***A Industrial Oriented Mini Project Report***

*On*

WATER QUALITY ANALYSIS USING MACHINE LEARNING

*Submitted in partial fulfilment of the requirements for the award*

*of the Degree of B. Tech.*

*In*

***Department of Information Technology***

*By*

*Muppi likitha-22911A1237*

***Under the guidance of***

Dr. D. Marlene Grace Verghese

Professor - Department of IT



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**VIDYA JYOTHI INSTITUTE OF TECHNOLOGY**

(An Autonomous Institution)

**[Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH]**

### 2024 – 2025



**CERTIFICATE**

This is to certify that the project report entitled **“Water Quality Analysis Using Machine Learning”** submitted by ***Madupathi Sravanthi -22911A1227*** to Vidya Jyothi Institute of Technology (An Autonomous Institution), Hyderabad, in partial fulfilment for the award of the degree of **B. Tech. in Information Technology** a *bonafide* record of project work carried out by us under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institution or University for the award of any degree.

**Internal Guide Head of the Department**

**Dr. D. Marlene Grace Verghese Dr. A. Obulesu**

**Professor Professor**

**Department of Information Technology Department of Information Technology**

**External Examine**

### DECLARATION

I declare that this project report titled **Water Quality Analysis Using Machine Learning** submitted in partial fulfilment of the requirements for the award of the degree of B. Tech in Information Technology is a record of original work carried out by us under the supervision of **Dr. D. Marlene Grace Verghese**, and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University.

Muppi Likitha-22911A1237

**Date:**

### ACKNOWLEDGEMENTS

I extend my heartfelt gratitude to my Internal Guide **Dr. D. Marlene Grace Verghese**, Professor, for her invaluable mentorship and guidance throughout the course of this project. Her expertise, encouragement, and insightful feedback played a crucial role in shaping our approach and refining our methodology.

I am deeply grateful to **Dr. A. Obulesu**, Head of the Department, and all the faculty members of the department for their constant support and guidance throughout our academic journey.

My sincere thanks to our beloved **Dean of Accreditation and Rankings, Dr. A. Padmaja Madam**, and **Academic Co-ordinator, Smt. G. Srilatha Madam**, whose unwavering support proved indispensable during the course.

I also express my heartfelt thanks to our **Principal, Dr. A. Srujana madam**, for her encouragement and continuous support throughout this endeavour.

I would like to sincerely thank our **College Management** for providing the necessary infrastructure and facilities that enabled us to carry out our work effectively.

Finally, I would like to express my deepest gratitude to my **Parents Madupathi Nagaiah Swamy, Madupathi Sangeetha** whose unwavering love, guidance, and support have been the foundation of our success and aspirations.

*Muppi Likitha-22911A1237*

**Table of Contents**

|  |  |
| --- | --- |
| **Description** | **Page No.** |
| **ABSTRACT** | **i** |
| **LIST OF FIGURES** | **ii** |
| **LIST OF TABLES** | **iii** |
| **1. Introduction** | **1** |
| 1.1Introduction | 2 |
| 1.2 Problem description | 2 |
| 1.3Existing System | 2-3 |
| 1.4 Proposed System | 3 |
| 1.5Objectives | 3-4 |
| **2. Literature Survey** | **5** |
| **3. System Analysis** | **6** |
| 3.1Feasibility Study | 6-7 |
| 3.2Software Requirements Specification | 7 |
| 3.2.1Functional Requirements | 7 |
| 3.2.2Non Functional Requirements | 8 |
| 3.3Software & Hardware Requirements | 8 |
| **4. System Design** | **9** |
| 4.1 Architecture | 9 |
| 4.2 UML Diagrams | 10-13 |
| 4.3 Modules | 14 |
| **5. Implementation & Results** | **15** |
| 5.1 Methodology | 15-16 |
| 5.2 Sample Code | 16-23 |
| 5.3 Output Screenshots | 23-33 |
| **6. Testing & Validation** | **34** |
| 6.1 Introduction | 34 |
| 6.2 Test case Scenarios | 34 |
| **7. Conclusion & Future Enhancement** | **35** |
| **8. Bibliography** | **36** |

## ABSTRACT

Access to safe drinking water is one of the essential needs of all human beings. From a legal point of view, access to drinking water is one of the fundamental human rights. Many factors affect water quality. One of the main areas of research in machine learning is the analysis of water quality. It is also known as water potability analysis because our task here is to understand all the factors that affect water potability and train a machine learning model that can classify whether a specific water sample is safe or unfit for consumption. In this project, we use Python and various libraries such as numpy, pandas, seaborn and matplotlib to implement a machine learning model for predicting whether a water sample is safe or unsuitable for consumption based on features such as ph, Hardness, Solids, Chloramines, Sulfate, Conductivity, Organic\_carbon, Trihalomethanes, Turbidity of the Water.

