassium, calcium, magnesium, sulphates, phosphates, nitrates lead, copper, cadmium, cobalt, iron and zinc. Total viable count, coliforms, fecal coliforms and Escherichia coli were also part of the study.

Advantage:

• Needed in rapidly change in system.

Disadvantage:

• Drinking water quality is poorly managed and monitored.

• water pressure is low in Pakistan supply systems.

[2 This article describes design and application of feed-forward, fully-connected, three-layer perceptron neural network model for computing the water quality index (WQI)(1) for Kinta River (Malaysia). The modeling efforts showed that the optimal network architecture was 23-34-1 and that the best WQI predictions were associated with the quick propagation (QP) training algorithm; a learning rate of 0.06; and a QP coefficient of 1.75. The WQI predictions of this model had significant, positive, very high correlation ($r=0.977$, $p<0.01$) with the measured WQI values, implying that the model predictions explain around 95.4% of the variation in the measured WQI values.

Advantage:

• Saving in bulk water requirement.

Disadvantage:

• Possible long-term degradation of water quality for possible saline intrusion.

Tirabassi

[3] A study was initiated to predict water quality index (WQI) using artificial neural networks (ANNs) with respect to the concentrations of 16 groundwater quality variables collected from 47 wells and springs in Andimeshk during 2006–2013 by the Iran's Ministry of Energy. Such a prediction has the potential to reduce the computation time and effort and the possibility of error in the calculations. For this purpose, three ANN's algorithms including ANNs with early stopping, Ensemble of ANNs and ANNs with Bayesian regularization were utilized. The application of these algorithms for this purpose is the first study in its type in Iran.

Advantage:

• Non toxic humans ,non residue left behind.

Disadvantage:

• Strong oxidizer may causes hydro genic minerals nutrients to precipitate, reducing bioavailability.

[4] Naive Bayes is one of the most efficient and effective inductive learning algorithms for machine learning and data mining. Its competitive performance in classification is surprising, because the conditional independence assumption on which it is based, is rarely true in real world applications. An open question is: what is the true reason for the surprisingly good performance of naive Bayes in classification? In this paper, we propose a novel explanation on the superb classification performance of naive Bayes.

Advantage:

• Naive Bayes is suitable for solving multi-class prediction problems.

• If its assumption of the independence of features holds true, it can perform better than other models and requires much less training data.

Disadvantage:

• Naive Bayes assumes that all predictors (or features) are independent, rarely happening in real life. This limits the applicability of this algorithm in real-world use cases.

2.2 REFERENCES:

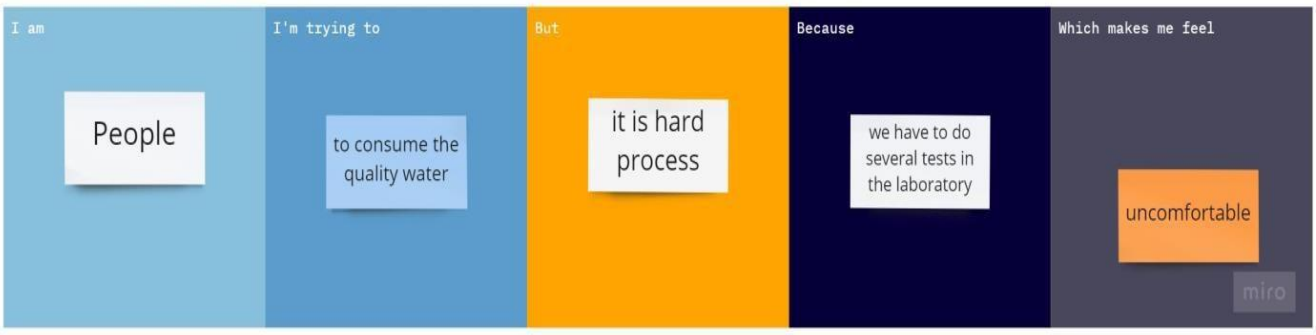
[1] Drinking Water Quality in Capital City of Pakistan Shahid Mehmood1, Asif Ahmad1, Anwaar Ahmed1, Nauman Khalid2 and Tariq Javed3.

[2] Gazzaz, N.M.; Yusoff, M.K.; Aris, A.Z.; Juahir, H.; Ramli, M.F. Artificial neural network modeling of the water quality index for Kinta River (Malaysia) using water quality variables as predictors. Mar. Pollut. Bull.2012, 64, 2409–2420.

[3] Sakizadeh, M. Artificial intelligence for the prediction of water quality index in groundwater systems. .Model. Earth Syst. Environ. 2016.

[4] Zhang, H. The optimality of naive Bayes. AA 2004.

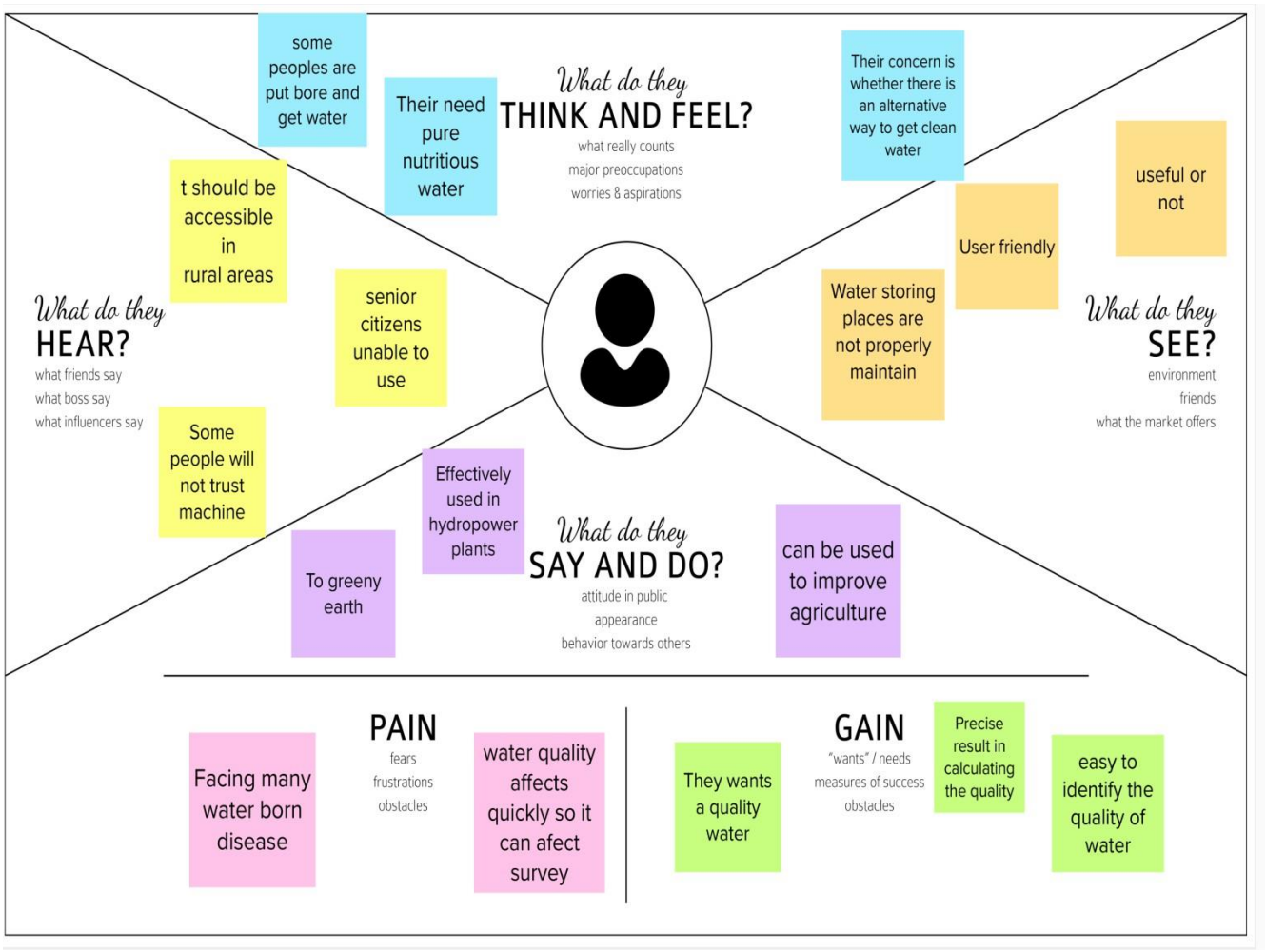
2.3 PROBLEM STATEMENT DEFINITION:



Problem Statement (PS)	**I am (Customer)**	**I'm trying to**	**But**	**Because**	**Which makes me feel**
PS-1	People	To consume the quality water	It is hard process	We have To do several tests in the laboratory	uncomfortable

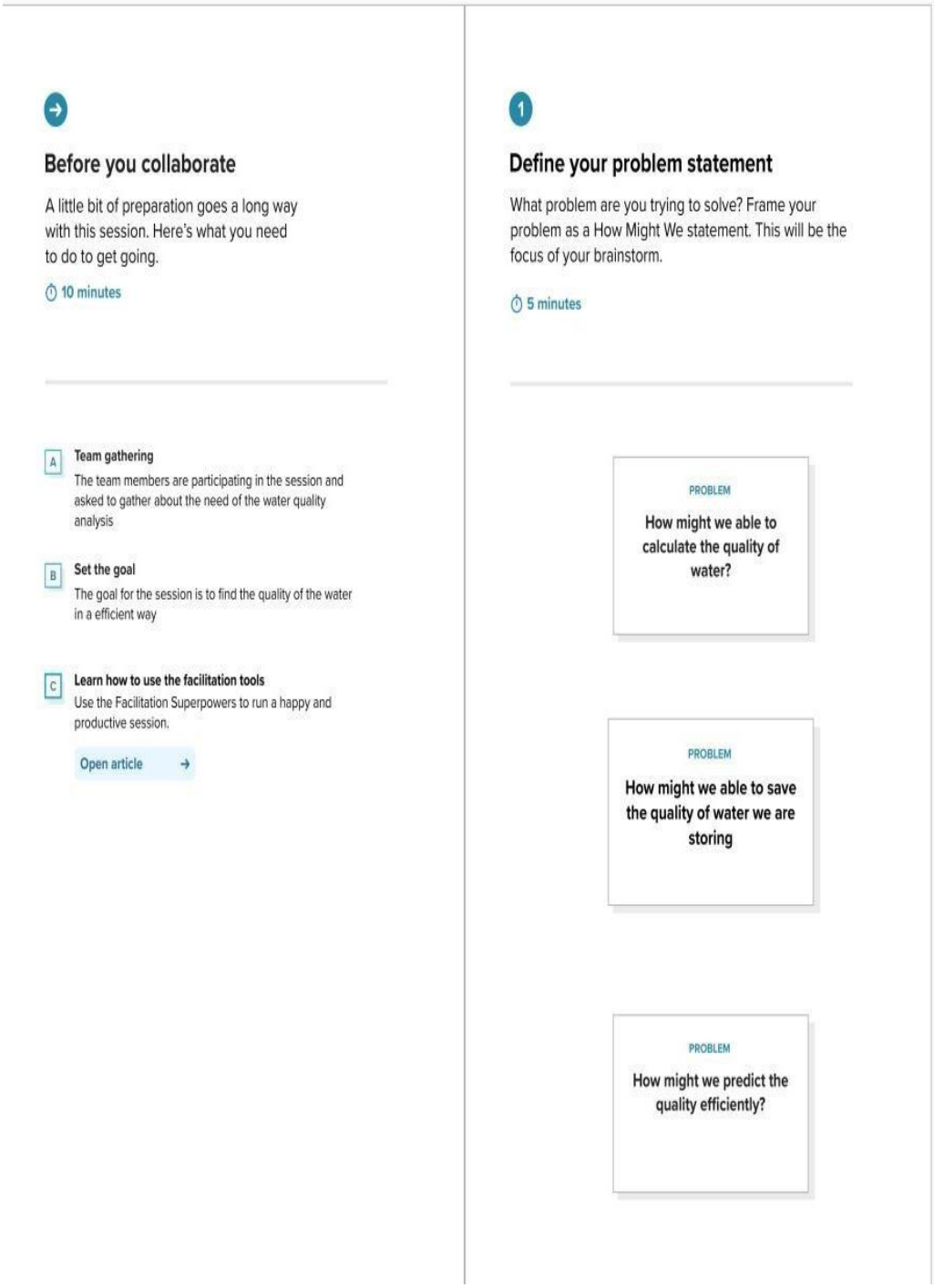
3. IDEATION & PROPOSED SOLUTION:

3.1 EMPATHY MAP CANVAS:

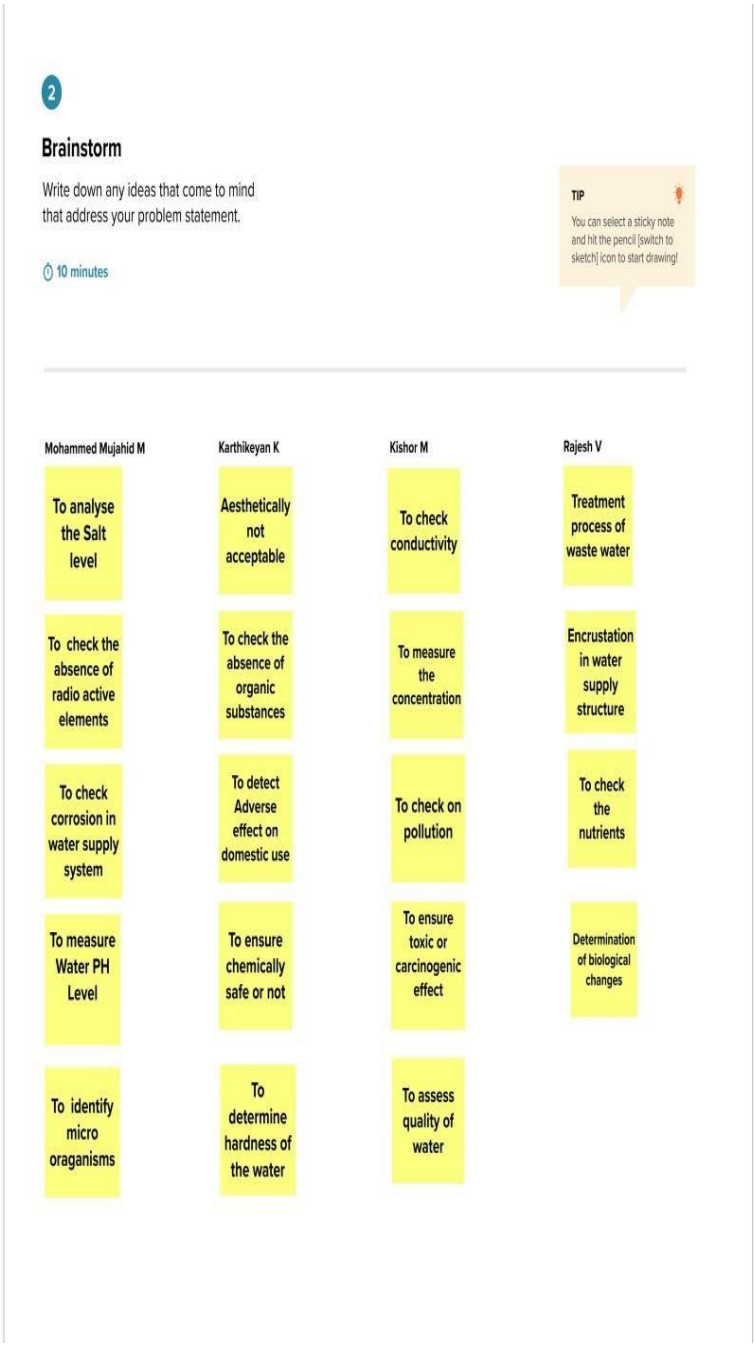


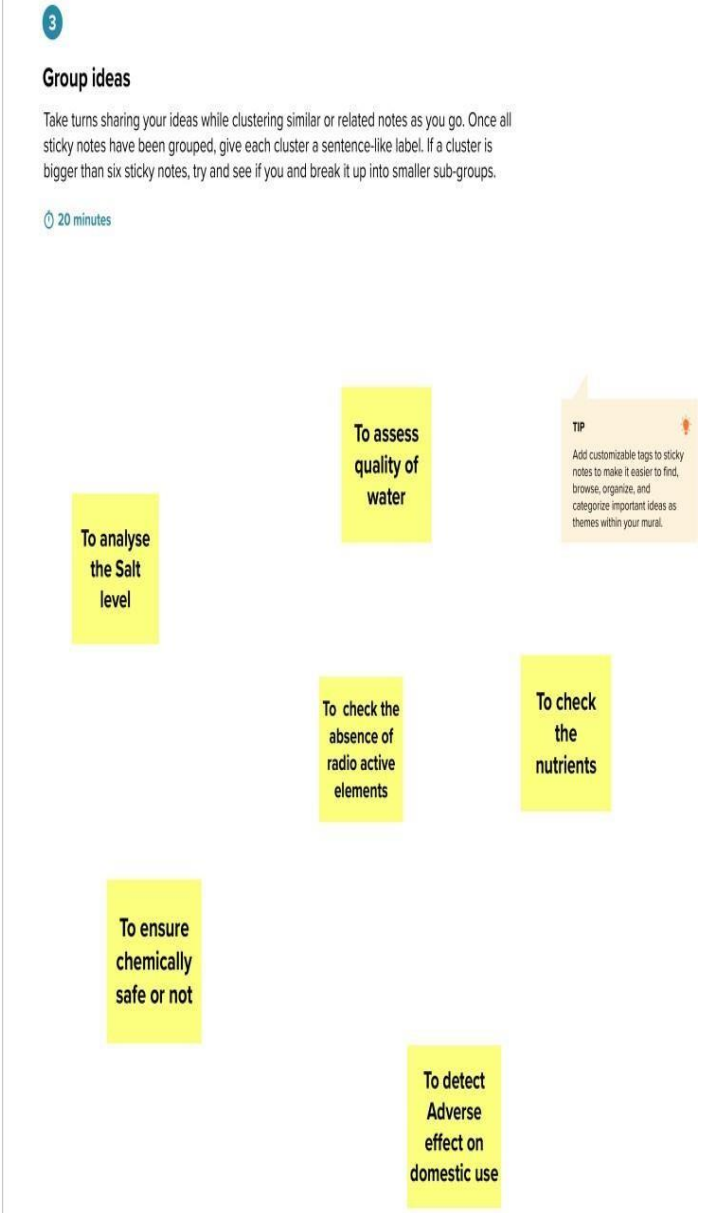
3.2 IDEATION & BRAINSTORMING:

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



Step-2: Brainstorm, Idea Listing and Grouping





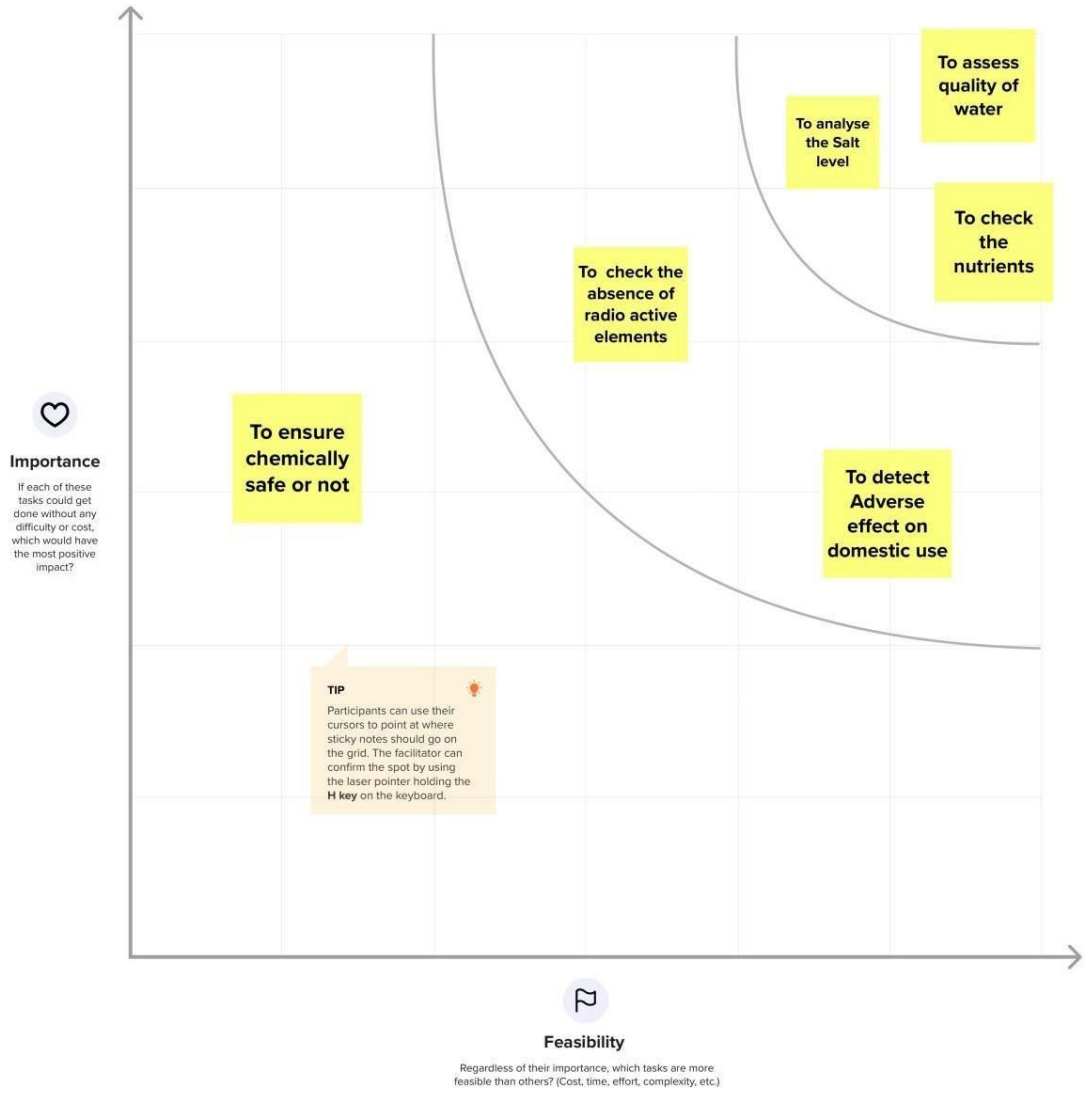
Step-3: Idea Prioritization

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

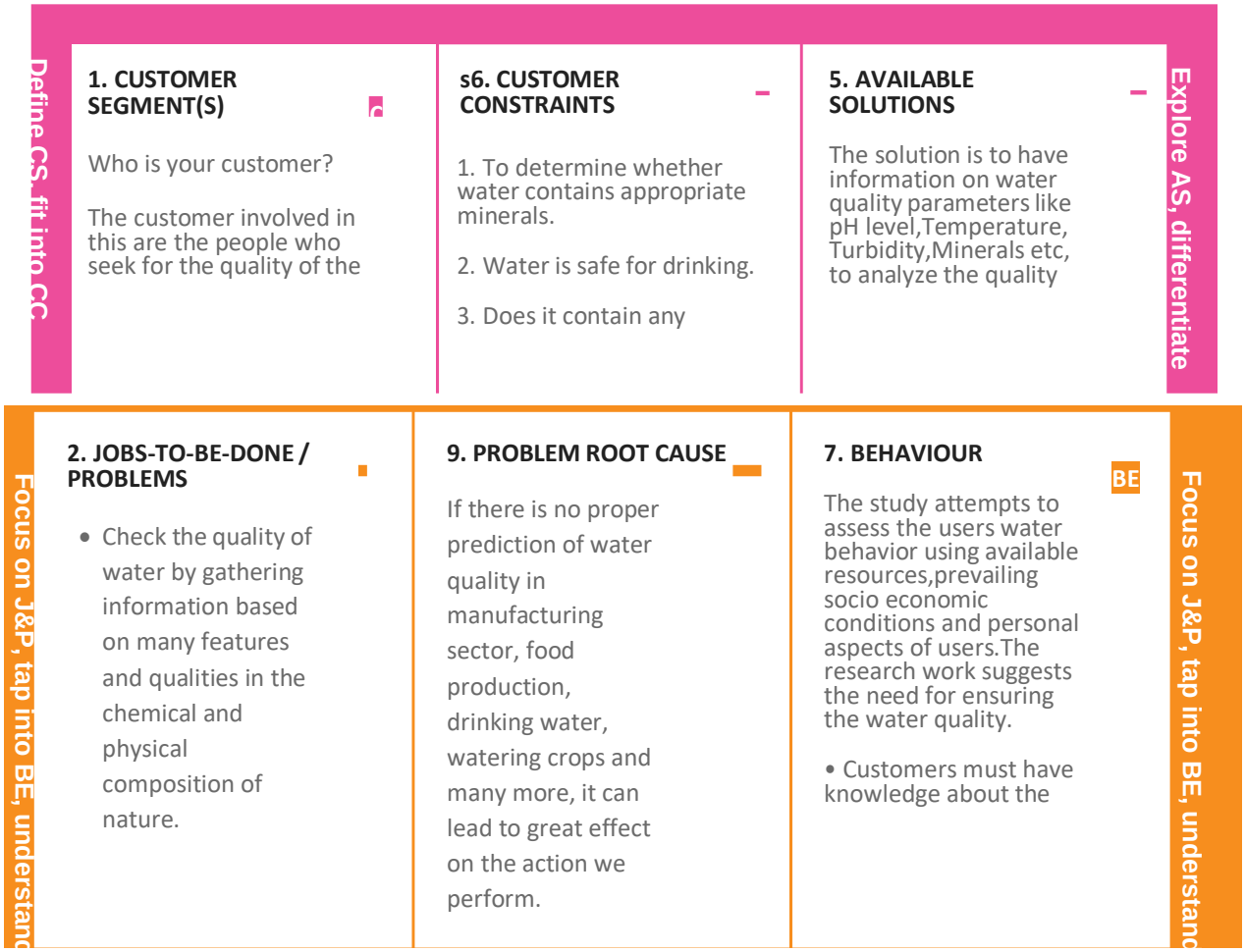


3.3 PROPOSED SOLUTION:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The problem which we are going to solve is that the people want to consume the quality water and that is taken care.
2.	Idea / Solution description	We are using the machine learning technique to find and predict that the water is in the quality to consume or not. The many parameters are used to determine the quality.
3.	Novelty / Uniqueness	The more accurate results are maintained by using the model evaluation technique.
4.	Social Impact / Customer Satisfaction	Helps people to better categorise the available water for various usage depending upon the analysis for which water conservation can be Practiced
5.	Business Model (Revenue Model)	Industries that provide sanitation facilities and products (like water purifiers, quality testers, etc..) can deploy this solution to provide more waste water treatment plants, better insights in health concerns and there may also be an increase in awareness

		and demand for better water quality and availability. People will start looking for treatments related to water borne diseases as the awarenesss increases.
6.	Scalability of the Solution	This system is enriched with all the testing environment, it is scalable to test all the water available in the globe.

3.4 PROBLEM SOLUTION FIT:



3. TRIGGERS

The water available is needed to be classified for its best usage on its constituents for various purpose. To analyze it we can use ML prediction about the water.

4. EMOTIONS: BEFORE / AFTER

BEFORE: Without appropriate technology to analyze the water quality, lead to various diseases.

AFTER: Now it is easy to evaluate the quality of water with the help of this application.

10. YOUR SOLUTION

1. It cluster the parameter like temperature, turbidity, hardness, pH level,

and dissolved minerals in the water.

2. It also evaluate the effort of substantial nutrients loads on overall

water quality.

3. Accurate model can be selected based on the outcome in the

model evaluation.

8. CHANNELS of BEHAVIOUR

8.1 ONLINE

People can make use of ML prediction to provide the various

characteristic of water as input and make it predict the proper

use of water usage depending upon the predefined learnings to machine.

8.2 OFFLINE

It makes easy to provide the measurements of water to the

machine and to predict the usage of quality of water for better use.

4. REQUIREMENT ANALYSIS:

4.1 FUNCTIONAL REQUIREMENTS:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Form	User can enter the information of the water bodies and using those details we can able to predict the quality.
FR-2	Executive administration	Regulation of monitoring the water environment status and regulatory compliance like pollution event emergency management, and it includes two different functions: early warning/forecast monitoring.
FR-3	Data handling	File contains water quality metrics for different waterbodies.
FR-4	Quality analysis	Analyze with the acquired information of the water across various water quality indicator like (PH, Turbidity, TDS, Temperature) using different models.
FR-5	Model prediction	Confirming based on water quality index and shows the machine learning prediction (Good, Partially Good, Poor) with the percentage of presence of various parameter.

4.2 NON-FUNCTIONAL REQUIREMENTS:

Following are the non-functional requirements of the proposed solution.

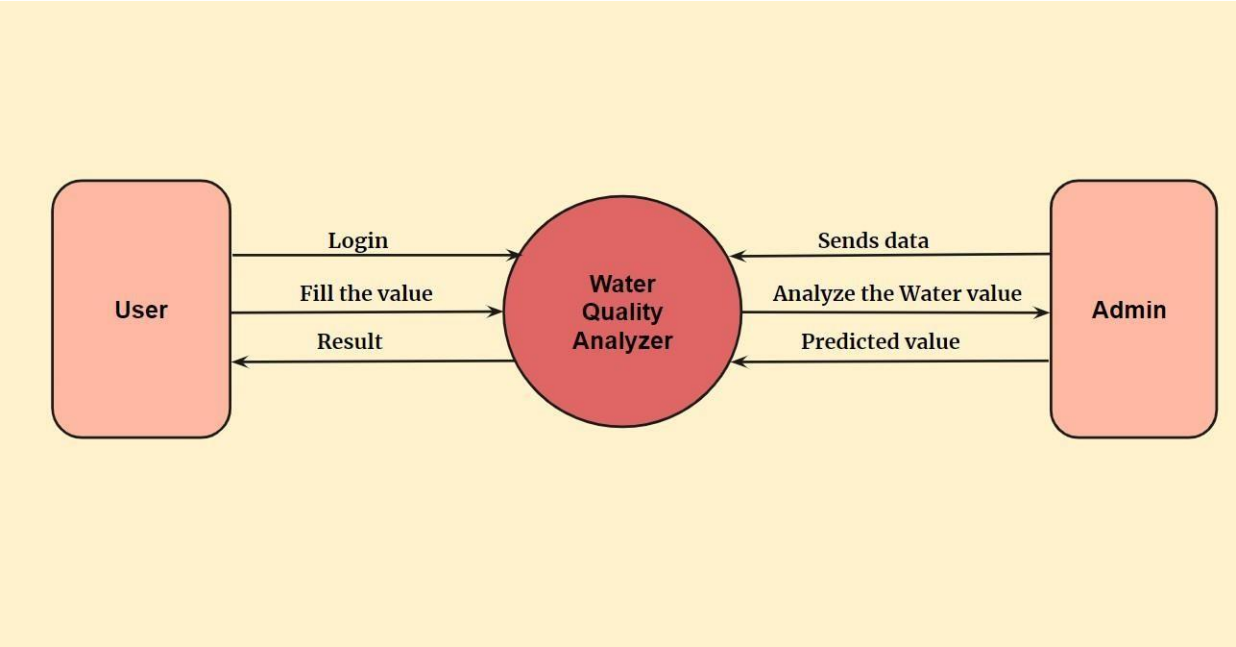
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The system provides a natural interaction with the users. Accurate water quality prediction with short time analysis and provide prediction safe to drink or not using some parameters and provide a great significance for water environment protection.
NFR-2	**Reliability**	The system is very reliable as it can last for long period of time when it is well maintained. The model can be extended in large scale by increasing the datasets.
NFR-3	**Performance**	Our system should run on 32-bit (x86) or 64-bit (x64) Dual-core 2.66-GHZ or faster processor. It should not exceed 2 GB RAM.
NFR-4	**Availability**	The system should be available for the duration of the user access the system until the user terminate the access. The system response to request of the user in less time and the recovery is done is less time.

NFR-5	**Scalability**	It provides an efficient outcome and has the ability to increase or decrease the performance of the system based on the datasets.

