PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTREPRENEURSHIP

TEAM ID : PNT2022TMID17967

PROJECT : EFFICIENT WATER QUALITY ANALYSIS AND PREDICTION USING MACHINE LEARNING

TEAM MEMBERS:

AISWERYA .T
ISWARYA .S
RITHIKA .K
VIGNESHWARI .M
GOKILA .V

TABLE OF CONTENTS

1. **INTRODUCTION**
   - 1.1 Project Overview
   - 1.2 Purpose
2. **LITERATURE SURVEY**
   - 2.1 Existing problem
   - 2.2 References
   - 2.3 Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
   - 3.1 Empathy Map Canvas
   - 3.2 Ideation & Brainstorming
   - 3.3 Proposed Solution
   - 3.4 Problem Solution fit
4. **REQUIREMENT ANALYSIS**
   - 4.1 Functional requirement
   - 4.2 Non-Functional requirements
5. **PROJECT DESIGN**
   - 5.1 Data Flow Diagrams
   - 5.2 Solution & Technical Architecture
   - 5.3 User Stories
6. **PROJECT PLANNING & SCHEDULING**
   - 6.1 Sprint Planning & Estimation
   - 6.2 Sprint Delivery Schedule
7. **CODING & SOLUTIONING**
8. **TESTING**
   - 8.1 Test Cases
   - 8.2 User Acceptance Testing
9. **RESULTS**
   - 9.1 Performance Metrics
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**
    - GitHub & Project Demo Link

1. INTRODUCTION

1.1 Project Overview

Water is the most important source for sustaining all kinds of life. Natural water resources and aquifers are being polluted due to indiscriminate urbanization and industrialization; as a result, it may be contaminated with physical, chemical, and biological impurities. As reported, 80% of the diseases are water borne diseases. Several criteria are used to measure the quality of water, including the quantity of salt (or salinity), bacteria levels, the percentage of dissolved oxygen or the number of particles suspended in the water (turbidity). Good water quality implies that harmful substances (pollutants) are absent from the water, and needed substances (oxygen, nutrients) are present. The traditional and common estimation of water quality has been Laboratory analysis which is time consuming and not very practical. This method can be processed efficiently by applying machine learning algorithms and big data tools. Machine learning (ML) is a field of inquiry devoted to understanding and building methods that 'learn', that is, methods that leverage data to improve performance on some set of tasks.

1.2 Purpose

The quality of water is a major concern for people living in urban areas. The quality of water serves as a powerful environmental determinant and a foundation for the prevention and control of waterborne diseases. However, predicting the urban water quality is a challenging task since the water quality varies in urban spaces non-linearly and depends on multiple factors, such as meteorology, water usage patterns, and land uses. The purpose of this project is to Predict Water Quality by considering all water quality standard indicators.

2. LITERATURE SURVEY

2.1 Existing problem

For testing the water quality, we must conduct lab tests on the water which is costly and time- consuming as well. So, in this paper, we propose an alternative approach using artificial intelligence to predict water quality. This method uses a significant and easily available water quality index which is set by the WHO (World Health Organization). The data taken in this paper is taken from the PCPB India which includes 3277 examples of the distinct wellspring. In this paper, WQI (Water Quality Index) is calculated using AI techniques. So, in future work, we can integrate this with an IoT based framework to study large datasets and to expand our study to a larger scale. By using that it can predict the water quality fast and more accurately than any other IoT framework. That IoT framework system uses some limits for the sensor to check the parameters like ph,

Temperature, Turbidity, and so on. And further after reading this parameter pass these readings to the Arduino microcontroller and ZigBee handset for further prediction.
Laboratory methods or DIY kits are used to measure the quality of the water. The most accurate findings are obtained via laboratory testing, which examines numerous parameters and takes the longest. Test strips and other at-home test kits offer quick results but have lower accuracies. Municipalities and bottled water firms are among the sources of water that frequently post their water quality data online for public use. The proposed system can be put into place by including basic parameters checking and can be expanded by incorporating various features related to water quality. The tested water quality parameters must meet standards set by their local governments, which are frequently influenced by international standards set by industry or water quality organizations like the World Health Organization (WHO). Since constant monitoring may significantly reduce water pollution, these kinds of quality monitoring systems will aid society in achieving a more secure future. Implementation will be far simpler with less functionalities.

2.2 References

- **Hadi Mohammed, Hoese Michel Tornyeviadzi, Razak Seidu, "Emulating process-based water quality modelling in water source reservoirs using machine learning", Journal of Hydrology Volume 609, June 2022, 127675.**

  Demonstrated the potential of machine learning model (Long Short-Term Memory(LSTM)) . A Hydro dynamic and water quality model was first calibrated to predict time series, profiles, and contours of water variables namely Eschericha coli(E.coli), faecal coliforms, zinc, and lead concentrations. The results obtained were combined with the input data to train a suite of LSTM models to emulate the results achieved with the process-based modeling.

- **Xudong Jia, "Detecting Water Quality Using KNN, Bayesian and Decision Tree", 2022 Asia Conference on Algorithms, Computing and Machine Learning (CACML)**

  Proposed a model using sklearn K Nearest Neighbor(KNN), Bayesian and decision tree.These models are used to classify water quality data. Comparison results show that decision tree algorithm performs best among the three supervised classification algorithms.

- **Umair Ahmed, Rafia Mumtaz, Hirra Anwar, Asad A.Shah, Rabia Irfan and Jose Garcia-Nieto, "Efficient Water Quality Prediction Using Supervised Machine Learning", MDPI 24 October 2019.**

  Proposed a methodology which employs four input parameters namely temperature, turbidity, pH and total dissolved solids.

- **Illa Iza Suhana Shamsuddin, Zalinda Othman, and Nor Samsiah Sani, "Water Quality Index Classification Based on Machine Learning:A Case from the Langat River Basin Model", Water 2022, 14,2939.**

  Proposed three machine learning models Artificial Neural Networks (ANN), Decision Trees (DT), and Support Vector Machines (SVM) to classify river water quality. Comparative performance analysis between the three models indicates that the SVM is the best model for predicting river water quality

- **Tianan Deng, Kwok-Wing Chau, Huan-Feng Duan, "Machine learning based marine water quality prediction for coastal hydro-environment management",Journal of Environmental Management Volume 284, 15 April 2021, 112051.**

  Proposed two different ML methods – Artificial Neural Networks (ANN) and Support Vector Machine (SVM) – are implemented and improved by introducing different hybrid learning algorithms for the simulations and comparative analysis.

- **Md Galal Uddin, Stephen Nash, Mir Talas Mahammad Diganta, Azizur Rahman, Agnieszka I. Olbert , "Robust machine learning algorithms for predicting coastal water quality index", Journal of Environmental Management Volume 321, 1 November 2022, 115923.**