i

## LIST OF FIGURES

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **FIG.NO** | **Figure Name** | **PAGE.NO** |
| 1 | 4.1 | System architecture for water quality analysis | 9 |
| 2 | 4.2 | Class Diagram for water quality analysis | 10 |
| 3 | 4.3 | Usecase Diagram for water quality analysis | 11 |
| 4 | 4.4 | Activity Diagram for water quality analysis | 12 |
| 5 | 4.5 | Sequence Diagram for water quality analysis | 13 |
| 6 | 5.1 | Dataset represents the Rows and Columns of the dataset and information in it | 23 |
| 7 | 5.2 | Entering the Ph Value. | 23 |
| 8 | 5.3 | Entering the HardnessValue | 25 |
| 9 | 5.4 | Entering the solids Value | 25 |
| 10 | 5.5 | Entering the chloramines Value | 25 |
| 11 | 5.6 | Entering the Sulphates Value | 25 |
| 12 | 5.7 | Entering the Conductivity value | 25 |
| 13 | 5.8 | Entering the organic carbon value | 25 |
| 14 | 5.9 | Entering the trihalomethanes value | 26 |
| 15 | 5.10 | Entering the Turbidity value | 26 |
| 16 | 5.11 | Output Screenshots | 26 |

ii

**LIST OF TABLES**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **TABLE.NO** | Table name | **PAGE.NO** |
| 1 | 6.1 | Test Case Scenarios | 34 |

iii

## CHAPTER-01 INTRODUCTION

### Introduction

Water quality is a critical aspect of environmental health, directly impacting the well-being of ecosystems and human populations. Rapid industrialization, urbanization, and agricultural practices have heightened concerns about the quality of water resources worldwide. In response to these challenges, this project aims to leverage machine learning techniques to analyze and predict water quality parameters, providing a valuable tool for environmental monitoring and decision- making. The project focuses on the analysis of key water quality indicators such as pH, turbidity, dissolved oxygen, and more. By harnessing the power of machine learning algorithms, we seek to develop a predictive model that can assess and forecast water quality based on historical data. The integration of a user interface enhances accessibility, allowing stakeholders, researchers, and the general public to interact with and interpret the findings in a user-friendly manner.

### Key Features:

* **User Friendly Interface:** Design an intuitive and user-friendly interface accessible to a wide range of users, including researchers, policymakers, and the general public.
* **Real-Time News Updates:** Implement real-time data visualization to display current water quality parameters. Use charts, graphs, and maps to convey information in a visually appealing and understandable manner.
* **Personalized Content:** Provide the ability to analyze historical water quality data trend and callow users to explore how water quality parameters have changed over time.
* **Predictive Analytics:** Integrate machine learning models to predict future water quality basic historical data and current conditions. Display predictions in an accessible format for users to understand potential trends.
* **Parameter Selection:** Allow users to select specific water quality parameters for detailed analysis Customization options enhance the applicability of the tool across diverse scenarios.
* **Data Export and Download:** Enable users to export analyzed data or reports for further offline analysis or documentation and provide download options in common formats (CSV, PDF, etc.).

### Problem description

Access to clean water is a basic human right, but growing pollution from industries, agriculture, and urban areas makes maintaining water quality a challenge. Traditional monitoring methods are slow, manual, and not suited for real-time insights. This project proposes a machine learning-based system to predict water quality using key parameters like pH, turbidity, and dissolved oxygen. By analyzing historical data, the system can forecast contamination risks and support timely action. An interactive interface allows users to explore and download water quality trends, promoting transparency, awareness, and smarter water management.

### Existing Systems

Currently, water quality analysis mainly relies on manual testing and chemical analyses conducted in laboratories. These methods can be expensive, require skilled personnel, and are subject to human error. Additionally, the results are often not immediately available, hindering rapid responses to water quality issues. Yet, there i a disconnect between the data quality, data gathering and data analysis due to the lack of standardized approaches for data collection and processing, spatio - temporal variation of key parameters in water bodies and new contaminants.

