# 1.INTRODUCTION

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 fulfil 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 behaviour of each individual variable independently is not possible in practise to accurately describe water quality on a spatial or temporal basis. The more challenging method is to combine the values of a group of physical and chemical variables into a single value . A quality value function (usually linear) represented the equivalence between the variable and its quality level was included in the index for each variable. These functions were created using direct measurements of a substance's concentration or the value of a physical variable derived from water sample studies. The major goal of this research is to examine how machine learning algorithms may be used to predict water quality..

## 1.1 PROJECT OVERVIEW

With the rapid increase in the volume of data on the aquatic environment, machine learning has become an important tool for data analysis, classification, and prediction. Unlike traditional models used in water-related research, data-driven models based on machine learning can efficiently solve more complex nonlinear problems. In water environment research, models and conclusions derived from machine learning have been applied to the construction, monitoring, simulation, evaluation, and optimization of various water treatment and management systems. Additionally, machine learning can provide solutions for water pollution control , water quality improvement, and wateshed ecosystem security management. In this review, we describe the cases in which machine learning algorithms have been applied to evaluate the water quality in different water environments, such as surface water, groundwater, drinking water, sewage, and seawater. Furthermore, we propose possible future applications of machine learning approaches to water environments.

## 1.2 PURPOSE

Groundwater is an important source of drinking water. As such, ensuring the safety of groundwater is essential to human health. Machine learning has extensive application prospects in groundwater analysis, including the assessment and prediction of groundwater quality and pollution sources.

The objective of water quality monitoring is to obtain quantitative information on the physical, chemical, and biological characteristics of water via statistical sampling.

## 2. ITERATURE SURVEY

A literature survey or a literature review in a project report is that section which shows the various analyses and research made in the field of your interest and the results already published, taking into account the various parameters of the project and the extent of the project.

It is the most important part of your report as it gives you a direction in the area of your research. It helps you set a goal for your analysis - thus giving you your problem statement.

When you write a literature review in respect of your project, you have to write the researches made by various analysts - their methodology (which is basically their abstract) and the conclusions they have arrived at. You should also give an account of how this research has influenced your thesis.Descriptive papers may or may not contain reviews, but analytical papers will contain reviews. A literature review must contain at least 5 - 7 published researches in your field of interest.

## 2.1 EXISITNG PROBLEM

The main problem lies here. For testing the water quality we have to 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 Organisation). 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 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.

## 2.2 REFERENCES


[1] Ritabrata Roy, An Introduction to Water Quality Analysis personalization ,2019.

[2] Sai Sreeja Kurra , Sambangi Geethika Naidu , Sravani Chowdala , Sree Chithra Yellanki , Dr. B. Esther Sunanda Water Quality Prediction Using Machine Learning , 2022

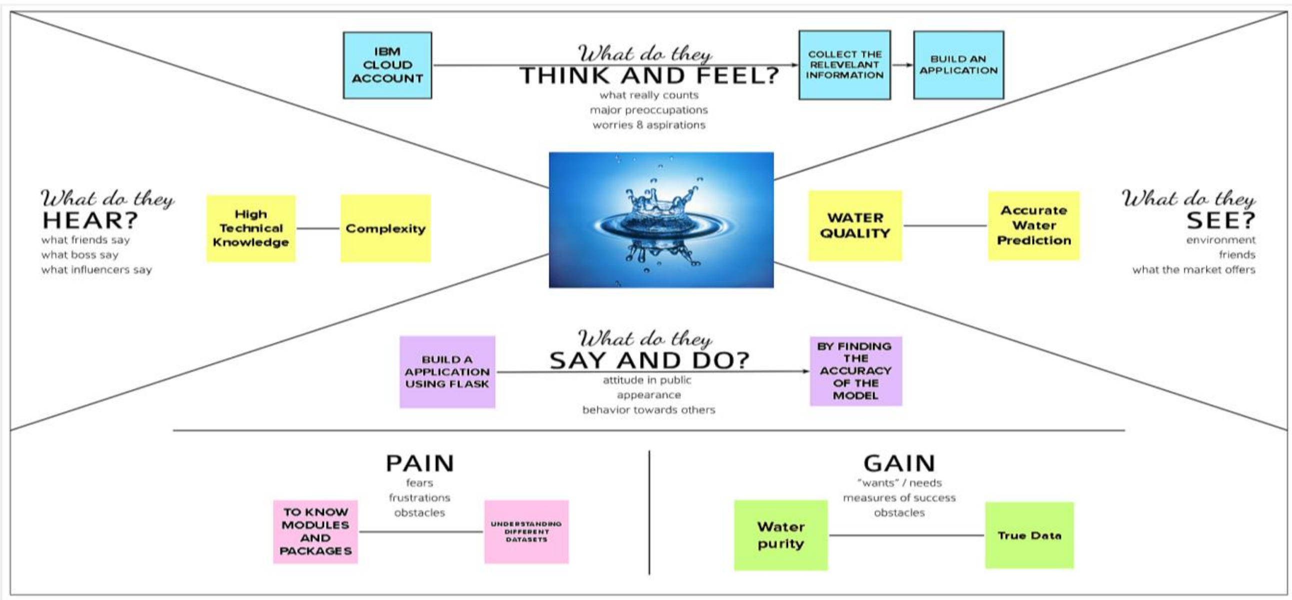
[3] Umair Ahmed , Rafia Mumtaz , Hirra Anwar , Asad A. Shah , Rabia Irfan and José García- v Nieto Efficient Water Quality Prediction Using Supervised Machine Learning ,2019.


## 2.3 PROBLEM STATEMENT DEFINITION

Water is considered as a vital resource that affects various aspects of human health and lives. 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, so this project aims at building a Machine Learning (ML) model to Predict Water Quality by considering all water quality standard indicators.