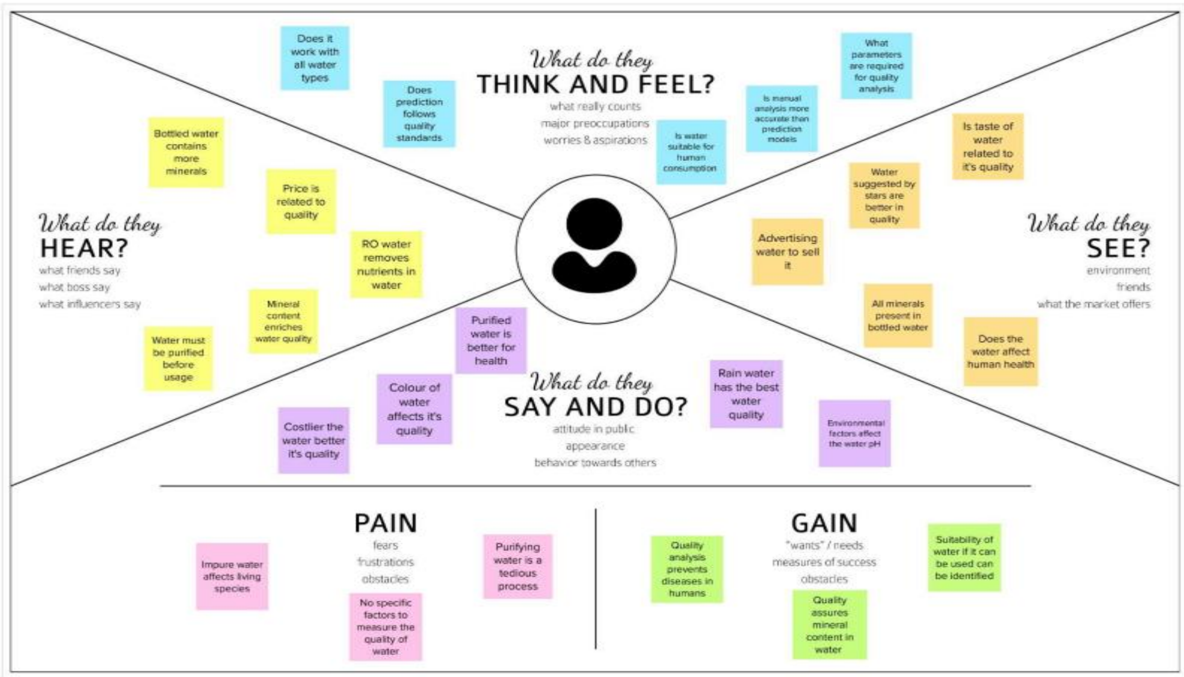
  Proposed eight commonly used algorithms, namely Random Forest (RF), Decision Tree (DT), K Nearest Neighbors (KNN), Extreme Gradient Boosting (XGB), Extra Tree (ExT), Support Vector Machine (SVM), Linear Regression (LR), and Gaussian Naïve Bayes (GNB). DT, ExT, and GXB models could be effective, robust and significantly reduce model uncertainty in predicting WQIs.

2.3 Problem Statement Definition

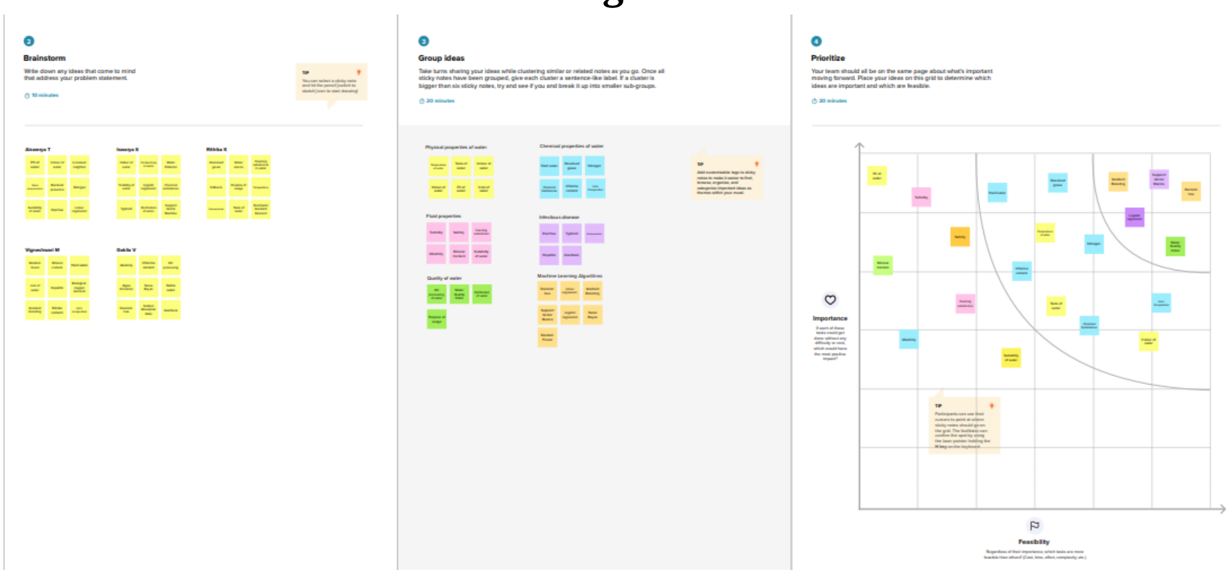
- Water makes up about 70% of the earth's surface and is one of the most important sources vital to sustaining life.
- Rapid urbanization and industrialization have led to a deterioration of water quality at an alarming rate, resulting in harrowing diseases.
- The quality of water is a major concern for people living in urban areas. The quality of water serves as a powerful environmental determinant and a foundation for the prevention and control of waterborne diseases.
- However, predicting the urban water quality is a challenging task since the water quality varies in urban spaces non-linearly and depends on multiple factors, such as meteorology, water usage patterns, and land uses.

3 IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming



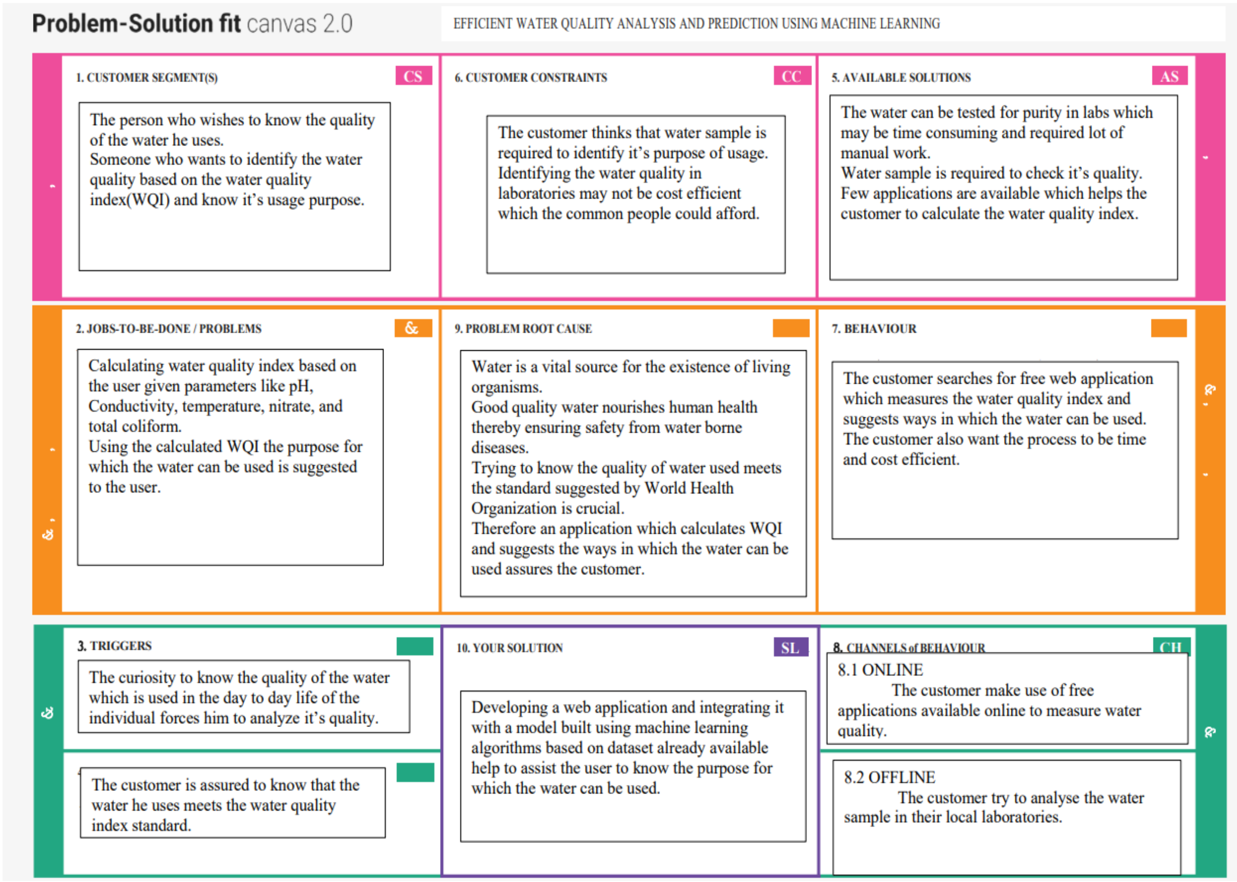
3.3 Proposed Solution

Proposed Solution Template:

The project team shall fill in the following information in the proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Predicting the urban water quality is a challenging task since the water quality varies in urban spaces non-linearly and depends on multiple factors, such as meteorology, water usage patterns, and land uses, so this project aims at building a Machine Learning (ML) model to Predict Water Quality by considering all water quality standard indicators.
2.	Idea / Solution description	A web application is designed to get water parameters and analyze them based on the model generated using machine learning and the quality of water is analyzed and displayed to the user.
3.	Novelty / Uniqueness	The model is built by Stacking classifier algorithms above meta classifier which helps to improve the prediction accuracy.
4.	Social Impact / Customer Satisfaction	The application built helps to ensure whether the water consumed by the customer, satisfies the requirements of the water quality index as provided by World Health Organization(WHO).
5.	Business Model (Revenue Model)	The application is to be used by common people. Therefore, the incorporation of advertisements is a source of revenue.
6.	Scalability of the Solution	The built application may also be used to train larger datasets, thereby enabling it to be used for industrial purposes.

3.4 Problem Solution fit


Problem-Solution fit canvas 2.0

EFFICIENT WATER QUALITY ANALYSIS AND PREDICTION USING MACHINE LEARNING

1. CUSTOMER SEGMENT(S) — CS

The person who wishes to know the quality of the water he uses.
Someone who wants to identify the water quality based on the water quality index(WQI) and know it's usage purpose.

6. CUSTOMER CONSTRAINTS — CC

The customer thinks that water sample is required to identify it's purpose of usage.
Identifying the water quality in laboratories may not be cost efficient which the common people could afford.

5. AVAILABLE SOLUTIONS — AS

The water can be tested for purity in labs which may be time consuming and required lot of manual work.
Water sample is required to check it's quality.
Few applications are available which helps the customer to calculate the water quality index.

2. JOBS-TO-BE-DONE / PROBLEMS — J&P

Calculating water quality index based on the user given parameters like pH, Conductivity, temperature, nitrate, and total coliform.
Using the calculated WQI the purpose for which the water can be used is suggested to the user.

9. PROBLEM ROOT CAUSE — RC

Water is a vital source for the existence of living organisms.
Good quality water nourishes human health thereby ensuring safety from water borne diseases.
Trying to know the quality of water used meets the standard suggested by World Health Organization is crucial.
Therefore an application which calculates WQI and suggests the ways in which the water can be used assures the customer.

7. BEHAVIOUR — BE

The customer searches for free web application which measures the water quality index and suggests ways in which the water can be used.
The customer also want the process to be time and cost efficient.

3. TRIGGERS — TR

The curiosity to know the quality of the water which is used in the day to day life of the individual forces him to analyze it's quality.

4. EMOTIONS: BEFORE / AFTER — EM

The customer is assured to know that the water he uses meets the water quality index standard.

10. YOUR SOLUTION — SL

Developing a web application and integrating it with a model built using machine learning algorithms based on dataset already available help to assist the user to know the purpose for which the water can be used.

8. CHANNELS of BEHAVIOUR — CH

8.1 ONLINE
The customer make use of free applications available online to measure water quality.

8.2 OFFLINE
The customer try to analyse the water sample in their local laboratories.

4 REQUIREMENT ANALYSIS

4.1 Functional requirement

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	**User Input**	Users are required to give chemical components of their water, which they need to test. The chemical components such as Temperature, pH, Dissolved Oxygen, Coliform, Biochemical oxygen demand, Conductivity, and Nitratenan details.
FR-2	**Display output**	Based on the range of water quality index available, given water sample values are classified and predicted the final result as (excellent, good, marginal, poor).

4.2 Non-Functional requirements

Non-functional Requirements:

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	System is such that it stands up to the customers expectation. When an application is usable, users can easily navigate its interface. The native user can also use the system effectively, without any difficulties. Users can easily determine what a feature is and what it can do.
NFR-2	**Security**	Various forms of questions are asked for calculating water quality index(wqi) and are securely stored in database.
NFR-3	**Reliability**	Consider recording the number of critical failures a system experiences during testing to check its reliability. Tracking the time between critical failures can help you understand the reliability of a system. If the number of failures is low, it means that the system operates properly.
NFR-4	**Performance**	User can interact with the system by providing some of details which is required for calculating the index. Response of the operation is good and fast.