### Disadvantages:

* Not integrated with Machine Learning or Deep Learning algorithms for automated predictions.
* Time-consuming due to manual sample collection, transport, and lab-based analysis.
* High operational costs for equipment, chemical reagents, and expert labor.
* Delayed results, which hinder immediate action in response to contamination.
* Human errors during sample handling or testing can lead to inaccurate outcomes.
* Lack of scalability to monitor large water bodies or multiple sites simultaneously.
* Inconsistent data collection due to absence of standardized methods or protocols.
* Limited frequency of testing, which may miss short-term pollution spikes.
* Inability to detect new or emerging contaminants without expensive testing extensions.
* No real-time monitoring, making it difficult to track changes dynamically.

### Proposed System

The proposed method is to build a machine learning model for Water quality. The process carries from data collection where the past data related to Water qualities are collected. Data mining is a commonly used technique for processing enormous data in the domain. The water if found before proper treatment can save lives. Machine learning is now applied and mostly used in health care where it reduces the manual effort and better model makes error less which leads in saving the life. The data analysis is done on the dataset proper variable identification done that is both the dependent variables and independent variables are found. Then proper machine learning algorithm are applied on the dataset where the pattern of data is learnt. After applying different algorithms, a better algorithm is used for the prediction of outcome

### Advantages:

### Enhances decision-making through data-driven insights and predictive analytics.

### Reduces dependency on manual testing and laboratory procedures.

### Enables early detection of contamination, allowing timely intervention and mitigation.

### Provides scalable and automated solutions for large-scale water quality monitoring.

### Capable of handling non-linear and complex relationships among water quality parameters.

### Improves accuracy and consistency in water potability classification.

### Allows integration with sensor data or IoT devices for continuous monitoring.

### Facilitates cost-effective long-term monitoring by reducing the need for frequent lab tests.

### 1.5 Objectives

The primary objective of this project is to develop an intelligent, user-friendly, and efficient system for predicting and analyzing water quality using machine learning techniques. The specific objectives include:

* To analyze historical water quality data and identify significant parameters such as pH, turbidity, dissolved oxygen, conductivity, and more that influence water quality.
* To develop a machine learning model that can accurately predict future water quality conditions based on historical and real-time data inputs.
* To build an interactive user interface for visualizing water quality trends and predictions.
* To provide users with the ability to download analyzed data for research and public awareness.
* To support timely and informed decision-making for environmental agencies and policymakers.

## CHAPTER – 02

## LITERATURE SURVEY

1. Bhumika Nandwana, Satyanarayan Tazi, Sheifalee Trivedi, Dinesh Kumar, and Santosh Kumar Vipparthi. “A survey paper on hand gesture recognition”. In: 2017 7th International Conference on Communication Systems and Network Technologies (CSNT). 2017, pp. 147–

152. DOI: 10.1109/CSNT.2017.8418527.

1. Harpreet Kaur and Jyoti Rani. “A review: Study of various techniques of Hand gesture recognition”. In: 2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES). 2016, pp. 1–5. DOI: 10.1109/ICPEICES.2016.7853514.
2. Jayesh S. Sonkusare, Nilkanth B. Chopade, Ravindra Sor, and Sunil L. Tade. “A Review on Hand Gesture Recognition System”. In: 2015 International Conference on Computing Communication Control and Automation. 2015, pp. 790–794. DOI: 10.1109/ICCUBEA.2015.158.
3. Hamid A. Jalab and Herman. K. Omer. “Human computer interface using hand gesture recognition based on neural network”. In: 2015 5th National Symposium on Information Technology: Towards New Smart World (NSITNSW). 2015, pp. 1–6. DOI: 10.1109/NSITNSW.2015.7176405.
4. Hung-Yuan Chung, Yao-Liang Chung, and Wei-Feng Tsai. “An Efficient Hand Gesture Recognition System Based on Deep CNN”. In: 2019 IEEE International Conference on Industrial Technology (ICIT). 2019, pp. 853– 858. DOI: 10.1109/ICIT.2019.8755038.
5. Zoltan Vamossy, Andras Toth, and Balazs Benedek. “Virtual Hand - Hand Gesture Recognition System”. In: 2007 5th International Symposium on Intelligent Systems and Informatics. 2007, pp. 97–102. DOI: 10.1109/ SISY.2007.4342632**.**

**CHAPTER – 03**

**SYSTEM ANALYSIS**

This project aims to develop a predictive system that utilizes machine learning to analyze water quality data. The system focuses on key water quality parameters such as pH, turbidity, and dissolved oxygen. Through data preprocessing, feature selection, model training, and performance evaluation, the system provides accurate predictions and visual insights into water conditions.