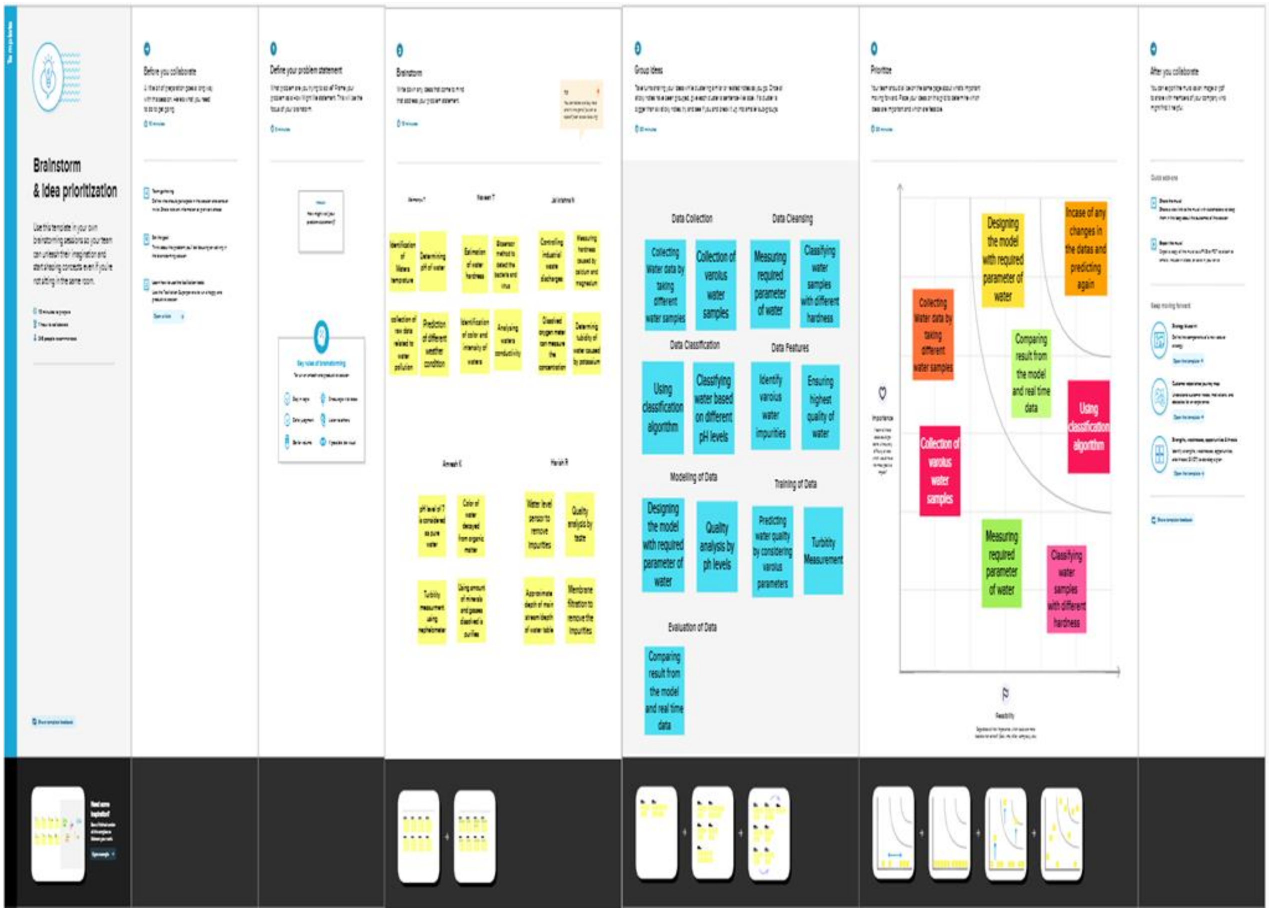
## 3. IDEATION & PROPOSED SOLUTION

### 3.1 EMPATHY MAP CANVAS




### 3.2 IDEATION AND BRAINSTORMING

Ideation refers to the whole creative process of coming up with and communicating new ideas. It can take many different forms, from coming up with a totally new idea to combining multiple existing ideas to create a new process or organizational system. Ideation is similar to a practice known as brainstorming.




### 3.3 PROPOSED SOLUTION

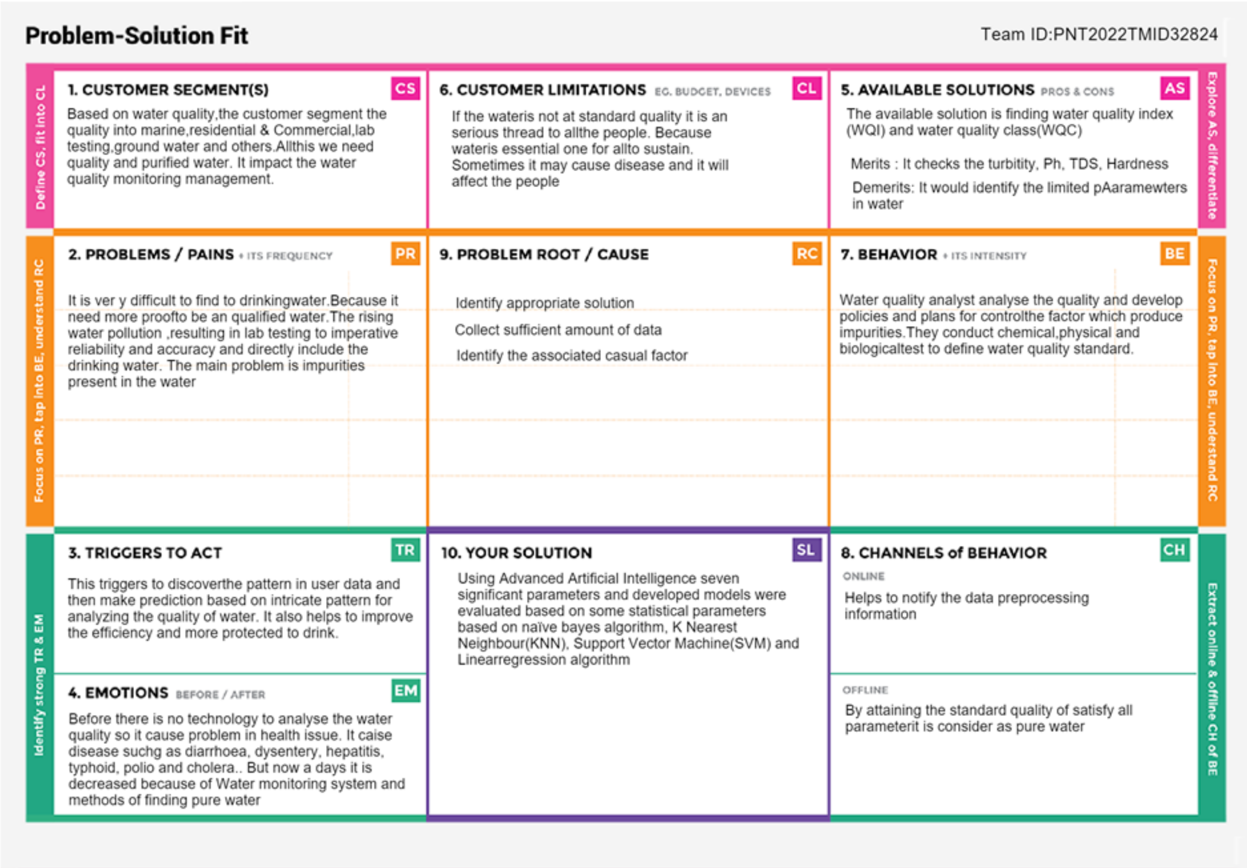
Proposed solution should relate the current situation to a desired result and describe the benefits that will accrue when the desired result is achieved. So, begin your proposed solution by briefly describing this desired result.

| S.No. | Parameter | Description |
|---|---|---|
| 1. | Problem Statement (Problem to be solved) | Efficient Water Quality Analysisand Prediction using Machine Learning. |

| | | |
|---|---|---|
| 2. | Idea / Solution description | For the WQI prediction, artificial neural network models, namely nonlinear autoregressive neural network (NARNET) and long short-term memory (LSTM) deep learning algorithm, have been developed. In addition, three machine learning algorithms, namely, support vector machine (SVM), K- nearest neighbour (K-NN), and Naive Bayes, have been used for the WQC forecasting.<br><br>The used dataset has 7 significant parameters, and the developed modelswere evaluated based on some statistical<br><br>parameters |
| 3. | Novelty / Uniqueness | In previous they find water quality with help of WQI and WQC. Now the solution is find with help of advanced artificial intelligence and it include seven<br><br>parameters |
| 4. | Social Impact / Customer Satisfaction | During the last years, water quality has been threatened by various pollutants. Therefore, modelling and predicting water quality have become very important in controlling water pollution. In this work, advanced artificial intelligence (AI) algorithms are developed to predict water quality index (WQI) and water quality classification (WQC).This is the |

| | | |
|---|---|---|
| | | impact of this statement. |

## 3.4 PROBLEM SOLUTION FIT




# 4. REQUIREMENT ANALYSIS

## 4.1 FUNCTIONAL REQUIREMENT

Functional requirements are product features or functions that developers must implement to enable users to accomplish their tasks. So, it’s important to make them clear both for the development team and the stakeholders. Generally, functional requirements describe system behavior under specific conditions.

| FR No. | Functional Requirement (Epic) | Sub Requirement (Story / Sub-Task) |
|---|---|---|
| FR-1 | User registration | Registration through GmailCreate an account<br>Follow the instructions |
| FR-2 | User Confirmation | Confirmation via Emailand it is predicted by water levelsensor |
| FR-3 | Interface sensor | Interface sensor andWater level sensorproduces thedetection of clean drinking water |
| FR-4 | Accessing datasets | Datasets are collected by data preprocessing method. |
| FR-5 | Mobile application | The efficient of water quality is analyzed, the mobileapplication is notused . |

## 4.2 NON-FUNCTIONAL REQUIREMENT

Nonfunctional requirements, not related to the system functionality, rather define how the system should perform.