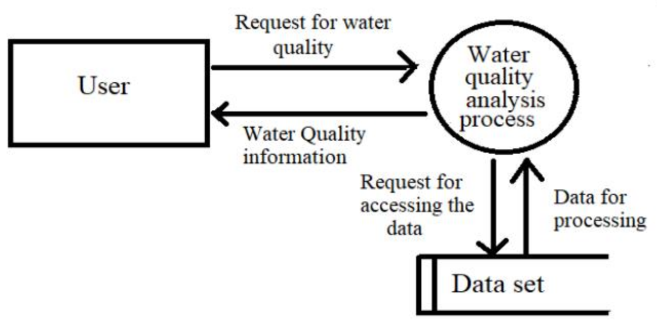
5 PROJECT DESIGN

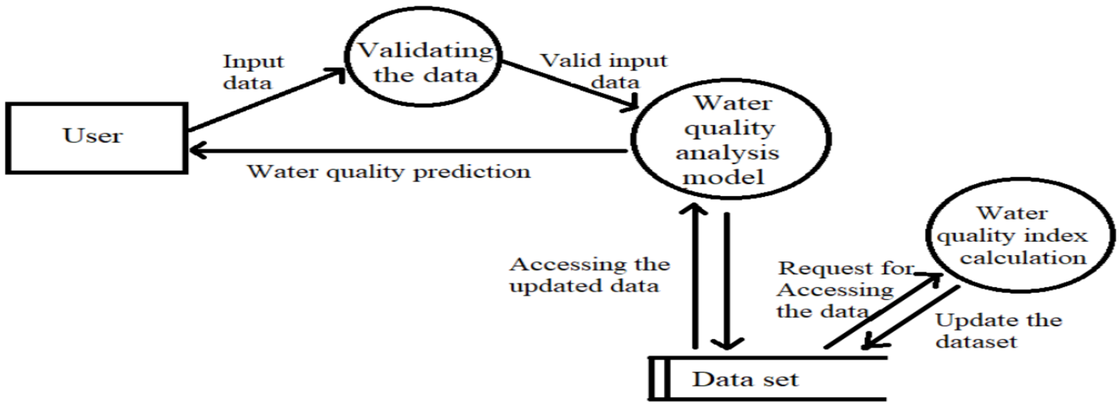
5.1 Data Flow Diagrams

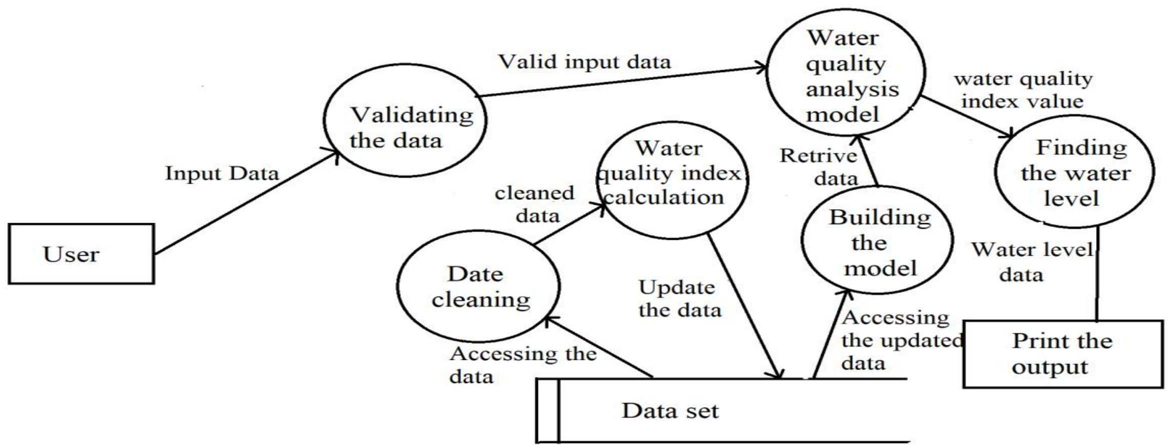
Data Flow Diagrams:

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

DFD Level 0

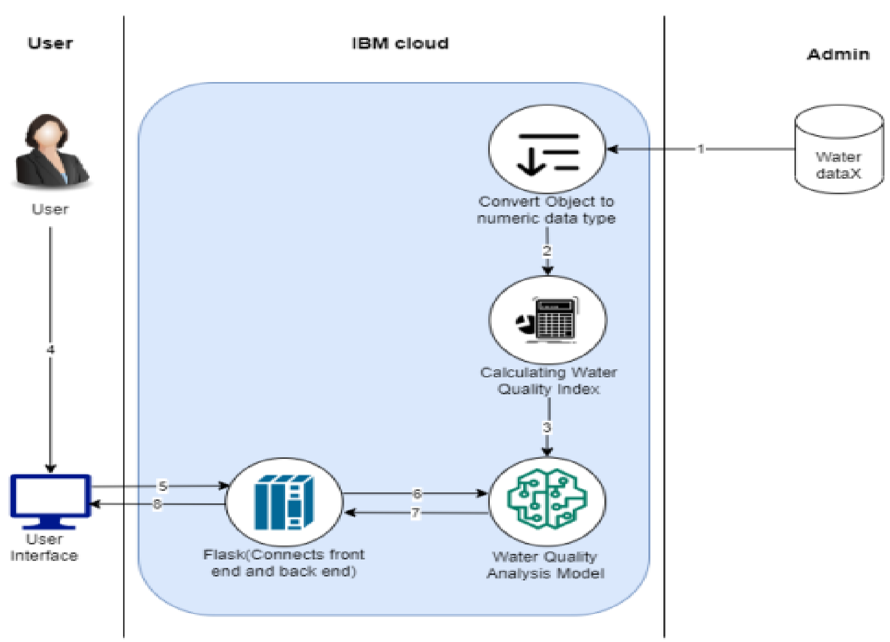




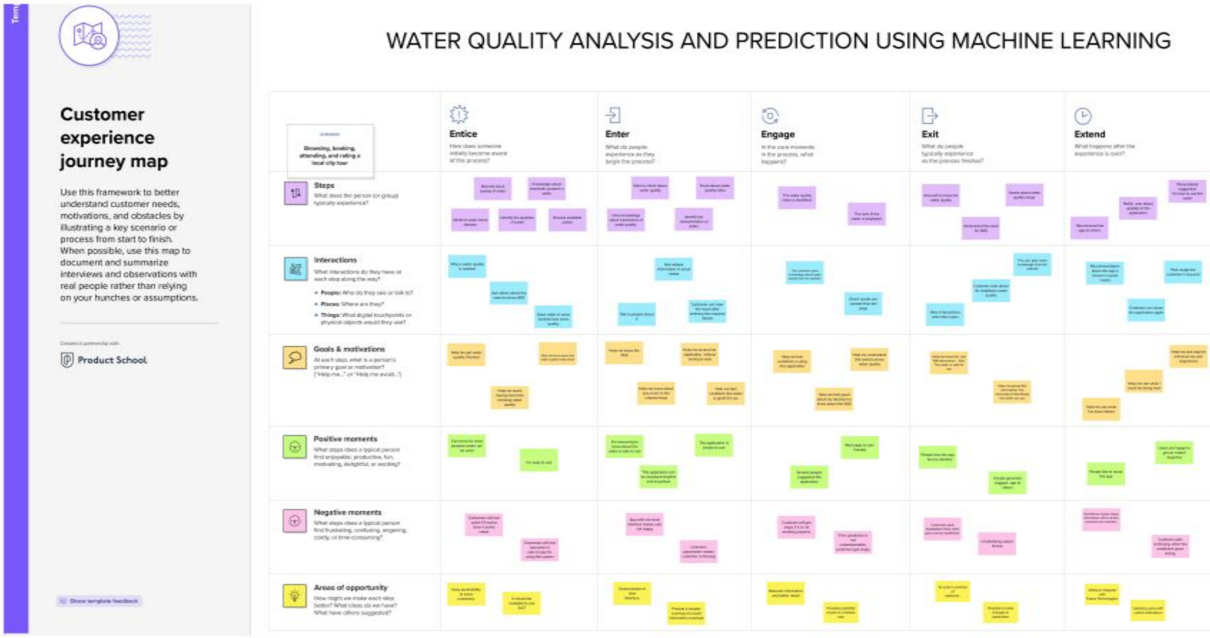


5.2 Solution & Technical Architecture

Efficient Water Quality Analysis and Prediction using Machine Learning



5.3 User Stories



6 PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Pre-processing	USN-1	The water quality dataset is pre-processed like the missing values are replaced with mean of the corresponding attributes.	5	Low	1
Sprint-2	Water Quality Index Calculation	USN-3	The water quality index is calculated using the standard formula. This formula requires some features such as pH, DO, coliform, conductivity, BOD, nitratine. Finally, the WQI is updated in the dataset.	10	Medium	2
Sprint-3	Building Predictive Model	USN-4	Using the updated dataset, a random forest classifier is used to create a prediction model with high accuracy.	20	High	3
Sprint-4	Finding Water Quality Level	USN-5	Finding the water quality level based on the predicted output of the model.	15	High	1
Sprint-4	User Interface (HTML Page)	USN-6	Designing the user interface to get input from user and show the WQI and water quality level.	15	High	2
Sprint-4	Connecting with Cloud	USN-7	The completed module is shifted into cloud.	15	High	3

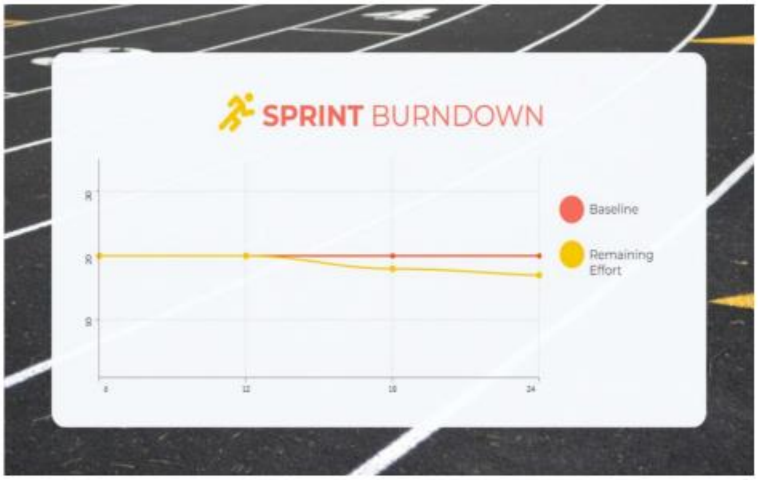
6.2 Sprint Delivery Schedule

Project Tracker, Velocity & Burndown Chart: (4 Marks)