#### Key Highlights:

* Collects and analyzes historical water quality data
* Applies machine learning algorithms for prediction
* Provides an interactive user interface for data input and visualization
* Supports data export in various formats
* Enhances decision-making through timely forecasts

### Feasibility Study

In order to evaluate if the project can be done in the given time frame, we are using theMachine Learning models, where we cover the feasibility of the project from a technological, economical, and legal perspective. Those perspectives would help us have a broad vision of the requirements and implications related to the project. We also discuss in this section the methodology used in conducting the project. Analysis is the process of finding the best solution to the problem. System analysis is the process by which we learn about the existing problems, define objects and requirements and evaluate the solutions. It is the way of thinking about the organization and the problem it involves, a set of technologies that helps in solving these problems. Feasibility study plays an important role in system analysis which gives the target for design and development.

### Technical Feasibility

The technologies and libraries relevant to Machine Learning and would be used to develop this project.

#### Hardware and Software Requirements:

Analyze whether the organization's budget and timeframe permit the purchase of the technology and software required for data analysis and machine learning.

#### Technology Proficiency:

To find out if the team has the technical know-how necessary to create,deploy and maintain the forecasting system, evaluate the team's technical knowledge.

### Economical feasibility

This study is carried out to check the economic impact that the system will have on the organization. Since the project is Machine learning based, the cost spent in executing this project would not demand cost for softwares and related products, as most of the products are open source and free to use. Hence the project would consumed minimal cost and is economically feasible.

#### Cost-Benefit Analysis:

Conduct a detailed cost-benefit analysis to determine if the potential benefits, such as cost savings and increased revenue, outweigh the project's costs, including development, implementation and maintenance.

### Requirements Specification

### Functional Requirements

Functional requirements specify which features or functions should be included in a system in order to satisfy user expectations. The functional requirements define the connection between the inputs and outputs, based on the premise. To obtain the output, all the operations that must be carried out on the input data must be specified.

The system should be able to provide these functionalities efficiently.

* **Resource Visualization:** The visualizations should be self-explanatory which can be easily understood by the user. There will be line plots and graphs which can be used as an effective measure while devising any new program.
* ML algorithm should be able to predict the output efficiently and accurately. Predict the water quality parameters in an accurate manner..

### 3.2.2 Non Functional Requirements

Non-functional requirements describe features, characteristics, and capacity of the system and they may constraints the boundaries of the proposed system.

The following are the non-functional requirements that are essential depending on the performance, cost, and control and give security efficiency and services.

Based on the above- explained non-functional prerequisites, they are as follows:

* User friendly
* The system should provide better accuracy.
* To perform efficiently with better throughput and response time

### Software & Hardware Requirements

**Software Requirements**

* **Operating System:** Windows 11
* **Frontend:** Python 3
* **Backend:** Sql

### Hardware Requirements

### RAM: 2GB

* **Processor:** Pentium IV
* **Hard Disk:** 2GB

**CHAPTER-04**

**SYSTEM DESIGN**

Our project presents a solution that leverages real-time camera input to recognize hand gestures and identify the number being shown by the user. The system uses computer vision techniques combined with machine learning or deep learning models to interpret the position and movement of the user's hand. This allows it to accurately determine numeric gestures (e.g., showing fingers to represent numbers from 0 to 5 or more) and respond with appropriate outputs.

This reference design adopts a simple yet effective approach to detect gestures based on:

* **Static gestures**, such as holding up a certain number of fingers steadily for a short duration.
* **Dynamic gestures**, such as swiping the hand left, right, up, or down across the camera frame to trigger specific actions.

The camera continuously captures frames, and these images are processed using techniques like:

* **Background subtraction** to isolate the hand.
* **Contour detection** to identify the hand's shape.
* **Convex hull and defects** to count fingers.
* **Time-based hold detection** to distinguish between a quick movement and an intentional gesture.

Once a gesture is recognized, the system maps it to a predefined command or result, such as displaying the number, executing a function, or interacting with an application.

This system is particularly useful in scenarios where:

* Touch-free interaction is needed (e.g., for hygiene or accessibility reasons).
* A lightweight and intuitive input method is required for embedded systems, smart devices, or educational tools.

