```
PNT2022TMID36055
```

EFFICIENT WATER QUALITY ANALYSIS AND PREDICTION USING MACHINE LEARNING

NALAIYA THIRAN PROJECT BASED LEARNING ON PROFESSIONAL READLINESS FOR INNOVATION, EMPLOYNMENT AND ENTERPRENEURSHIP

A PROJECT REPORT

Submitted

By

Yashwanth V [2019506117]

Aditya Kumar [2019506007]

Sneha Pillai [2019506091]

Arun Prasad [2019506013]

Thanigai Vasan [2019506104]

```
```
PNT2022TMID36055

1. INTRODUCTION

1.1 Project overview:

Water is one of the most fundamental needs to support people's livelihood. Access to water and sanitation is even recognised by the United Nations as human rights, which reflects the crucial role of water in human lives. However, as the availability and sources of clean water is a critical issue that must be addressed, mankind is confronting a number of difficulties. Diseases including cholera, typhoid, dysentery, diarrhoea, and polio can all be spread by contaminated water. The World Health Organization (WHO) reports that a large portion of the 485.0000 health-related fatalities per year are due to contaminated water, poor sanitation, and poor hygiene.
Determining the water quality is one of the most challenging and the need of the hour . Machine Learning techniques can prove to be beneficial in predicting the quality of water.

1.2 Purpose

Water quality refers to the suitability of water for different uses according to its physical, chemical, biological, and organoleptic (taste-related) properties. It is especially important to understand and measure water quality as it directly impacts human consumption and health, industrial and domestic use, and the natural environment.

Water quality is determined by a set of physical and chemical parameters such as PH, temperature, B.O.D and so on . The acceptable and unacceptable values for each variable must then be established. If the parameters values lead to a range of WQI values it classifies the quality of the water sample. Hence, using these chemical and physical parameters as input to find WQI Value will lead to a precise calculation.

This project, aims to develop a machine learning model to determine the water quality index(WQI) of water samples. The model takes in the values of different contributing factors pH value, conductivity, hardness, dissolved oxygen etc. Machine Learning algorithms like KNN, SVM , Random Forests and Logistic Regression have been used to help find the WQI value for the water sample.

2. LITERATURE SURVEY

2.1 Existing problem

1. Water sample collection and laboratory analysis are time and resource intensive processes. Numerous machine learning methods, including multivariate linear regression (MLR) and artificial neural network (ANN) models, have been suggested in the last ten years to solve the issue. It has been demonstrated that the adaptive neuro-fuzzy inference system (ANFIS) is a useful tool for extracting the complex linear and non-linear relationships concealed in datasets. Despite having good performance in predicting water quality, the ANFIS model. Stratified sampling and wavelet denoising techniques are used, and the results are presented together with a comparison of the deep prediction performance of the MLR, ANN, and ANFIS models.
   Disadvantages: Because of the complex linear and nonlinear interactions in the water quality dataset, simple linear regression analysis cannot predict water quality with any degree of accuracy. When the input parameters are uncertain, ANN models are unable to articulate the non-linear relationship concealed in the dataset.

2. In order to establish a reliable strategy for forecasting water quality as accurately as feasible, various AI algorithms are evaluated to handle Water Quality data collected over an extended period of time. The Water Quality data was classified using the Water Quality Index by a number of machine learning classifiers and their stacking ensemble models (WQI). Support Vector Machine (SVM), Random Forest (RF), Logistic Regression (LR), Decision Tree (DT), CATBoost, XGBoost, and Multilayer Perceptron were among the classifiers that were examined (MLP). 1679 samples and associated meta-data were collected over a nine-year period and were part of the dataset used in the study. Additionally, Receiver Operating Characteristic curves (ROC) and precision-recall curves were utilised to evaluate the effectiveness of the different classifiers. **Disadvantages:** In comparison to linear regression, random forests do not offer as much insight into the coefficients. Since the required training time is longer when a large data collection is found, SVM does not perform effectively. The feature selection process for decision trees is automatic, and the model is non-parametric. It costs money because DT takes time to process and train the model. DT is also insufficient when using regression and forecasting continuous values. LRM should not be utilized if there are less observations than features because this could lead to overfitting.

3. The best model and algorithms for pattern extraction and prediction were evaluated and categorised using the water quality results using Knowledge Discovery in Databases (KDD). The goal of the study is to use data mining techniques to extract information from the dataset for assessing and categorising the water quality based on various characteristics. Four well-known data mining techniques, such as CBA, SVMs, NB, and KNN, are used in a sequence of phases that include data selection of water quality metrics, cleaning, and normalisation. For model prediction and pattern extraction, many methods were used as classifiers. These included Gradient Boosted Trees, Decision Trees, Random Forests, Generalized Linear Models, Naive Bayes, and the Deep Learning algorithm.

4. The evaluation and forecasting of water quality research findings are reviewed in this publication. In order to analyse water quality, the article categorises and contrasts big data analytics methodologies in use and big data-based prediction models. The structures and networks in the human brain are attempted to be simulated by an artificial **neural network model**. Nodes in the design of neural networks either produce a signal or don't, often according to a sigmoid activation function. Radial basis functions are used in radial basis function network models as activation functions. Radial basis functions of the inputs and neuron parameters are combined linearly as the network's output. The accuracy, robustness, issue kinds, sample size, efficiency, and simplicity of **RBFN** are its overall strengths. An unsupervised learning method called a **Deep Belief Network (DBN)** has many layers of hidden units. Although the units are not connected, the layers are. When trained in an unsupervised manner, DBN can learn to probabilistically recreate its inputs. One of the best methods for knowledge discovery and data mining is the **decision tree** model. Contrary to the prior model, this one employ supervised learning, which links observations about an item to predictions about its intended value. Artificial neural networks and the decision tree method are combined in the **improved decision tree model**. The grouping of data is this approach's key benefit. The **least squares support vector machine model** offers a supervised learning method that involves solving a group of linear equations to arrive at the solution.

   Disadvantages: These models have a number of drawbacks, including problems with data quality and validation, the need for research on big data quality assurance, real-time monitoring of water quality, and supervision of water resources. Research and development of real-time water quality monitor and evaluation systems that support water quality evaluation and analysis on various levels, as well as challenges with big data modelling for dynamic water quality monitor and analysis at the different levels for smart cities.

5. A variety of models, including the Adaptive Neuro-Fuzzy Inference System (ANFIS), Radial Basis Function Neural Networks (RBF-ANN), and Multi-Layer Perceptron Neural Network (MLP-ANN), were used to set up a water quality prediction model for better water resource management. Two scenarios—Scenario 1 and Scenario 2—were presented during these procedures. In Scenario 1, a prediction model is built for each station's water quality characteristics; in Scenario 2, a prediction model is built based on the value of the same parameter at the preceding station. **ANFIS**, which allowed for the realisation of a highly non-linear mapping, is regarded as being more effective than conventional linear approaches for producing non-linear time series. ANFIS is a multi-layer feed-forward network that uses fuzzy reasoning and neural network learning methods to help map the input space to the output space. The **MLP-ANN**, which has multiple layers of neurons, is a feed-forward network because the output of one neuron is transmitted to its neighbouring layer's input. As an alternative, the **RBF-ANN**, which has capabilities comparable to those of the MLP-ANN, is frequently used for stringent interpolation problems in space with multiple dimensions.

6. This study's objective is to create a water quality prediction model utilising Artificial Neural Networks (ANN) and time-series analysis to incorporate water quality parameters. Historical data on water quality are used in this study. Mean-Squared Error (MSE), Root Mean Squared Error (RMSE), and Regression Analysis are the performance evaluation metrics used to gauge how well the model is doing. ANN has received widespread recognition as a tool for classifying complicated information, including those pertaining to environmental dynamics. It can effectively explain the non-linear relationship between the intricate water quality statistics.

7. The objective of the study is to analyse and predict the quality of water using machine learning algorithms such as Linear Regression and Stacked Denoising Autoencoder as well as using neural networks such as Deep Belief Networks and Multi Layer Perceptron Network. The data collected for the study are pH, dissolved oxygen, turbidity, chlorine, etc which is collected from Krishna river basin near Chaskaman. After data collection, three clusters were created for 3 seasons:

Summer, Winter, Monsoon. On experimenting, turbidity showed to have the highest variability among all the parameters. It is affected during the monsoon season most. pH does not have much variation in data and hence, it is stable compared to dissolved oxygen and turbidity.

8. The popularity of deep learning (DL), an advanced branch of machine learning (ML) for artificial intelligence, has grown recently. Convolutional neural networks (CNN) and long short-term memory (LSTM) are two popular DL models. The LSTM is a subtype of recurrent neural network (RNN) that stores, processes, and represents extended sequential data in hidden memory. Each neuron in a layer of the CNN is connected to a tiny local region of neurons in the input data by means of a series of convolutional layers. This is accomplished by sliding a filter, which is a weight matrix.
   In recent years, hybrid neural networks—which combine the benefits of multiple networks—have drawn more and more attention. Examine how well DL models—LSTM, CNN, and hybrid CNN-LSTM models—can predict variables related to water quality in comparison to more conventional ML models (decision tree and support vector regression). Compare the effectiveness of the LSTM and CNN approaches for predicting short-term changes in water quality. Create a hybrid (CNN-LSTM) model that combines the benefits of the CNN and LSTM approaches.

2.2 References:

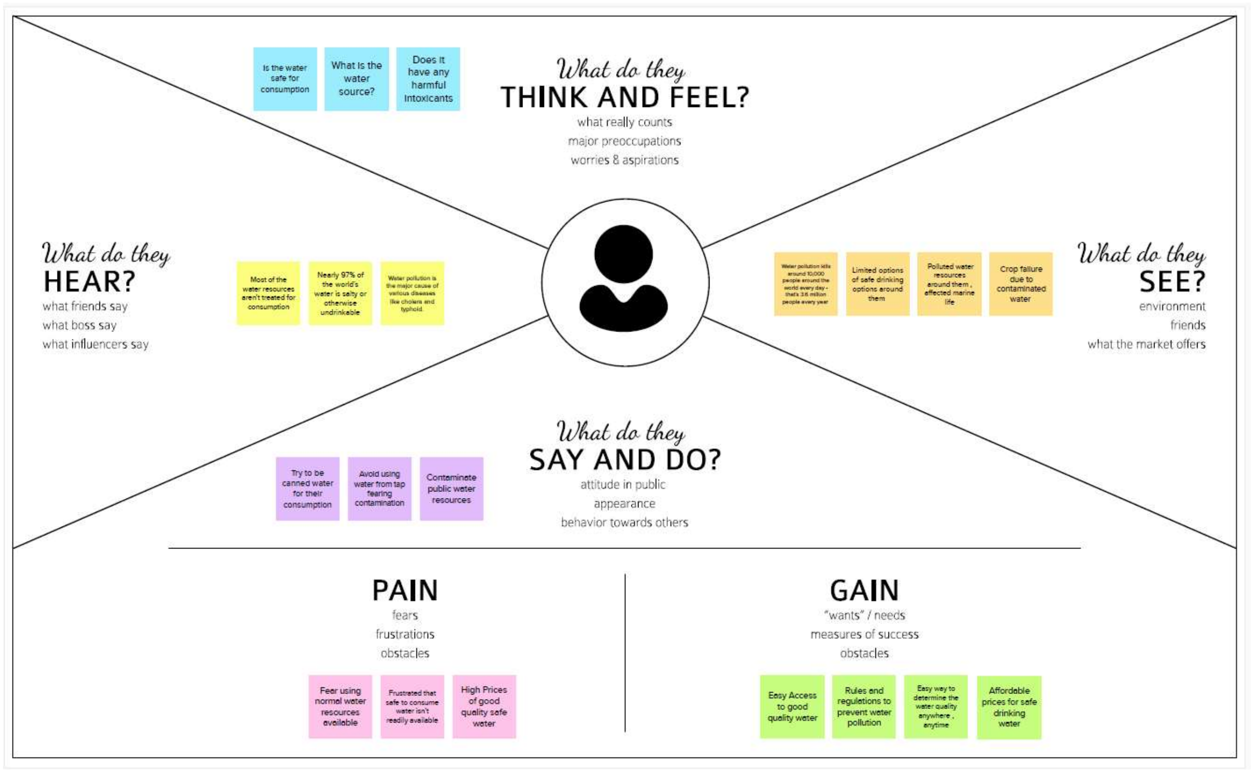
1. Water quality prediction based on machine learning techniques - Zhao Fu
2. Efficient Water Quality Prediction Using Supervised Machine Learning - Umair Ahmed.
3. Water quality classification using machine learning algorithms -
4. Data-driven Water Quality Analysis and Prediction: A Survey - Gaganjot Kaur Kang, Jerry Zeyu Gao, Gang Xie.
5. Machine learning methods for better water quality prediction - Ali Najah Ahmeda
6. Predicting and analyzing water quality using Machine Learning: A comprehensive model - Yafra Khan, Chai Soo See.
7. Predictive Analysis of Water Quality Parameters using Deep Learning - Khanchan Khare

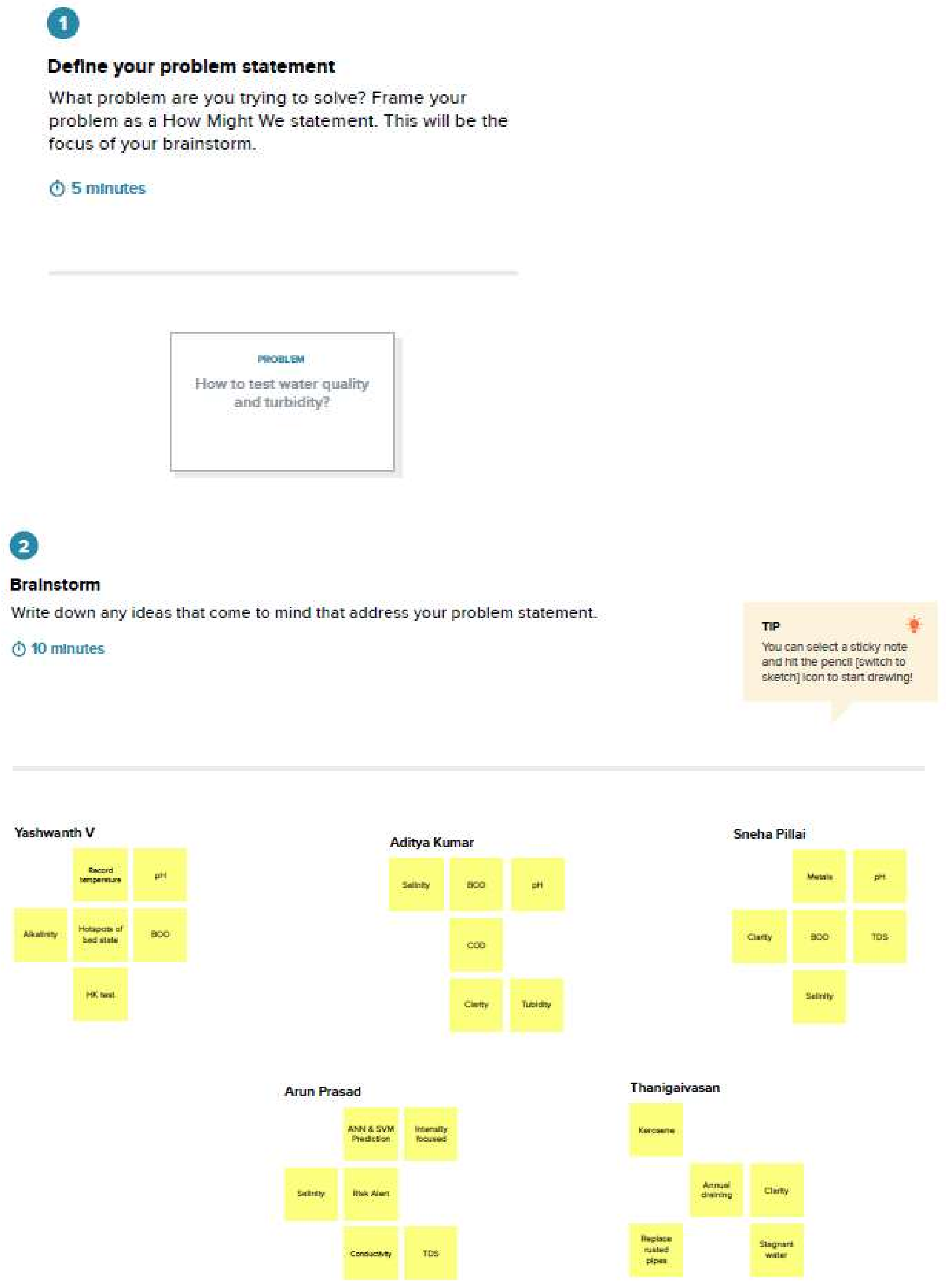
```
PNT2022TMID36055
```

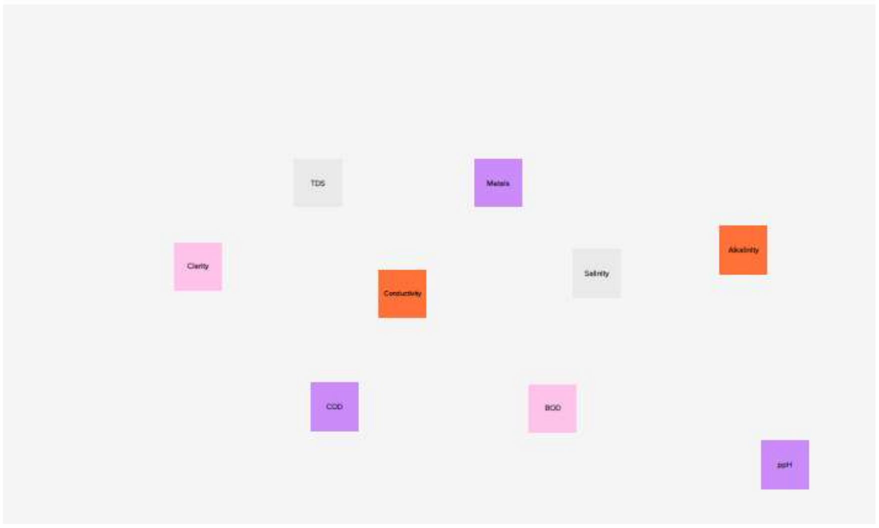
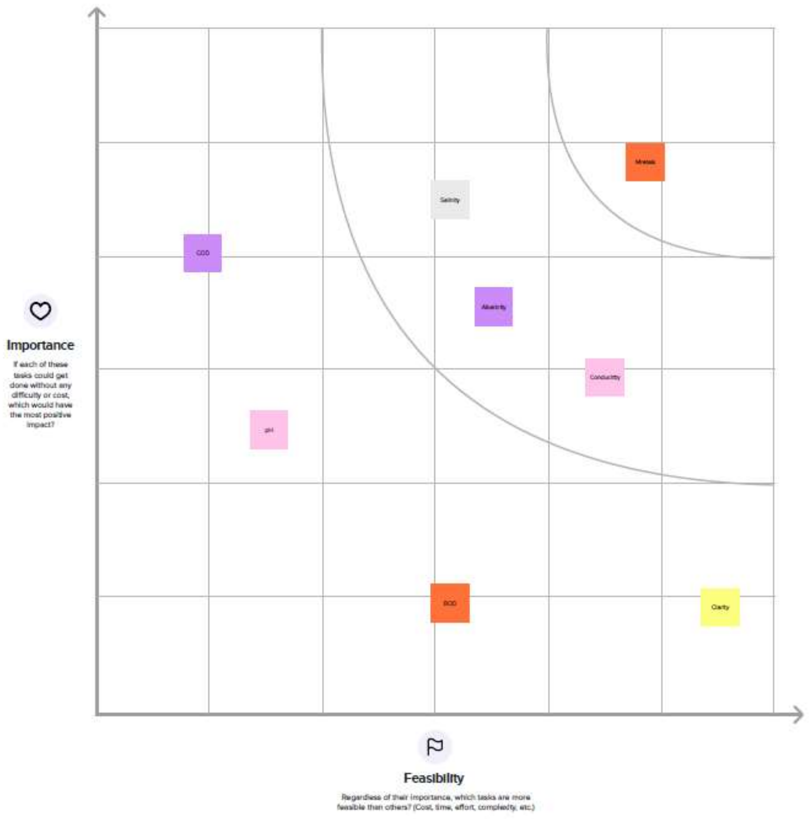
8. Short-term water quality variable prediction using a hybrid CNN–LSTM deep learning model - Rahim Barzegar, Mohammad Taghi Aalami, Jan Adamowski.