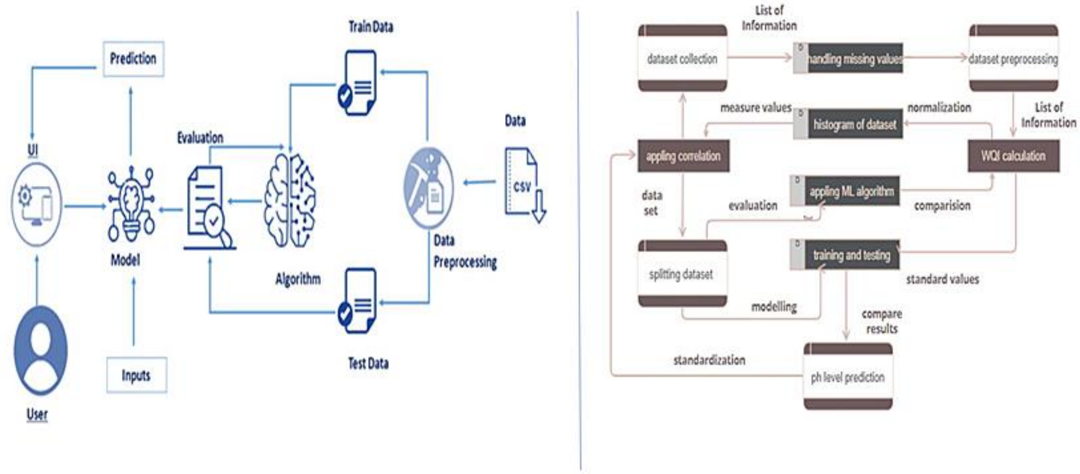
| FR No. | Non-Functional Requirement | Description |
|---|---|---|
| NFR-1 | **Usability** | This project is useful for all human being by predictingapurified water. |

| | | |
|---|---|---|
| NFR-2 | **Security** | We have designed thisproject to securethe peoplefromdrinking the impurity water. |
| NFR-3 | **Reliability** | This project will help everyone in protecting their health. Accurate water quality prediction is the basisof water environment management and is of great significance forwater environment protection. |
| NFR-4 | **Performance** | This system uses different sensors for monitoring the water quality by determine pH,Turbidity,conductivity and temperature. The<br><br>data preprocessing access the dataset. With the useof this we predict the quality water. |
| NFR-5 | **Availability** | By developing and deploying resilient hardware andsoftware we can analyze the drinking water . |

# 5. PROJECT DESIGN

## 5.1 Data Flow Diagrams

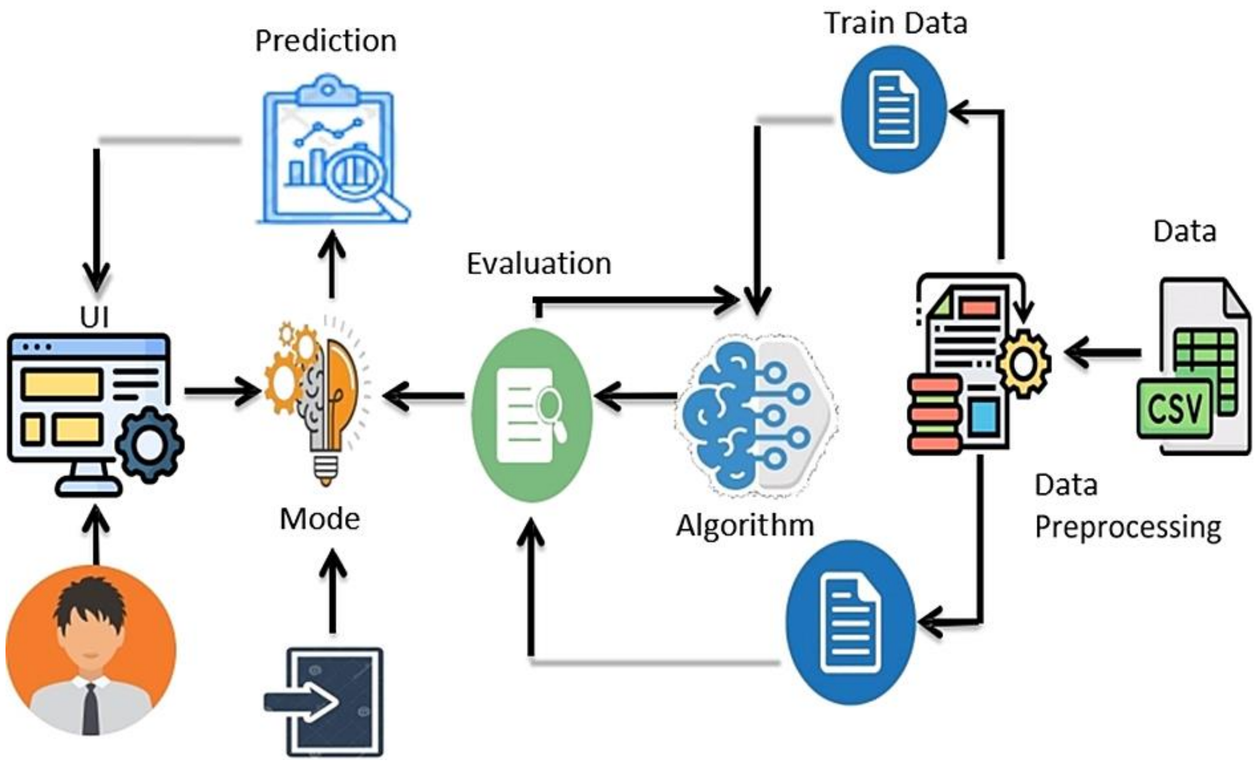
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.




## 5.2 SOLUTION AND TECHNINCAL ARCHITECTURE

- Createand log in to the IBM WatsonStudio.
- Uploadthe Jupiter notebook and start running it.
- Download the dataset.
- Pandasis used for reading the data and performing initial data exploration.
- Matplotlib and Seaborn are used for visualizing the data.
- Scikit-Learn is used for model development.

Use the IBM Watson Machine Learning feature to deploy and access the model to generate employee attrition classification




### 5.3 USER STORIES

A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.

| User Type | Functional Requirement(Epic) | User Story Number | User Story / Task | Acceptance criteria | Priority | Release |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| Developer | Data Preparation | USN -1 | Collecting waterdataset and pre-processing it | Handle missing values, outliers, null values and so on | High | Sprint-1 |
| | Model Building | USN -2 | Create a ML modelto predict waterquality | Fitting data in perfect model | Medium | Sprint-1 |
| | Model Evaluation | USN -3 | Calculate the performance, errorrateand complexity of ML model | Above 80% performance | Medium | Sprint-1 |
| | Model Deployment | USN -5 | Using flaskand deploy modelfinally inIBM cloud using IBMstorage and WatsonStudio | Working in a proper manner | Medium | Sprint-2 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Customer | Registration | USN -5 | As a user, I can registerfor the applicationby entering my email, password, and confirmingmy password | I can access my account /dashboard | Medium | Sprint-3 |
| | Confirmation | USN -6 | As a user, I will receive confirmationemail once I have registered for the application | I can receiveconfirmation email &click confirm | Low | Sprint-3 |
| | Login | USN -7 | As a user, I can loginto the applicationby entering email &password | I am accessing my account | Medium | Sprint-3 |
| | Dashboard | USN -8 | As a user, I can use the application by entering water data | Iam accessing my dashboard | High | Sprint-4 |

# 6. PROJECT PLANNING & SCHEDULING

## 6.1 SPRINT PLANNING AND ESTIMATION

| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Story Points | Priority | Team Members |
|---|---|---|---|---|---|---|
| Sprint-1 | Data Collection | USN-1 | Collecting downloading dataset for preprocessing | 10 | High | Naveen Abimanyu |
| | | USN-2 | Data pre-processing formats the dataand handles the missing data in the dataset | 10 | High | Naveen Abimanyu |
| Sprint-2 | Model Building | USN-3 | Calculate theWater Quality Index(WQI) using specified formula for every parameter. | 20 | Medium | Harish,Jai Krishna, Amresh |
| | | USN-4 | Splitting the data intotraining and testingdata set from the entire dataset. | 10 | High | Jai Krishna, Amresh |
| Sprint-3 | Training and Testing | USN-5 | Training the model usingRandom Forest Regression algorithm and testing the performance of the model | 10 | Medium | Naveen Abimanyu |
| Sprint-4 | Implementation of Web | USN-6 | Implementing the webpage for collecting the data | 10 | High | Naveen,Abimanyu,Harish,Jai krishna,Amresh |
| | | USN-7 | Deploying the model using IBM Cloudand IBMWatsonStudio | 10 | Medium | Naveen AbimanyuHarish ,Jai Krishna, Amresh |