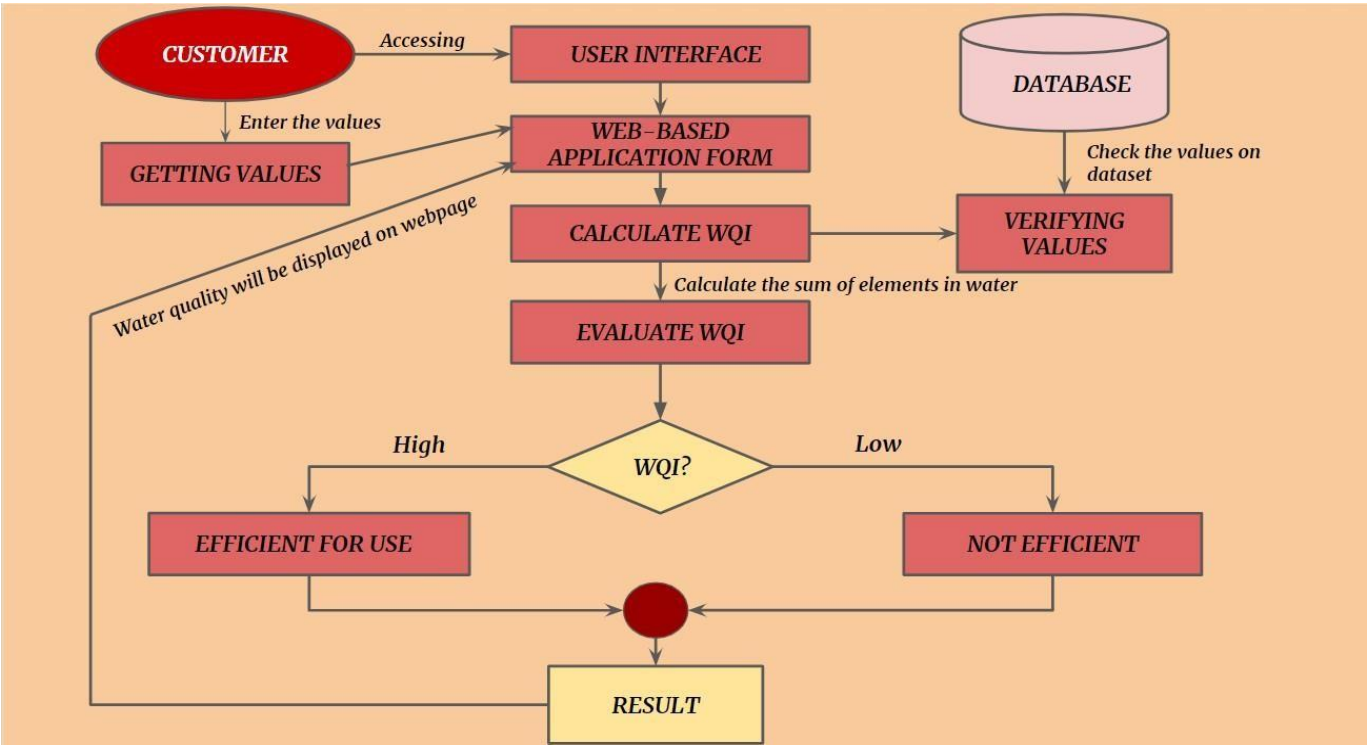
5. PROJECT DESIGN:

5.1 DATA FLOW DIAGRAMS:

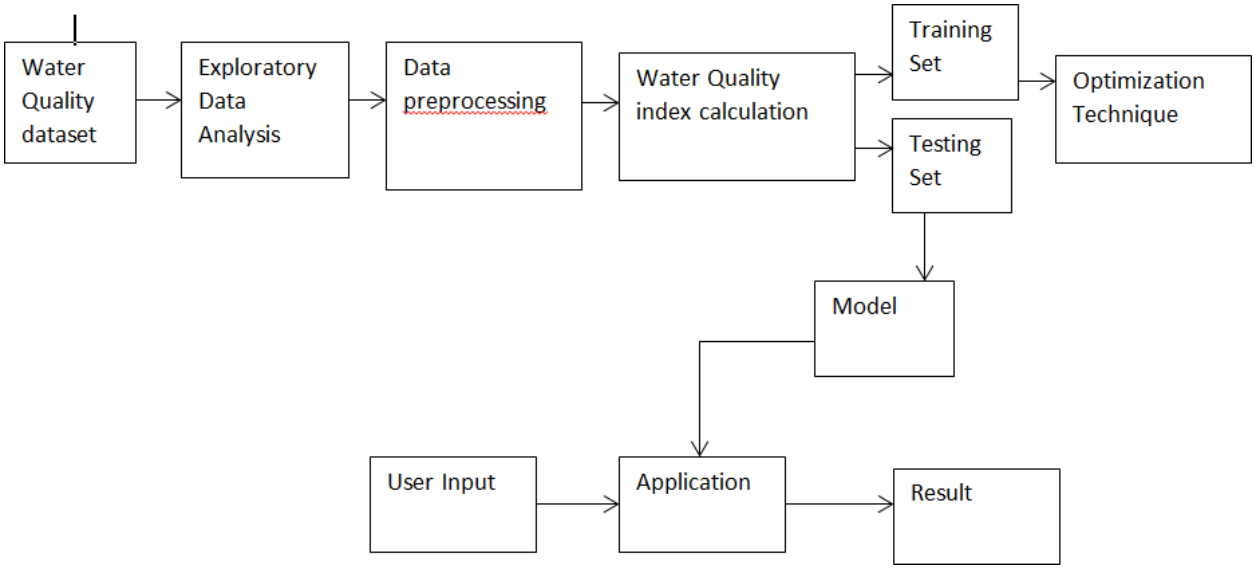
DFD LEVEL 0:



DFD LEVEL 1:



5.2 SOLUTION & TECHNICAL ARCHITECTURE:



5.3 USER STORIES:

User type	Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Priority
Admin	Sprint-1	Data Preparation	USN-1	Collecting water dataset and pre-processing it	High
Admin	Sprint-1	Handling Missing values	USN-2	Handle all the missing values in the Dataset	High
Admin	Sprint-1	Calculate the Water Quality Index	USN-3	Calculate the water quality index using the collected dataset	High
Admin	Sprint-1	Data Visualization	USN-4	Visualize the data using the histogram and heatmaps.	Medium
Admin	Sprint-2	Model Building	USN-5	Create an ML model to predict Water quality	High
Admin	Sprint-3	Model Evaluation	USN-6	Calculate the performance, error rate, and complexity of the ML model and evaluate the dataset based on the parameter that the	High

User type	Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Priority
				dataset consists of.	
Admin	Sprint-3	Model Deployment	USN-7	As a user, I need to deploy the model and need to find the results.	Medium
User	Sprint-3	Web page (Form)	USN-8	As a user, I can use the application by entering the water dataset to analyze or predict the results.	High
Admin	Sprint-4	Flask App	USM-9	Flask app should be created to act as an interface between the frontend and model	High

6. PROJECT PLANNING AND SCHEDULING:

6.1. SPRINT PLANNING &ESTIMATION:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Preparation	USN-1	Collecting water dataset and pre-processing it	5	High	Sukumar G Sarvesh Kumar Y
Sprint-1	Handling Missing values	USN-2	Handle all the missing values in the dataset	5	High	
Sprint-1	Calculate the Water Quality Index	USN-3	Calculate the water quality index using the collected dataset	5	High	A S Lokesh Reddy S Venkata Krishna
Sprint-1	Data Visualization	USN-4	Visualize the data using the histogram and heatmaps.	5	Medium	
Sprint-2	Model Building	USN-5	Create an ML model to predict waterquality	20	High	Sukumar G, A S Lokesh Reddy
Sprint-3	Model Evaluation	USN-6	Calculate the performance, error rate, and complexity of the ML model and evaluate thedataset based on the parameter that the dataset	5	High	

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
			consists of.			
Sprint-3	Model Deployment	USN-7	As a user, I need to deploy the model and need to find the results.	10	Medium	S Venkata Krishna, Sarvesh Kumar Y, Sukumar G
Sprint-3	Web page (Form)	USN-8	As a user, I can use the application by entering the water dataset to analyze or predict the results.	5	High	A S Lokesh Reddy, Sarvesh Kumar Y, Sukumar G
Sprint-4	Flask App	USM-9	Flask app should be created to act as an interface between the frontend and model	20	High	Sarvesh Kumar Y, S Venkata Krishna, A S Lokesh Reddy

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	20	05 Nov 2022

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

6.3 REPORTS FROM JIRA:

7.CODING & SOLUTIONING:

7.1 FEATURE 1:

The proposed system is the machine learning model where we could able to predict the quality of the water from giving the necessary details regarding the water body. This part deals with creating a model from the random forest algorithm. With the dataset we will be finding out the water quality index and using that we split the data into the training and testing set. Then the model will be created using the splitted data. After the model is created the accuracy of the model will be determined and model is deployed in the pickle. There is also another method to deploy a model using the IBM cloud.