2.3. Problem Statement Definition

Problem Statement (PS)	I am (Customer)	I am trying to	But	Because	Which makes me feel
PS-1	Government Metro Water Resource Engineer	To make sure The public have access to good quality water	There is no manual or easy way to do this.	No access to equipments and other devices to do so.	worried
PS-2	Drinking Water Supplier	To make sure my customers get quality water.	Due to limited resources no way to do this .	All the resources are costly and cant be accessible easily	Worried

```
PNT2022TMID36055

# 3.IDEATION & PROPOSED SOLUTION

## 3.1 Empathy Map Canvas



What do they
THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
Is the water safe for consumption
What is the water source?
Does it have any harmful Intoxicants
What do they
HEAR?
what friends say
what boss say
what influencers say
Most of the water resources aren't treated for consumption
Nearly 97% of the world's water is salty or otherwise undrinkable
Water pollution is the major cause of various diseases like cholera and typhoid.
What do they
SEE?
environment
friends
what the market offers
Water pollution kills around 10,000 people around the world every day - that's 3.6 million people every year
Limited options of safe drinking options around them
Polluted water resources around them , effected marine life
Crop failure due to contaminated water
What do they
SAY AND DO?
attitude in public
appearance
behavior towards others
Try to be canned water for their consumption
Avoid using water from tap fearing contamination
Contaminate public water resources
PAIN
fears
frustrations
obstacles
Fear using normal water resources available
Frustrated that safe to consume water isn't readily available
High Prices of good quality safe water
GAIN
"wants" / needs
measures of success
obstacles
Easy Access to good quality water
Rules and regulations to prevent water pollution
Easy way to determine the water quality anywhere , anytime
Affordable prices for safe drinking water



```

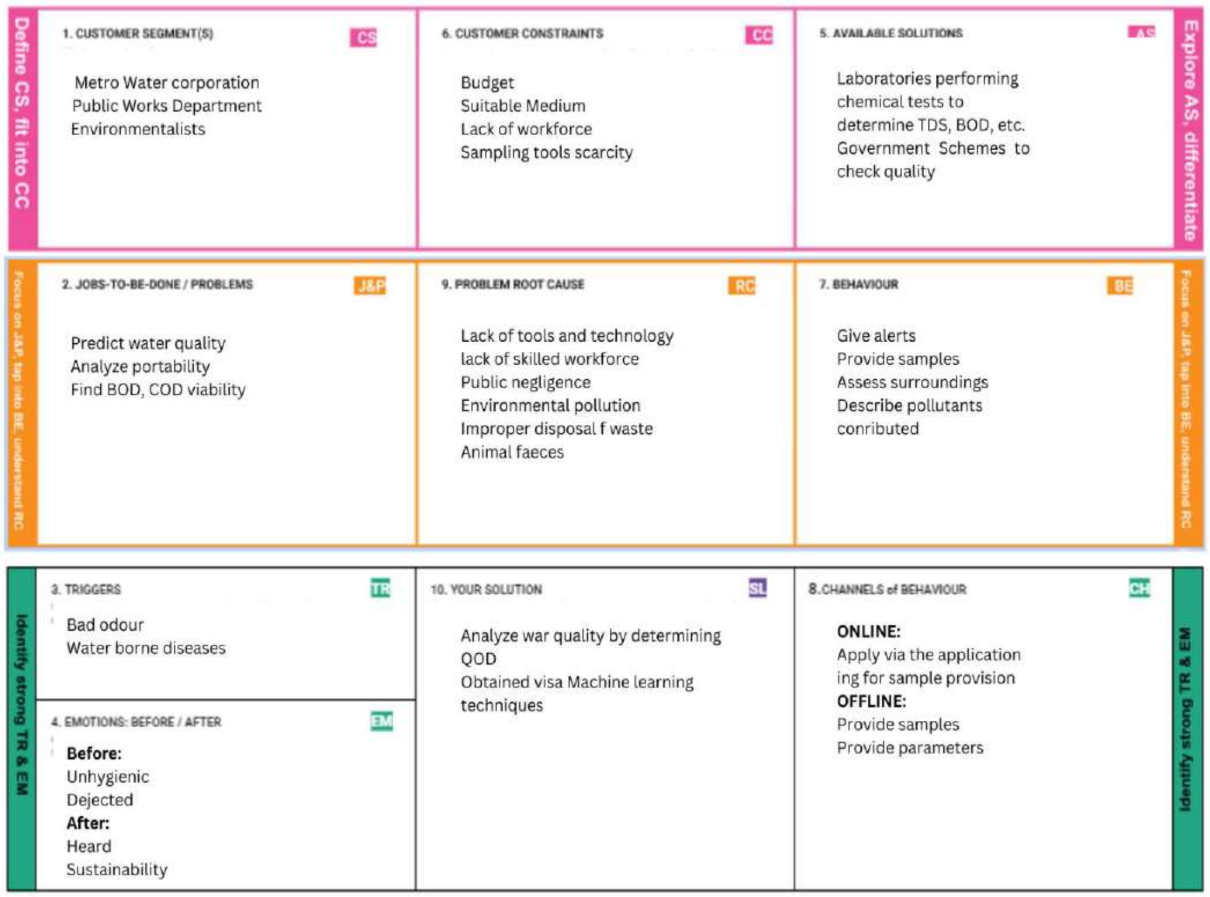
```
PNT2022TMID36055

## 3.2 Ideation & Brainstorming



1

**Define your problem statement**

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

5 minutes

PROBLEM

How to test water quality and turbidity?

2

**Brainstorm**

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

10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Yashwanth V

Record temperature

pH

Alkalinity

Hotspots of bad state

BOD

HK test

Aditya Kumar

Salinity

BOD

pH

COD

Clarity

Tubidity

Sneha Pillai

Metals

pH

Clarity

BOD

TDS

Salinity

Arun Prasad

ANN & SVM Prediction

Internally focused

Salinity

Risk Alert

Conductivity

TDS

Thanigaivasan

Kerosene

Annual draining

Clarity

Replace rusted pipes

Stagnant water
```

```

PNT2022TMID36055

3

**Group Ideas**

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

20 minutes




```

3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Water quality parameters like BOD, COD, alkalinity, salinity, TDS, etc. to be analysed to formulate WQI (Water Quality Index) for each attribute using a formula for which the proportionality constant value 'k' will be deduced. Suitability for the usage of the water for different entities will be deduced based on the WQI calibrated. Change of 'k' values for different utilisation domains must be observed. This is done via machine learning techniques.
2.	Idea / Solution description	Water quality parameters like BOD, COD, alkalinity, salinity, TDS, etc. to be analysed to formulate WQI (Water Quality Index) for each attribute using a formula for which the proportionality constant value 'k' will be deduced. Suitability for the usage of the water for different entities will be deduced based on the WQI calibrated. Change of 'k' values for different utilisation domains must be observed. This is done via machine learning techniques.
3.	Novelty / Uniqueness	WQI is calculated and it's ranges were normalised. It was then subject to classification into categories. Categorising was based on the type of suitability for consumers based on its purity. The best k value for the best accuracy was found using a plot. Random Forest classifier was used for this as we had customised this as a classification problem which

		yielded the best accuracy.
4.	Social Impact / Customer Satisfaction	The water is collected from the water body and analysed by means of its Water Quality Index (WQI). Thus, people can assess the cleanliness of the water bodies whose water they consume and become more proactive and alert towards this issue which can help avert many water-borne diseases like typhoid, cholera, diarrhoea, etc. Thus, there is a tremendous social impact and customer satisfaction that is achieved on testing and predicting the quality of water.
5.	Business Model (Revenue Model)	In future , a premium paid feature which enables the user to easily find out the harmful effects that can be caused by the water body and also categorises the nearby water bodies for different usage capabilities (Crop production, human consumption , marine life and so on)
6.	Scalability of the Solution	The water which is initially collected from small water sources like wells and taps can be expanded to bigger water bodies like rivers, ponds, and other freshwater sources and readings can be taken at multiple areas of the water body and analysed thoroughly.

```
PNT2022TMID36055

## 3.4 Problem Solution fit



Define CS, fit into CC

**1. CUSTOMER SEGMENT(S)** CS

Metro Water corporation
Public Works Department
Environmentalists

**6. CUSTOMER CONSTRAINTS** CC

Budget
Suitable Medium
Lack of workforce
Sampling tools scarcity

**5. AVAILABLE SOLUTIONS** AS

Laboratories performing chemical tests to determine TDS, BOD, etc.
Government Schemes to check quality

Explore AS, differentiate

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

**2. JOBS-TO-BE-DONE / PROBLEMS** J&P

Predict water quality
Analyze portability
Find BOD, COD viability

**9. PROBLEM ROOT CAUSE** RC

Lack of tools and technology
lack of skilled workforce
Public negligence
Environmental pollution
Improper disposal f waste
Animal faeces

**7. BEHAVIOUR** BE

Give alerts
Provide samples
Assess surroundings
Describe pollutants conributed

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

Identify strong TR & EM

**3. TRIGGERS** TR

Bad odour
Water borne diseases

**4. EMOTIONS: BEFORE / AFTER** EM

**Before:**
Unhygienic
Dejected
**After:**
Heard
Sustainability

**10. YOUR SOLUTION** SL

Analyze war quality by determining QOD
Obtained visa Machine learning techniques

**8. CHANNELS of BEHAVIOUR** CH

**ONLINE:**
Apply via the application ing for sample provision
**OFFLINE:**
Provide samples
Provide parameters

Identify strong TR & EM
```

```
```

4.REQUIREMENT ANALYSIS

4.1 Functional Requirements

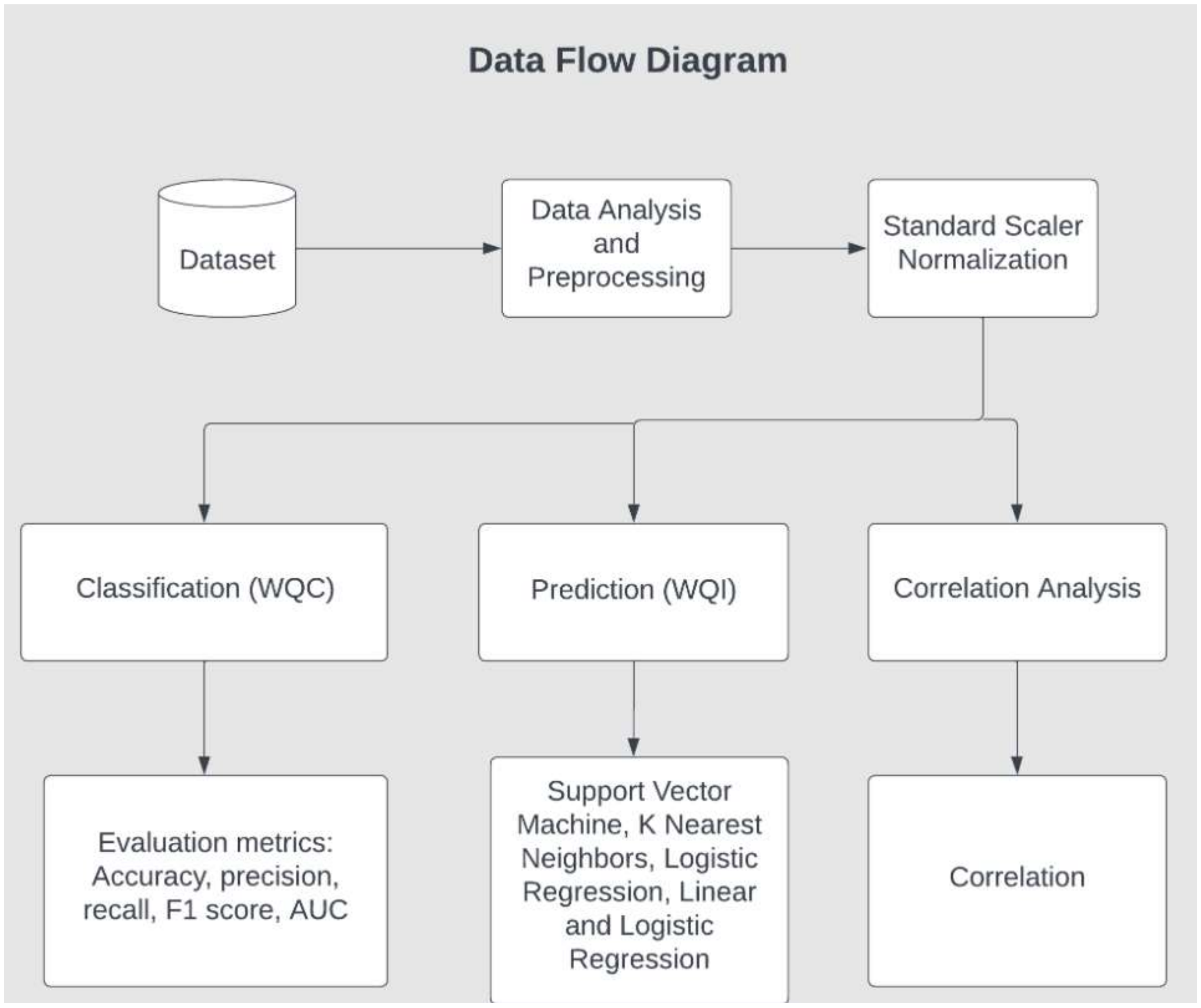
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Dataset Collection	Download the provided csv file containing the details of different parameters of set of water samples.
FR-2	Data Pre-processing	Pre – Processing of data using various machine learning pre-processing techniques, such as remove duplicates.
FR-3	Building ML Model	Build and train the model using IBM cloud.
FR-4	WQI Prediction	On basis of the values provided by the user , the ML Model predicts the value of the WQI.
FR-5	Reporting	Result of the water quality analysis will be displayed to the user.

4.2 Non-Functional Requirement

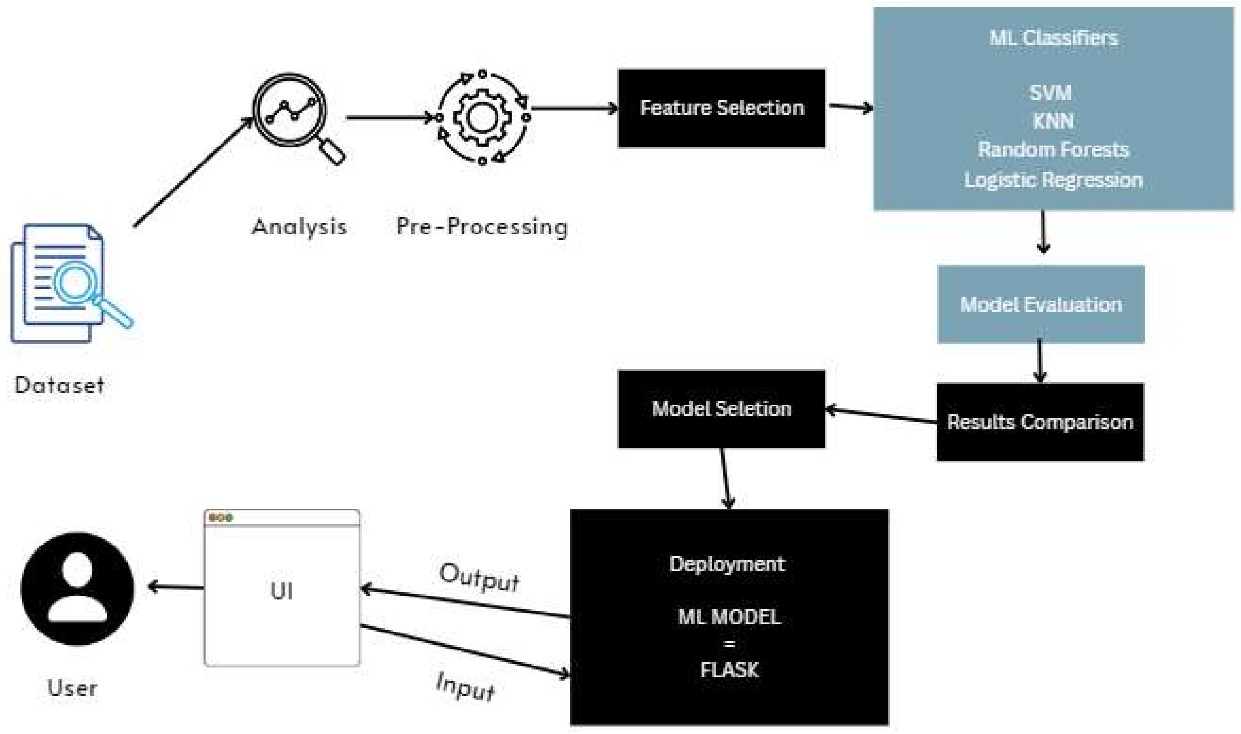
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	User can easily find the WQI of the water sample with just entering few water data input values.
NFR-2	Reliability	The ML Model is very reliable in reporting the WQI value if the data inputs are correct.
NFR-3	Performance	Higher performance compared to pre existing and manual methods.
NFR-4	Availability	Through the proposed webpage the WQI analysis can be done anytime and anywhere.
NFR-5	Scalability	User can analyze and measure the WQI easily.