# 6.2 SPRINT DELIVERY SCHEDULE

| Sprint | Total Story Points | Duration | Sprint Start Date | Sprint End Date (Planned) | Story Points Completed (as on Planned | Sprint Release Date(Actual) |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | **EndDate)** | |
| 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-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 | 19 Nov 2022 |

# 7. CODING & SOLUTIONING

## 7.1 FEATURE 1

### Data collection

Data mining techniques require domain knowledge in order to generate predictions. For water quality applications, it is vital to understand how various water quality parameters influence water quality. This information can come from a domain expert or historical data collections. For the forecasting task, two types of data sets were used: a carefully created huge synthetic data set and an available real data set

| STATION C | LOCATION | STATE | Temp | D.O. (mg/l | PH | CONDUCT | B.O.D. (mg | NITRATEN | FECAL COL | TOTAL CO | year |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1393 | DAMANGA | DAMAN & | 30.6 | 6.7 | 7.5 | 203 | NAN | 0.1 | 11 | 27 | 2014 |
| 1399 | ZUARI AT I | GOA | 29.8 | 5.7 | 7.2 | 189 | 2 | 0.2 | 4953 | 8391 | 2014 |
| 1475 | ZUARI AT I | GOA | 29.5 | 6.3 | 6.9 | 179 | 1.7 | 0.1 | 3243 | 5330 | 2014 |
| 3181 | RIVER ZUA | GOA | 29.7 | 5.8 | 6.9 | 64 | 3.8 | 0.5 | 5382 | 8443 | 2014 |
| 3182 | RIVER ZUA | GOA | 29.5 | 5.8 | 7.3 | 83 | 1.9 | 0.4 | 3428 | 5500 | 2014 |
| 1400 | MANDOVI | GOA | 30 | 5.5 | 7.4 | 81 | 1.5 | 0.1 | 2853 | 4049 | 2014 |
| 1476 | MANDOVI | GOA | 29.2 | 6.1 | 6.7 | 308 | 1.4 | 0.3 | 3355 | 5672 | 2014 |
| 3185 | RIVER MAI | GOA | 29.6 | 6.4 | 6.7 | 414 | 1 | 0.2 | 6073 | 9423 | 2014 |
| 3186 | RIVER MAI | GOA | 30 | 6.4 | 7.6 | 305 | 2.2 | 0.1 | 3478 | 4990 | 2014 |
| 3187 | RIVER MAI | GOA | 30.1 | 6.3 | 7.6 | 77 | 2.3 | 0.1 | 2606 | 4301 | 2014 |
| 1543 | RIVER KAL | GOA | 27.8 | 7.1 | 7.1 | 176 | 1.2 | 0.1 | 4573 | 7817 | 2014 |
| 1548 | RIVER ASS | GOA | 27.9 | 6.7 | 6.4 | 93 | 1.4 | 0.1 | 2147 | 3433 | 2014 |
| 2276 | RIVER BICI | GOA | 29.3 | 7.4 | 6.8 | 121 | 1.7 | 0.4 | 11633 | 18125 | 2014 |
| 2275 | RIVER CHA | GOA | 29.2 | 6.9 | 7 | 620 | 1.1 | 0.1 | 3500 | 6300 | 2014 |
| 3189 | RIVER CHA | GOA | 30 | 6 | 7.5 | 72 | 1.6 | 0.2 | 4995 | 9517 | 2014 |
| 1546 | RIVER KHA | GOA | 29 | 7.3 | 7 | 247 | 1.5 | 0.2 | 1095 | 2453 | 2014 |
| 2270 | RIVER KHA | GOA | 29.1 | 7.3 | 7 | 188 | 1 | 0.1 | 1286 | 3048 | 2014 |
| 2272 | RIVER KUS | GOA | 28.7 | 7 | 6.9 | 224 | 1.2 | 0.3 | 3896 | 6742 | 2014 |
| 1545 | RIVER MAI | GOA | 28.7 | 7.3 | 6.7 | 144 | 1.5 | 0.1 | 1940 | 3052 | 2014 |
| 2274 | RIVER MAI | GOA | 29.5 | 5.3 | 6.8 | 319 | 1.8 | 0.3 | 6458 | 10250 | 2014 |
| 2271 | RIVER SAL | GOA | 29 | 6.3 | 6.4 | 79 | 1.6 | 1.4 | 7592 | 12842 | 2014 |
| 2273 | RIVER SAL | GOA | 29.4 | 5.4 | 7.6 | 39 | 1.4 | 0.1 | 3176 | 6367 | 2014 |
| 3183 | RIVER SAL | GOA | 28.3 | 2.2 | 6.5 | 322 | 4.7 | 1.2 | 11210 | 14920 | 2014 |
| 3184 | RIVER SAL | GOA | 30.1 | 5.2 | 7.1 | 192 | 2.6 | 0.3 | 5073 | 8925 | 2014 |
| 3190 | RIVER SIN | GOA | 30.3 | 5.6 | 7.5 | 282 | 1.8 | 0.1 | 3205 | 5082 | 2014 |
| 3191 | RIVER SIN | GOA | 30.5 | 5.5 | 7.4 | 275 | 1.5 | 0.1 | 4698 | 8625 | 2014 |
| 1547 | RIVER TAL | GOA | 29.1 | 7.3 | 6.7 | 55 | 1.4 | 0.1 | 2638 | 4003 | 2014 |
| 3188 | RIVER TIR | GOA | 30.1 | 6.5 | 7.5 | 415 | 2 | 0.1 | 864 | 1538 | 2014 |

## Data Preprocessing

The processing phase is very important in data analysis to improve the data quality. In this phase, the WQI has been calculated from the most significant parameters of the dataset. Then, water samples have been classified on the basis of the WQI values. For obtaining superior accuracy, the -score method has been used as a data normalization technique.

```
In [34]: #Feature Scaling
         from sklearn.preprocessing import StandardScaler
         sc = StandardScaler()
         x_train = sc.fit_transform(x_train)
         x_test = sc.transform(x_test)
```

## Water Quality Index Calculation

To measure water quality, WQI is used to be calculated using various parameters that significantly affect WQ [40–42]. In this study, a published dataset is considered to test the proposed model, and seven significant water quality parameters are included. The WQI has been calculated using the following formula:

$$\text{WQI} = \frac{\sum_{i=1}^{N} q_i \times w_i}{\sum_{i=1}^{N} w_i},$$

where: is the total number of parameters included in the WQI calculations is the quality rating scale for each parameter  calculated by equation (2) below, and  is the unit weight for each parameter calculated by equation (3).

$$q_i = 100 \times \left( \frac{V_i - V_{\text{Ideal}}}{S_i - V_{\text{Ideal}}} \right),$$

where: is the measured value of parameter  in the tested water samples is the ideal value of parameter  in pure water (0 for all parameters except  and ), and  is the recommended standard value of parameter  .

$$w_i = \frac{K}{S_i},$$

Performance Measures Results True Positives (TP) are when the model predicts the positive class properly. True Negatives (TN) is one of the components of a confusion matrix designed to demonstrate how classification algorithms work. Positive outcomes that the model predicted incorrectly are known as False Positives (FP). False Negatives (FN) are negative outcomes that the model predicts negative class. Accuracy is the most basic and intuitive performance metric, consisting of the ratio of successfully predicted observations to total observations.