### 4.1 Architecture

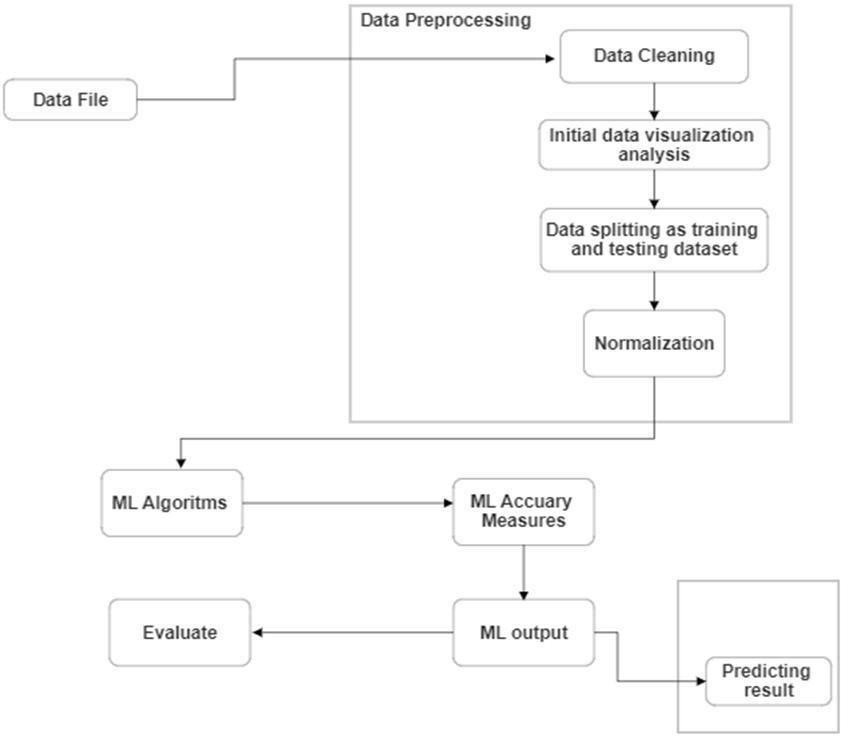
****

Fig 4.1 system architecture for water quality analysis

### UML Diagrams

* **Class Diagram for water quality analysis**

The class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. A collection of class diagrams represents the whole system. Classes are composed of three things: names, attributes, and operation.

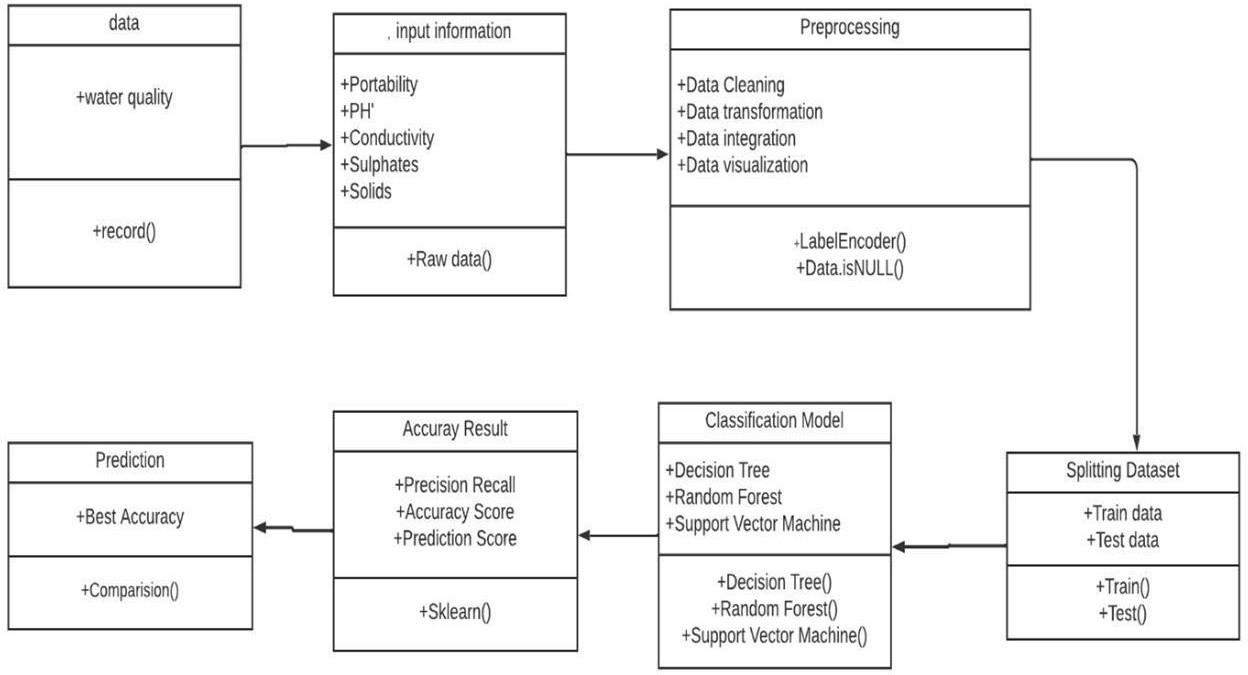
****

Fig 4.2 Class Diagram for water quality analysis

### Use case Diagram for water quality analysis

A use case diagram is a graphical depiction of a user's possible interactions with a system. Usecase diagrams show various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