5. PROJECT DESIGN

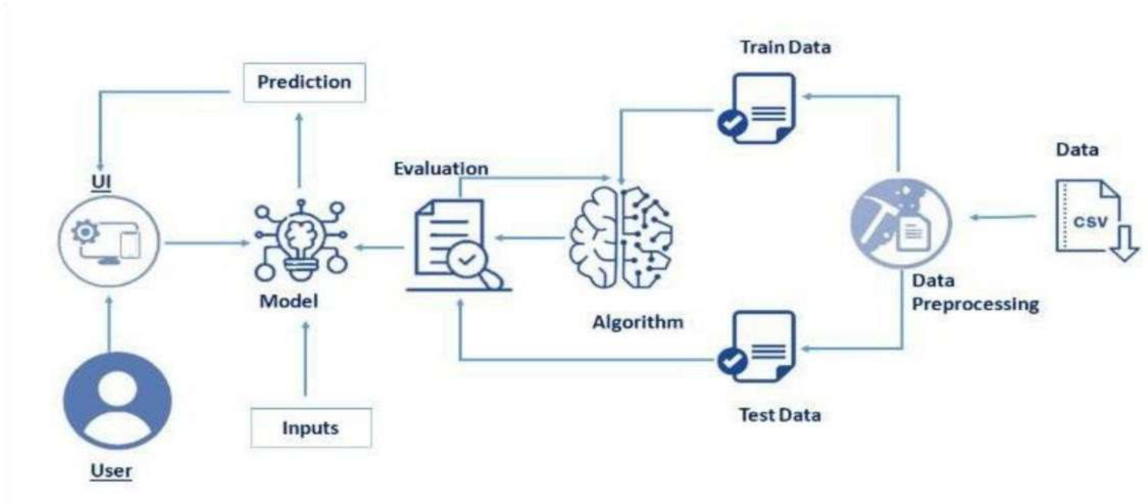
5.1. Data Flow Diagrams


Data Flow Diagram
Dataset
Data Analysis and Preprocessing
Standard Scaler Normalization
Classification (WQC)
Prediction (WQI)
Correlation Analysis
Evaluation metrics: Accuracy, precision, recall, F1 score, AUC
Support Vector Machine, K Nearest Neighbors, Logistic Regression, Linear and Logistic Regression
Correlation

5.2 Solution and Technical Architecture



Technical Architecture:



5.3 User Stories

User type	Functional requirement	User story no	User story/Task	Acceptance criteria	Priorit y	Release
Customer	Dataset Collection and Pre-Processing	USN - 1	As a user, I can know which parameters influence.	I can get high quality results	High	Sprint 1
Customer	Model Building	USN - 2	As a user, I can enter values to predict the water quality.	I can get quick results.	High	Sprint 2
Customer	Training and Testing	USN – 3	As a user, I will get high accuracy value of WQI.	I can check the water quality with precision.	High	Sprint 3
Customer	Implementati on of webpage	USN - 4	As a user, I can find WQI anywhere anytime	I can see quick results on the web page.	High	Sprint 4

6. PROJECT PLANNING & SCHEDULING

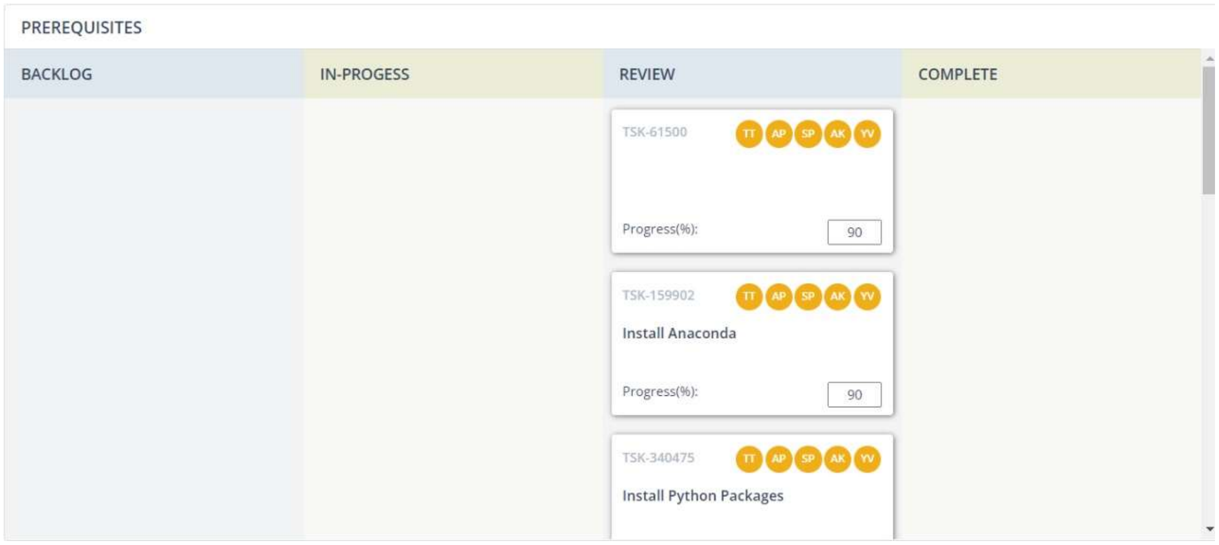
6.1.Sprint Planning & Estimation

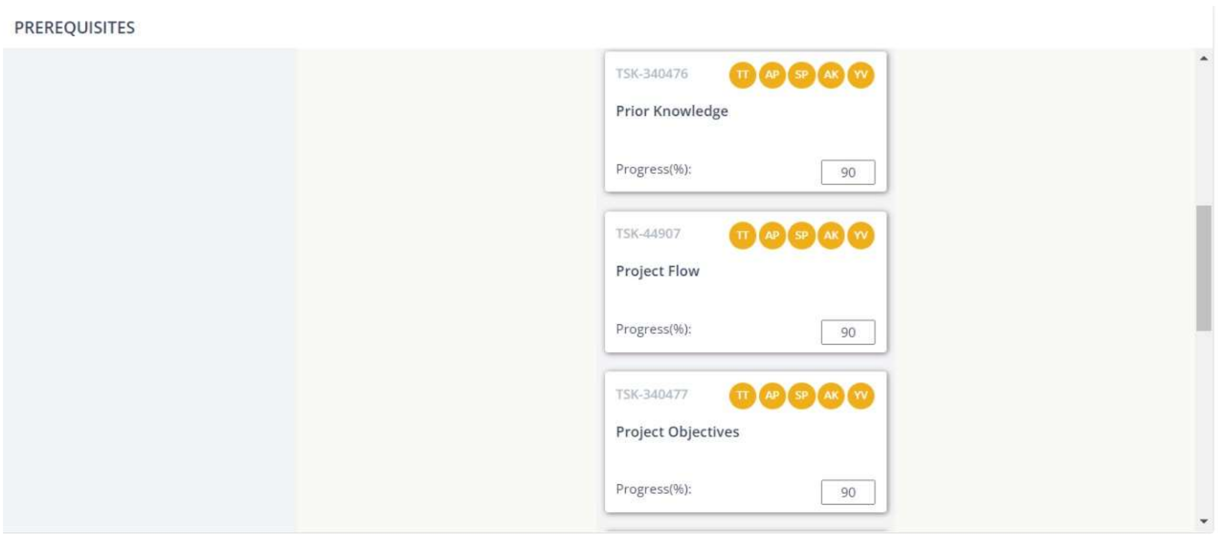
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint1	Data Collection	USN-1	Collect dataset to pre-process.	10	High	Yashwanth V
Sprint1		USN-1	Data pre-processing-formats the data and handles the missing Data.	10	Medium	Thanigaivasan T
Sprint2	Model Building	USN-1,2	Calculate the Water Quality Index (WQI) using given formula for every parameter.	10	High	Sneha Pillai Yashwanth V
Sprint2		USN-1,2	Splitting the data into training and testing data .	10	High	Aditya Kumar Arun Prasad A
Sprint3	Training and Testing	USN-1,2	Training the model using ML algorithm sand testing the performance of the model	20	High	Yashwanth V Arun Prasad A
Sprint4	Implementation of Web page	USN-1,2	Implementing the web page for collecting the data from user	10	High	Sneha Pillai Aditya Kumar
Sprint4		USN-1,2	Deploying the model using IBM Cloud and IBM Watson Studio	10	Medium	Thanigaivasan T Yashwanth V Sneha Pillai

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3Reports from JIRA



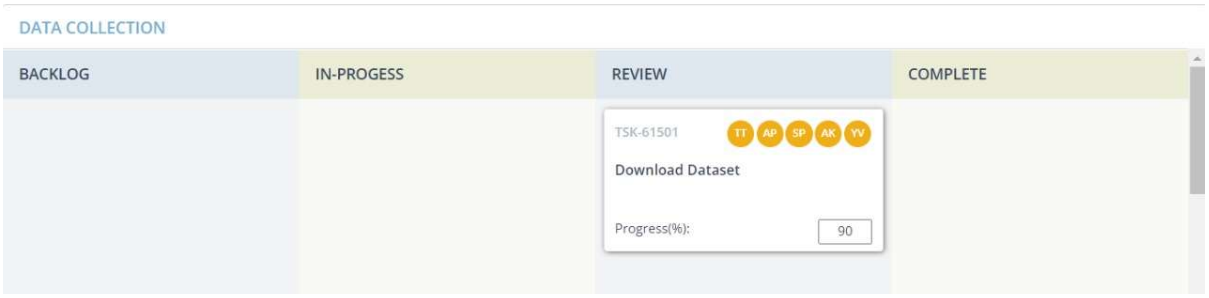


PREREQUISITES

TSK-340476 TT AP SP AK YV
Prior Knowledge
Progress(%): 90

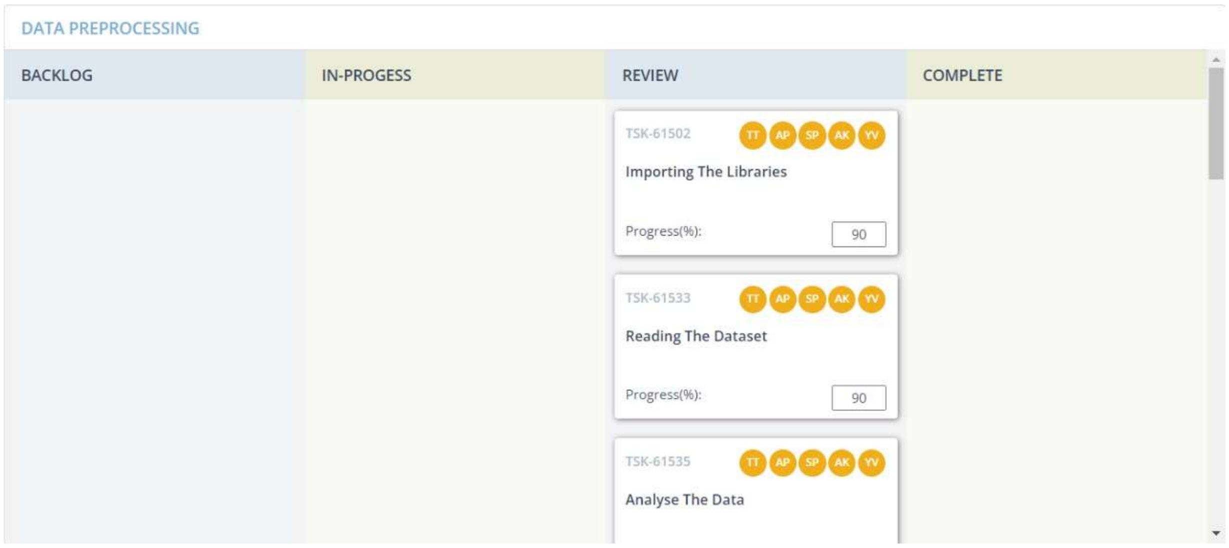
TSK-44907 TT AP SP AK YV
Project Flow
Progress(%): 90

TSK-340477 TT AP SP AK YV
Project Objectives
Progress(%): 90



DATA COLLECTION

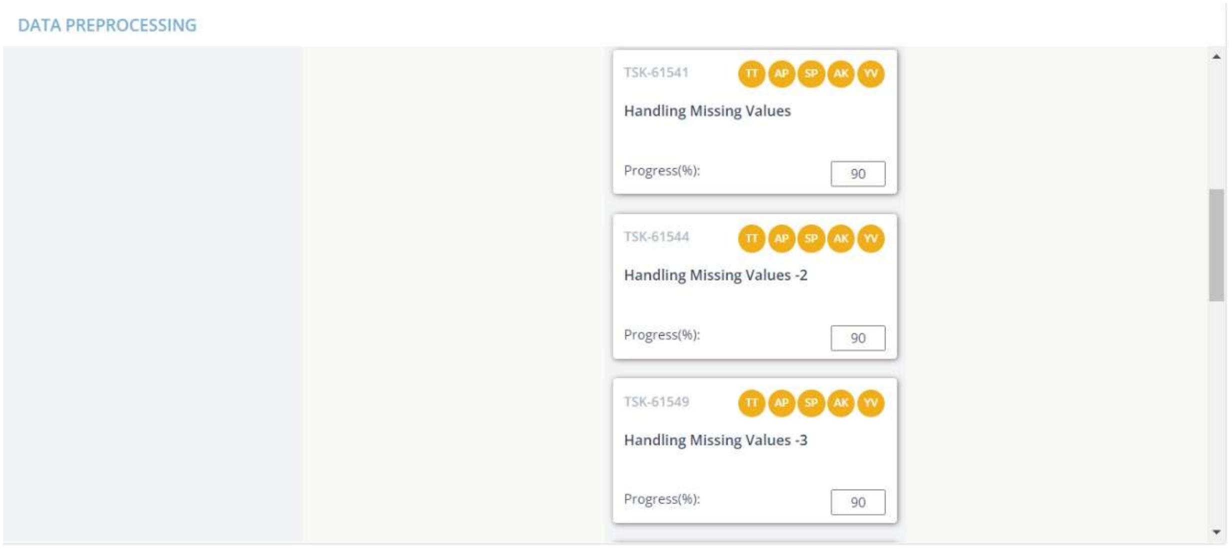
BACKLOG	IN-PROGESS	REVIEW	COMPLETE
		TSK-61501 TT AP SP AK YV Download Dataset Progress(%): 90	



DATA PREPROCESSING

BACKLOG	IN-PROGESS	REVIEW	COMPLETE
		TSK-61502 TT AP SP AK YV Importing The Libraries Progress(%): 90	
		TSK-61533 TT AP SP AK YV Reading The Dataset Progress(%): 90	
		TSK-61535 TT AP SP AK YV Analyse The Data	

PNT2022TMID53565

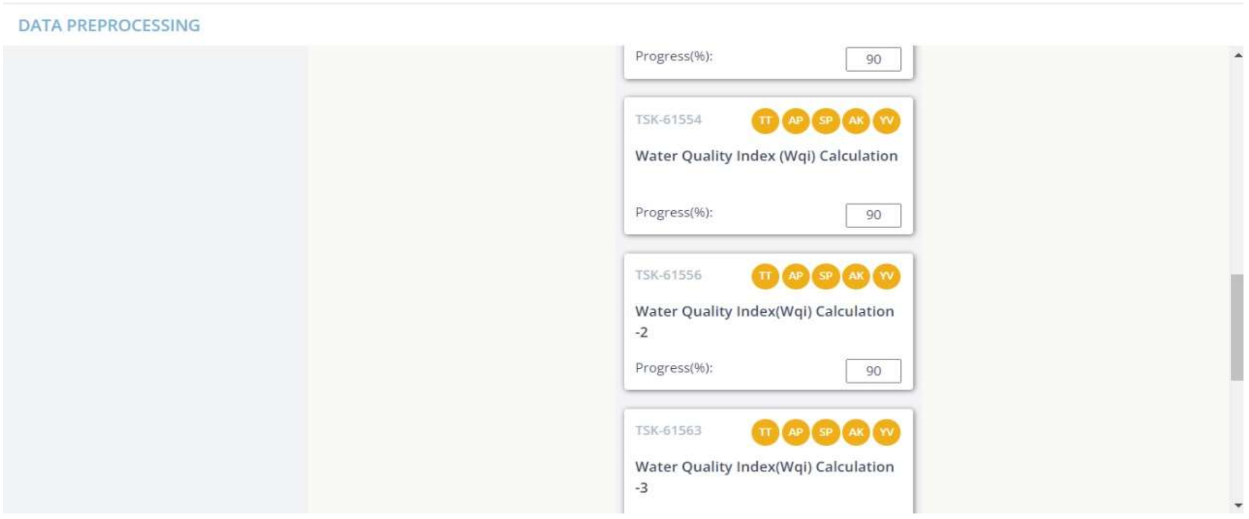


DATA PREPROCESSING

TSK-61541 TT AP SP AK YV
Handling Missing Values
Progress(%): 90

TSK-61544 TT AP SP AK YV
Handling Missing Values -2
Progress(%): 90

TSK-61549 TT AP SP AK YV
Handling Missing Values -3
Progress(%): 90



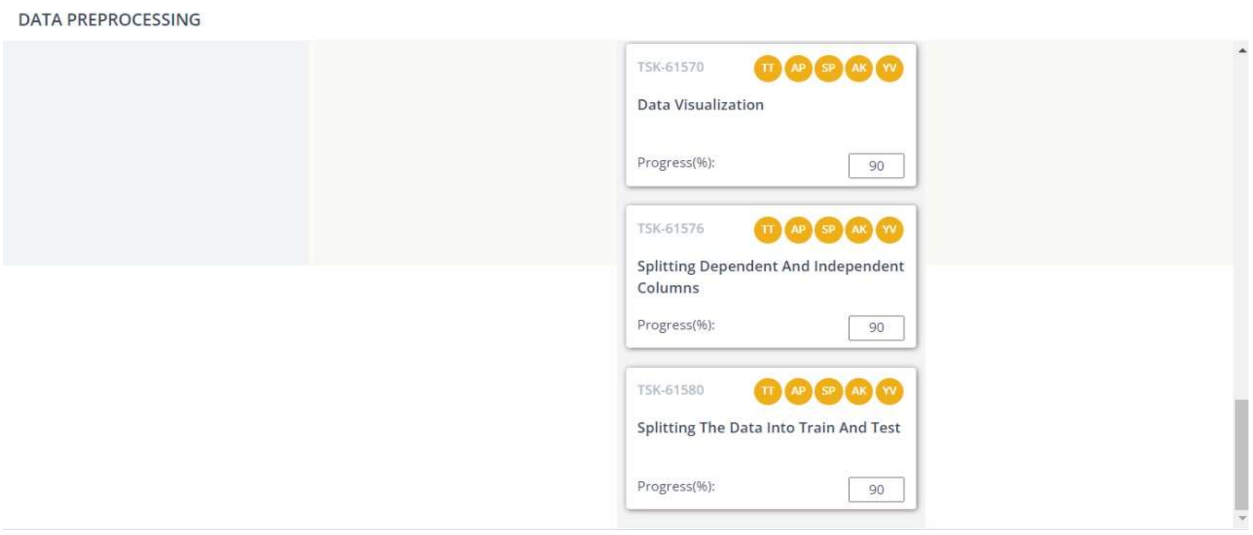
DATA PREPROCESSING

Progress(%): 90

TSK-61554 TT AP SP AK YV
Water Quality Index (Wqi) Calculation
Progress(%): 90

TSK-61556 TT AP SP AK YV
Water Quality Index(Wqi) Calculation -2
Progress(%): 90

TSK-61563 TT AP SP AK YV
Water Quality Index(Wqi) Calculation -3



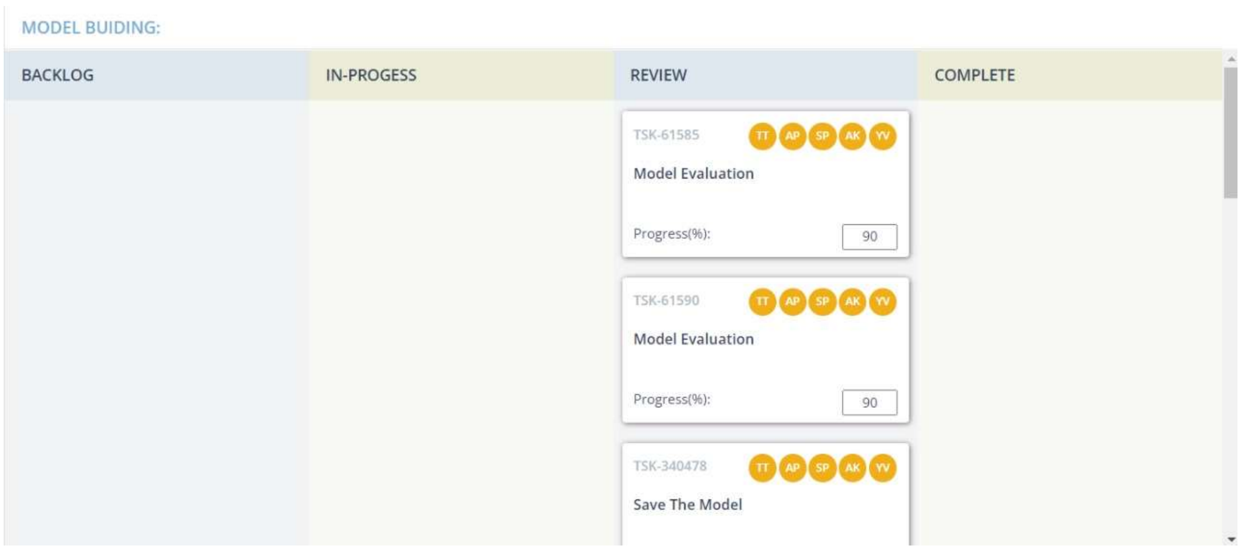
DATA PREPROCESSING

TSK-61570 TT AP SP AK YV
Data Visualization
Progress(%): 90

TSK-61576 TT AP SP AK YV
Splitting Dependent And Independent Columns
Progress(%): 90