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	28 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	04 Oct 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	11 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	15 Nov 2022

Burndown Chart:

	A	B	C	D	E
1	Days	**6**	**12**	**18**	**24**
2	**Total story points**	20	20	20	20
3	**Story points complet**	20	20	18	17



7 CODING & SOLUTIONING

Importing the libraries

```
In [2]: import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        import warnings
```

Reading the dataset

In [4]:
```python
data=pd.read_csv(r'D:/PRIEE/water_dataX.csv',encoding='latin1')
data
```

Out[4]:

	STATION CODE	LOCATIONS	STATE	Temp	D.O. (mg/l)	PH	CONDUCTIVITY (µmhos/cm)	B.O.D. (mg/l)	NITRATENAN N+ NITRITENANN (mg/l)	C (MP
0	1393	DAMANGANGA AT D/S OF MADHUBAN, DAMAN	DAMAN & DIU	30.6	6.7	7.5	203	NAN	0.1	
1	1399	ZUARI AT D/S OF PT. WHERE KUMBARJRIA CANAL JOI...	GOA	29.8	5.7	7.2	189	2	0.2	
2	1475	ZUARI AT PANCHAWADI	GOA	29.5	6.3	6.9	179	1.7	0.1	
3	3181	RIVER ZUARI AT BORIM BRIDGE	GOA	29.7	5.8	6.9	64	3.8	0.5	
4	3182	RIVER ZUARI AT MARCAIM JETTY	GOA	29.5	5.8	7.3	83	1.9	0.4	
...	...	...	...	...	...	...	...	...	...	

Analyze the data

In [5]:
```python
data.head()
```

Out[5]:

	STATION CODE	LOCATIONS	STATE	Temp	D.O. (mg/l)	PH	CONDUCTIVITY (µmhos/cm)	B.O.D. (mg/l)	NITRATENAN N+ NITRITENANN (mg/l)	FECAL COLIFORM (MPN/100ml)	TOTAL COLIFORM (MPN/100ml)Mean	year
0	1393	DAMANGANGA AT D/S OF MADHUBAN, DAMAN	DAMAN & DIU	30.6	6.7	7.5	203	NAN	0.1	11	27	2014
1	1399	ZUARI AT D/S OF PT. WHERE KUMBARJRIA CANAL JOI...	GOA	29.8	5.7	7.2	189	2	0.2	4953	8391	2014
2	1475	ZUARI AT PANCHAWADI	GOA	29.5	6.3	6.9	179	1.7	0.1	3243	5330	2014
3	3181	RIVER ZUARI AT BORIM BRIDGE	GOA	29.7	5.8	6.9	64	3.8	0.5	5382	8443	2014
4	3182	RIVER ZUARI AT MARCAIM JETTY	GOA	29.5	5.8	7.3	83	1.9	0.4	3428	5500	2014

In [6]:
```python
data.describe()
```

Out[6]:

	year
count	1991.000000
mean	2010.038172
std	3.057333
min	2003.000000
25%	2008.000000
50%	2011.000000
75%	2013.000000
max	2014.000000

```
In [7]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1991 entries, 0 to 1990
Data columns (total 12 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   STATION CODE                        1991 non-null   object
 1   LOCATIONS                           1991 non-null   object
 2   STATE                               1991 non-null   object
 3   Temp                                1991 non-null   object
 4   D.O. (mg/l)                         1991 non-null   object
 5   PH                                  1991 non-null   object
 6   CONDUCTIVITY (µmhos/cm)             1991 non-null   object
 7   B.O.D. (mg/l)                       1991 non-null   object
 8   NITRATENAN N+ NITRITENANN (mg/l)    1991 non-null   object
 9   FECAL COLIFORM (MPN/100ml)          1991 non-null   object
 10  TOTAL COLIFORM (MPN/100ml)Mean      1991 non-null   object
 11  year                                1991 non-null   int64
dtypes: int64(1), object(11)
memory usage: 186.8+ KB
```

```
In [8]: data.shape

Out[8]: (1991, 12)
```

Handling missing values 1

```
In [9]: data.isnull().any()

Out[9]: STATION CODE                        False
        LOCATIONS                           False
        STATE                               False
        Temp                                False
        D.O. (mg/l)                         False
        PH                                  False
        CONDUCTIVITY (µmhos/cm)             False
        B.O.D. (mg/l)                       False
        NITRATENAN N+ NITRITENANN (mg/l)    False
        FECAL COLIFORM (MPN/100ml)          False
        TOTAL COLIFORM (MPN/100ml)Mean      False
        year                                False
        dtype: bool
```

Handling missing values 2

```
In [10]: data.dtypes
```

```
Out[10]: STATION CODE                          object
LOCATIONS                             object
STATE                                 object
Temp                                  object
D.O. (mg/l)                           object
PH                                    object
CONDUCTIVITY (µmhos/cm)               object
B.O.D. (mg/l)                         object
NITRATENAN N+ NITRITENANN (mg/l)      object
FECAL COLIFORM (MPN/100ml)            object
TOTAL COLIFORM (MPN/100ml)Mean        object
year                                   int64
dtype: object
```

```
In [11]: 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
```

```
Out[11]: STATION CODE                           object
LOCATIONS                              object
STATE                                  object
Temp                                  float64
D.O. (mg/l)                           float64
PH                                    float64
CONDUCTIVITY (µmhos/cm)               float64
B.O.D. (mg/l)                         float64
NITRATENAN N+ NITRITENANN (mg/l)      float64
FECAL COLIFORM (MPN/100ml)             object
TOTAL COLIFORM (MPN/100ml)Mean        float64
year                                    int64
dtype: object
```

```
n [13]: data.isnull().sum()
```

```
ut[13]: STATION CODE                          0
LOCATIONS                             0
STATE                                 0
Temp                                 92
D.O. (mg/l)                          31
PH                                    8
CONDUCTIVITY (µmhos/cm)              25
B.O.D. (mg/l)                        43
NITRATENAN N+ NITRITENANN (mg/l)    225
FECAL COLIFORM (MPN/100ml)            0
TOTAL COLIFORM (MPN/100ml)Mean      132
year                                  0
dtype: int64
```

Handling missing values 3

```
In [14]: 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['B.O.D. (mg/l)'].fillna(data['B.O.D. (mg/l)'].mean(),inplace=True)
         data['CONDUCTIVITY (µmhos/cm)'].fillna(data['CONDUCTIVITY (µmhos/cm)'].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)
```

```
In [15]: data.drop(['FECAL COLIFORM (MPN/100ml)'],axis=1,inplace=True)
```

```
In [17]: 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 Calculation

```
In [18]: 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)))))
```

```
In [19]: 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)))))
```

```
In [21]: 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)))))
```

```
In [22]: data['nbdo']=data.do.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)))))
```

