## Project Report

**Team ID:** PNT2022TMID45216

**Team Leader:** Keerthana M

**Team Members:** Shanmugapriya M

Lathika R

Jeeva P

Banupriya K

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

Learning

# INTRODUCTION:

* 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.

# 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 factor 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.

# LITERATURE SURVEY:

* 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

1. 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.

1. 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.

# 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.

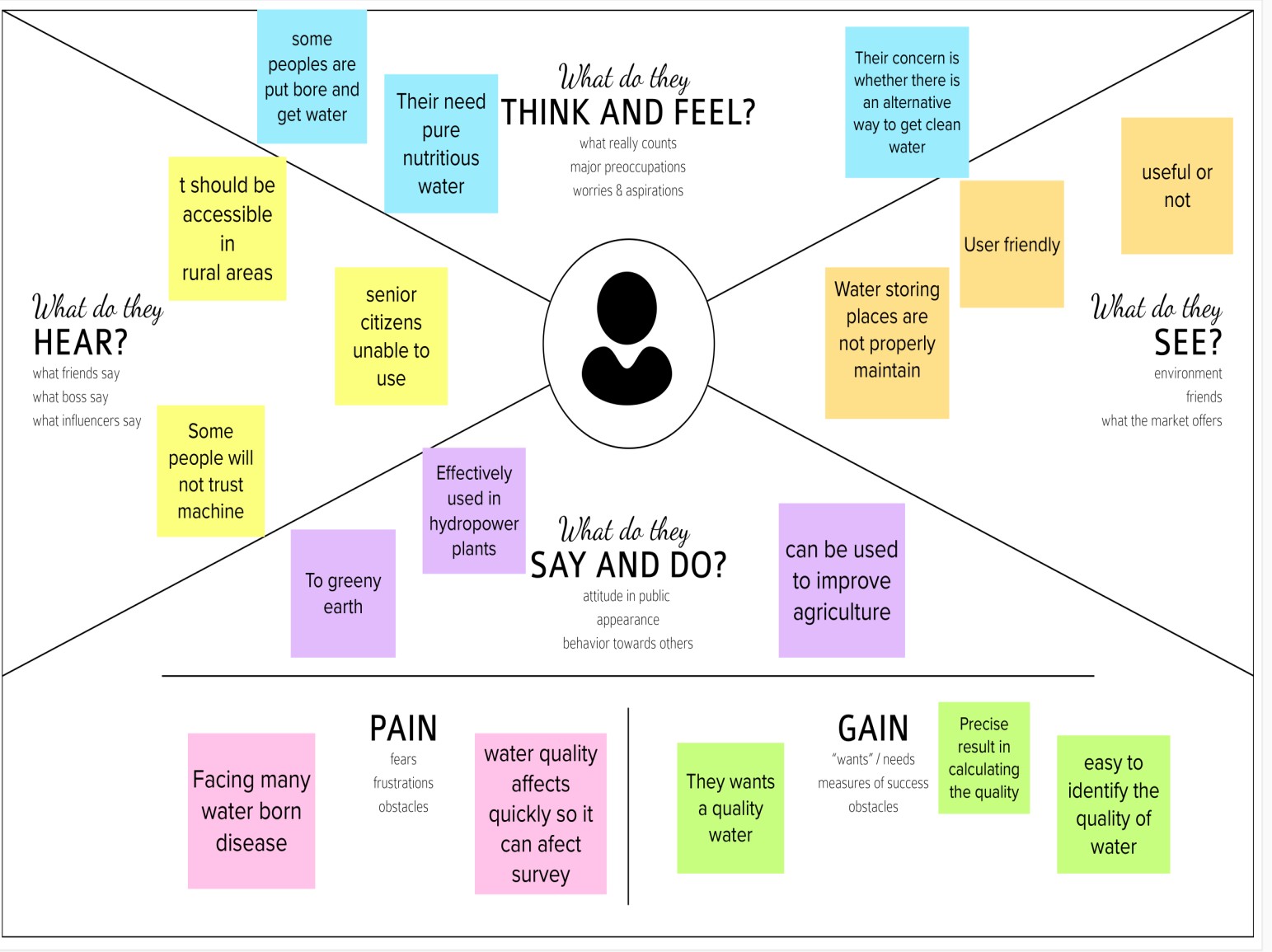
# PROBLEM STATEMENT DEFINITION:



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 |

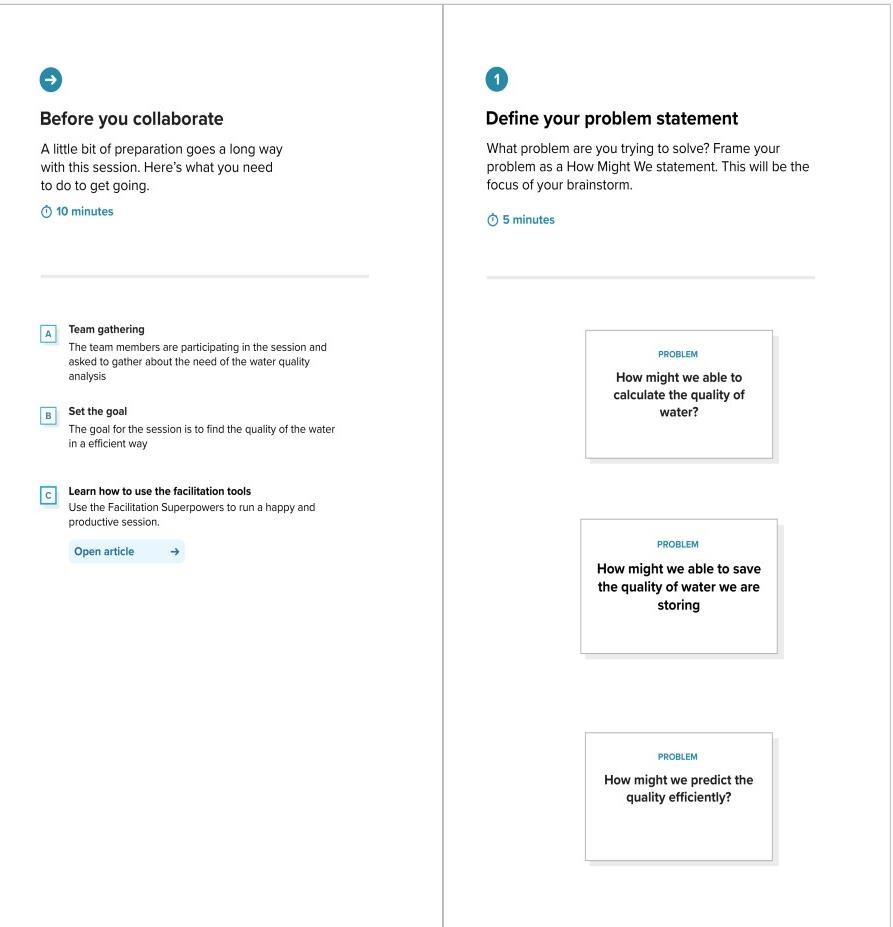
1. **IDEATION & PROPOSED SOLUTION:**

# EMPATHY MAP CANVAS:

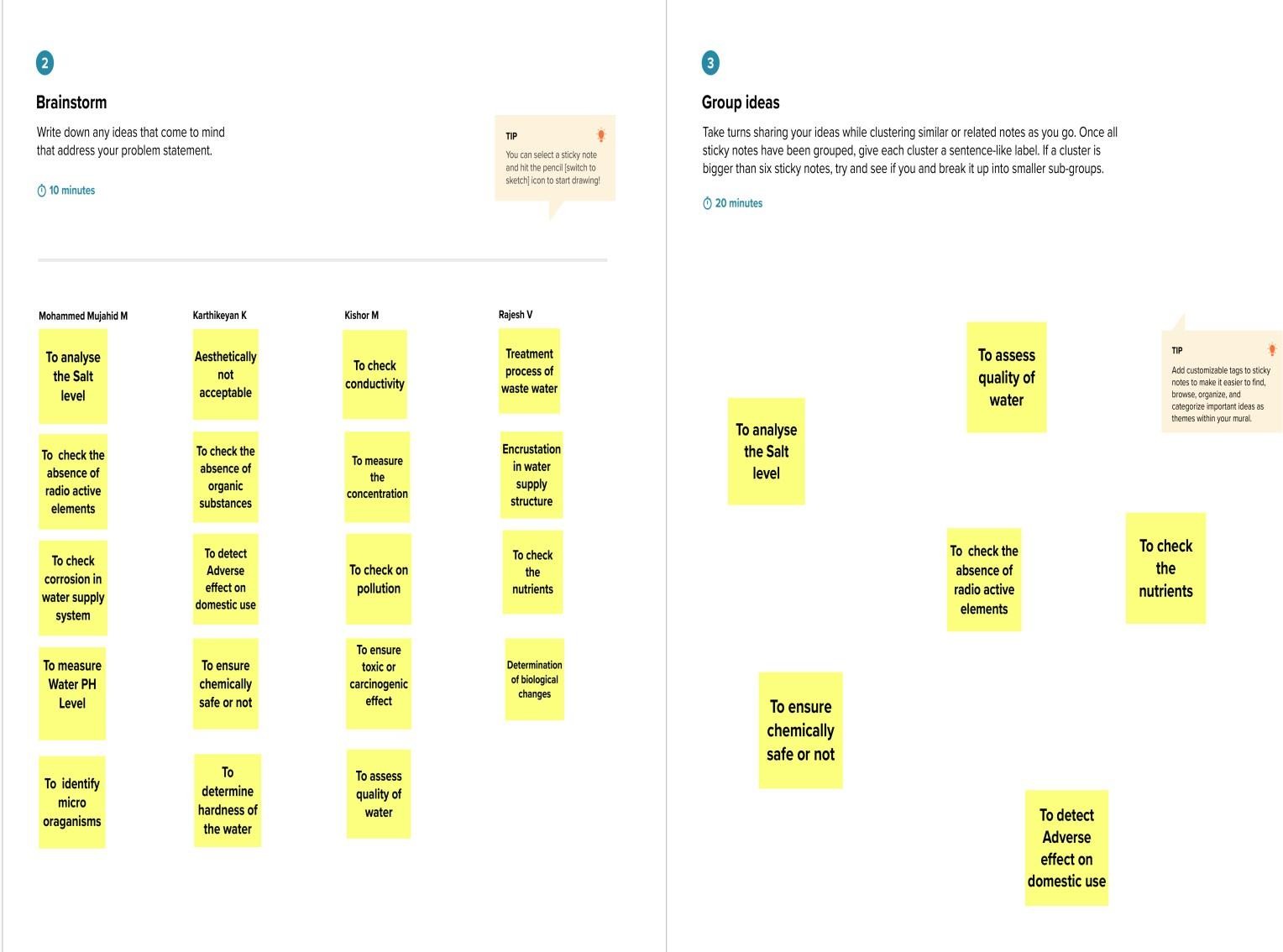


* 1. **IDEATION & BRAINSTORMING:**

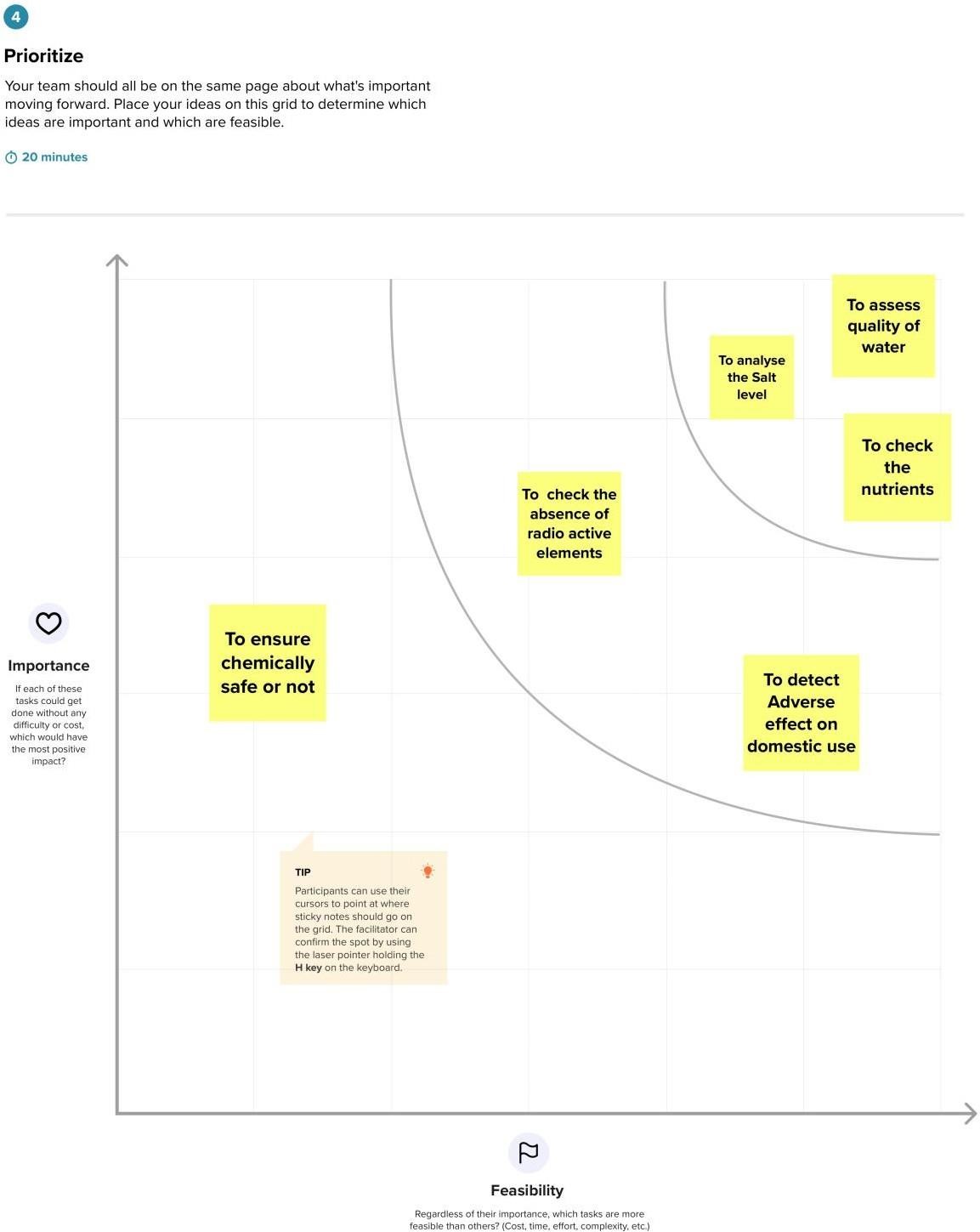
# Step-1: Team Gathering, Collaboration and Select the Problem Statement



**Step-2: Brainstorm, Idea Listing and Grouping**



# Step-3: Idea Prioritization



* 1. **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 categories 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  awareness 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. |

# PROBLEM SOLUTION FIT:

**s6. CUSTOMER CONSTRAINTS**

**C**

**1. CUSTOMER SEGMENT(S)**

**Define CS, fit into CC**

* + 1. To determine whether water contains appropriate minerals.

Who is your customer?

The customer involved in this are the people who seek for the quality of the

* + 1. Water is safe for drinking.
    2. Does it contain any

1. **AVAILABLE SOLUTIONS**

The solution is to have information on water quality parameters like pH level, Temperature, Turbidity, Minerals etc., to analyze the quality

**Explore AS, differentiate**

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

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

* + - Check the quality of water by gathering information based on many features and qualities in the chemical and physical composition of nature.