TSK-61580 TT AP SP AK YV
Splitting The Data Into Train And Test
Progress(%): 90

PNT2022TMID53565



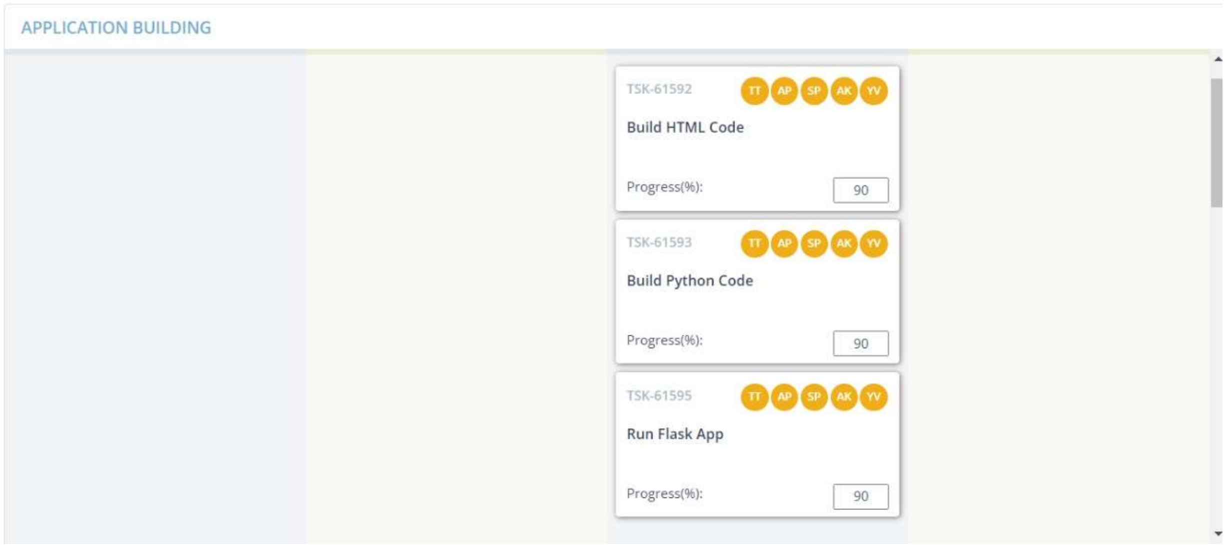
MODEL BUIDING:

BACKLOG | IN-PROGESS | REVIEW | COMPLETE

TSK-61585 TT AP SP AK YV
Model Evaluation
Progress(%): 90

TSK-61590 TT AP SP AK YV
Model Evaluation
Progress(%): 90

TSK-340478 TT AP SP AK YV
Save The Model

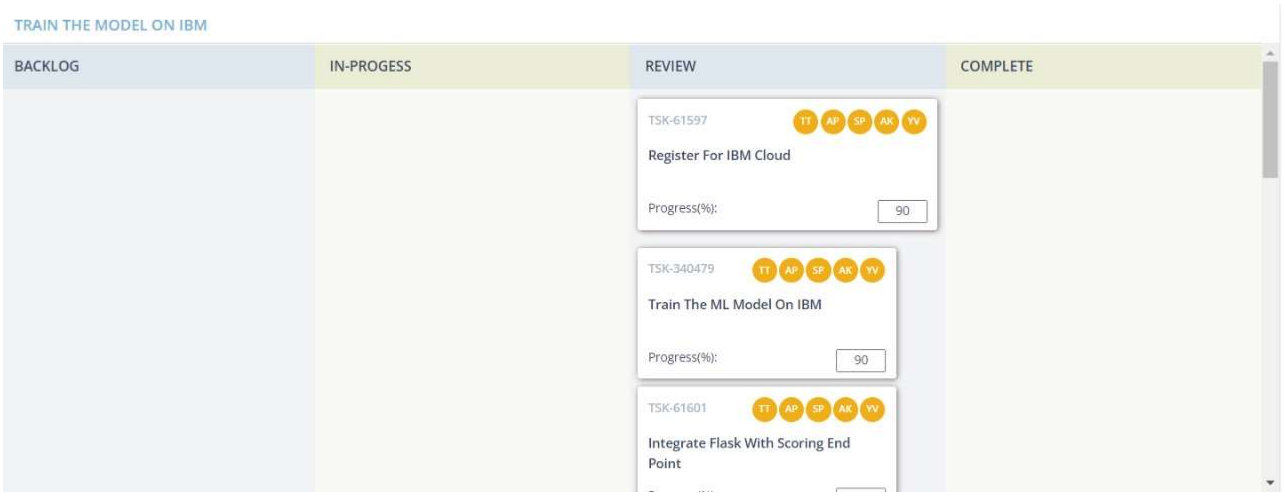


APPLICATION BUILDING

TSK-61592 TT AP SP AK YV
Build HTML Code
Progress(%): 90

TSK-61593 TT AP SP AK YV
Build Python Code
Progress(%): 90

TSK-61595 TT AP SP AK YV
Run Flask App
Progress(%): 90



TRAIN THE MODEL ON IBM

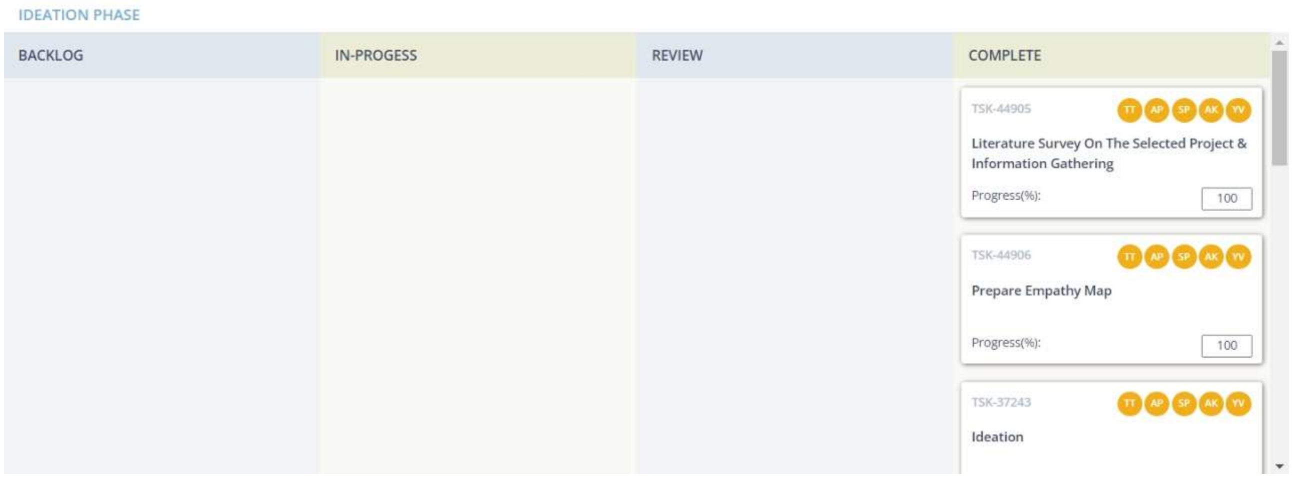
BACKLOG | IN-PROGESS | REVIEW | COMPLETE

TSK-61597 TT AP SP AK YV
Register For IBM Cloud
Progress(%): 90

TSK-340479 TT AP SP AK YV
Train The ML Model On IBM
Progress(%): 90

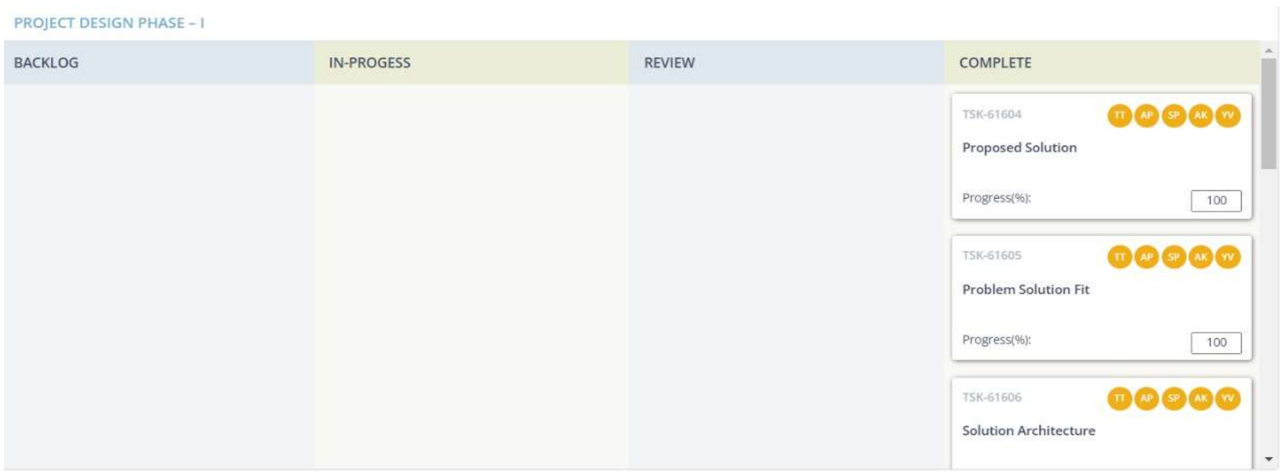
TSK-61601 TT AP SP AK YV
Integrate Flask With Scoring End Point

PNT2022TMID53565



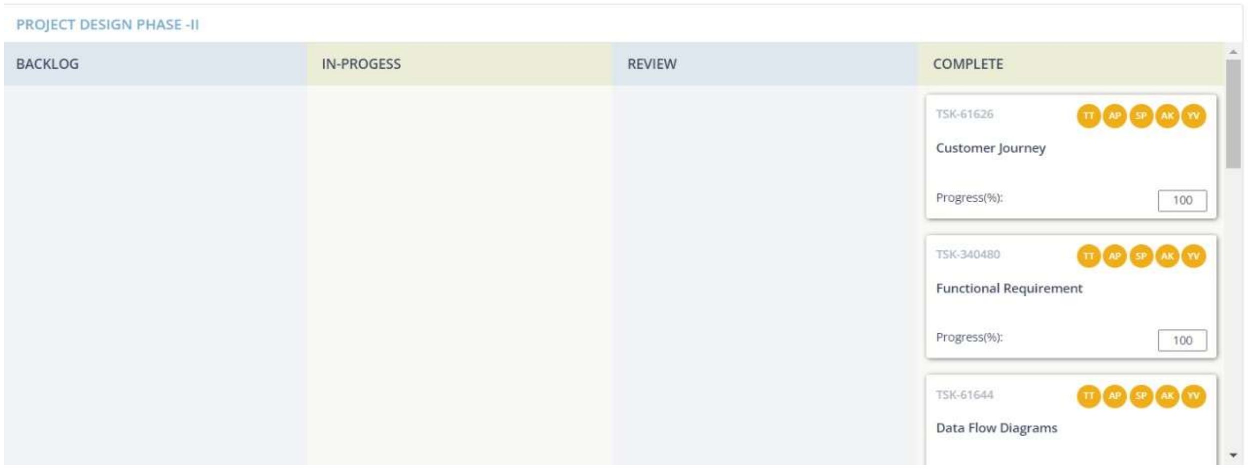
IDEATION PHASE

BACKLOG	IN-PROGESS	REVIEW	COMPLETE
			TSK-44905 TT AP SP AK YV Literature Survey On The Selected Project & Information Gathering Progress(%): 100
			TSK-44906 TT AP SP AK YV Prepare Empathy Map Progress(%): 100
			TSK-37243 TT AP SP AK YV Ideation



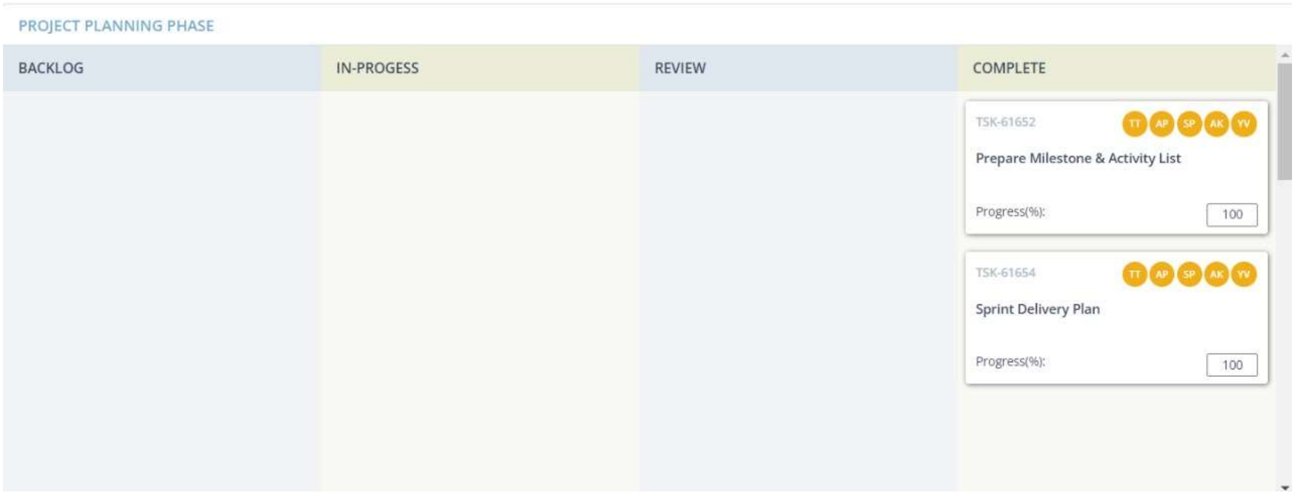
PROJECT DESIGN PHASE – I

BACKLOG	IN-PROGESS	REVIEW	COMPLETE
			TSK-61604 TT AP SP AK YV Proposed Solution Progress(%): 100
			TSK-61605 TT AP SP AK YV Problem Solution Fit Progress(%): 100
			TSK-61606 TT AP SP AK YV Solution Architecture



PROJECT DESIGN PHASE -II

BACKLOG	IN-PROGESS	REVIEW	COMPLETE
			TSK-61626 TT AP SP AK YV Customer Journey Progress(%): 100
			TSK-340480 TT AP SP AK YV Functional Requirement Progress(%): 100
			TSK-61644 TT AP SP AK YV Data Flow Diagrams


PROJECT PLANNING PHASE

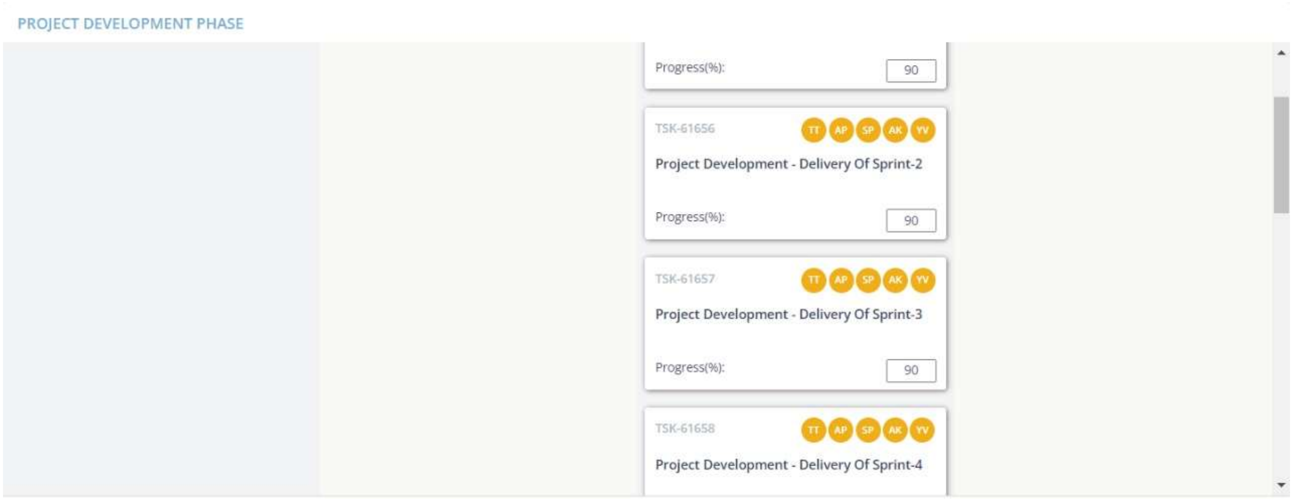
BACKLOG | IN-PROGESS | REVIEW | COMPLETE

TSK-61652

Prepare Milestone & Activity List
Progress(%): 100

TSK-61654

Sprint Delivery Plan
Progress(%): 100


PROJECT DEVELOPMENT PHASE

Progress(%): 90

TSK-61656

Project Development - Delivery Of Sprint-2
Progress(%): 90

TSK-61657

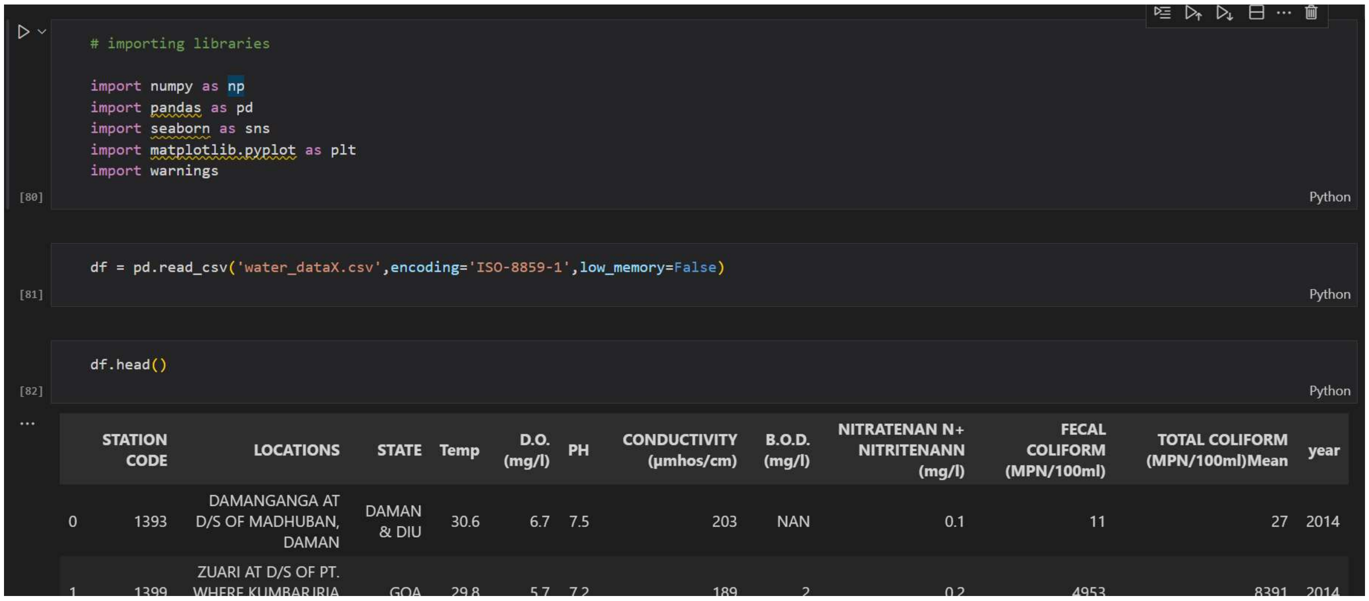
Project Development - Delivery Of Sprint-3
Progress(%): 90