```
In [23]: 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)))))
```

Water Quality Index Calculation 2

In [23]:
```python
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)))))
```

In [24]:
```python
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)))))
```

Water Quality Index Calculation 3

In [25]:
```python
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
```

Out[25]:

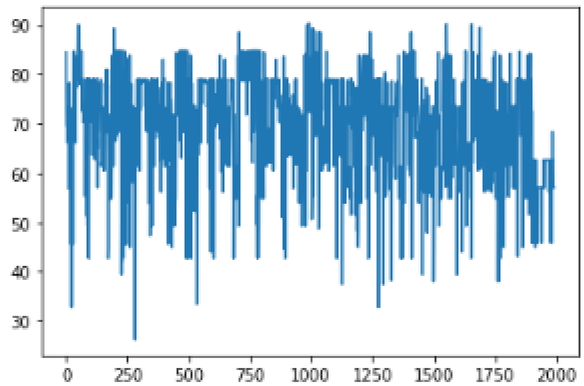
	station	location	state	Temp	do	ph	co	bod	na	tc	...	nbdo	nec	nna	wph	wdo	wbdo	wec	wna	wco
0	1393	DAMANGANGA AT D/S OF MADHUBAN, DAMAN	DAMAN & DIU	30.600000	6.7	7.5	203.0	6.940049	0.100000	27.0	...	60	60	100	16.5	28.10	14.04	0.54	2.8	22.48
1	1399	ZUARI AT D/S OF PT. WHERE KUMBARJRIA CANAL JOI...	GOA	29.800000	5.7	7.2	189.0	2.000000	0.200000	8391.0	...	80	60	100	16.5	22.48	18.72	0.54	2.8	11.24
2	1475	ZUARI AT PANCHAWADI	GOA	29.500000	6.3	6.9	179.0	1.700000	0.100000	5330.0	...	60	60	100	13.2	28.10	14.04	0.54	2.8	11.24
3	3181	RIVER ZUARI AT BORIM BRIDGE	GOA	29.700000	5.8	6.9	64.0	3.800000	0.500000	8443.0	...	80	100	100	13.2	22.48	18.72	0.90	2.8	11.24
4	3182	RIVER ZUARI AT MARCAIM JETTY	GOA	29.500000	5.8	7.3	83.0	1.900000	0.400000	5500.0	...	80	80	100	16.5	22.48	18.72	0.72	2.8	11.24
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1986	1330	TAMBIRAPARANI AT ARUMUGANERI, TAMILNADU	NAN	26.209814	7.9	738.0	7.2	2.700000	0.518000	202.0	...	60	100	100	0.0	28.10	14.04	0.90	2.8	16.86
1987	1450	PALAR AT VANIYAMBADI WATER SUPPLY HEAD WORK, T...	NAN	29.000000	7.5	585.0	6.3	2.600000	0.155000	315.0	...	60	100	100	0.0	28.10	14.04	0.90	2.8	16.86
1988	1403	GUMTI AT U/S SOUTH TRIPURA,TRIPURA	NAN	28.000000	7.6	98.0	6.2	1.200000	1.623079	570.0	...	60	100	100	0.0	28.10	14.04	0.90	2.8	11.24

In [26]:
```python
average=data.groupby('year')['wqi'].mean()
average.head()
```

Out[26]:
```
year
2003    58.900455
2004    54.270000
2005    69.043361
2006    68.363429
2007    68.773000
Name: wqi, dtype: float64
```

Data Visualization:

```
In [32]: plt.plot(data['wqi'])
Out[32]: [<matplotlib.lines.Line2D at 0x212ac4a3640>]
```



Splitting dependent and independent columns:

```
In [35]: x=data.iloc[:,0:7].values
         y=data.iloc[:,7:].values

In [36]: x.shape
Out[36]: (1991, 7)

In [37]: y.shape
Out[37]: (1991, 1)
```

Splitting data into train and test:

```
In [38]: from sklearn.model_selection import train_test_split
         X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.3,random_state=10)
         Y_train1d=np.ravel(Y_train)
```

Building Predictive Model:

Random Forest Regression:

```
[ ] from sklearn.ensemble import RandomForestRegressor
    regressor1 = RandomForestRegressor(n_estimators = 100, random_state = 0)
    regressor1.fit(X_train,Y_train1d)

    RandomForestRegressor(random_state=0)

    Y_pred1 = regressor1.predict(X_test)

[ ] result1 = regressor1.score(X_test, Y_test)
    print("Accuracy - test set: %.2f%%" % (result1*100.0))

    Accuracy - test set: 97.98%

[ ] from sklearn import metrics
    print('MAE:',metrics.mean_absolute_error(Y_test,Y_pred1))
    print('MSE:',metrics.mean_squared_error(Y_test,Y_pred1))
    print('RMSE:',np.sqrt(metrics.mean_squared_error(Y_test,Y_pred1)))

    MAE: 0.47559899665555844
    MSE: 2.012301841939794
    RMSE: 1.4185562526526023
```

Linear Regression:

```
[ ] from sklearn.linear_model import LinearRegression
    regressor = LinearRegression()
    regressor.fit(X_train,Y_train)

    LinearRegression()

[ ] Y_pred = regressor.predict(X_test)

[ ] result = regressor.score(X_test, Y_test)
    print("Accuracy - test set: %.2f%%" % (result))

    Accuracy - test set: 0.28%

[ ] 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)))

    MAE: 6.837882116994348
    MSE: 72.12112265184501
    RMSE: 8.492415595803411
```

Decision Tree Regression:

```
[ ] from sklearn.tree import DecisionTreeRegressor
    regressor2 = DecisionTreeRegressor(random_state = 0)
    regressor2.fit(X_train, Y_train)

    DecisionTreeRegressor(random_state=0)

    Y_pred2 = regressor2.predict(X_test)

[ ] result2 = regressor2.score(X_test, Y_test)
    print("Accuracy - test set: %.2f%%" % (result2*100.0))

    Accuracy - test set: 96.39%

[ ] from sklearn import metrics
    print('MAE:',metrics.mean_absolute_error(Y_test,Y_pred2))
    print('MSE:',metrics.mean_squared_error(Y_test,Y_pred2))
    print('RMSE:',np.sqrt(metrics.mean_squared_error(Y_test,Y_pred2)))

    MAE: 0.4438127090301296
    MSE: 3.5931063545150494
    RMSE: 1.895549090505189
```

Finding Water Quality Level:

Finding the water quality level based on the predicted output of the model.

```
> WaterPrediction > sample.py > login
import joblib
import numpy as np
import flask
#from flask_core import CORS
from flask import 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 = "8HrhmjnJlOEE0j_lz0tsFmxjw0_xq882lNskkHHnPe9Z"
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.Flask(__name__,static_url_path='')
#CORS(app)
#model = pickle.load(open('wqi1.pkl','rb'))
@app.route('/',methods=['GET'])
def home() :
    return render_template("quality.html")
```

```python
@app.route('/login',methods =['POST'])
def login():
    year=int(request.form["year"])
    do=float(request.form["do"])
    ph=float(request.form["ph"])
    co=float(request.form["co"])
    bod=float(request.form["bod"])
    na=float(request.form["na"])
    tc=float(request.form["tc"])
    total=[[year,do,ph,co,bod,na,tc]]

    payload_scoring = {"input_data": [{"field": [['year','do','ph','co','bod','na','tc']], "values": total}]}

    response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/1d5e319e-68d7-47a8-be14-9edd2f04dbcb/predictions?version=2022-10-28', json=paylo
    headers={'Authorization':'Bearer'+mltoken})
    print(response_scoring)
    predictions=response_scoring.json()
    y_pred=predictions['predictions'][0]['values'][0][0]
    print("final prediction",y_pred)

    if(y_pred>=95 and y_pred<=100):
        return render_template("quality.html",showcase='Excellent, The predicted value is'+ str(y_pred))
    elif(y_pred>=89 and y_pred<=94):
        return render_template("quality.html",showcase='Very Good, The predicted value is'+ str(y_pred))
    elif(y_pred>=80 and y_pred<=88):
        return render_template("quality.html",showcase='Good, The predicted value is'+ str(y_pred))
    elif(y_pred>=65 and y_pred<=79):
        return render_template("quality.html",showcase='Fair, The predicted value is'+ str(y_pred))
    elif(y_pred>=45 and y_pred<=64):
        return render_template("quality.html",showcase='Marginal, The predicted value is'+ str(y_pred))
    else:
        return render_template("quality.html",showcase='Poor, The predicted value is'+ str(y_pred))

if __name__ == '__main__':
    app.run()
```

User Interface (HTML Page):

Designing the user interface to get input from user and show the WQI and water quality level.