**9. PROBLEM ROOT CAUSE**

If there is no proper prediction of water quality in manufacturing sector, food production, drinking water, watering crops and many more, it can lead to great effect on the action we perform.

1. **BEHAVIOUR**

**BE**

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

The study attempts to assess the users water behavior using available resources, prevailing socio-economic conditions and personal aspects of users. The research work suggests the need for ensuring the water quality.

* Customers must have knowledge about the

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **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. | **10. YOUR SOLUTION**   1. It clusters the parameter like temperature, turbidity, hardness, pH level,   and dissolved minerals in the water.   1. It also evaluates the effort of substantial nutrients loads on overall   water quality.   1. Accurate model can be selected based on the outcome in the   model evaluation. | 1. **CHANNELS of BEHAVIOUR**    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.   * 1. **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. 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. |

# REQUIREMENT ANALYSIS:

* + 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. |

# NON-FUNCTIONAL REQUIREMENTS:

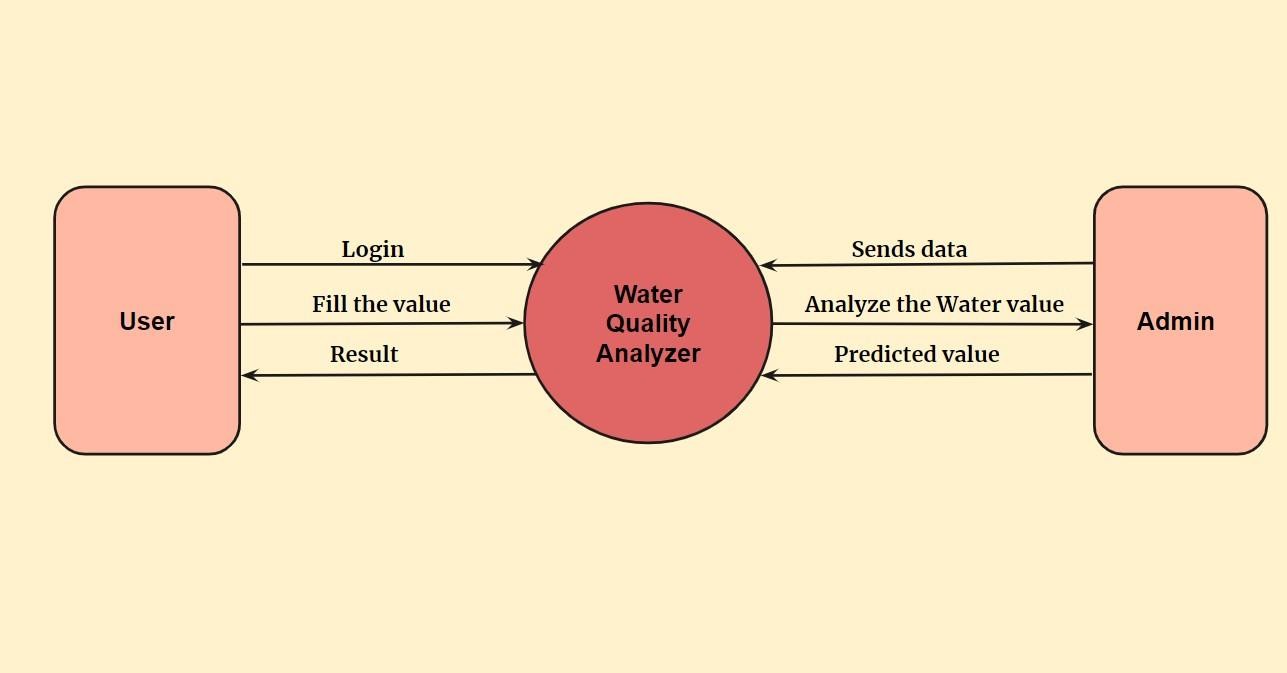
Following are the non-functional requirements of the proposed solution.

|  |  |  |
| --- | --- | --- |
| **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. |