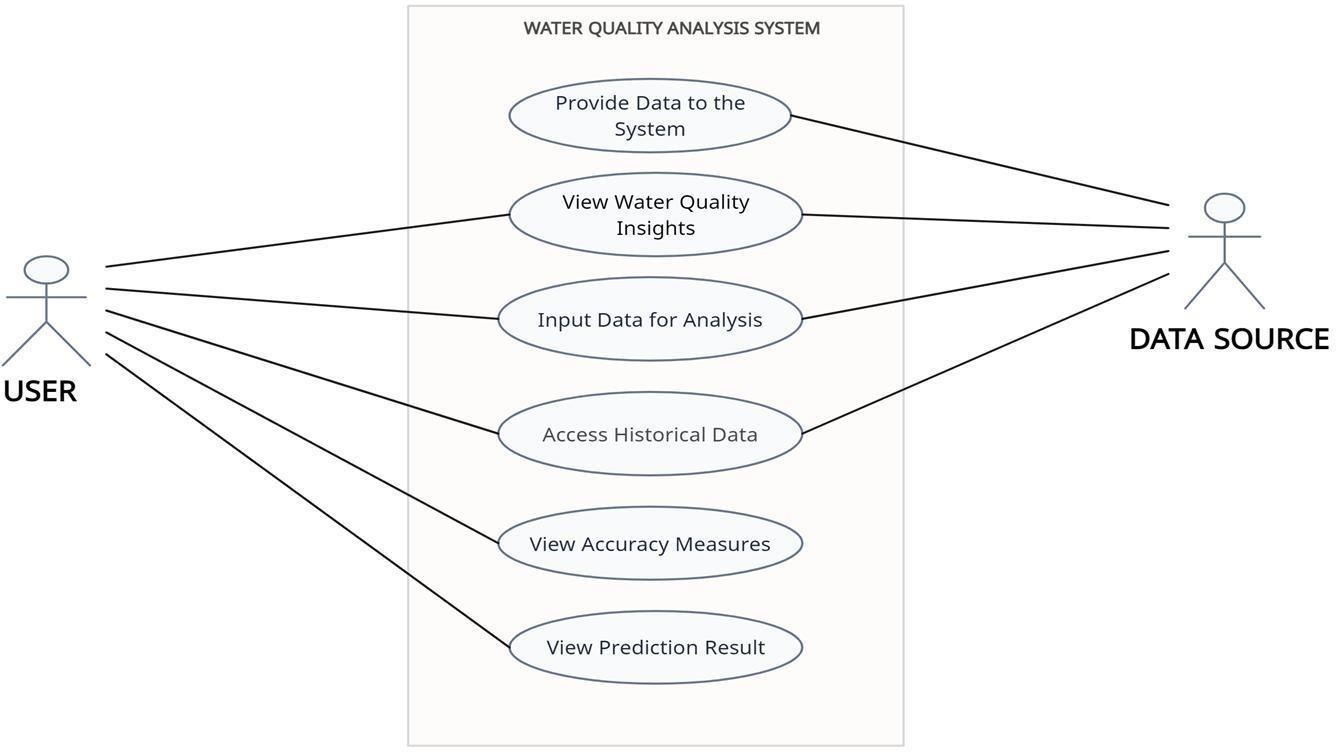
****

Fig 4.3 Usecase Diagram for water quality analysis

### Activity Diagram for water quality analysis

### An activity diagram for water quality analysis visually represents the workflow of activities involved in analyzing water quality, from data collection to reporting results.

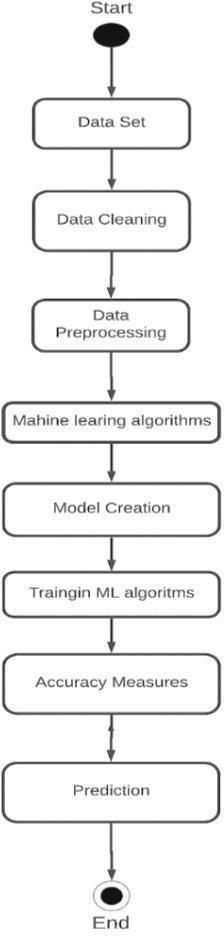
****

Fig 4.4 Activity Diagram for water quality analysis

### Sequence Diagram for water quality analysis

### A sequence diagram for water quality analysis illustrates the interactions between users and the system components over time, detailing how data is collected, processed, and reported.