The below code is the model created from the random forest algorithm,

```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
regressor.fit(x_train, y_train)
y_pred = regressor.predict(x_test)
```

The model can be deployed in the pickle file using the below code

```
import pickle
pickle.dump(regressor,open('wqi.pkl', 'wb'))
```

The flask app is created to act as an interface to predict the quality from the details the user is giving,

```
import numpy as np
from flask import Flask,render_template,request
import pickle
```

```
app= Flask(__name__)
model=pickle.load(open(r'C:\Users\Admin\Desktop\final_project\App\wqi.pkl','rb')
)
@app.route('/')
def home() :
  return render_template("web.html")
@app.route('/login',methods = ['POST'])
def login() :
  do = request.form["do"]
  ph = request.form["ph"]
  co = request.form["co"]
  bod = request.form["bod"]
  tc = request.form["tc"]
  na = request.form["na"]
  total = [[float(do),float(ph),float(co),float(bod),float(na),float(tc)]]
  y_pred = model.predict(total)
  y_pred=y_pred[[0]]
  if(y_pred >= 95 and y_pred<=100):
    pred = 'Excellent, The Predicted Value Is'+ str(y_pred)
  elif(y_pred >= 89 and y_pred<=94):
    pred = 'Very Good, The Predicted Value Is'+ str(y_pred)
  elif(y_pred >= 80 and y_pred<=88):
    pred = 'Good, The Predicted Value Is'+ str(y_pred)
  elif(y_pred >= 65 and y_pred<=79):
    pred = 'Fair, The Predicted Value Is'+ str(y_pred)
  elif(y_pred >= 45 and y_pred<=64):
    pred = 'Marginal, The Predicted Value Is'+ str(y_pred)
```

```
  else:
    pred = 'Poor, The Predicted Value Is'+ str(y_pred)

  return render_template('web.html', output='{}'.format(pred))

if__name__== '__main__':
     app.run(debug = True)
```

7.2 FEATURE 2:

The model is deployed in the IBM cloud with the following code.

```
from ibm_watson_machine_learning import APIClient
wml_credentials = {
   "apikey":"628i3j3-GpGkQuV9e6LMj86WTz4sURkK3mgnsN1J4YZN",
   "url":"https://us-south.ml.cloud.ibm.com"
}
wml_client = APIClient(wml_credentials)
wml_client.spaces.list()
space_id = "b399e114-ceed-4f9c-a056-275bdfbe2c3c"
wml_client.set.default_space(space_id)
wml_client.software_specifications.list()
MODEL_NAME = 'Water-Quality'
DEPLOYMENT_NAME = 'water-quality-prediction'
DEPLOY_MODEL = regressor
software_spec_uid= wml_client.software_specifications.get_id_by_name('runtime-
22.1-py3.9')
model_props = {
   wml_client.repository.ModelMetaNames.NAME: MODEL_NAME,
```

```
    wml_client.repository.ModelMetaNames.TYPE: 'scikit-learn_1.0',
    wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:
software_spec_uid
}
model_details = wml_client.repository.store_model(
    model=DEPLOY_MODEL,
    meta_props=model_props,
    training_data=x_train,
    training_target=y_train
)
```

The flask app for the IBM deployed model is,

```
import numpy as np
from flask import Flask,render_template,request
import pickle

import requests

# NOTE: you must manually set API_KEY below using information retrieved
from your IBM Cloud account.
API_KEY = "628i3j3-GpGkQuV9e6LMj86WTz4sURkK3mgnsN1J4YZN"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey":
 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]
```

```
header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}


app= Flask(__name__)

@app.route('/')
def home() :
  return render_template("web.html")
@app.route('/login',methods = ['POST'])
def login() :
  do = request.form["do"]
  ph = request.form["ph"]
  co = request.form["co"]
  bod = request.form["bod"]
  tc = request.form["tc"]
  na = request.form["na"]
  total = [[float(do),float(ph),float(co),float(bod),float(na),float(tc)]]

  payload_scoring = {"input_data": [{"fields": [['f0','f1','f2','f3','f4','f5']], "values":
total}]}
  response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/2ad7e145-2d11-434d-9fc2-
8e0a355734ed/predictions?version=2022-11-15', json=payload_scoring,
headers={'Authorization': 'Bearer ' + mltoken})
  print("Scoring response")
  print(response_scoring.json())
  pred=response_scoring.json()
```

```
  res=pred['predictions'][0]['values'][0][0]
  print(res)
  return render_template('web.html', output='{}'.format(res))

if__name__== '__main__':
    app.run(debug = True,port=5010)
```

8. TESTING:

8.1. TEST CASES:

- Verify that the user could able to use that web page.
- Verify that the user could able to enter the value.
- Verify that the values entered by the user are computed.
- Verify that the user could able to see the predicted value.

8.2.USER ACCEPTANCE TESTING:

1. **Defect Analysis**

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	7	4	2	3	17
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	10	2	4	15	31
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	3	2	1	6
Totals	20	12	13	21	66

2. **Test Case Analysis**

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	6	0	0	6

Client Application	45	0	0	45
Security	2	0	0	2
Outsource Shipping	2	0	0	2
Exception Reporting	7	0	0	7
Final Report Output	3	0	0	3
Redirecting	1	0	0	1

9. RESULTS:

9.1. PERFORMANCE METRICS:

The performance metrics are the accuracy of the model and the errors that the model is predicting. The MAE(Mean Absolute Error), MSE(Mean Squared Error) and RMSE(Root Mean Squared Error) are 0.9892681704260707, 5.557973864661655 and 2.3575355489709278 respectively. The accuracy of the model is 96.97034933666699. These are the factors that determine the performance of the model.

10. ADVANTAGES AND DISADVANTAGES:

Advantages:

- Speeds up manual testing and improve the overall quality
- It provides more test coverage to the test cases whenever there is a change or update in application
- This model in software testing evaluates test cases and various error incidents in a short span of time.
- Reduces Ignored Bugs Probability
- This offers advanced features such as more revenues at a reduced cost, enhanced user experience, competitive positioning in the industry as the company delivers a high-quality product.

Disadvantages:

- These applications automate the majority of tedious and repetitive tasks
- When it comes to processing data, the scale of data generated far exceeds the human capacity to understand and analyze it.

11. CONCLUSION:

One of the most important resources for survival is water, and WQI measures the quality of water. Traditionally, one must undergo an expensive and time-consuming lab analysis to test the purity of the water. This project investigated a machine learning approach to forecast water quality using basic, readily accessible water quality data and produces the accuracy of 96.97034933666699.

12. FUTURE SCOPE:

The fundamental issue that machine learning models with unbalanced datasets encounter is overfitting. In the future, generalized results are expected to be obtained utilizing mixed features and a balanced dataset. In addition, we want to forecast water quality and automatically extract features using deep learning and a sizable dataset.

13. APPENDIX:

SOURCE CODE:

Efficient_water_quality_analysis.ipynb:

```
# **Importing libraries**
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
import sklearn
# **Reading Dataset**
data=pd.read_csv(r'C:\Users\Admin\Desktop\final_project\Dataset\water_dataX.cs
v',encoding='ISO-8859-1',low_memory=False)
# **Analyse the data**
data.head()
data.describe()
data.info()
data.shape
# **Handling Missing Values_1**
data.isnull().any()
data.isnull().sum()
data.dtypes
data['Temp']=pd.to_numeric(data['Temp'],errors='coerce')
data['D.O. (mg/l)']=pd.to_numeric(data['D.O. (mg/l)'],errors='coerce')
data['PH']=pd.to_numeric(data['PH'],errors='coerce')
data['B.O.D. (mg/l)']=pd.to_numeric(data['B.O.D. (mg/l)'],errors='coerce')
```

```
data['CONDUCTIVITY (µmhos/cm)']=pd.to_numeric(data['CONDUCTIVITY (µmhos/cm)'],errors='coerce')
data['NITRATENAN N+NITRITENANN (mg/l)']=pd.to_numeric(data['NITRATENAN N+ NITRITENANN (mg/l)'],errors='coerce')
data['TOTAL COLIFORM (MPN/100ml)Mean']=pd.to_numeric(data['TOTAL COLIFORM (MPN/100ml)Mean'],errors='coerce')
data.dtypes
data.isnull().sum()
data['Temp'].fillna(data['Temp'].mean(),inplace=True)
data['D.O. (mg/l)'].fillna(data['D.O. (mg/l)'].mean(),inplace=True)
data['PH'].fillna(data['PH'].mean(),inplace=True)
data['CONDUCTIVITY (µmhos/cm)'].fillna(data['CONDUCTIVITY (µmhos/cm)'].mean(),inplace=True)
data['B.O.D. (mg/l)'].fillna(data['B.O.D. (mg/l)'].mean(),inplace=True)
data['NITRATENAN N+ NITRITENANN (mg/l)'].fillna(data['NITRATENAN N+ NITRITENANN (mg/l)'].mean(),inplace=True)
data['TOTAL COLIFORM (MPN/100ml)Mean'].fillna(data['TOTAL COLIFORM (MPN/100ml)Mean'].mean(),inplace=True)
data.drop(["FECAL COLIFORM (MPN/100ml)"],axis=1,inplace=True)
data=data.rename(columns = {'D.O. (mg/l)': 'do'})
data=data.rename(columns = {'CONDUCTIVITY (µmhos/cm)': 'co'})
data=data.rename(columns = {'B.O.D. (mg/l)': 'bod'})
data=data.rename(columns = {'NITRATENAN N+ NITRITENANN (mg/l)': 'na'})
data=data.rename(columns = {'TOTAL COLIFORM (MPN/100ml)Mean': 'tc'})
data=data.rename(columns = {'STATION CODE': 'station'})
data=data.rename(columns = {'LOCATIONS': 'location'})
```

```
data=data.rename(columns = {'STATE': 'state'})
data=data.rename(columns = {'PH': 'ph'})
# **Water Quality Index (WQI) Calculation**
#calculation of pH
data['npH']=data.ph.apply(lambda x: (100 if(8.5>=x>=7)
                         else(80 if(8.6>=x>=8.5) or (6.9>=x>=6.8)
                          else (60 if(8.8>=x>=8.6) or (6.8>=x>=6.7)
                              else(40 if(9>=x>=8.8) or (6.7>=x>=6.5)
                                  else 0)))))
#calculation of dissolved oxygen
data['ndo']=data.do.apply(lambda x: (100 if(x>=6)
                         else(80 if(6>=x>=5.1)
                          else (60 if(5>=x>=4.1)
                              else(40 if(4>=x>=3)
                                  else 0)))))
#calculation of total coliform
data['nco']=data.tc.apply(lambda x: (100 if(5>=x>=0)
                         else(80  if(50>=x>=5)
                          else (60 if(500>=x>=50)
                              else(40 if(10000>=x>=500)
                                  else 0)))))
#calculation of B.D.O
data['nbdo']=data.bod.apply(lambda x:(100 if(3>=x>=0)
                         else(80 if(6>=x>=3)
                          else (60 if(80>=x>=6)
                              else(40 if(125>=x>=80)
                                  else 0)))))
```

```
#calculation of electric conductivity
data['nec']=data.co.apply(lambda x:(100 if(75>=x>=0)
                          else(80 if(150>=x>=75)
                           else (60 if(225>=x>=150)
                              else(40 if(300>=x>=225)
                                 else 0)))))
#calculation of nitrate
data['nna']=data.na.apply(lambda x:(100 if(20>=x>=0)
                          else(80 if(50>=x>=20)
                           else (60 if(100>=x>=50)
                              else(40 if(200>=x>=100)
                                 else 0)))))
#Calculation of Water Quality Index WQI
data['wph']=data.npH*0.165
data['wdo']=data.ndo*0.281
data['wbdo']=data.nbdo*0.234
data['wec']=data.nec*0.009
data['wna']=data.nna*0.028
data['wco']=data.nco*0.281
data['wqi']=data.wph+data.wdo+data.wbdo+data.wec+data.wna+data.wco
data
#Calculation of overall WQI for each year
average = data.groupby('year')['wqi'].mean()
average.head()
# # Data Visualization
sns.distplot(data['ph'])
plt.show()
```

```
data.hist(figsize=(14,14))
plt.show()
plt.figure(figsize=(13,8))
sns.heatmap(data.corr(),annot=True,cmap='terrain')
plt.show()
# **Splitting Dependent and Independent Columns**
data.head()
data.drop(['location','station','state'],axis =1,inplace=True)
data.head()
x=data.iloc[:,1:7].values
x.shape
y=data.iloc[:,-1:].values
y.shape
print(x)
print(y)
# **Splitting the Data Into Train and Test**
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,random_state=10)
# ##**Random_Forest_Regression**
#Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
regressor.fit(x_train, y_train)
```

```
y_pred = regressor.predict(x_test)
from sklearn import metrics
print('MAE:',metrics.mean_absolute_error(y_test,y_pred))
print('MSE:',metrics.mean_squared_error(y_test,y_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
#accuracy of the model
metrics.r2_score(y_test, y_pred)
# ##**Save The Model**
import pickle
pickle.dump(regressor,open('wqi.pkl', 'wb'))
```

Efficient_water_quality_analysis_2.ipynb:

```
# **Importing libraries**
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
import sklearn
# **Reading Dataset**
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='h8h1AiRjDQe6G8CwNquXaWxUXo_hBMwaM_v7UC2hhrrl',
    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 = 'efficientwaterqualityanalysisandp-donotdelete-pr-rdu73l4mduwakc'
object_key = 'water_data1.txt'

streaming_body_1 = cos_client.get_object(Bucket=bucket,
Key=object_key)['Body']
data=pd.read_csv(streaming_body_1)
# Your data file was loaded into a botocore.response.StreamingBody object.
# Please read the documentation of ibm_boto3 and pandas to learn more about the
possibilities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/
# **Analyse the data**
data.head()
data.describe()
data.info()
data.shape
# **Handling Missing Values_1**
data.isnull().any()
data.isnull().sum()
```

```
data.dtypes
data['Temp']=pd.to_numeric(data['Temp'],errors='coerce')
data['D.O. (mg/l)']=pd.to_numeric(data['D.O. (mg/l)'],errors='coerce')
data['PH']=pd.to_numeric(data['PH'],errors='coerce')
data['B.O.D. (mg/l)']=pd.to_numeric(data['B.O.D. (mg/l)'],errors='coerce')
data['CONDUCTIVITY (µmhos/cm)']=pd.to_numeric(data['CONDUCTIVITY
(µmhos/cm)'],errors='coerce')
data['NITRATENAN N+NITRITENANN
(mg/l)']=pd.to_numeric(data['NITRATENAN N+ NITRITENANN
(mg/l)'],errors='coerce')
data['TOTAL COLIFORM (MPN/100ml)Mean']=pd.to_numeric(data['TOTAL
COLIFORM (MPN/100ml)Mean'],errors='coerce')
data.dtypes
data.isnull().sum()
data['Temp'].fillna(data['Temp'].mean(),inplace=True)
data['D.O. (mg/l)'].fillna(data['D.O. (mg/l)'].mean(),inplace=True)
data['PH'].fillna(data['PH'].mean(),inplace=True)
data['CONDUCTIVITY (µmhos/cm)'].fillna(data['CONDUCTIVITY
(µmhos/cm)'].mean(),inplace=True)
data['B.O.D. (mg/l)'].fillna(data['B.O.D. (mg/l)'].mean(),inplace=True)
data['NITRATENAN N+ NITRITENANN (mg/l)'].fillna(data['NITRATENAN N+
NITRITENANN (mg/l)'].mean(),inplace=True)
data['TOTAL COLIFORM (MPN/100ml)Mean'].fillna(data['TOTAL COLIFORM
(MPN/100ml)Mean'].mean(),inplace=True)
data.drop(["FECAL COLIFORM (MPN/100ml)"],axis=1,inplace=True)
data=data.rename(columns = {'D.O. (mg/l)': 'do'})
data=data.rename(columns = {'CONDUCTIVITY (µmhos/cm)': 'co'})
```

```
data=data.rename(columns = {'B.O.D. (mg/l)': 'bod'})
data=data.rename(columns = {'NITRATENAN N+ NITRITENANN (mg/l)': 'na'})
data=data.rename(columns = {'TOTAL COLIFORM (MPN/100ml)Mean': 'tc'})
data=data.rename(columns = {'STATION CODE': 'station'})
data=data.rename(columns = {'LOCATIONS': 'location'})
data=data.rename(columns = {'STATE': 'state'})
data=data.rename(columns = {'PH': 'ph'})
# **Water Quality Index (WQI) Calculation**
#calculation of pH
data['npH']=data.ph.apply(lambda x: (100 if(8.5>=x>=7)
                        else(80 if(8.6>=x>=8.5) or (6.9>=x>=6.8)
                          else (60 if(8.8>=x>=8.6) or (6.8>=x>=6.7)
                             else(40 if(9>=x>=8.8) or (6.7>=x>=6.5)
                                else 0)))))
#calculation of dissolved oxygen
data['ndo']=data.do.apply(lambda x: (100 if(x>=6)
                        else(80 if(6>=x>=5.1)
                          else (60 if(5>=x>=4.1)
                             else(40 if(4>=x>=3)
                                else 0)))))
#calculation of total coliform
data['nco']=data.tc.apply(lambda x: (100 if(5>=x>=0)
                        else(80  if(50>=x>=5)
                          else (60 if(500>=x>=50)
                             else(40 if(10000>=x>=500)
                                else 0)))))
#calculation of B.D.O
```

```
data['nbdo']=data.bod.apply(lambda x:(100 if(3>=x>=0)
                            else(80 if(6>=x>=3)
                              else (60 if(80>=x>=6)
                                 else(40 if(125>=x>=80)
                                    else 0)))))
#calculation of electric conductivity
data['nec']=data.co.apply(lambda x:(100 if(75>=x>=0)
                            else(80 if(150>=x>=75)
                              else (60 if(225>=x>=150)
                                 else(40 if(300>=x>=225)
                                    else 0)))))
#calculation of nitrate
data['nna']=data.na.apply(lambda x:(100 if(20>=x>=0)
                            else(80 if(50>=x>=20)
                              else (60 if(100>=x>=50)
                                 else(40 if(200>=x>=100)
                                    else 0)))))
#Calculation of Water Quality Index WQI
data['wph']=data.npH*0.165
data['wdo']=data.ndo*0.281
data['wbdo']=data.nbdo*0.234
data['wec']=data.nec*0.009
data['wna']=data.nna*0.028
data['wco']=data.nco*0.281
data['wqi']=data.wph+data.wdo+data.wbdo+data.wec+data.wna+data.wco
data
#Calculation of overall WQI for each year
```

```
average = data.groupby('year')['wqi'].mean()
average.head()
# # Data Visualization
sns.distplot(data['ph'])
plt.show()
data.hist(figsize=(14,14))
plt.show()
plt.figure(figsize=(13,8))
sns.heatmap(data.corr(),annot=True,cmap='terrain')
plt.show()
# **Splitting Dependent and Independent Columns**
data.head()
data.drop(['location','station','state'],axis =1,inplace=True)
data.head()
x=data.iloc[:,1:7].values
x.shape
y=data.iloc[:,-1:].values
y.shape
print(x)
print(y)
# **Splitting the Data Into Train and Test**
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,random_state=10)
# ##**Random_Forest_Regression**
#Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