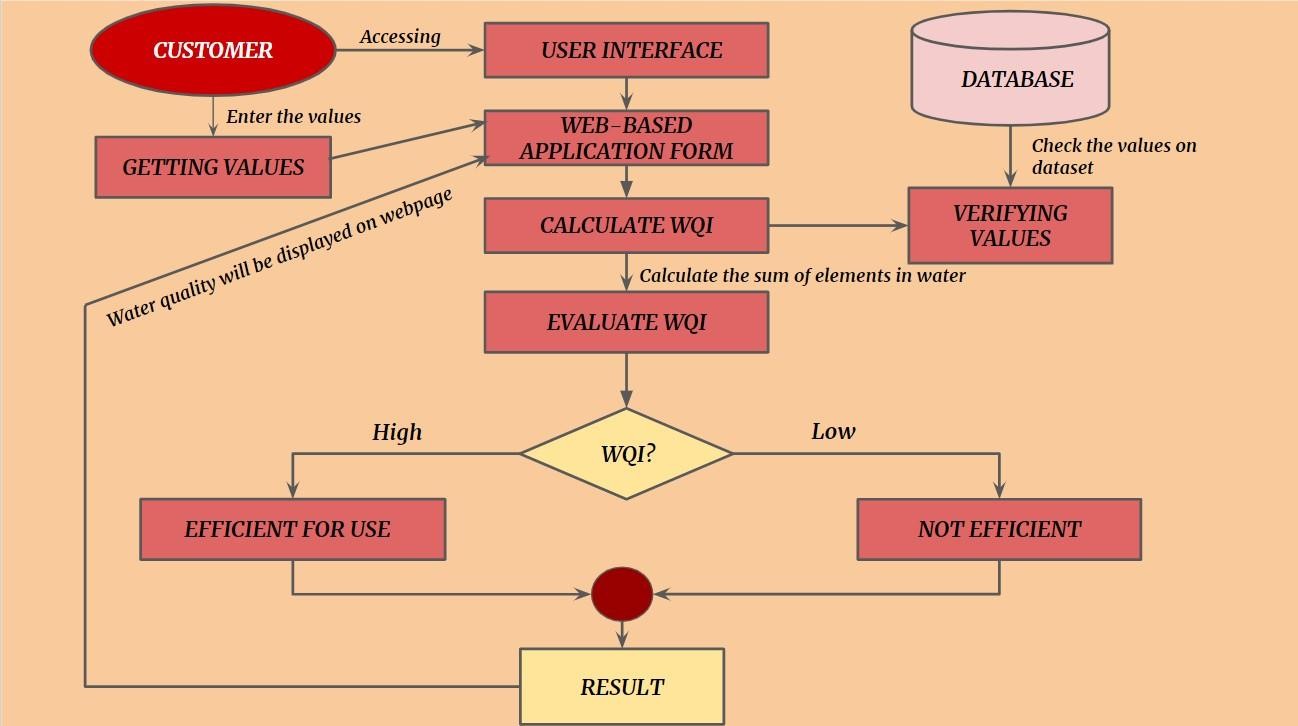
|  |  |  |
| --- | --- | --- |
| NFR-5 | **Scalability** | It provides an efficient outcome and has the ability to increase or decrease the performance of the system  Based on the datasets. |

# PROJECT DESIGN:

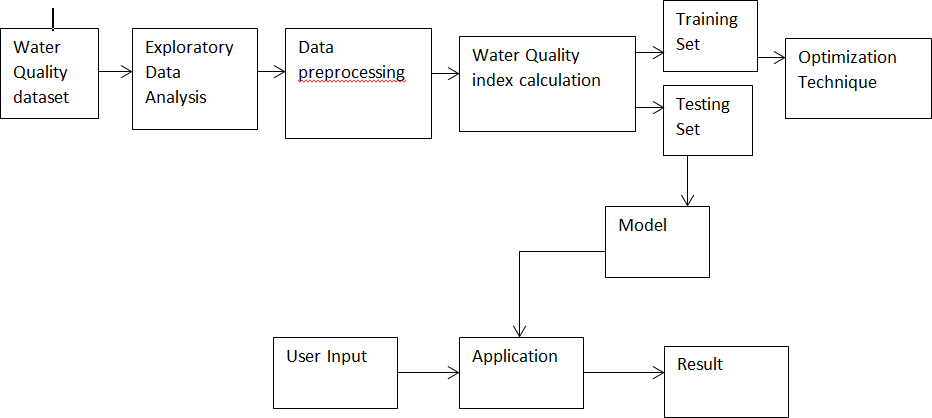
* + 1. **DATA FLOW DIAGRAMS: DFD LEVEL 0:**