```html
<html>
<head>
  <link rel = "stylesheet" href="{{url_for('static',filename='css/style.css')}}">
</head>
<body>

<div class="bg-img">

<center><h1 style="color:rgb(182, 0, 73)">Water Quality Prediction</h1></center>
<image src="{{url_for('static',filename = 'image/OIP5.jpg')}}" ></image>
  <form action="/login" method = "post" class="container">
    <center><input type="text" name="year" placeholder="Enter year"/>
        <input type="text" name="do" placeholder="Enter D.O"/>
        <input type="text" name="ph" placeholder="Enter PH"/>
        <input type="text" name="co" placeholder="Enter Conductivity"/>
        <input type="text" name="bod" placeholder="Enter B.O.D"/>
        <input type="text" name="na" placeholder="Enter Nitratenen"/>
        <input type="text" name="tc" placeholder="Enter Total Coliform"/>
        <button type="submit" class="btn">Predict</button>
        <div class="bor"><center><b><font color="red" size=5>{{showcase}}</font></b></center></div>
        </center>
  </form>
</div>
</body>
</html>
```

Connecting with Cloud:

The completed module is shifted into cloud.

IBM Deployment

```
!pip install -U ibm-watson-machine-learning
```

```
Requirement already satisfied: ibm-watson-machine-learning in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.0.256)
Requirement already satisfied: pandas<1.5.0,>=0.24.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (1.3
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (1.26.7)
Requirement already satisfied: ibm-cos-sdk==2.11.* in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (2.11.
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (2022.9.24)
Requirement already satisfied: packaging in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (21.3)
Requirement already satisfied: importlib-metadata in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (4.8.2)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (2.26.0)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (0.3.3)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (0.8.9)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm
```

In [52]:
```
from ibm_watson_machine_learning import APIClient
import json
```

Authenticate and set space

In [55]:
```
wml_credentials={
    "apikey":"8HrhmjnJlOEE0j_lz0tsFmxjw0_xq882lNskkHHnPe9Z",
    "url":"https://us-south.ml.cloud.ibm.com"
}
```

In [56]:
```
wml_client=APIClient(wml_credentials)
```

In [58]:
```
wml_client.spaces.list()
```

```
Note: 'limit' is not provided. Only first 50 records will be displayed if the number of records exceed 50
------------------------------------  -------------  ------------------------
ID                                    NAME           CREATED
d71a7b7f-9611-47f9-a953-bf5ea7b2c931  Water Quality  2022-10-28T06:08:51.915Z
------------------------------------  -------------  ------------------------
```

In [59]:
```
SPACE_ID="d71a7b7f-9611-47f9-a953-bf5ea7b2c931"
```

In [60]:
```
wml_client.set_default_space(SPACE_ID)
```

Out[60]: 'SUCCESS'

Save and Deploy the model

In [64]:
```
import sklearn
sklearn.__version__
```

Out[64]: '1.0.2'

In [65]:
```
MODEL_NAME='Water Quuality'
DEPLOYMENT_NAME='Water Quality'
DEMO_MODEL=regressor1
```

In [66]:
```
software_spec_uid=wml_client.software_specifications.get_id_by_name('runtime-22.1-py3.9')
```

In [70]:
```
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
}
```

In [73]:
```
model_details = wml_client.repository.store_model(
    model=DEMO_MODEL,
    meta_props=model_props,
    training_data=X_train,
    training_target=Y_train1d
)
```

In [74]:
```
model_details
```

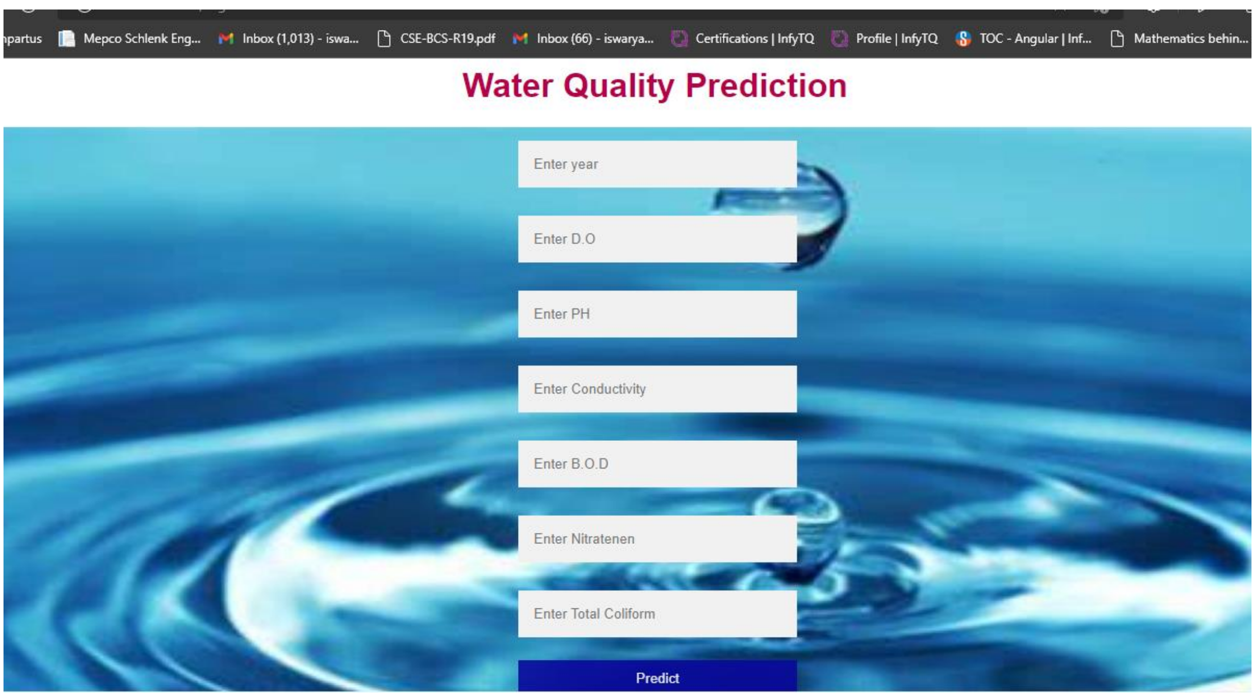
```
In [76]: deployment_props={
             wml_client.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NAME,
             wml_client.deployments.ConfigurationMetaNames.ONLINE:{}
         }
```

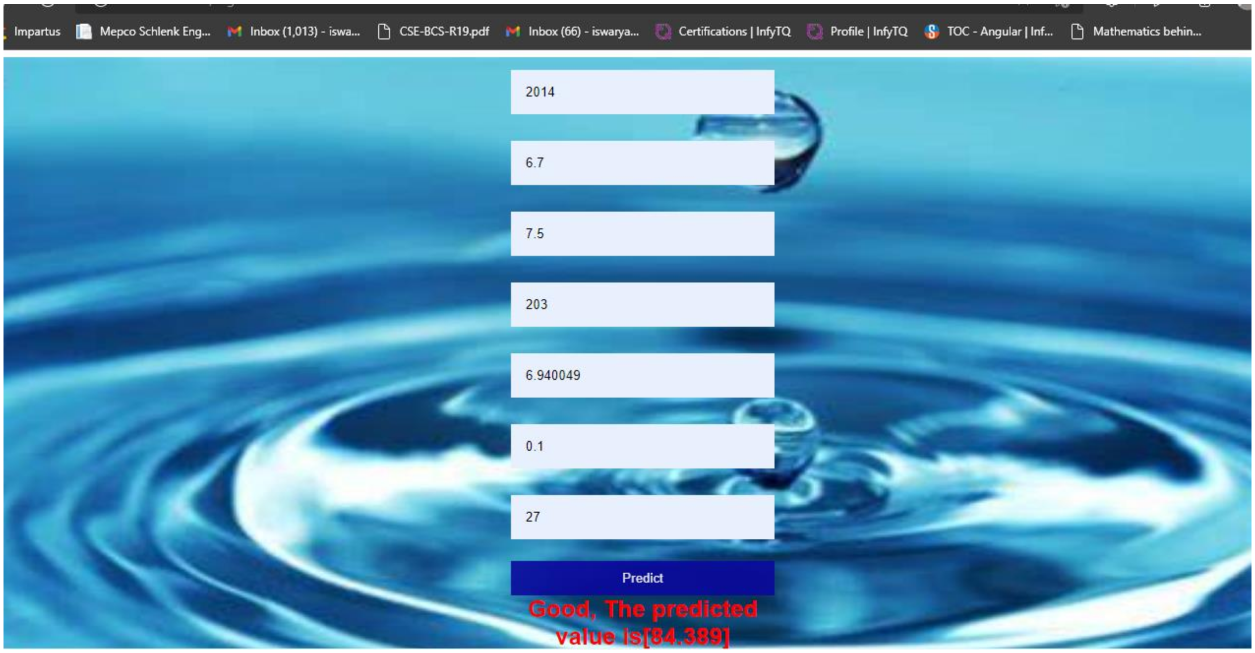
```
In [78]: deployment=wml_client.deployments.create(
             artifact_uid=model_id,
             meta_props=deployment_props
         )
```