TSK-61658

Project Development - Delivery Of Sprint-4

7.CODING & SOLUTIONING

7.1 Model Prediction



```
# importing libraries

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
```

[80] Python

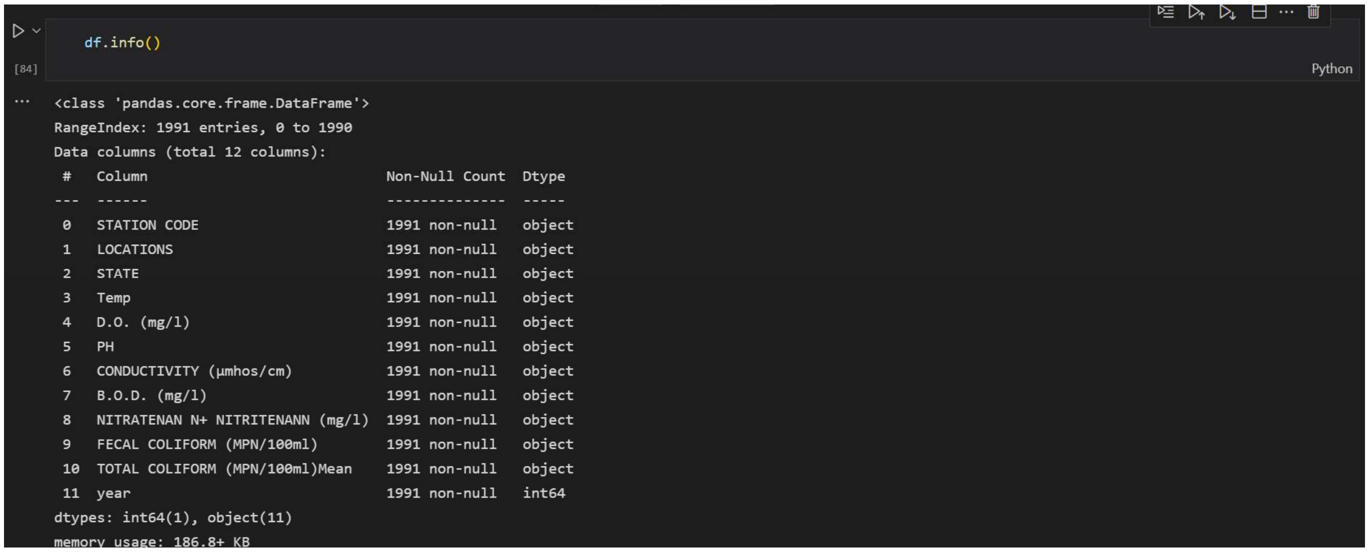
```
df = pd.read_csv('water_dataX.csv',encoding='ISO-8859-1',low_memory=False)
```

[81] Python

```
df.head()
```

[82] Python

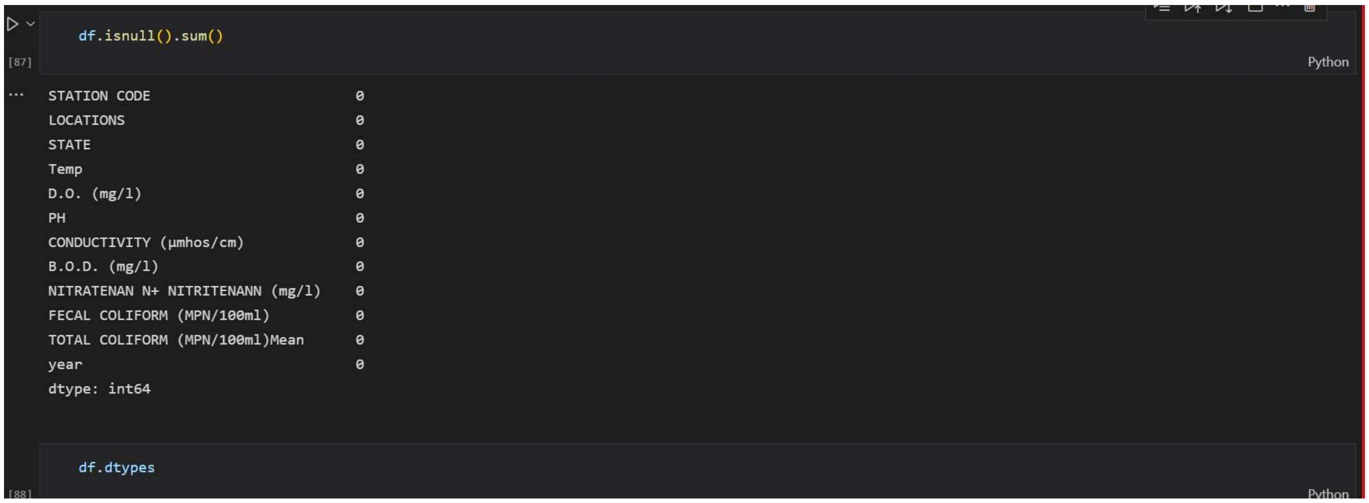
	STATION CODE	LOCATIONS	STATE	Temp	D.O. (mg/l)	PH	CONDUCTIVITY (µmhos/cm)	B.O.D. (mg/l)	NITRATENAN N+ NITRITENANN (mg/l)	FECAL COLIFORM (MPN/100ml)	TOTAL COLIFORM (MPN/100ml)Mean	year
0	1393	DAMANGANGA AT D/S OF MADHUBAN, DAMAN	DAMAN & DIU	30.6	6.7	7.5	203	NAN	0.1	11	27	2014
1	1399	ZUARI AT D/S OF PT. WHERE KUMBARJRIA	GOA	29.8	5.7	7.2	189	2	0.2	4953	8391	2014



```
df.info()
```

[84] Python

```
<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
```

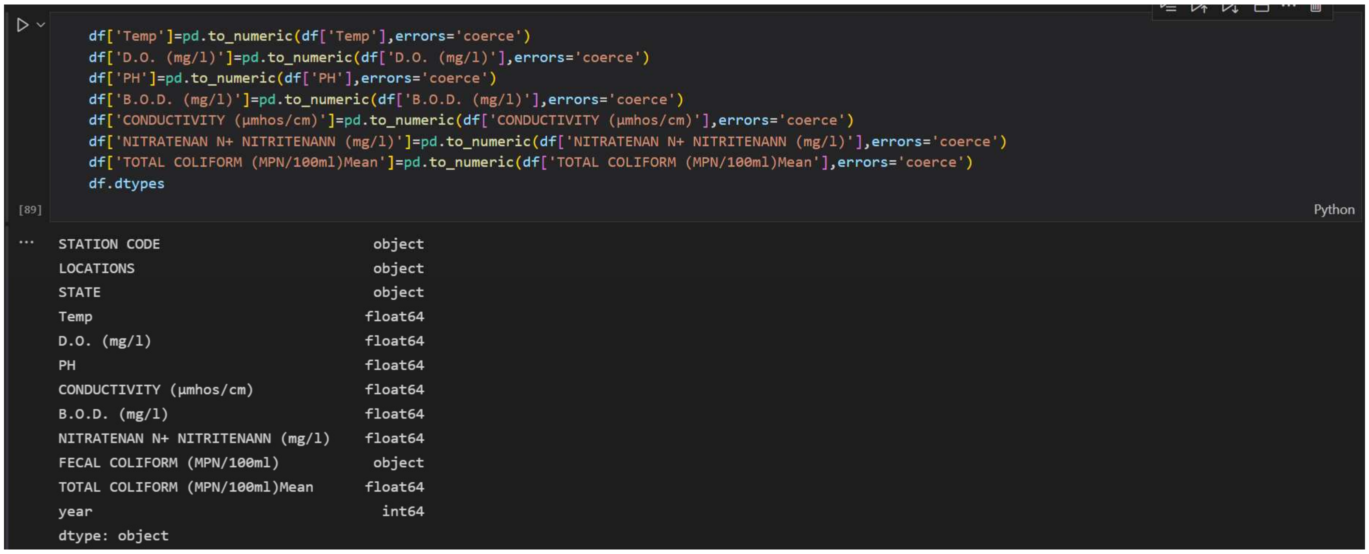


```
df.isnull().sum()
```

[87]

```
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
```

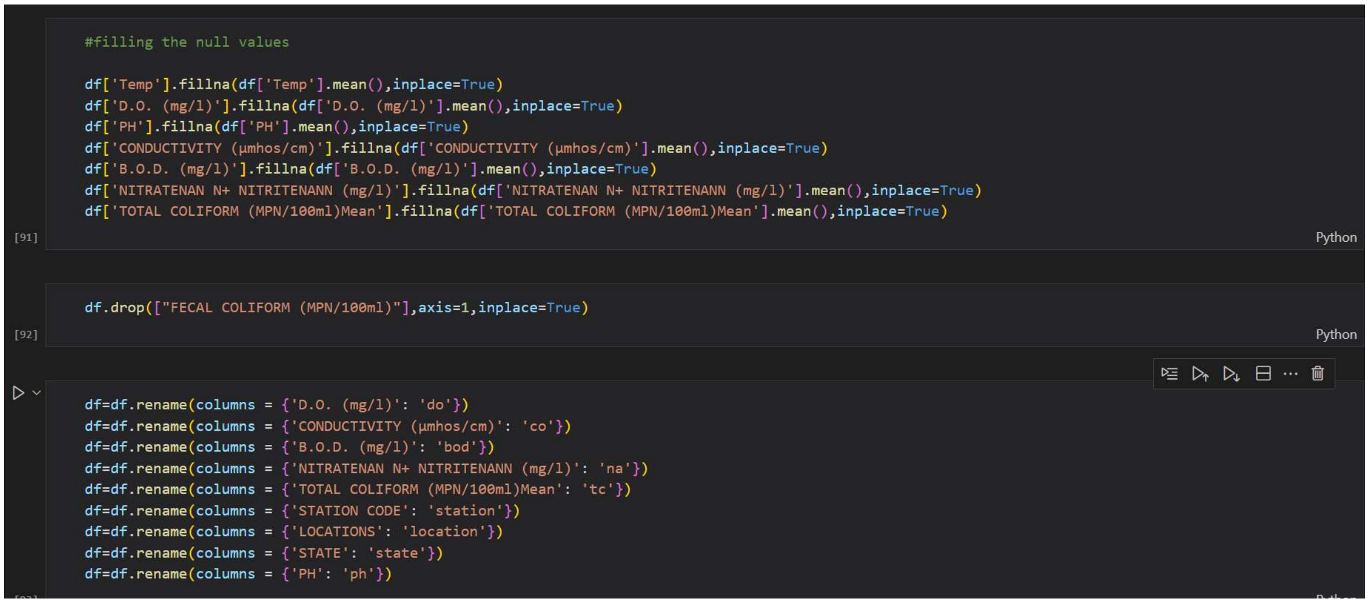
```
df.dtypes
```



```
df['Temp']=pd.to_numeric(df['Temp'],errors='coerce')
df['D.O. (mg/l)']=pd.to_numeric(df['D.O. (mg/l)'],errors='coerce')
df['PH']=pd.to_numeric(df['PH'],errors='coerce')
df['B.O.D. (mg/l)']=pd.to_numeric(df['B.O.D. (mg/l)'],errors='coerce')
df['CONDUCTIVITY (µmhos/cm)']=pd.to_numeric(df['CONDUCTIVITY (µmhos/cm)'],errors='coerce')
df['NITRATENAN N+ NITRITENANN (mg/l)']=pd.to_numeric(df['NITRATENAN N+ NITRITENANN (mg/l)'],errors='coerce')
df['TOTAL COLIFORM (MPN/100ml)Mean']=pd.to_numeric(df['TOTAL COLIFORM (MPN/100ml)Mean'],errors='coerce')
df.dtypes
```

[89]

```
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
```



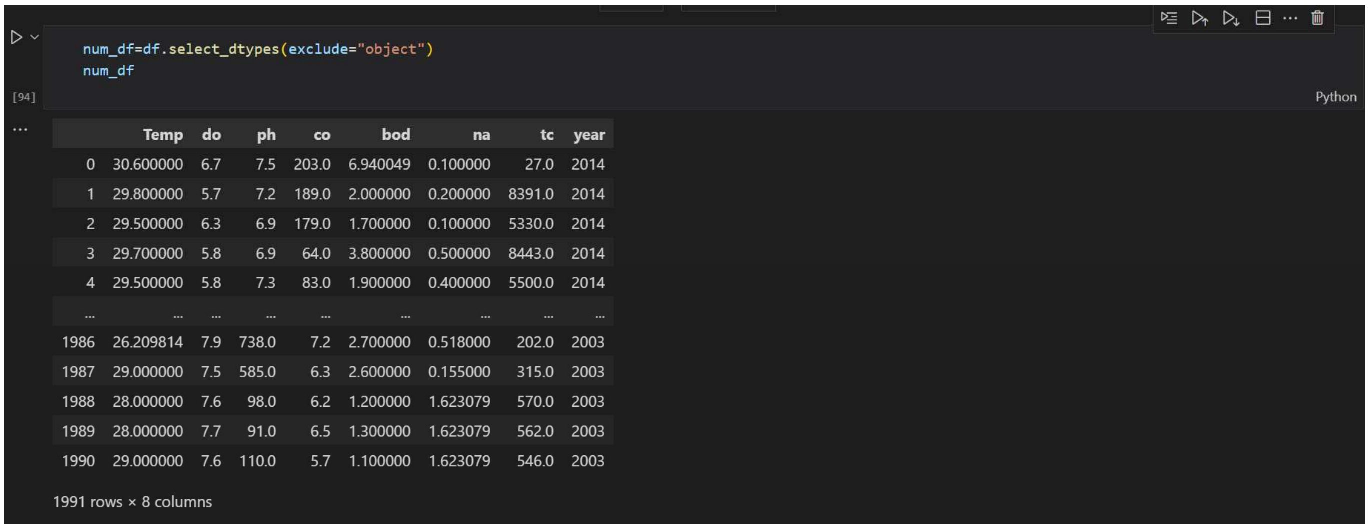
```
#filling the null values

df['Temp'].fillna(df['Temp'].mean(),inplace=True)
df['D.O. (mg/l)'].fillna(df['D.O. (mg/l)'].mean(),inplace=True)
df['PH'].fillna(df['PH'].mean(),inplace=True)
df['CONDUCTIVITY (µmhos/cm)'].fillna(df['CONDUCTIVITY (µmhos/cm)'].mean(),inplace=True)
df['B.O.D. (mg/l)'].fillna(df['B.O.D. (mg/l)'].mean(),inplace=True)
df['NITRATENAN N+ NITRITENANN (mg/l)'].fillna(df['NITRATENAN N+ NITRITENANN (mg/l)'].mean(),inplace=True)
df['TOTAL COLIFORM (MPN/100ml)Mean'].fillna(df['TOTAL COLIFORM (MPN/100ml)Mean'].mean(),inplace=True)
```

[91]

```
df.drop(["FECAL COLIFORM (MPN/100ml)"],axis=1,inplace=True)
```

[92]

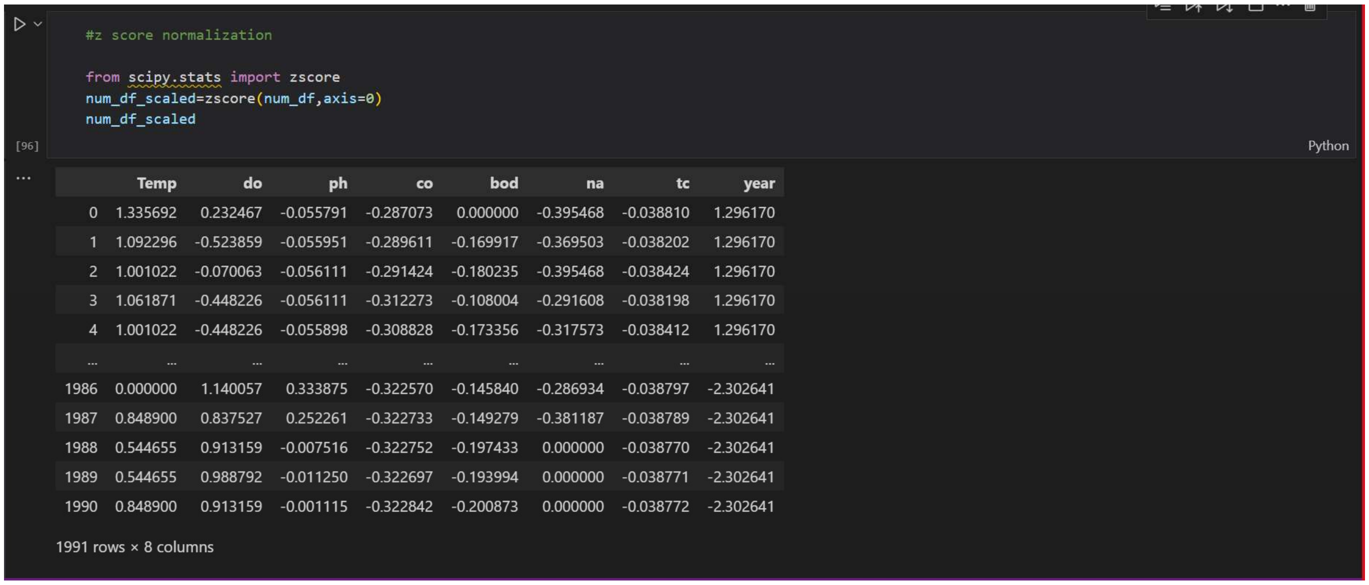
```
df=df.rename(columns = {'D.O. (mg/l)': 'do'})
df=df.rename(columns = {'CONDUCTIVITY (µmhos/cm)': 'co'})
df=df.rename(columns = {'B.O.D. (mg/l)': 'bod'})
df=df.rename(columns = {'NITRATENAN N+ NITRITENANN (mg/l)': 'na'})
df=df.rename(columns = {'TOTAL COLIFORM (MPN/100ml)Mean': 'tc'})
df=df.rename(columns = {'STATION CODE': 'station'})
df=df.rename(columns = {'LOCATIONS': 'location'})
df=df.rename(columns = {'STATE': 'state'})
df=df.rename(columns = {'PH': 'ph'})
```



```
num_df=df.select_dtypes(exclude="object")
num_df
```

[94] Python

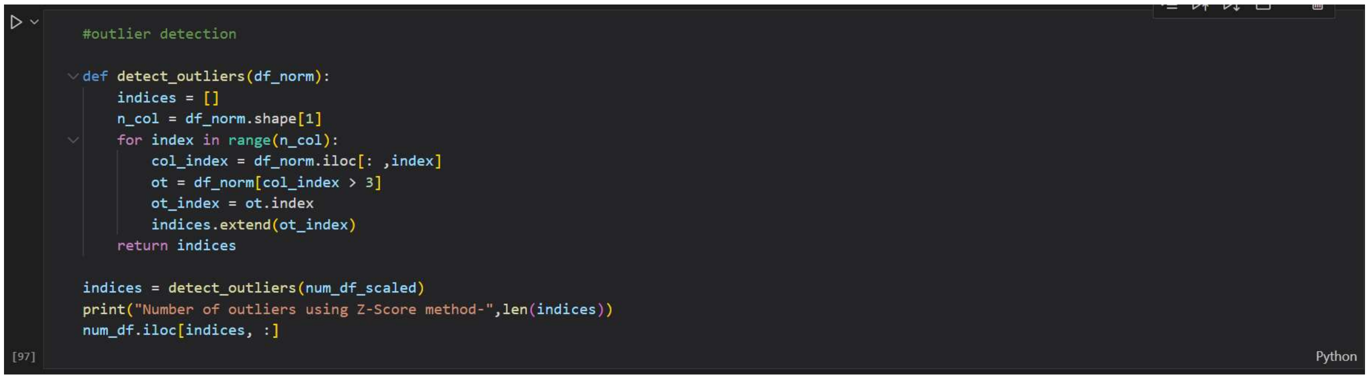
```
           Temp   do     ph     co       bod        na      tc  year
0     30.600000  6.7    7.5  203.0  6.940049  0.100000    27.0  2014
1     29.800000  5.7    7.2  189.0  2.000000  0.200000  8391.0  2014
2     29.500000  6.3    6.9  179.0  1.700000  0.100000  5330.0  2014
3     29.700000  5.8    6.9   64.0  3.800000  0.500000  8443.0  2014
4     29.500000  5.8    7.3   83.0  1.900000  0.400000  5500.0  2014
...         ...  ...    ...    ...       ...       ...     ...   ...
1986  26.209814  7.9  738.0    7.2  2.700000  0.518000   202.0  2003
1987  29.000000  7.5  585.0    6.3  2.600000  0.155000   315.0  2003
1988  28.000000  7.6   98.0    6.2  1.200000  1.623079   570.0  2003
1989  28.000000  7.7   91.0    6.5  1.300000  1.623079   562.0  2003
1990  29.000000  7.6  110.0    5.7  1.100000  1.623079   546.0  2003

1991 rows × 8 columns
```



```
#z score normalization

from scipy.stats import zscore
num_df_scaled=zscore(num_df,axis=0)
num_df_scaled
```