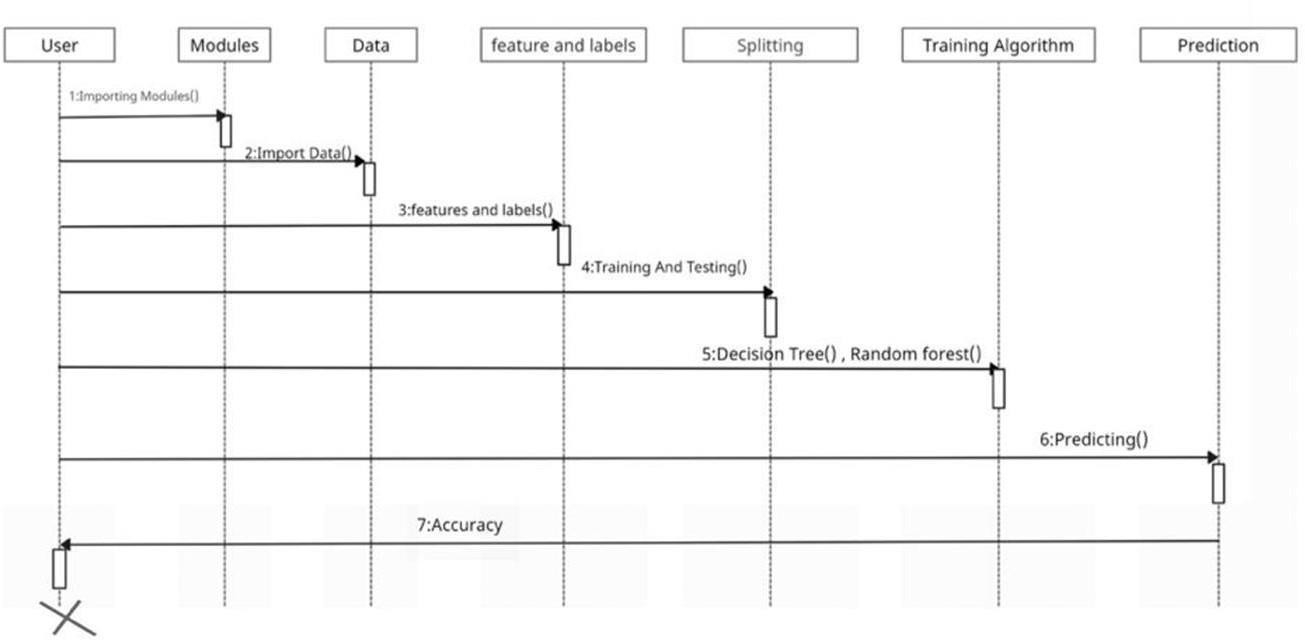


Fig 4.5 Sequence Diagram for water quality analysis

### Modules

The project involves several interconnected modules designed to efficiently predict the potability of water, leveraging machine learning techniques and user-friendly interfaces. The **Data Collection and Preprocessing** module serves as the foundation by first loading the dataset, which may include both structured and unstructured data. Missing values within the dataset are handled through imputation techniques or by removing incomplete records, ensuring that the dataset remains robust and ready for analysis. Additionally, the data is normalized or scaled to bring all features to a common scale, which is particularly important for models sensitive to feature magnitude, such as Support Vector Machines and K-Nearest Neighbors.

Once the data is preprocessed, the **Feature Selection** phase follows, where relevant and significant features are chosen to represent the data, while irrelevant or redundant features are removed to enhance the model’s performance. This step ensures that the model is not overwhelmed by irrelevant information, improving both the efficiency and accuracy of predictions. Feature engineering, such as the creation of new composite features, can also be incorporated to better capture the underlying patterns in the data.

The **Model Training and Evaluation** module is at the core of the project, where various machine learning algorithms like Support Vector Machine (SVM), Random Forest (RF), Decision Trees (DT), and K-Nearest Neighbors (KNN) are applied to the dataset. Each model is trained and tested to assess its performance based on metrics such as accuracy, precision, recall, F1 score, and computational efficiency. Cross-validation is often employed to ensure the model generalizes well to unseen data. Based on these performance metrics, the best-performing model is selected to be used in the subsequent prediction stage.

In the **Prediction Module**, the trained model is used to predict water potability for new, unseen input data provided by the user. This module accepts user inputs, processes them through the trained model, and generates predictions about the potability of water. The system can be designed to give a clear and actionable recommendation on whether the water is safe to drink or not based on the trained model’s output.