```
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
regressor.fit(x_train, y_train)
y_pred = regressor.predict(x_test)
from sklearn import metrics
print('MAE:',metrics.mean_absolute_error(y_test,y_pred))
print('MSE:',metrics.mean_squared_error(y_test,y_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
#accuracy of the model
metrics.r2_score(y_test, y_pred)
get_ipython().system('pip install -U ibm-watson-machine-learning')
from ibm_watson_machine_learning import APIClient
wml_credentials = {
   "apikey":"628i3j3-GpGkQuV9e6LMj86WTz4sURkK3mgnsN1J4YZN",
   "url":"https://us-south.ml.cloud.ibm.com"
}
wml_client = APIClient(wml_credentials)
wml_client.spaces.list()
space_id = "b399e114-ceed-4f9c-a056-275bdfbe2c3c"
wml_client.set.default_space(space_id)
wml_client.software_specifications.list()
MODEL_NAME = 'Water-Quality'
DEPLOYMENT_NAME = 'water-quality-prediction'
DEPLOY_MODEL = regressor
```

```
software_spec_uid =wml_client.software_specifications.get_id_by_name('runtime-
22.1-py3.9')
software_spec_uid
model_props = {
    wml_client.repository.ModelMetaNames.NAME: MODEL_NAME,
    wml_client.repository.ModelMetaNames.TYPE: 'scikit-learn_1.0',
    wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:
software_spec_uid
}
model_details = wml_client.repository.store_model(
    model=DEPLOY_MODEL,
    meta_props=model_props,
    training_data=x_train,
    training_target=y_train
)
model_details
```

web.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Water Quality Analysis and Prediction</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
```

```
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></
script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></scrip
t>
</head>
<body>
<!-- Navigation Bar-->
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
  <!-- Links -->
  <ul class="navbar-nav">
    <li class="nav-item">
      <a class="nav-link" href="https://used-car-price-prediction-
k.herokuapp.com">Home</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">About</a>
    </li>
  </ul>
</nav>
<br>
<div class="container">
  <!--<img align="middle" src="https://image.freepik.com/free-vector/by-my-car-
illustration-concept_114360-870.jpg"> -->
  <h1>Water Quality Prediction</h1>
  <p style="font-style: italic;">Fill the Details to predict the quality of water</p>
```

```
    <!-- User Input Form-->
    <form action="/login" method="POST">
        <div class="form-group">
            <label for="do">Dissolved Oxygen:</label>
            <input type="text" step="0.01" class="form-control"
placeholder="Dissolved Oxygen" id="Dissolved Oxygen" name="do"
required><br>

            <label for="ph">PH:</label>
            <input type="text" step="0.01" class="form-control" placeholder="PH"
id="PH" name="ph" required><br>

            <label for="co">Carbon Monoxide:</label>
            <input type="text" step="0.01" class="form-control" placeholder="Carbon
Monoxide" id="Carbon Monoxide" name="co" required><br>
            <label for="bo">Biochemical Oxygen Demand:</label>
            <input type="text" step="0.01" class="form-control"
placeholder="Biochemical Oxygen Demand" id="Biochemical Oxygen Demand"
name="bod" required><br>
            <label for="na">Sodium:</label>
            <input type="text" step="0.01" class="form-control"
placeholder="Sodium" id="Sodium" name="na" required><br>
            <label for="tc">Technetium:</label>
            <input type="text" step="0.01" class="form-control"
placeholder="Technetium" id="Technetium" name="tc" required><br>
            <div class="button-group" style="margin-top:15px;"><br>
```

```
            <button type="submit" name="submit" class="btn btn-
primary">Submit</button>
          </div>
        </div>
    </form>
</div>
<!-- After Prediction -->
<div class="container">
  <h2></h2>
  <div class="alert alert-info" role="alert">
    <strong></strong> {{output}}
  </div>
</div>
</body>
</html>
```

app.py:

```
import numpy as np
from flask import Flask,render_template,request
import pickle
app= Flask(__name__)
model=pickle.load(open(r'C:\Users\Admin\Desktop\final_project\App\wqi.pkl','rb')
)
@app.route('/')
def home() :
  return render_template("web.html")
@app.route('/login',methods = ['POST'])
def login() :
```

```
  do = request.form["do"]
  ph = request.form["ph"]
  co = request.form["co"]
  bod = request.form["bod"]
  tc = request.form["tc"]
  na = request.form["na"]
  total = [[float(do),float(ph),float(co),float(bod),float(na),float(tc)]]
  y_pred = model.predict(total)
  y_pred=y_pred[[0]]
  if(y_pred >= 95 and y_pred<=100):
    pred = 'Excellent, The Predicted Value Is'+ str(y_pred)
  elif(y_pred >= 89 and y_pred<=94):
    pred = 'Very Good, The Predicted Value Is'+ str(y_pred)
  elif(y_pred >= 80 and y_pred<=88):
    pred = 'Good, The Predicted Value Is'+ str(y_pred)
  elif(y_pred >= 65 and y_pred<=79):
    pred = 'Fair, The Predicted Value Is'+ str(y_pred)
  elif(y_pred >= 45 and y_pred<=64):
    pred = 'Marginal, The Predicted Value Is'+ str(y_pred)
  else:
    pred = 'Poor, The Predicted Value Is'+ str(y_pred)

  return render_template('web.html', output='{}'.format(pred))

if__name__== '__main__':
    app.run(debug = True)
```

app-ibm.py:

```
import numpy as np
from flask import Flask,render_template,request
import pickle

import requests

# NOTE: you must manually set API_KEY below using information retrieved
from your IBM Cloud account.
API_KEY = "628i3j3-GpGkQuV9e6LMj86WTz4sURkK3mgnsN1J4YZN"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey":
 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}



app= Flask(__name__)

@app.route('/')
def home() :
  return render_template("web.html")
@app.route('/login',methods = ['POST'])
def login() :
  do = request.form["do"]
  ph = request.form["ph"]
```

```
  co = request.form["co"]
  bod = request.form["bod"]
  tc = request.form["tc"]
  na = request.form["na"]
  total = [[float(do),float(ph),float(co),float(bod),float(na),float(tc)]]

  payload_scoring = {"input_data": [{"fields": [['f0','f1','f2','f3','f4','f5']], "values":
total}]}
  response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/2ad7e145-2d11-434d-9fc2-
8e0a355734ed/predictions?version=2022-11-15', json=payload_scoring,
headers={'Authorization': 'Bearer ' + mltoken})
  print("Scoring response")
  print(response_scoring.json())
  pred=response_scoring.json()
  res=pred['predictions'][0]['values'][0][0]
  print(res)
  return render_template('web.html', output='{}'.format(res))

if__name__== '__main__':
    app.run(debug = True,port=5010)
```

GITHUB LINK:
IBM-EPBL/IBM-Project-51089-1660970835: Efficient Water Quality Analysis & Prediction using Machine Learning (github.com)