```
#######################################################################################

Synchronous deployment creation for uid: '7a2f0798-3c58-4252-8684-bf47e99c33dd' started

#######################################################################################


initializing
Note: online_url is deprecated and will be removed in a future release. Use serving_urls instead.

ready


------------------------------------------------------------------------------------------------
Successfully finished deployment creation, deployment_uid='1d5e319e-68d7-47a8-be14-9edd2f04dbcb'
------------------------------------------------------------------------------------------------
```

8 TESTING

8.1 Test Cases





8.2 User Acceptance Testing

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the Water Quality Analysis and Prediction Using Machine Learning project at the time of the release to User Acceptance Testing (UAT).

2. 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	8	3	2	1	14
Duplicate	1	0	2	0	3
External	1	2	0	2	5
Fixed	9	2	2	15	28
Not Reproduced	0	1	1	0	2
Skipped	0	0	1	1	2
Won't Fix	0	0	0	0	0
Totals	19	8	8	19	54

3. 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	8	0	0	8
Client Application	50	0	0	50
Security	1	0	0	1
Outsource Shipping	2	0	0	2
Exception Reporting	6	0	0	6
Final Report Output	3	0	0	3
Version Control	2	0	0	2

9 RESULTS

9.1 Performance Metrics

Model Performance Testing:

S.No.	Parameter	Values	Screenshot
1.	Metrics	**Regression Model:** **Linear Regression:** MAE - 6.837882116994348, MSE - 72.12112265184501, RMSE - 8.492415595803411, R2 score - 0.27583949127715235 **Decision Tree:** MAE - 0.4438127090301296, MSE - 3.5931063545150494, RMSE - 1.895549090505189, R2 score - 0.9639220019058543 **Random Forest:** MAE - 0.47559899665555844, MSE - 2.012301841939794, RMSE - 1.4185562526526023, R2 score - 0.9797946915968346	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))) print('r2_score:',metrics.r2_score(Y_test,Y_pred)) MAE: 6.837882116994348 MSE: 72.12112265184501 RMSE: 8.492415595803411 r2_score: 0.27583949127715235 from sklearn import metrics print('MAE:',metrics.mean_absolute_error(Y_test,Y_pred2)) print('MSE:',metrics.mean_squared_error(Y_test,Y_pred2)) print('RMSE:',np.sqrt(metrics.mean_squared_error(Y_test,Y_pred2))) print('r2_score:',metrics.r2_score(Y_test,Y_pred2)) MAE: 0.4438127090301296 MSE: 3.5931063545150494 RMSE: 1.895549090505189 r2_score: 0.9639220019058543 from sklearn import metrics print('MAE:',metrics.mean_absolute_error(Y_test,Y_pred1)) print('MSE:',metrics.mean_squared_error(Y_test,Y_pred1)) print('RMSE:',np.sqrt(metrics.mean_squared_error(Y_test,Y_pred1))) print('r2_score:',metrics.r2_score(Y_test,Y_pred1)) MAE: 0.47559899665555844 MSE: 2.012301841939794 RMSE: 1.4185562526526023 r2_score: 0.9797946915968346
2.	Tune the Model	Hyperparameter Tuning	All the features are required for WQI calculation.So hyperparameter tuning is not applicable.

10 ADVANTAGES & DISADVANTAGES

ADVANTAGES

- water quality prediction helps in controlling Water Pollution
- To predict the water is safe or not
- Predicting potable water quality for water management and water pollution prevention.
- Water quality prediction convey the health of ecosystems, safety of human contact, extend of water pollution and condition of drinking water

DISADVANTAGES

- Training necessary Somewhat difficult to manage over time and with large data sets
- Requires manual operation to submit data, some configuration required
- Costly, usually only feasible under Exchange Network grants Technical expertise and network server required
- Requires manual operation to submit data Cannot respond to data queries from other nodes, and therefore cannot interact with the Exchange Network Technical expertise and network server required

11CONCLUSION

The water quality is monitored and managed effectively because of the importance of drinking water. Water has a direct effect on our health. This adds more reason to test the quality of drinking water. Several boards of committees and protocols are established to check the quality of water. The assessment of water quality differs from origin to origin. Using machine learning techniques, the water quality is tested without any regular laboratory tests. By using Random Forest algorithm, we can evaluate the quality of water based on the attributes such as pH, BOD, DO, minerals, and coliform in the water. This model can be used for predicting the quality of water and can monitor the potability of the water. This model acts as a prototype for the IoT sensors and can make the model even more efficient to predict the quality of water and potability of water. Data cleaning and processing, missing value analysis, exploratory analysis, and model creation and evaluation were all part of the analytical process. The best accuracy on a public test set will be discovered, as will the highest accuracy score. This application can assist in determining the current state of water quality.

12 FUTURE SCOPE

In future works, we propose integrating the findings of this research in a large-scale IoT-based online monitoring system using only the sensors of the required parameters. The tested algorithms would predict the water quality immediately based on the real-time data fed from the IoT system.The proposed IoT system would employ the parameter sensors of pH, turbidity, temperature and TDS for parameter readings and communicate those readings using an Arduino microcontroller. It would identify poor quality water before it is released for consumption and alert concerned authorities. It will hopefully result in curtailment of people consuming poor quality water and consequently de-escalate harrowing diseases like typhoid and diarrhea. In this regard, the application of a prescriptive analysis from the expected values would lead to future facilities to support decision and policy makers.

13APPENDIX

GitHub Link:

https://github.com/IBM-EPBL/IBM-Project-2530-1658473480

Demo Link:

https://drive.google.com/file/d/1Vgj9RBfOHB_jM7FKcLmM3m5EHh1vP6v9/view?ts=63791654