# DFD LEVEL 1:



* + 1. **SOLUTION & TECHNICAL ARCHITECTURE:**



# 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  waterquality | High |
| Admin | 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 | 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 |

* 1. **PROJECT PLANNING AND SCHEDULING:**

# 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 | Keerthana M, Lathika R |
| 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 | Shanmugapriya M, Banupriya K |
| 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 | Keerthana M, Lathika R, Jeeva P |
| 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 dataset | 5 | High |

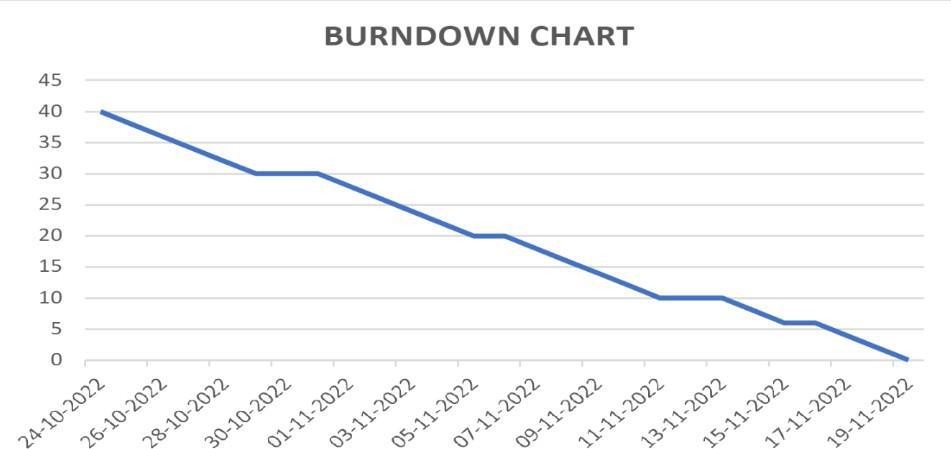
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Story Points** | **Priority** | **Team Members** |
|  |  |  | consists of. |  |  |  |
| Sprint- 3 | Model Deployment | USN-7 | As a user, I need to deploy the model and  need to find the results. | 10 | Medium | Keerthana M, Shanmugapriya M, Lathika R, Jeeva P, Banupriya K |
| 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 | Keerthana M, Shanmugapriya M, Lathika R, Jeeva P, Banupriya K |
| Sprint- 4 | Flask App | USM-9 | Flask app should be created to act as an interface between the frontend and  model | 20 | High | Keerthana M, Shanmugapriya M, Lathika R, Jeeva P, Banupriya K |

* 1. **SPRINT DELIVERY SCHEDULE:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Total Story Points** | **Durati on** | **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** | **Durati on** | **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:



* 1. **CODING & SOLUTIONING:**

# 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)

# 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)

# TESTING:

* 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.

# USER ACCEPTANCE TESTING:

## 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 |

## Test Case Analysis

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Section** | **Total Cases** | **Not Tested** | **Fa il** | **Pas s** |
| 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 |

# 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.

# 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:

* This 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.

# 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.

# FUTURE SCOPE:

The fundamental issue that machine learning models with unbalanced datasets encounter is overfitting. In the future, generalised 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.

# 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:

**https://github.com/IBM-EPBL/IBM-Project-44497-1660724900 PROJECT DEMOSTRATION LINK:**

**https://drive.google.com/drive/folders/1rC1jLKYYh-vLm4qe307dVTkP8ahqx03n**