[96] Python

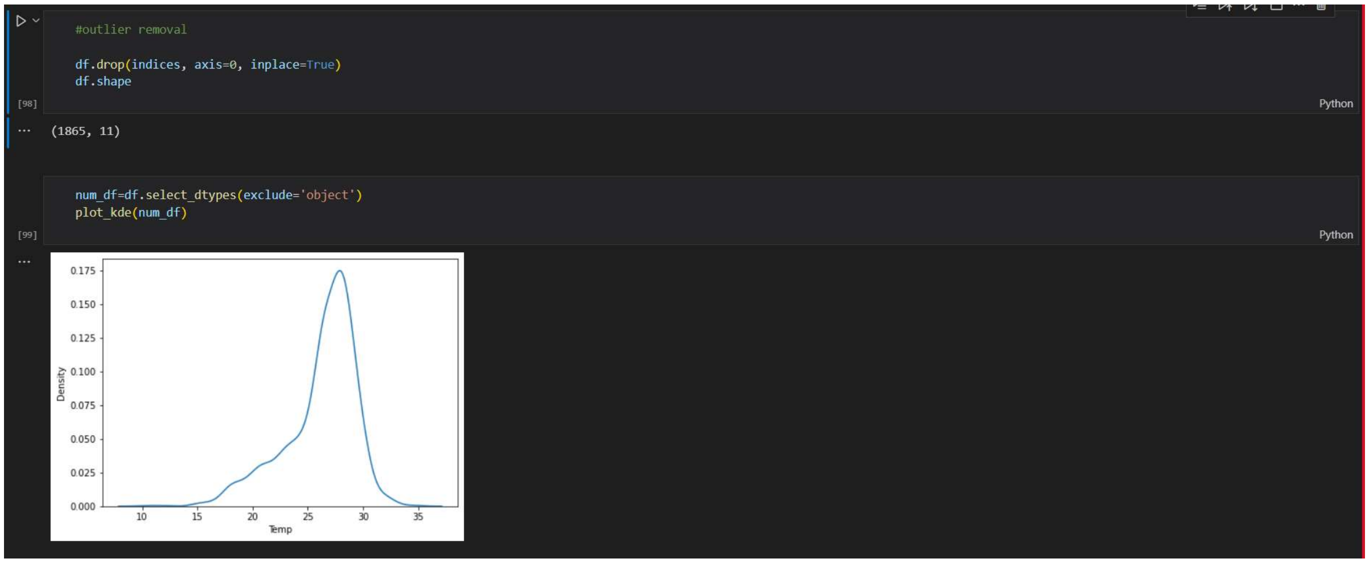
```
          Temp        do         ph         co        bod        na         tc       year
0     1.335692  0.232467  -0.055791  -0.287073   0.000000 -0.395468  -0.038810   1.296170
1     1.092296 -0.523859  -0.055951  -0.289611  -0.169917 -0.369503  -0.038202   1.296170
2     1.001022 -0.070063  -0.056111  -0.291424  -0.180235 -0.395468  -0.038424   1.296170
3     1.061871 -0.448226  -0.056111  -0.312273  -0.108004 -0.291608  -0.038198   1.296170
4     1.001022 -0.448226  -0.055898  -0.308828  -0.173356 -0.317573  -0.038412   1.296170
...        ...       ...        ...        ...        ...       ...        ...        ...
1986  0.000000  1.140057   0.333875  -0.322570  -0.145840 -0.286934  -0.038797  -2.302641
1987  0.848900  0.837527   0.252261  -0.322733  -0.149279 -0.381187  -0.038789  -2.302641
1988  0.544655  0.913159  -0.007516  -0.322752  -0.197433  0.000000  -0.038770  -2.302641
1989  0.544655  0.988792  -0.011250  -0.322697  -0.193994  0.000000  -0.038771  -2.302641
1990  0.848900  0.913159  -0.001115  -0.322842  -0.200873  0.000000  -0.038772  -2.302641

1991 rows × 8 columns
```



```
#outlier detection

def detect_outliers(df_norm):
    indices = []
    n_col = df_norm.shape[1]
    for index in range(n_col):
        col_index = df_norm.iloc[: ,index]
        ot = df_norm[col_index > 3]
        ot_index = ot.index
        indices.extend(ot_index)
    return indices

indices = detect_outliers(num_df_scaled)
print("Number of outliers using Z-Score method-",len(indices))
num_df.iloc[indices, :]
```

[97] Python

PNT2022TMID53565



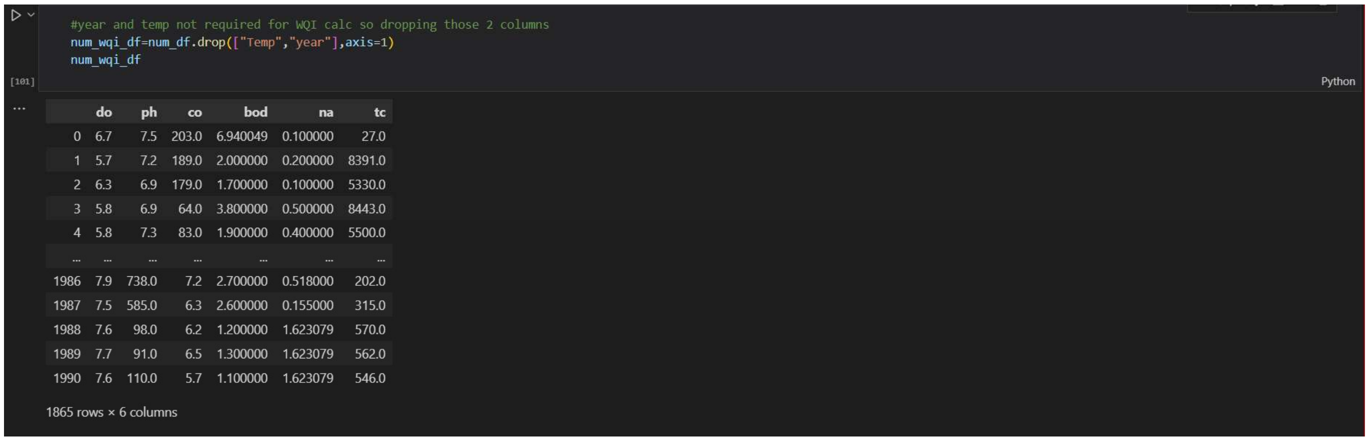
```
#outlier removal

df.drop(indices, axis=0, inplace=True)
df.shape
```

[98] Python

```
(1865, 11)
```

```
num_df=df.select_dtypes(exclude='object')
plot_kde(num_df)
```

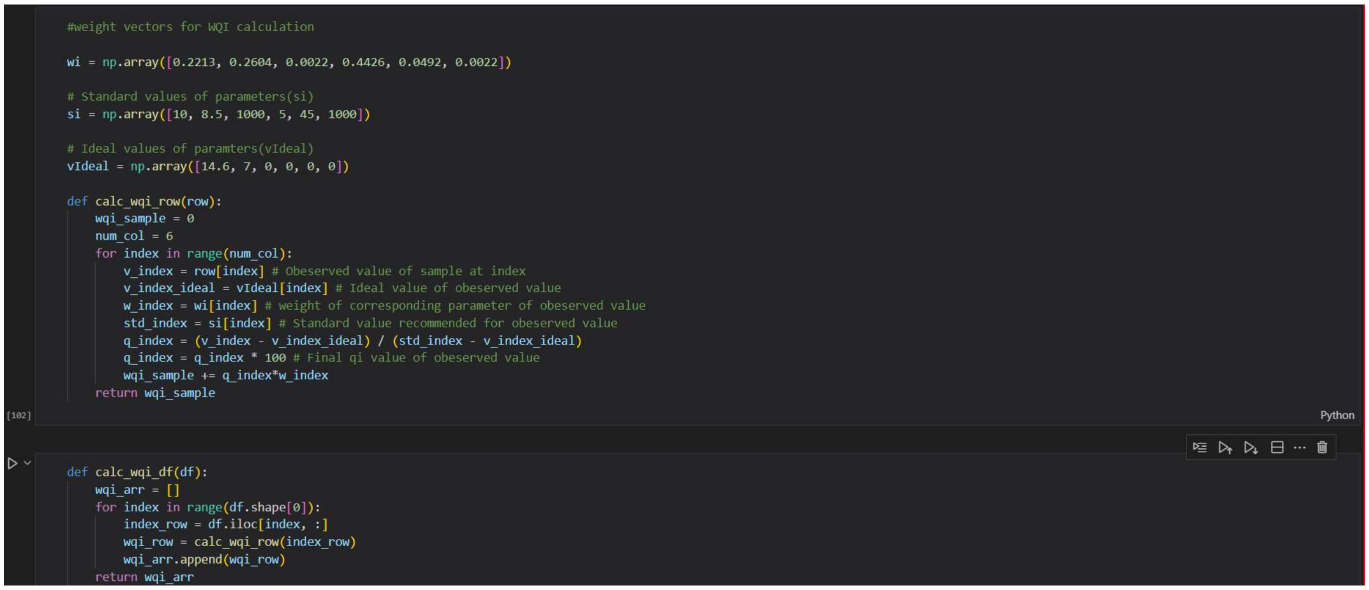
[99] Python



```
#year and temp not required for WQI calc so dropping those 2 columns
num_wqi_df=num_df.drop(["Temp","year"],axis=1)
num_wqi_df
```

[101] Python

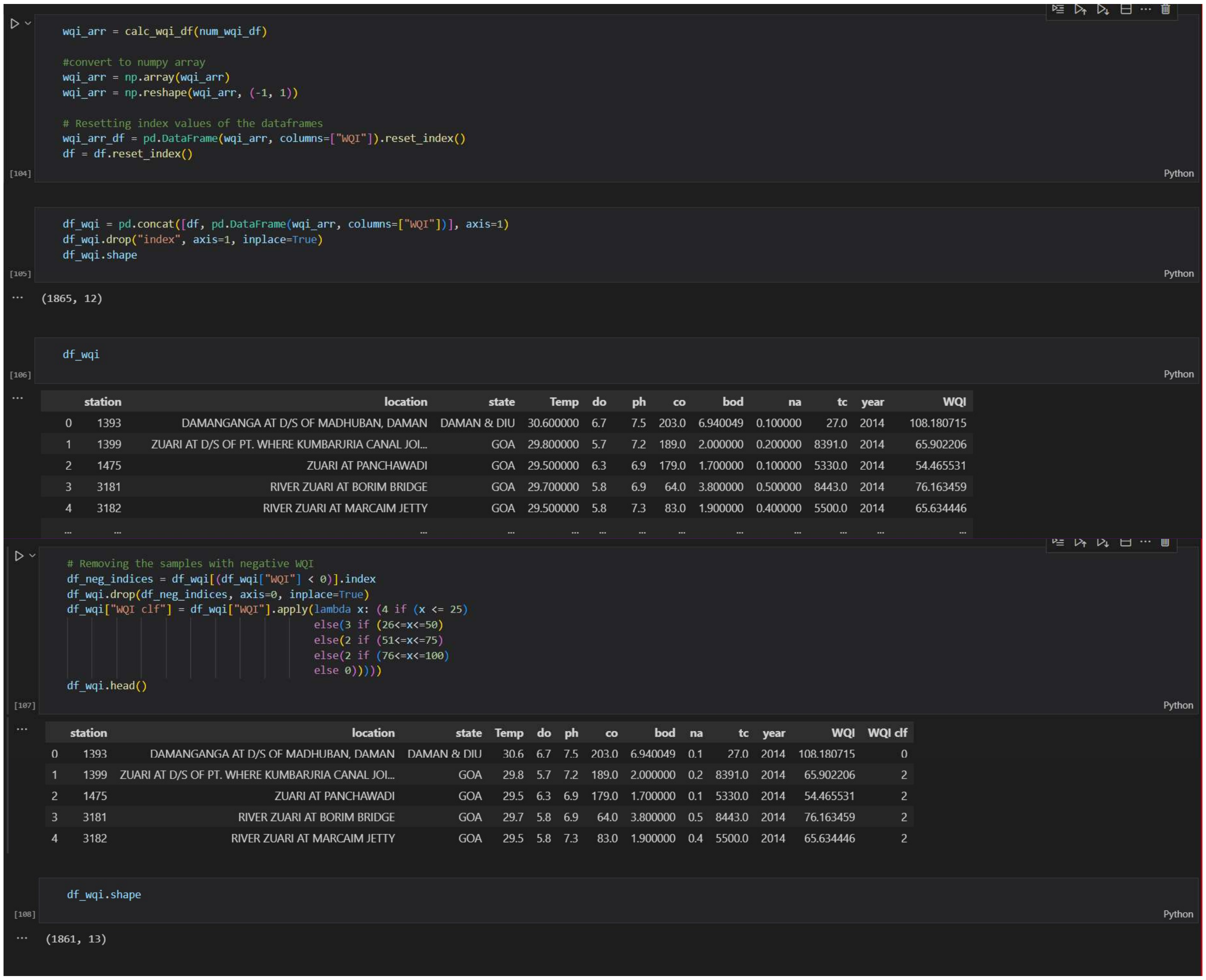
```
        do     ph     co       bod        na      tc
0      6.7    7.5  203.0  6.940049  0.100000    27.0
1      5.7    7.2  189.0  2.000000  0.200000  8391.0
2      6.3    6.9  179.0  1.700000  0.100000  5330.0
3      5.8    6.9   64.0  3.800000  0.500000  8443.0
4      5.8    7.3   83.0  1.900000  0.400000  5500.0
...    ...    ...    ...       ...       ...     ...
1986   7.9  738.0    7.2  2.700000  0.518000   202.0
1987   7.5  585.0    6.3  2.600000  0.155000   315.0
1988   7.6   98.0    6.2  1.200000  1.623079   570.0
1989   7.7   91.0    6.5  1.300000  1.623079   562.0
1990   7.6  110.0    5.7  1.100000  1.623079   546.0

1865 rows × 6 columns
```



```
#weight vectors for WQI calculation

wi = np.array([0.2213, 0.2604, 0.0022, 0.4426, 0.0492, 0.0022])

# Standard values of parameters(si)
si = np.array([10, 8.5, 1000, 5, 45, 1000])

# Ideal values of paramters(vIdeal)
vIdeal = np.array([14.6, 7, 0, 0, 0, 0])

def calc_wqi_row(row):
    wqi_sample = 0
    num_col = 6
    for index in range(num_col):
        v_index = row[index] # Obeserved value of sample at index
        v_index_ideal = vIdeal[index] # Ideal value of obeserved value
        w_index = wi[index] # weight of corresponding parameter of obeserved value
        std_index = si[index] # Standard value recommended for obeserved value
        q_index = (v_index - v_index_ideal) / (std_index - v_index_ideal)
        q_index = q_index * 100 # Final qi value of obeserved value
        wqi_sample += q_index*w_index
    return wqi_sample
```

[102] Python

```
def calc_wqi_df(df):
    wqi_arr = []
    for index in range(df.shape[0]):
        index_row = df.iloc[index, :]
        wqi_row = calc_wqi_row(index_row)
        wqi_arr.append(wqi_row)
    return wqi_arr
```



```
[104]
wqi_arr = calc_wqi_df(num_wqi_df)

#convert to numpy array
wqi_arr = np.array(wqi_arr)
wqi_arr = np.reshape(wqi_arr, (-1, 1))

# Resetting index values of the dataframes
wqi_arr_df = pd.DataFrame(wqi_arr, columns=["WQI"]).reset_index()
df = df.reset_index()
```

```
[105]
df_wqi = pd.concat([df, pd.DataFrame(wqi_arr, columns=["WQI"])], axis=1)
df_wqi.drop("index", axis=1, inplace=True)
df_wqi.shape
```

```
(1865, 12)
```

```
[106]
df_wqi
```

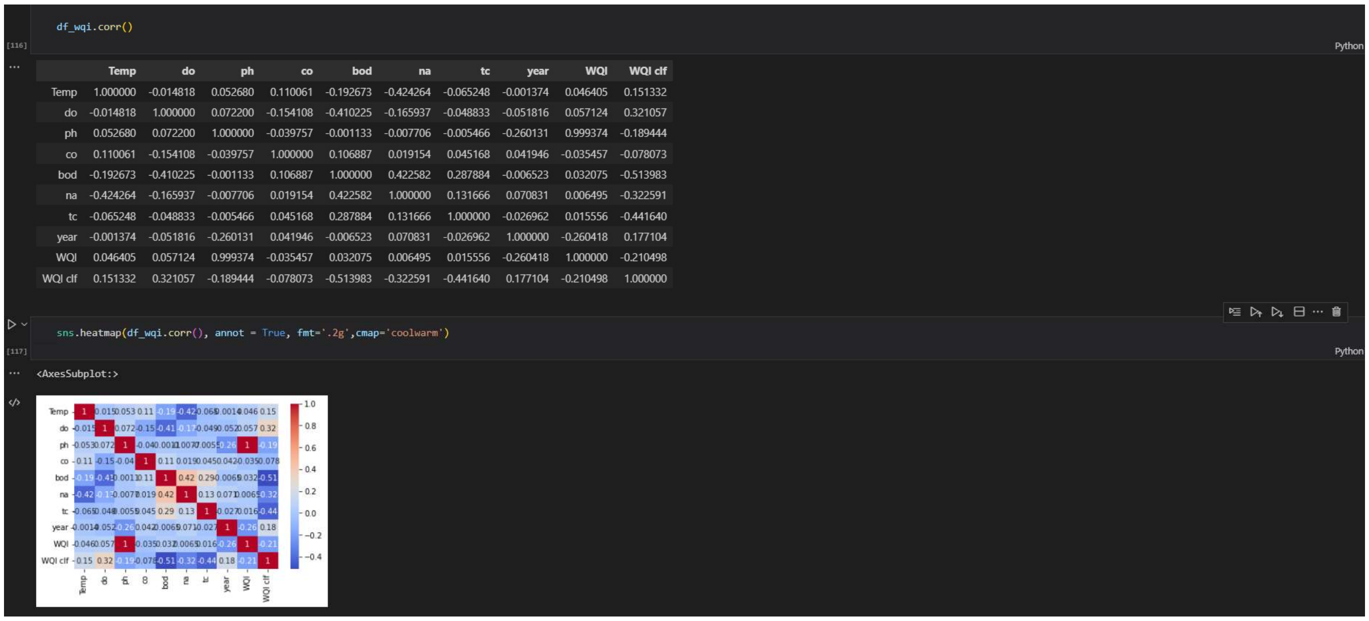
	station	location	state	Temp	do	ph	co	bod	na	tc	year	WQI
0	1393	DAMANGANGA AT D/S OF MADHUBAN, DAMAN	DAMAN & DIU	30.600000	6.7	7.5	203.0	6.940049	0.100000	27.0	2014	108.180715
1	1399	ZUARI AT D/S OF PT. WHERE KUMBARJRIA CANAL JOI...	GOA	29.800000	5.7	7.2	189.0	2.000000	0.200000	8391.0	2014	65.902206
2	1475	ZUARI AT PANCHAWADI	GOA	29.500000	6.3	6.9	179.0	1.700000	0.100000	5330.0	2014	54.465531
3	3181	RIVER ZUARI AT BORIM BRIDGE	GOA	29.700000	5.8	6.9	64.0	3.800000	0.500000	8443.0	2014	76.163459
4	3182	RIVER ZUARI AT MARCAIM JETTY	GOA	29.500000	5.8	7.3	83.0	1.900000	0.400000	5500.0	2014	65.634446
...	...	...	...	...	...	...	...	...	...	...	...	...

```
[107]
# Removing the samples with negative WQI
df_neg_indices = df_wqi[(df_wqi["WQI"] < 0)].index
df_wqi.drop(df_neg_indices, axis=0, inplace=True)
df_wqi["WQI clf"] = df_wqi["WQI"].apply(lambda x: (4 if (x <= 25)
                                        else(3 if (26<=x<=50)
                                        else(2 if (51<=x<=75)
                                        else(2 if (76<=x<=100)
                                        else 0)))))
df_wqi.head()
```

	station	location	state	Temp	do	ph	co	bod	na	tc	year	WQI	WQI clf
0	1393	DAMANGANGA AT D/S OF MADHUBAN, DAMAN	DAMAN & DIU	30.6	6.7	7.5	203.0	6.940049	0.1	27.0	2014	108.180715	0
1	1399	ZUARI AT D/S OF PT. WHERE KUMBARJRIA CANAL JOI...	GOA	29.8	5.7	7.2	189.0	2.000000	0.2	8391.0	2014	65.902206	2
2	1475	ZUARI AT PANCHAWADI	GOA	29.5	6.3	6.9	179.0	1.700000	0.1	5330.0	2014	54.465531	2
3	3181	RIVER ZUARI AT BORIM BRIDGE	GOA	29.7	5.8	6.9	64.0	3.800000	0.5	8443.0	2014	76.163459	2
4	3182	RIVER ZUARI AT MARCAIM JETTY	GOA	29.5	5.8	7.3	83.0	1.900000	0.4	5500.0	2014	65.634446	2

```
[108]
df_wqi.shape
```

```
(1861, 13)
```



```
df_wqi.corr()
```