Accuracy = TP+TN/(TP+FP+FN+TN)

**Random_Forest_Regression**

```
In [34]: #Feature Scaling
         from sklearn.preprocessing import StandardScaler
         sc = StandardScaler()
         x_train = sc.fit_transform(x_train)
         x_test = sc.transform(x_test)
```

```
In [35]: 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)
```

## 7.2 FEATURE 2

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

```
*app.py - I:\Projects\WQP\app.py (3.10.1)*
File Edit Format Run Options Window Help
s
import numpy as np
from flask import Flask,render_template,request
import pickle
app= Flask(__name__)
model=pickle.load(open(r'I:\Projects\WQP\wqi.pkl','rb'))
@app.route('/')
def home() :
  return render_template("web.html")
@app.route('/login',methods = ['POST'])
def login() :
  year = request.form["year"]
  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):
    return render_template("web.html",showcase = 'Excellent, The Predicted Value Is'+ str(y_pred))
  elif(y_pred >= 89 and y_pred<=94):
    return render_template("web.html",showcase = 'Very Good, The Predicted Value Is'+ str(y_pred))
  elif(y_pred >= 80 and y_pred<=88):
    return render_template("web.html",showcase = 'Good, The Predicted Value Is'+ str(y_pred))
  elif(y_pred >= 65 and y_pred<=79):
    return render_template("web.html",showcase = 'Fair, The Predicted Value Is'+ str(y_pred))
  elif(y_pred >= 45 and y_pred<=64):
    return render_template("web.html",showcase = 'Marginal, The Predicted Value Is'+ str(y_pred))
  else:
    return render_template("web.html",showcase = 'Poor, The Predicted Value Is'+ str(y_pred))


if __name__ == '__main__':
     app.run(debug = True,port=5000)
Ln: 1 Col:
```

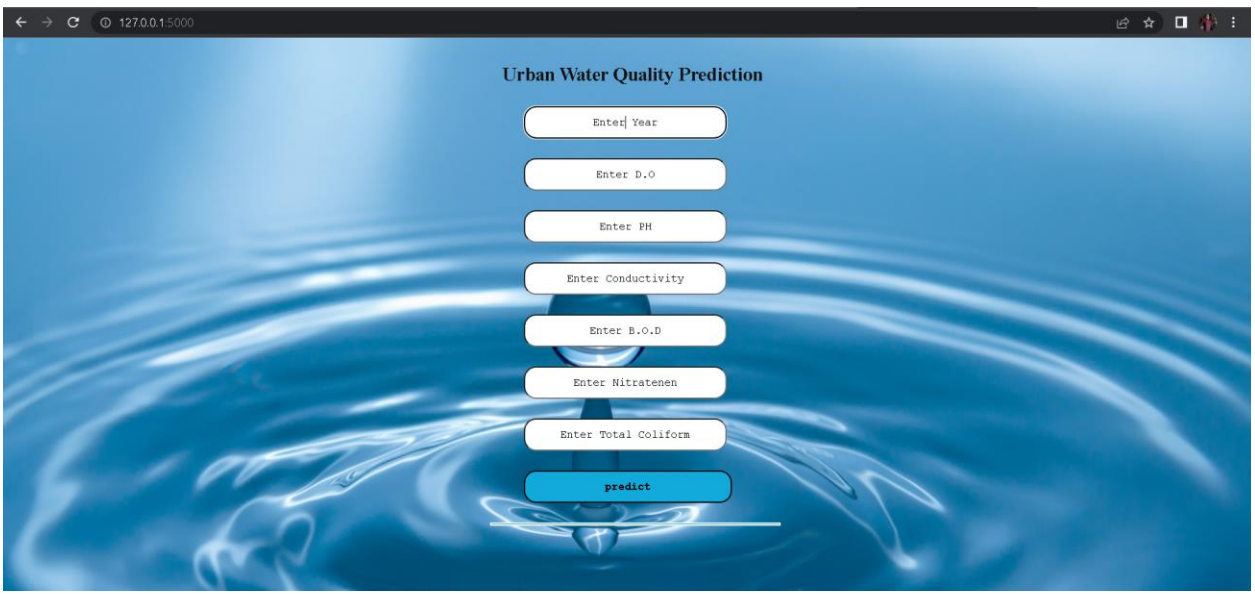
Running app.py

```
(base) I:\Projects\WQP>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 815-263-485
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [19/Nov/2022 09:55:23] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [19/Nov/2022 09:55:24] "GET / HTTP/1.1" 200 -
```

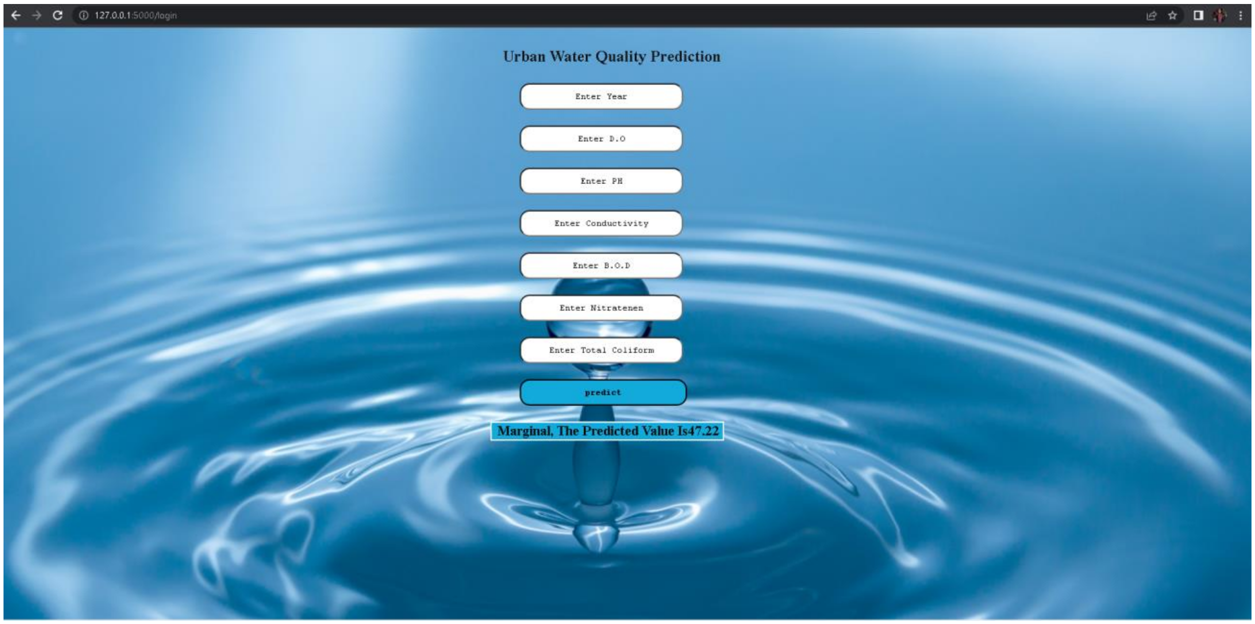
**8.TESTING**

Testing is the process of evaluating and verifying that a software product or application does what it is supposed to do. The benefits of testing include preventing bugs, reducing development costs and improving performance

**8.1 TEST CASES 1**



## TEST CASES 2




## 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 Efficient 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 | 10 | 4 | 2 | 3 | 20 |
| Duplicate | 1 | 0 | 3 | 0 | 4 |
| External | 2 | 3 | 0 | 1 | 6 |
| Fixed | 11 | 2 | 4 | 20 | 37 |

| | | | | | |
|---|---|---|---|---|---|
| Not Reproduced | 0 | 0 | 1 | 0 | 1 |
| Skipped | 0 | 0 | 1 | 1 | 2 |
| Won't Fix | 0 | 5 | 2 | 1 | 8 |
| Totals | 24 | 14 | 13 | 26 | 77 |

## 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 | 7 | 0 | 0 | 7 |
| Client Application | 51 | 0 | 0 | 51 |
| Security | 2 | 0 | 0 | 2 |
| Outsource Shipping | 3 | 0 | 0 | 3 |
| Exception Reporting | 9 | 0 | 0 | 9 |
| Final Report Output | 4 | 0 | 0 | 4 |
| Version Control | 2 | 0 | 0 | 2 |

# 9. RESULTS

## 9.1 PERFORMANCE METRICS

**Model Evaluation**

```
In [58]: 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: 0.9872080200501312
         MSE: 5.555095879699248
         RMSE: 2.3569250899634566

In [59]: #accuracy of the model
         metrics.r2_score(y_test, y_pred)

Out[59]: 0.96971918125809
```

# 10. ADVANTAGES & DISADVANTAGES

1.Whether it be for groundwater, surface water or open water, there are a number of reasons why it is important for you to undertake regular water quality testing. If you're wanting to create a solid foundation on which to build a broader water management plan, then investing in water quality testing should be your first point of action. This testing will also allow you to adhere to strict permit regulations and be in compliance with Australian laws.