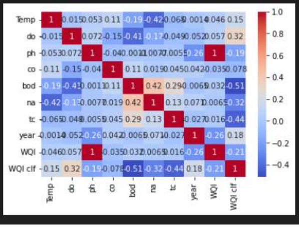
[116]

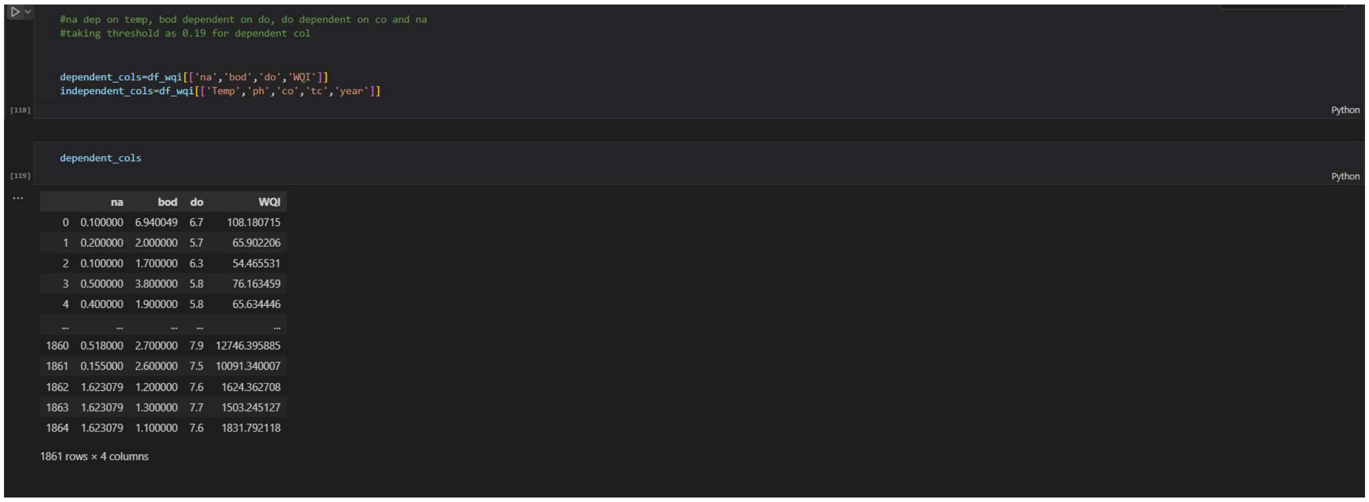
	Temp	do	ph	co	bod	na	tc	year	WQI	WQI clf
Temp	1.000000	-0.014818	0.052680	0.110061	-0.192673	-0.424264	-0.065248	-0.001374	0.046405	0.151332
do	-0.014818	1.000000	0.072200	-0.154108	-0.410225	-0.165937	-0.048833	-0.051816	0.057124	0.321057
ph	0.052680	0.072200	1.000000	-0.039757	-0.001133	-0.007706	-0.005466	-0.260131	0.999374	-0.189444
co	0.110061	-0.154108	-0.039757	1.000000	0.106887	0.019154	0.045168	0.041946	-0.035457	-0.078073
bod	-0.192673	-0.410225	-0.001133	0.106887	1.000000	0.422582	0.287884	-0.006523	0.032075	-0.513983
na	-0.424264	-0.165937	-0.007706	0.019154	0.422582	1.000000	0.131666	0.070831	0.006495	-0.322591
tc	-0.065248	-0.048833	-0.005466	0.045168	0.287884	0.131666	1.000000	-0.026962	0.015556	-0.441640
year	-0.001374	-0.051816	-0.260131	0.041946	-0.006523	0.070831	-0.026962	1.000000	-0.260418	0.177104
WQI	0.046405	0.057124	0.999374	-0.035457	0.032075	0.006495	0.015556	-0.260418	1.000000	-0.210498
WQI clf	0.151332	0.321057	-0.189444	-0.078073	-0.513983	-0.322591	-0.441640	0.177104	-0.210498	1.000000

```
sns.heatmap(df_wqi.corr(), annot = True, fmt='.2g',cmap='coolwarm')
```

[117]

```
<AxesSubplot:>
```





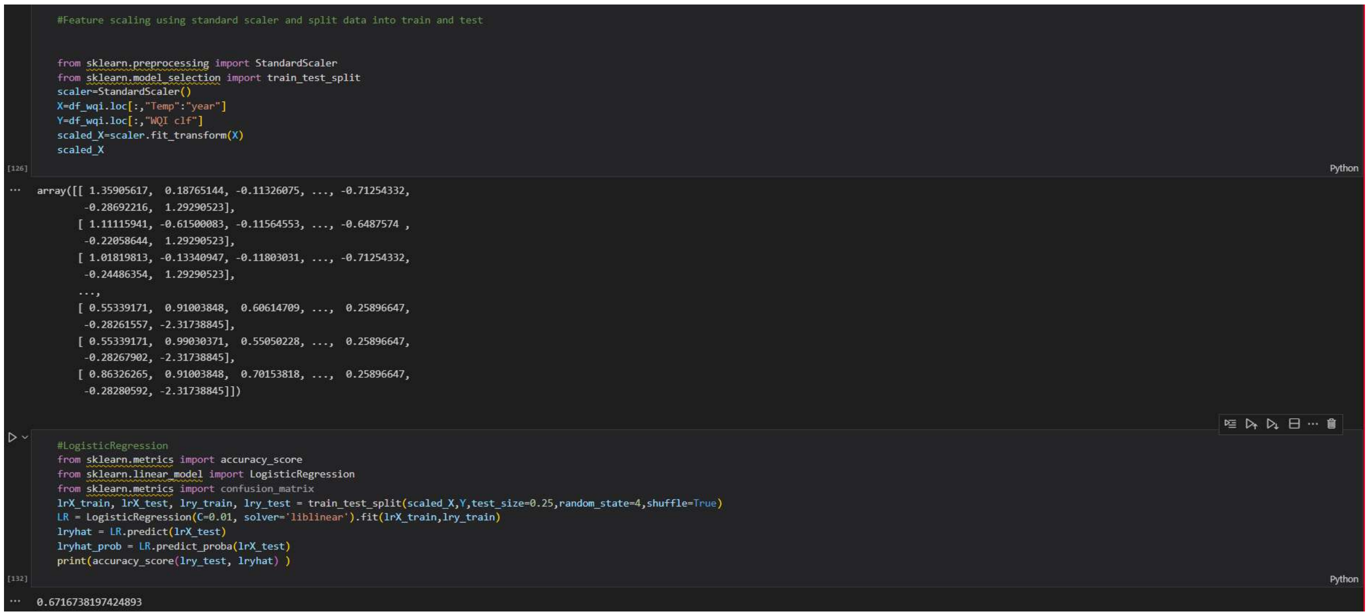
```
#na dep on temp, bod dependent on do, do dependent on co and na
#taking threshold as 0.19 for dependent col


dependent_cols=df_wqi[['na','bod','do','WQI']]
independent_cols=df_wqi[['Temp','ph','co','tc','year']]
```

[118]

```
dependent_cols
```

[119]

```
          na       bod   do           WQI
0   0.100000  6.940049  6.7    108.180715
1   0.200000  2.000000  5.7     65.902206
2   0.100000  1.700000  6.3     54.465531
3   0.500000  3.800000  5.8     76.163459
4   0.400000  1.900000  5.8     65.634446
...      ...       ...  ...           ...
1860 0.518000 2.700000  7.9  12746.395885
1861 0.155000 2.600000  7.5  10091.340007
1862 1.623079 1.200000  7.6   1624.362708
1863 1.623079 1.300000  7.7   1503.245127
1864 1.623079 1.100000  7.6   1831.792118

1861 rows × 4 columns
```

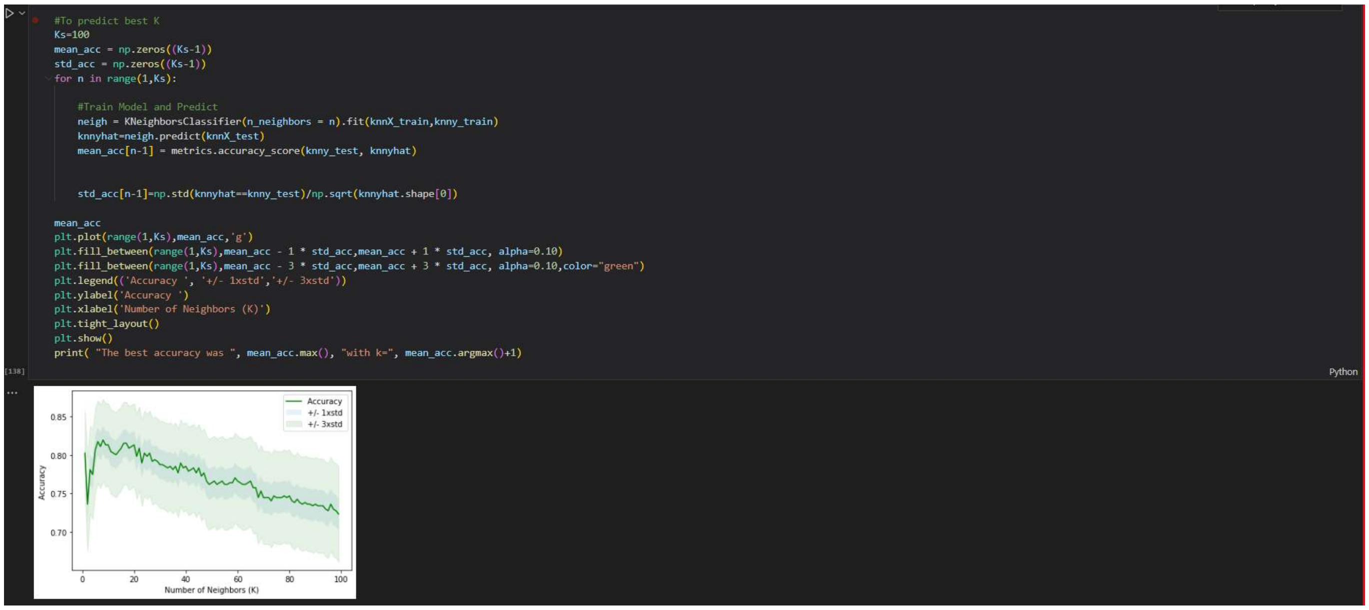


```
#Feature scaling using standard scaler and split data into train and test


from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
scaler=StandardScaler()
X=df_wqi.loc[:,"Temp":"year"]
Y=df_wqi.loc[:,"WQI clf"]
scaled_X=scaler.fit_transform(X)
scaled_X
```

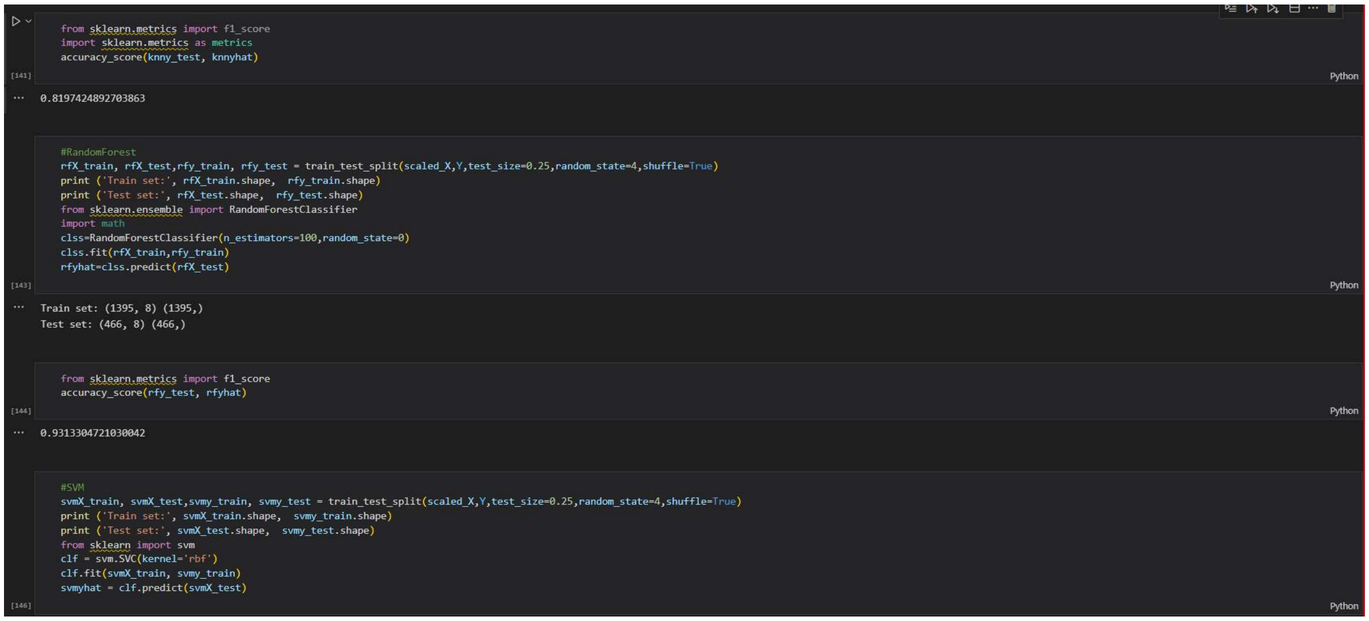
```
array([[ 1.35905617,  0.18765144, -0.11326075, ..., -0.71254332,
        -0.28692216,  1.29290523],
       [ 1.11115941, -0.61500083, -0.11564553, ..., -0.6487574 ,
        -0.22058644,  1.29290523],
       [ 1.01819813, -0.13340947, -0.11803031, ..., -0.71254332,
        -0.24486354,  1.29290523],
       ...,
       [ 0.55339171,  0.91003848,  0.60614709, ...,  0.25896647,
        -0.28261557, -2.31738845],
       [ 0.55339171,  0.99030371,  0.55050228, ...,  0.25896647,
        -0.28267902, -2.31738845],
       [ 0.86326265,  0.91003848,  0.70153818, ...,  0.25896647,
        -0.28280592, -2.31738845]])
```

```
#LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
lrX_train, lrX_test, lry_train, lry_test = train_test_split(scaled_X,Y,test_size=0.25,random_state=4,shuffle=True)
LR = LogisticRegression(C=0.01, solver='liblinear').fit(lrX_train,lry_train)
lryhat = LR.predict(lrX_test)
lryhat_prob = LR.predict_proba(lrX_test)
print(accuracy_score(lry_test, lryhat) )
```

```
0.6716738197424893
```



```
#To predict best K
Ks=100
mean_acc = np.zeros((Ks-1))
std_acc = np.zeros((Ks-1))
for n in range(1,Ks):

    #Train Model and Predict
    neigh = KNeighborsClassifier(n_neighbors = n).fit(knnX_train,knny_train)
    knnyhat=neigh.predict(knnX_test)
    mean_acc[n-1] = metrics.accuracy_score(knny_test, knnyhat)


    std_acc[n-1]=np.std(knnyhat==knny_test)/np.sqrt(knnyhat.shape[0])

mean_acc
plt.plot(range(1,Ks),mean_acc,'g')
plt.fill_between(range(1,Ks),mean_acc - 1 * std_acc,mean_acc + 1 * std_acc, alpha=0.10)
plt.fill_between(range(1,Ks),mean_acc - 3 * std_acc,mean_acc + 3 * std_acc, alpha=0.10,color="green")
plt.legend(('Accuracy ', '+/- 1xstd','+/- 3xstd'))
plt.ylabel('Accuracy ')
plt.xlabel('Number of Neighbors (K)')
plt.tight_layout()
plt.show()
print( "The best accuracy was ", mean_acc.max(), "with k=", mean_acc.argmax()+1)
```



```
[141]
from sklearn.metrics import f1_score
import sklearn.metrics as metrics
accuracy_score(knny_test, knnyhat)
```

```
0.8197424892703863
```

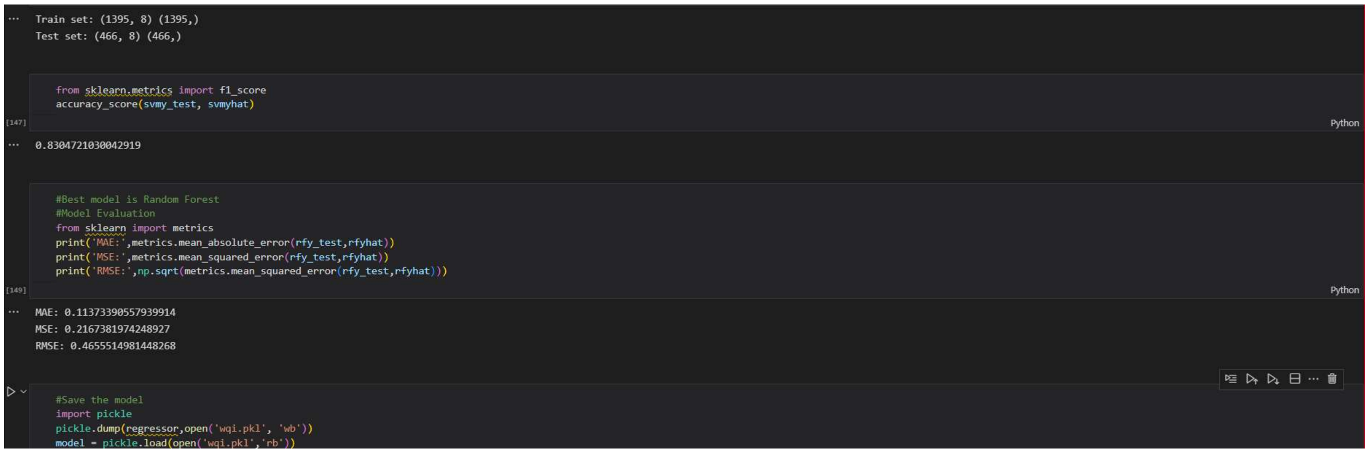
```
[143]
#RandomForest
rfX_train, rfX_test,rfy_train, rfy_test = train_test_split(scaled_X,Y,test_size=0.25,random_state=4,shuffle=True)
print ('Train set:', rfX_train.shape,  rfy_train.shape)
print ('Test set:', rfX_test.shape,  rfy_test.shape)
from sklearn.ensemble import RandomForestClassifier
import math
clss=RandomForestClassifier(n_estimators=100,random_state=0)
clss.fit(rfX_train,rfy_train)
rfyhat=clss.predict(rfX_test)
```

```
Train set: (1395, 8) (1395,)
Test set: (466, 8) (466,)
```

```
[144]
from sklearn.metrics import f1_score
accuracy_score(rfy_test, rfyhat)
```

```
0.9313304721030042
```

```
[146]
#SVM
svmX_train, svmX_test,svmy_train, svmy_test = train_test_split(scaled_X,Y,test_size=0.25,random_state=4,shuffle=True)
print ('Train set:', svmX_train.shape,  svmy_train.shape)
print ('Test set:', svmX_test.shape,  svmy_test.shape)
from sklearn import svm
clf = svm.SVC(kernel='rbf')
clf.fit(svmX_train, svmy_train)
svmyhat = clf.predict(svmX_test)
```



```
Train set: (1395, 8) (1395,)
Test set: (466, 8) (466,)
```

```
[147]
from sklearn.metrics import f1_score
accuracy_score(svmy_test, svmyhat)
```

```
0.8304721030042919
```