2.Identifying the health of your water will help you to discover where it may need some help. Ultimately, finding a source of pollution, or remaining proactive with your monitoring will enable you to save money in the long term. The more information that you can obtain will assist you with your decision on what product you may need to improve the condition of your water. Simply guessing and buying products based on a hunch or a general trend is ill-advised, as each body of water has unique properties that can only be discovered through testing.

3.Measuring the amount of dissolved oxygen in your water is another important advantage of water quality testing, as typically the less oxygen, the higher the water temperature, resulting in a more harmful environment for aquatic life. These levels do fluctuate slightly across the seasons, but regular monitoring of your water quality will allow you to discover trends over time, and whether there are other factors that may be contributing to the results you discover

There needs to be a more user-centric approach towards tackling the water quality issues, by using user friendly tools and an interactive environment so that the solution actually benefits in tackling water quality issues.

- Not all models have been able to numerically predict the magnesium absorption ratio (MAR) and the permeability index (PI), so classification models may be able to improve the accuracy of predictions

. • Internet Connectivity and times may be a problem, since data won't be updated

## 11. CONCLUSION

Potability determines the quality of water, which is one of the most important resources for existence. Traditionally, testing water quality required an expensive and time-consuming lab analysis. This study looked into an alternative machine learning method for predicting water quality using only a few simple water quality criteria. To estimate, a set of representative supervised machine learning algorithms was used. It would detect water of bad quality before it was released for consumption and notify the appropriate authorities It will hopefully reduce the number of individuals who drink low-quality water, lowering the risk of diseases like typhoid and diarrhea. In this case, using a prescriptive analysis based on projected values would result in future capabilities to assist decision and policy makers.

## 12. FUTURE SCOPE

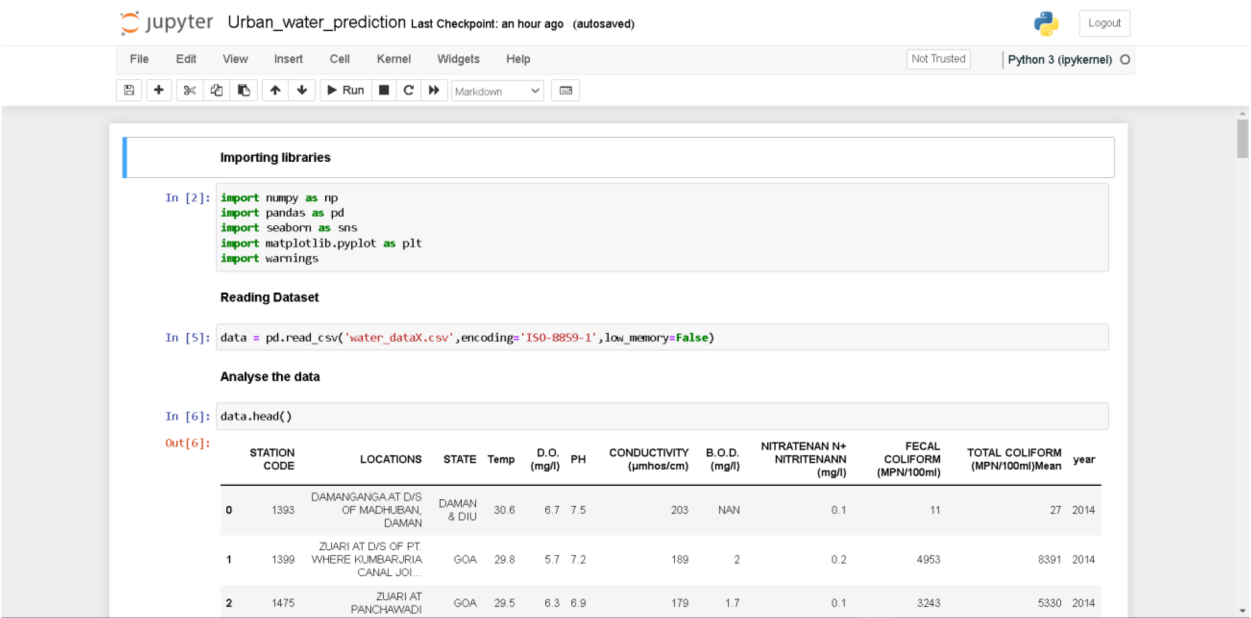
1. It helps to calculate large data

2. Easily calculate the purity of water

# 13.APPENDIX

## SOURCE CODE

Urban water prediction.ipynb


jupyter Urban_water_prediction Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

**Importing libraries**

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

**Reading Dataset**

In [5]:
```
data = pd.read_csv('water_dataX.csv',encoding='ISO-8859-1',low_memory=False)
```

**Analyse the data**

In [6]:
```
data.head()
```

Out[6]:

| | 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 |

jupyter Urban_water_prediction Last Checkpoint: an hour ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Markdown

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 [7]:
```
data.describe()
```

Out[7]:

| | 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 [8]:
```
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
```

jupyter Urban_water_prediction Last Checkpoint: a few seconds ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Markdown

Out[24]:

| | 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 | 84 |
| 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 | ... | 100 | 60 | 100 | 16.5 | 22.48 | 23.40 | 0.54 | 2.8 | 11.24 | 76 |
| 2 | 1475 | ZUARI AT PANCHAWADI | GOA | 29.500000 | 6.3 | 6.9 | 179.0 | 1.700000 | 0.100000 | 5330.0 | ... | 100 | 60 | 100 | 13.2 | 28.10 | 23.40 | 0.54 | 2.8 | 11.24 | 79 |
| 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 | 69 |
| 4 | 3182 | RIVER ZUARI AT MARCAIM JETTY | GOA | 29.500000 | 5.8 | 7.3 | 83.0 | 1.900000 | 0.400000 | 5500.0 | ... | 100 | 80 | 100 | 16.5 | 22.48 | 23.40 | 0.72 | 2.8 | 11.24 | 77 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 1986 | 1330 | TAMBIRAPARANI AT ARUMUGANERI, TAMILNADU | NAN | 26.209814 | 7.9 | 738.0 | 7.2 | 2.700000 | 0.518000 | 202.0 | ... | 100 | 100 | 100 | 0.0 | 28.10 | 23.40 | 0.90 | 2.8 | 16.86 | 72 |
| 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 | ... | 100 | 100 | 100 | 0.0 | 28.10 | 23.40 | 0.90 | 2.8 | 16.86 | 72 |
| 1988 | 1403 | GUMTI AT U/S SOUTH TRIPURA,TRIPURA | NAN | 28.000000 | 7.6 | 98.0 | 6.2 | 1.200000 | 1.623079 | 570.0 | ... | 100 | 100 | 100 | 0.0 | 28.10 | 23.40 | 0.90 | 2.8 | 11.24 | 66 |
| 1989 | 1404 | GUMTI AT D/S SOUTH TRIPURA, TRIPURA | NAN | 28.000000 | 7.7 | 91.0 | 6.5 | 1.300000 | 1.623079 | 562.0 | ... | 100 | 100 | 100 | 0.0 | 28.10 | 23.40 | 0.90 | 2.8 | 11.24 | 66 |

jupyter Urban_water_prediction Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Markdown

```
 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 [9]:
```
data.shape
```

Out[9]:
```
(1991, 12)
```

**Handling Missing Values**

In [10]:
```
data.isnull().any()
```

Out[10]:
```
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
```

jupyter Urban_water_prediction Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Markdown

```
NITRATENAN N+ NITRITENANN (mg/l)    False
FECAL COLIFORM (MPN/100ml)          False
TOTAL COLIFORM (MPN/100ml)Mean      False
year                                False
dtype: bool
```

In [11]:
```
data.isnull().sum()
```

Out[11]:
```
STATION CODE                        0
LOCATIONS                           0
STATE                               0
Temp                                0
D.O. (mg/l)                         0
PH                                  0
CONDUCTIVITY (µmhos/cm)             0
B.O.D. (mg/l)                       0
NITRATENAN N+ NITRITENANN (mg/l)    0
FECAL COLIFORM (MPN/100ml)          0
TOTAL COLIFORM (MPN/100ml)Mean      0
year                                0
dtype: int64
```

In [12]:
```
data.dtypes
```

Out[12]:
```
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
```

jupyter Urban_water_prediction Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

In [13]:
```python
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[13]:
```
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
```

In [14]:
```python
data.isnull().sum()
```

Out[14]:
```
STATION CODE                          0
LOCATIONS                             0
STATE                                 0
Temp                                 92
D.O. (mg/l)                          31
PH                                    8
CONDUCTIVITY (µmhos/cm)              25
```

jupyter Urban_water_prediction Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

Out[14]:
```
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
```

In [15]:
```python
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)
```

In [16]:
```python
data.drop(["FECAL COLIFORM (MPN/100ml)"],axis=1,inplace=True)
```

In [17]:
```python
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'})
```

jupyter Urban_water_prediction Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

```
data=data.rename(columns = {'LOCATIONS': 'location'})
data=data.rename(columns = {'STATE': 'state'})
data=data.rename(columns = {'PH': 'ph'})
```

**Water Quality Index (WQI) Calculation**

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

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

In [20]:
```
#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)))))
```

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

jupyter Urban_water_prediction Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

```
                                  else (60 if(80>=x>=6)
                                    else(40 if(125>=x>=80)
                                      else 0)))))
```

In [22]:
```
#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)))))
```

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

In [24]:
```
#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
```

Out[24]:

| | 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 & 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 |

jupyter Urban_water_prediction Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run | Markdown

| | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | KUMBARJRIA CANAL JOI... | | | | | | | | | | | | | | | | | | | | |
| **2** | 1475 | ZUARI AT PANCHAWADI | GOA | 29.500000 | 6.3 | 6.9 | 179.0 | 1.700000 | 0.100000 | 5330.0 | ... | 100 | 60 | 100 | 13.2 | 28.10 | 23.40 | 0.54 | 2.8 | 11.24 | 79 |
| **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 | 69 |
| **4** | 3182 | RIVER ZUARI AT MARCAIM JETTY | GOA | 29.500000 | 5.8 | 7.3 | 83.0 | 1.900000 | 0.400000 | 5500.0 | ... | 100 | 80 | 100 | 16.5 | 22.48 | 23.40 | 0.72 | 2.8 | 11.24 | 77 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **1986** | 1330 | TAMBIRAPARANI AT ARUMUGANERI, TAMILNADU | NAN | 26.209814 | 7.9 | 738.0 | 7.2 | 2.700000 | 0.518000 | 202.0 | ... | 100 | 100 | 100 | 0.0 | 28.10 | 23.40 | 0.90 | 2.8 | 16.86 | 72 |
| **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 | ... | 100 | 100 | 100 | 0.0 | 28.10 | 23.40 | 0.90 | 2.8 | 16.86 | 72 |
| **1988** | 1403 | GUMTI AT U/S SOUTH TRIPURA,TRIPURA | NAN | 28.000000 | 7.6 | 98.0 | 6.2 | 1.200000 | 1.623079 | 570.0 | ... | 100 | 100 | 100 | 0.0 | 28.10 | 23.40 | 0.90 | 2.8 | 11.24 | 66 |
| **1989** | 1404 | GUMTI AT D/S SOUTH TRIPURA, TRIPURA | NAN | 28.000000 | 7.7 | 91.0 | 6.5 | 1.300000 | 1.623079 | 562.0 | ... | 100 | 100 | 100 | 0.0 | 28.10 | 23.40 | 0.90 | 2.8 | 11.24 | 66 |
| **1990** | 1726 | CHANDRAPUR, AGARTALA D/S OF HAORA RIVER, TRIPURA | NAN | 29.000000 | 7.6 | 110.0 | 5.7 | 1.100000 | 1.623079 | 546.0 | ... | 100 | 100 | 100 | 0.0 | 28.10 | 23.40 | 0.90 | 2.8 | 11.24 | 66 |

1991 rows × 24 columns

jupyter Urban_water_prediction Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run | Markdown

In [25]:
```python
#Calculation of overall WQI for each year
average = data.groupby('year')['wqi'].mean()
average.head()
```

Out[25]:
```
year
2003    66.239545
2004    61.290000
2005    73.762689
2006    72.909714
2007    74.233000
Name: wqi, dtype: float64
```

**Splitting Dependent and Independent Columns**

In [26]:
```python
data.head()
data.drop(['location','station','state'],axis =1,inplace=True)
```

In [27]:
```python
data.head()
```

Out[27]:

| | Temp | do | ph | co | bod | na | tc | year | npH | ndo | ... | nbdo | nec | nna | wph | wdo | wbdo | wec | wna | wco | wqi |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 30.6 | 6.7 | 7.5 | 203.0 | 6.940049 | 0.1 | 27.0 | 2014 | 100 | 100 | ... | 60 | 60 | 100 | 16.5 | 28.10 | 14.04 | 0.54 | 2.8 | 22.48 | 84.46 |
| **1** | 29.8 | 5.7 | 7.2 | 189.0 | 2.000000 | 0.2 | 8391.0 | 2014 | 100 | 80 | ... | 100 | 60 | 100 | 16.5 | 22.48 | 23.40 | 0.54 | 2.8 | 11.24 | 76.96 |
| **2** | 29.5 | 6.3 | 6.9 | 179.0 | 1.700000 | 0.1 | 5330.0 | 2014 | 80 | 100 | ... | 100 | 60 | 100 | 13.2 | 28.10 | 23.40 | 0.54 | 2.8 | 11.24 | 79.28 |
| **3** | 29.7 | 5.8 | 6.9 | 64.0 | 3.800000 | 0.5 | 8443.0 | 2014 | 80 | 80 | ... | 80 | 100 | 100 | 13.2 | 22.48 | 18.72 | 0.90 | 2.8 | 11.24 | 69.34 |
| **4** | 29.5 | 5.8 | 7.3 | 83.0 | 1.900000 | 0.4 | 5500.0 | 2014 | 100 | 80 | ... | 100 | 80 | 100 | 16.5 | 22.48 | 23.40 | 0.72 | 2.8 | 11.24 | 77.14 |

5 rows × 21 columns

jupyter Urban_water_prediction Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Markdown

5 rows × 21 columns

In [28]:
```
x=data.iloc[:,1:7].values
```

In [29]:
```
x.shape
```

Out[29]:
```
(1991, 6)
```

In [30]:
```
y=data.iloc[:,-1:].values
y.shape
```

Out[30]:
```
(1991, 1)
```

In [31]:
```
print(x)
```

```
[[6.70000000e+00 7.50000000e+00 2.03000000e+02 6.94004877e+00
  1.00000000e-01 2.70000000e+01]
 [5.70000000e+00 7.20000000e+00 1.89000000e+02 2.00000000e+00
  2.00000000e-01 8.39100000e+03]
 [6.30000000e+00 6.90000000e+00 1.79000000e+02 1.70000000e+00
  1.00000000e-01 5.33000000e+03]
 ...
 [7.60000000e+00 9.80000000e+01 6.20000000e+00 1.20000000e+00
  1.62307871e+00 5.70000000e+02]
 [7.70000000e+00 9.10000000e+01 6.50000000e+00 1.30000000e+00
  1.62307871e+00 5.62000000e+02]
 [7.60000000e+00 1.10000000e+02 5.70000000e+00 1.10000000e+00
  1.62307871e+00 5.46000000e+02]]
```