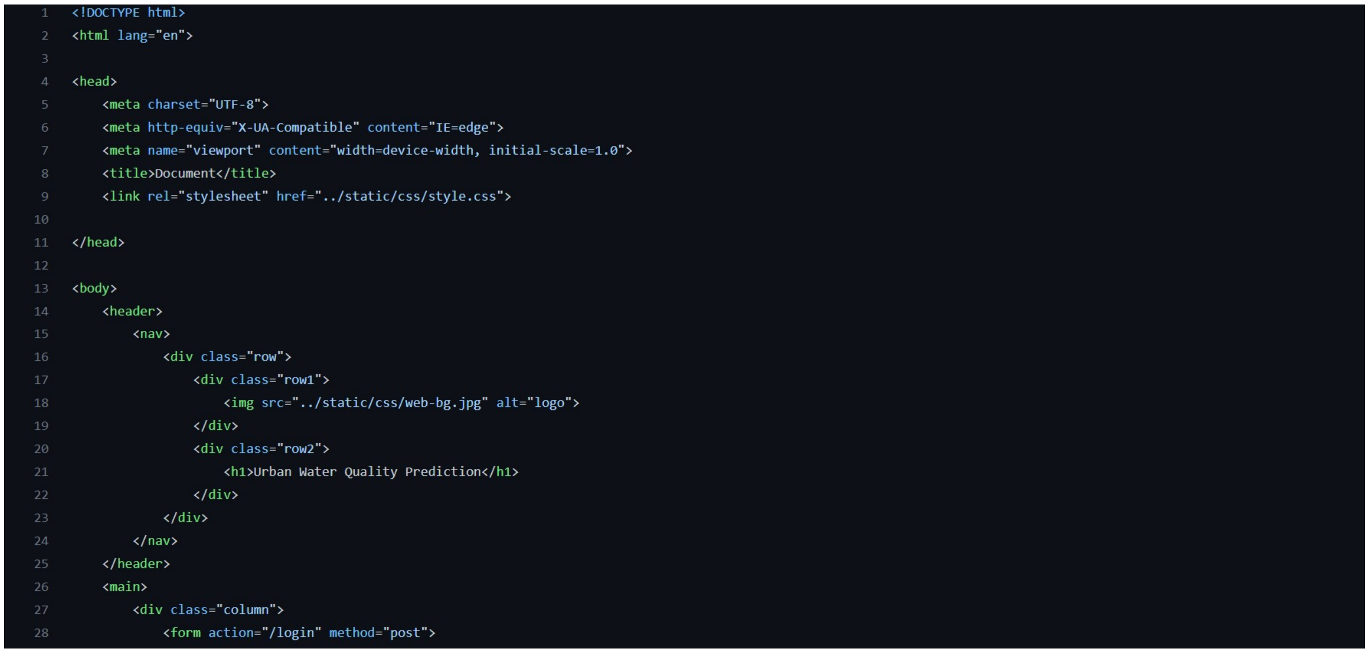
```
[149]
#Best model is Random Forest
#Model Evaluation
from sklearn import metrics
print('MAE:',metrics.mean_absolute_error(rfy_test,rfyhat))
print('MSE:',metrics.mean_squared_error(rfy_test,rfyhat))
print('RMSE:',np.sqrt(metrics.mean_squared_error(rfy_test,rfyhat)))
```

```
MAE: 0.11373390557939914
MSE: 0.2167381974248927
RMSE: 0.4655514981448268
```

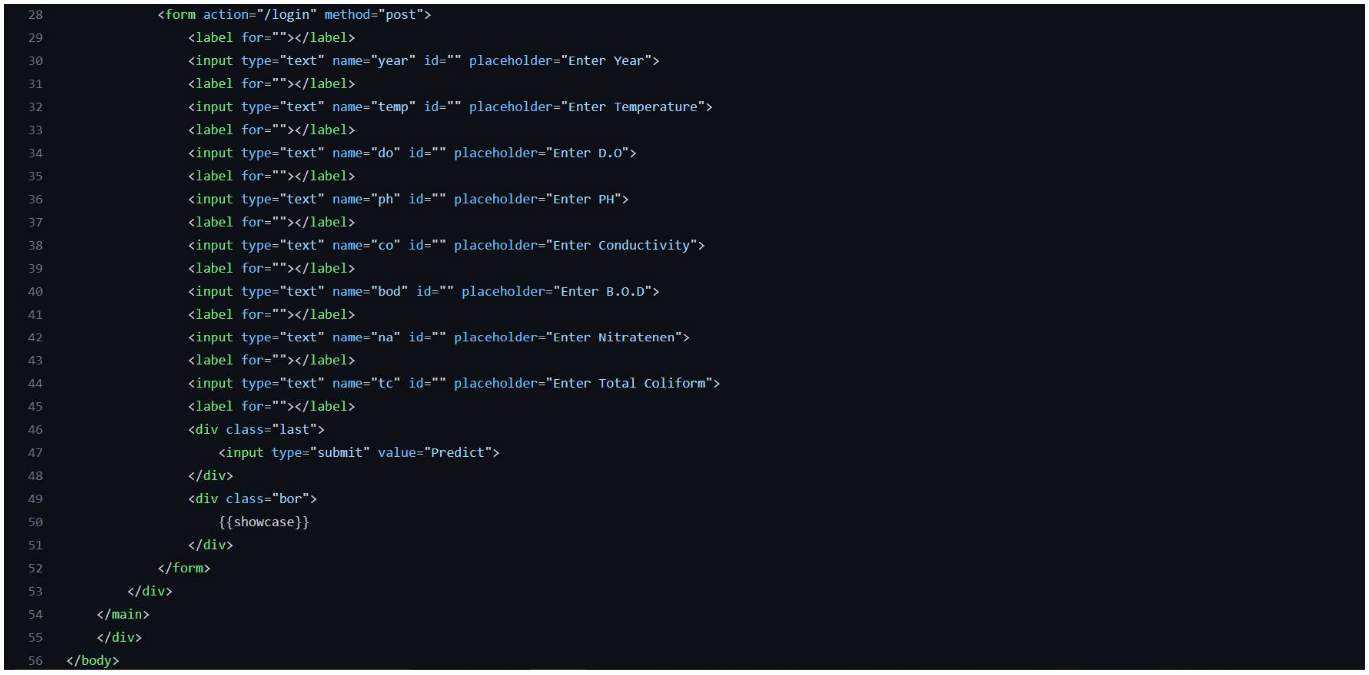
```
#Save the model
import pickle
pickle.dump(regressor,open('wqi.pkl', 'wb'))
model = pickle.load(open('wqi.pkl','rb'))
```

7.2Feature 2

Index.html



```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="../static/css/style.css">

</head>

<body>
    <header>
        <nav>
            <div class="row">
                <div class="row1">
                    <img src="../static/css/web-bg.jpg" alt="logo">
                </div>
                <div class="row2">
                    <h1>Urban Water Quality Prediction</h1>
                </div>
            </div>
        </nav>
    </header>
    <main>
        <div class="column">
            <form action="/login" method="post">
```



```html
            <form action="/login" method="post">
                <label for=""></label>
                <input type="text" name="year" id="" placeholder="Enter Year">
                <label for=""></label>
                <input type="text" name="temp" id="" placeholder="Enter Temperature">
                <label for=""></label>
                <input type="text" name="do" id="" placeholder="Enter D.O">
                <label for=""></label>
                <input type="text" name="ph" id="" placeholder="Enter PH">
                <label for=""></label>
                <input type="text" name="co" id="" placeholder="Enter Conductivity">
                <label for=""></label>
                <input type="text" name="bod" id="" placeholder="Enter B.O.D">
                <label for=""></label>
                <input type="text" name="na" id="" placeholder="Enter Nitratenen">
                <label for=""></label>
                <input type="text" name="tc" id="" placeholder="Enter Total Coliform">
                <label for=""></label>
                <div class="last">
                    <input type="submit" value="Predict">
                </div>
                <div class="bor">
                    {{showcase}}
                </div>
            </form>
        </div>
    </main>
    </div>
</body>
```

Backend.py

```
import requests
import numpy as np
from flask import Flask,render_template,request
import pickle
import collections
collections.Callable = collections.abc.Callable
# NOTE: you must manually set API_KEY below using information retrieved from your
IBM Cloud account.
API_KEY = "WtGCq52jbLpAf6KgOTSLokoZeKmNP_wAgP-7i6HxzyuF"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey": API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]
header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

app= Flask(__name__)
#model=pickle.load(open('wqi.pkl','rb'))
@app.route('/')
def home() :
  return render_template("index.html")
@app.route('/login',methods = ['POST'])
def login() :
  temp=request.form["temp"]
  do = request.form["do"]
  ph = request.form["ph"]
  co = request.form["co"]
  bod = request.form["bod"]
  na = request.form["na"]
  tc = request.form["tc"]
  year = request.form["year"]

  total = [[float(temp),float(do),float(ph),float(co),float(bod),float(na),float(tc),int(year)]]
  payload_scoring = {"input_data": [{"field":
[["temp","do","ph","co","bod","na","tc","year"]], "values": total}]}
  response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/b61b8352-d24c-4ede-b0f4-
9dcd50fffa28/predictions?version=2022-11-19',
json=payload_scoring,headers={'Authorization': 'Bearer ' + mltoken})
  print("Scoring response")
```

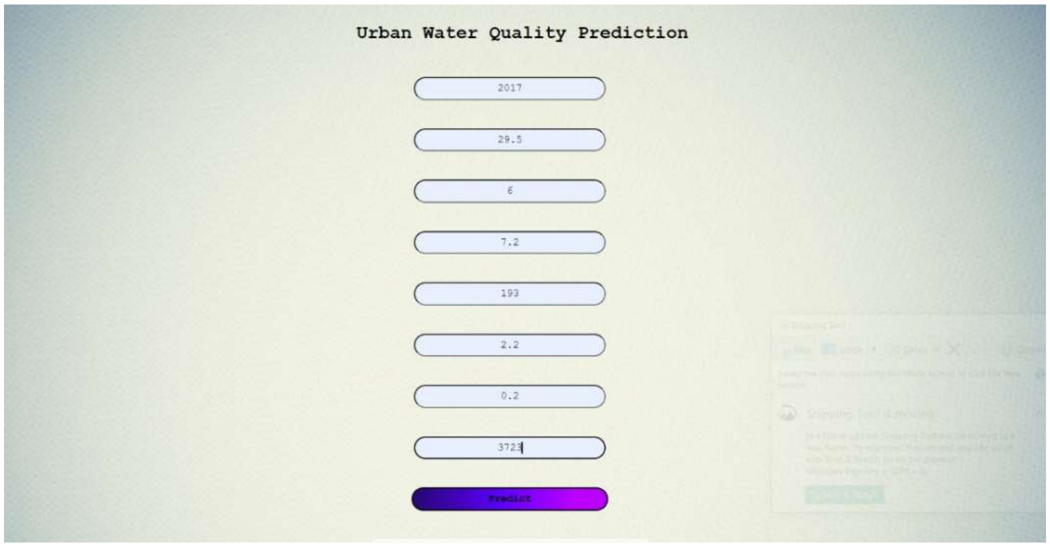
```
 y_pred=response_scoring.json()
 print(y_pred)
 wqi_class=y_pred['predictions'][0]['values'][0][0]
 if wqi_class==0:
   return render_template("index.html",showcase="The predicted water quality is Poor")
 elif wqi_class==2:
   return render_template("index.html",showcase="The predicted water quality is
Medium")
 elif wqi_class==3:
   return render_template("index.html",showcase="The predicted water quality is Good")
 elif wqi_class==4:
   return render_template("index.html",showcase="The predicted water quality is
Excellent")

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

Displaying backend.py.

8 TESTING

8.1 Test Cases

- Webpage working and Model Prediction



Urban Water Quality Prediction

2017

29.5

6

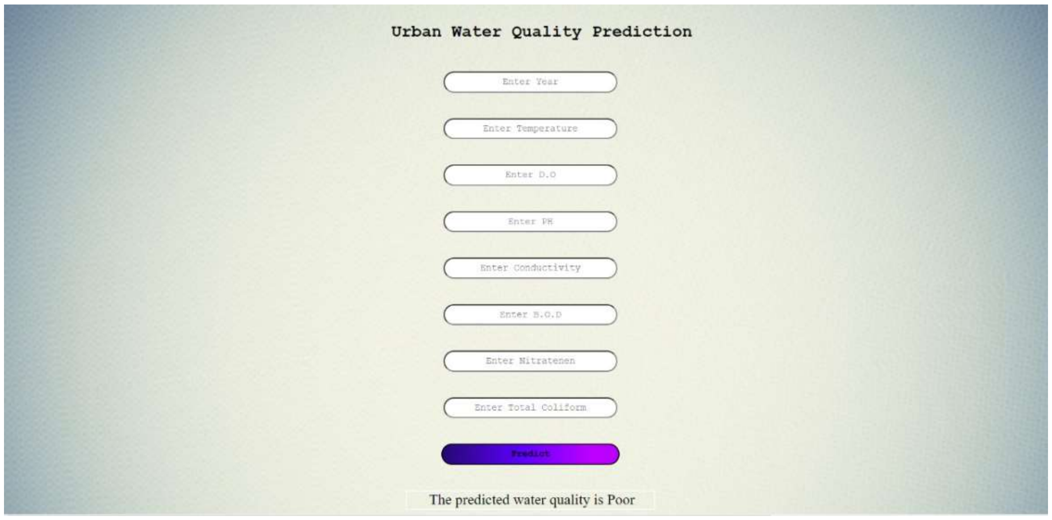
7.2

193

2.2

0.2

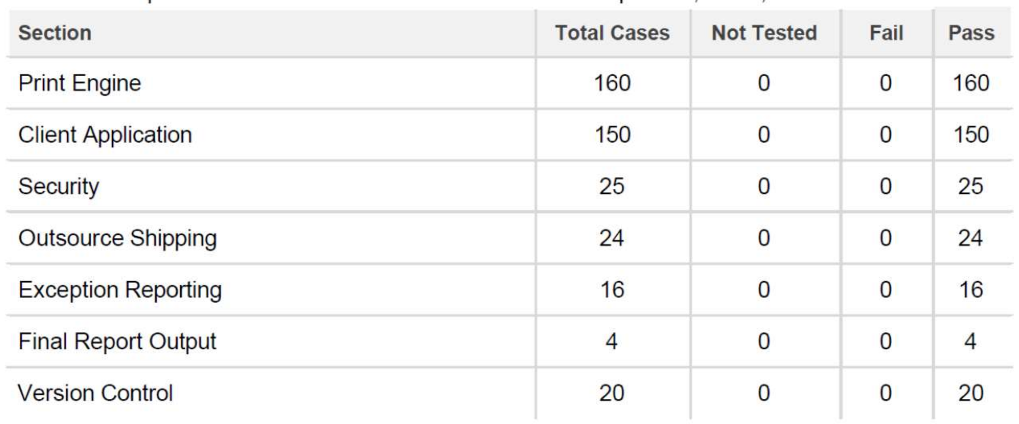
3723


Urban Water Quality Prediction

The predicted water quality is Poor

8.2 User Acceptance Testing

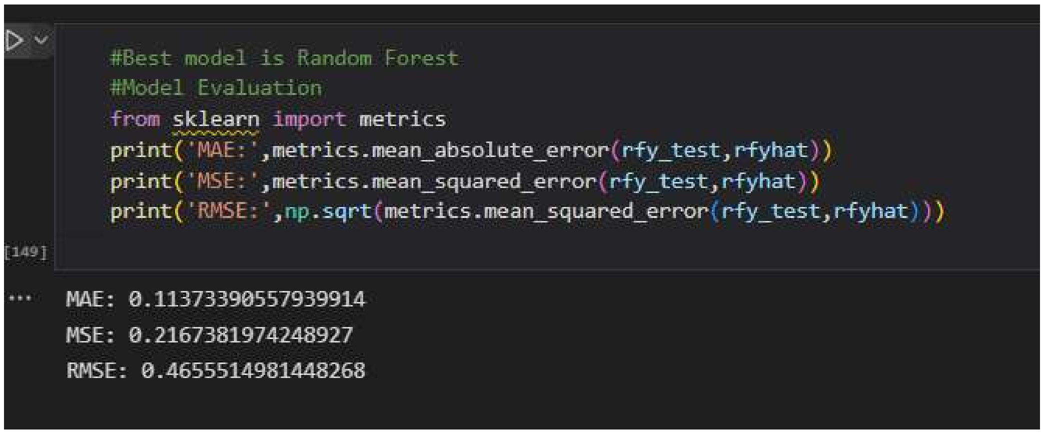
Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	8	3	1	2	14
Duplicate	2	0	3	0	5
External	2	3	0	0	5
Fixed	11	2	5	20	38
Not Reproduced	0	0	1	0	1
Skipped	1	0	1	0	2
Won't Fix	0	5	0	0	5
Totals	24	13	11	22	70



Section	Total Cases	Not Tested	Fail	Pass
Print Engine	160	0	0	160
Client Application	150	0	0	150
Security	25	0	0	25
Outsource Shipping	24	0	0	24
Exception Reporting	16	0	0	16
Final Report Output	4	0	0	4
Version Control	20	0	0	20

9.RESULTS

9.1Performance metrics



```
#Best model is Random Forest
#Model Evaluation
from sklearn import metrics
print('MAE:',metrics.mean_absolute_error(rfy_test,rfyhat))
print('MSE:',metrics.mean_squared_error(rfy_test,rfyhat))
print('RMSE:',np.sqrt(metrics.mean_squared_error(rfy_test,rfyhat)))
```

```
MAE: 0.11373390557939914
MSE: 0.2167381974248927
RMSE: 0.4655514981448268
```

10. ADVANTAGES AND DISADVANTAGES

Advantages :

- User can analyse the water quality anytime and anywhere.
- Quick Prediction of WQI.
- Free method to of predicting water quality.
- Can be used in any location
- No expensive physical equipment required
- Prediction done using many parameters that improves the precision

Disadvantages:

- Users have to input all the values of the parameters to obtain a accurate WQI.
- No Dynamic Prediction of WQI on basis of location
- No physical measurement used to capture the location specific data.

11.CONCLUSION

This project has successfully developed a web application that will predict the water quality by using machine learning techniques. Using the WQI formula , and with help of web application and a trained model WQI prediction is carried out. With the utilization of the web application and a trained model, the prediction of water quality has been done using the water quality index calculation that tells us the measure of the water quality using several parameters in different locations. The quality index was efficiently calculated on the test data entered by the user.

12. FUTURE SCOPE

This project aims to classify the type of usage the water can be used for in terms of the WQI index , like if it is fit for drinking , if it is fit for farming and other use cases of water and also use location to help users find water quality of water resources near them without having to fill in values.

13. APPENDIX

Source Code:

ML Code : https://github.com/IBM-EPBL/IBM-Project-1978-1658422148/blob/main/Project%20Development%20Phase/Sprint%204/Sprint_4.ipynb

Web Part : https://github.com/IBM-EPBL/IBM-Project-1978-1658422148/tree/main/Project%20Development%20Phase/Sprint%204/Sprint_4%20front%20end

Video demo : https://drive.google.com/file/d/15AZtiCuibmVx5hWa0pLHACNOtgmBhkyl/view

Github Project link : https://github.com/IBM-EPBL/IBM-Project-1978-1658422148