In [32]:
```
print(y)
```

```
[[84.46]
```

jupyter Urban_water_prediction Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Markdown

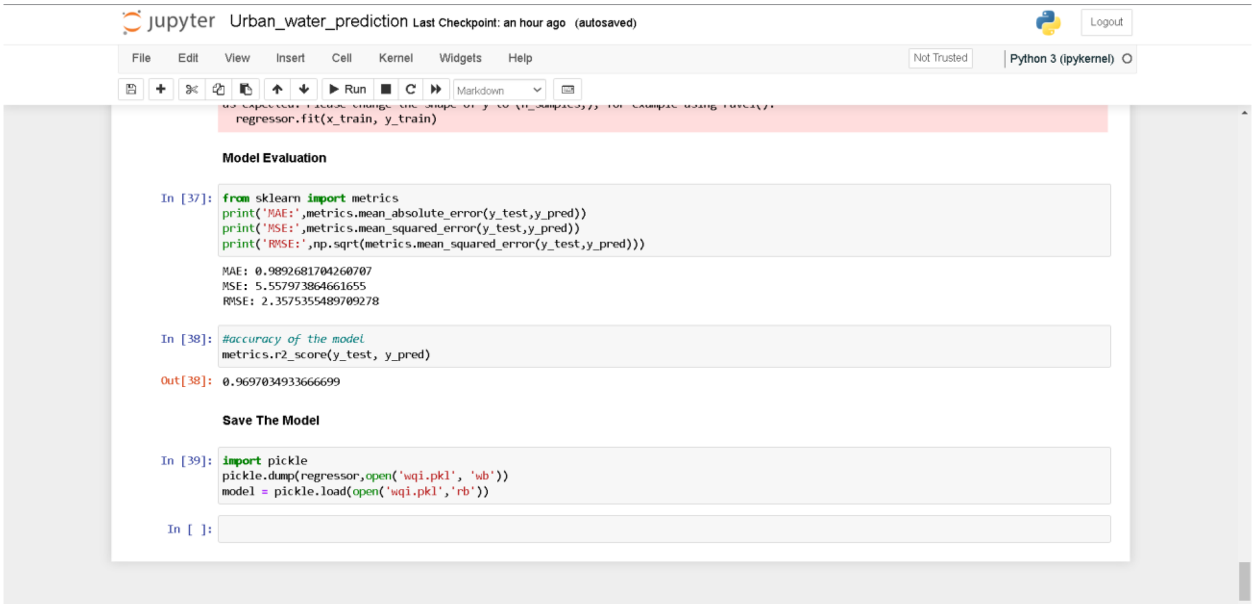
```
[[6.70000000e+00 7.50000000e+00 2.03000000e+02 6.94004877e+00
  1.00000000e-01 2.70000000e+01]
 [5.70000000e+00 7.20000000e+00 1.89000000e+02 2.00000000e+00
  2.00000000e-01 8.39100000e+03]
 [6.30000000e+00 6.90000000e+00 1.79000000e+02 1.70000000e+00
  1.00000000e-01 5.33000000e+03]
 ...
 [7.60000000e+00 9.80000000e+01 6.20000000e+00 1.20000000e+00
  1.62307871e+00 5.70000000e+02]
 [7.70000000e+00 9.10000000e+01 6.50000000e+00 1.30000000e+00
  1.62307871e+00 5.62000000e+02]
 [7.60000000e+00 1.10000000e+02 5.70000000e+00 1.10000000e+00
  1.62307871e+00 5.46000000e+02]]
```

In [32]:
```
print(y)
```

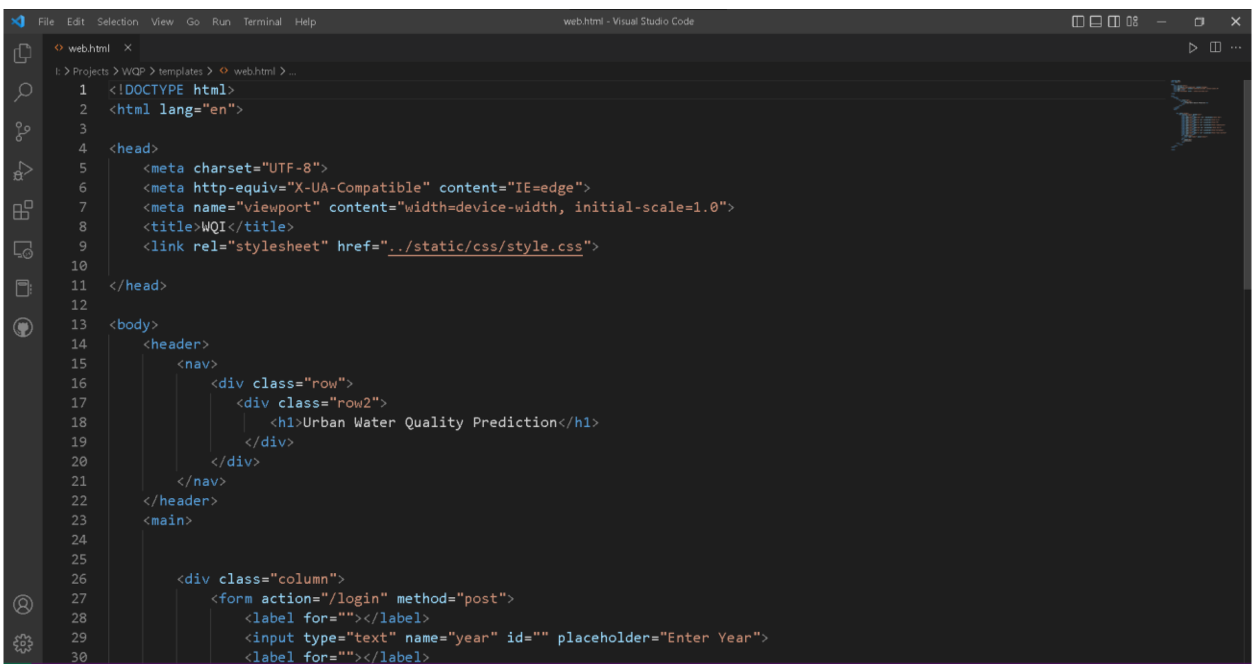

```
[[84.46]
 [76.96]
 [79.28]
 ...
 [66.44]
 [66.44]
 [66.44]]
```

**Splitting the Data Into Train and Test**

In [33]:
```
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)
```




Web.html




App.py

*app.py - I:\Projects\WQP\app.py (3.10.1)*

File Edit Format Run Options Window Help

```python
s
import numpy as np
from flask import Flask,render_template,request
import pickle
app= Flask(__name__)
model=pickle.load(open(r'I:\Projects\WQP\wqi.pkl','rb'))
@app.route('/')
def home() :
  return render_template("web.html")
@app.route('/login',methods = ['POST'])
def login() :
  year = request.form["year"]
  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):
    return render_template("web.html",showcase = 'Excellent, The Predicted Value Is'+ str(y_pred))
  elif(y_pred >= 89 and y_pred<=94):
    return render_template("web.html",showcase = 'Very Good, The Predicted Value Is'+ str(y_pred))
  elif(y_pred >= 80 and y_pred<=88):
    return render_template("web.html",showcase = 'Good, The Predicted Value Is'+ str(y_pred))
  elif(y_pred >= 65 and y_pred<=79):
    return render_template("web.html",showcase = 'Fair, The Predicted Value Is'+ str(y_pred))
  elif(y_pred >= 45 and y_pred<=64):
    return render_template("web.html",showcase = 'Marginal, The Predicted Value Is'+ str(y_pred))
  else:
    return render_template("web.html",showcase = 'Poor, The Predicted Value Is'+ str(y_pred))


if __name__ == '__main__':
     app.run(debug = True,port=5000)
```

Ln: 1 Col:

GitHub Link : [https://github.com/IBM-EPBL/IBM-Project-16249-1659610220](https://github.com/IBM-EPBL/IBM-Project-16249-1659610220)