The **Visualization and UI** module plays a crucial role in making the entire system user-friendly. A dashboard is created, featuring interactive charts and graphs to visualize the results of the model's predictions. The dashboard may display trends over time, distributions of water quality parameters, and model performance metrics. Additionally, the UI allows users to filter data based on various parameters and makes it easy to interact with the system. Users can also export the data in common formats like CSV or PDF, facilitating further analysis or documentation of the results.

# 

# CHAPTER-05

# IMPLEMENTATION

### 5.1 Methodology

### This project is developed primarily using the Python programming language, known for its simplicity, versatility, and powerful ecosystem of scientific libraries. Python is particularly well-suited for machine learning applications due to its ease of syntax, strong community support, and extensive collection of tools for data analysis, visualization, and GUI development. In addition, the project uses Tkinter, Python’s standard GUI library, for building an interactive user interface to make real-time water quality predictions.

### Several Python libraries are utilized throughout the project to handle various tasks:

### Pandas: A powerful data manipulation library used to load, clean, and preprocess the dataset. It allows for easy access to data structures like DataFrames and provides functions such as .head(), .info(), .describe(), and .isnull() to explore and manipulate data.

### NumPy: Used for numerical operations and array manipulation. It forms the foundation for many operations in machine learning workflows, including statistical calculations and matrix operations.

### Matplotlib and Seaborn: These libraries are used for data visualization. Matplotlib provides control over basic plots like histograms and line graphs, while Seaborn builds on it to produce more aesthetically pleasing and complex statistical visualizations like heatmaps and KDE plots.

### Scikit-learn (sklearn): The primary machine learning library used in the project. It offers a wide range of supervised learning models, data preprocessing tools, model evaluation metrics, and utilities like train\_test\_split, StandardScaler, cross\_val\_score, classification\_report, GridSearchCV, and more.

### Imbalanced-learn (imblearn): Although not central to this particular version, this library is often used for oversampling techniques like SMOTE to handle imbalanced datasets, which may be useful in future iterations.

### Joblib (from sklearn.externals): A lightweight library for efficiently saving and loading large Python objects like trained models and scalers

### The study begins with the importation of essential Python libraries such as Pandas, NumPy, Matplotlib, Seaborn, and Scikit-learn, which are used for data handling, visualization, and machine learning modeling. The dataset, containing various physicochemical parameters of water samples (like pH, sulfate, trihalomethanes, etc.), is loaded using pandas.read\_csv(). Initial exploratory commands such as .head(), .shape, .columns, .info(), and .describe() are used to understand the structure, size, and statistical summary of the dataset.,

### The next step involves data cleaning and preprocessing. Null values are identified using .isnull().sum() and visualized with a Seaborn heatmap. Columns with missing values—namely, 'ph', 'Sulfate', and 'Trihalomethanes'—are imputed with their respective mean values. Data distributions of key features are examined using histograms and KDE plots to understand their spread and skewness. Additionally, the correlation matrix is computed and visualized using a heatmap, helping identify relationships between features and reduce multicollinearity.

### To prepare the data for modeling, feature scaling is performed using StandardScaler to ensure all features contribute equally. The dataset is then split into training and testing subsets using train\_test\_split, ensuring class balance with stratify=Y. Multiple classification models are initialized, including Logistic Regression, Decision Tree, Random Forest, SVM, K-Nearest Neighbors, Extra Trees, Naive Bayes, Gradient Boosting, and AdaBoost. Cross-validation with 5 folds is conducted using cross\_val\_score, and the average accuracy of each model is recorded and compared.

### Following model evaluation, the top-performing models (Extra Trees, Random Forest, and SVM) are further trained and tested on the dataset. Their performance is assessed using classification metrics such as precision, recall, F1-score, and support, provided via the classification\_report. An ROC curve is also plotted for the Extra Trees Classifier to visualize its performance across different classification thresholds.

### To optimize the model further, hyperparameter tuning is carried out on the Random Forest Classifier using GridSearchCV with parameters like n\_estimators, min\_samples\_split, min\_samples\_leaf, and criterion. The best parameters are selected, and the optimized model is evaluated on the test set for its final classification performance.

### Finally, a user input interface is created using Python’s input() function to allow real-time predictions. The user is prompted to enter values for all relevant features (e.g., pH, Hardness, Solids, etc.). These inputs are then standardized using the previously fitted scaler and passed into the best-performing model to predict water potability. Based on the prediction, a message is displayed indicating whether the water is safe or not safe for consumption.

### 5.2 Sample Code

import pandas as pd #importing required libraries import matplotlib as plt

import seaborn as sns import warnings

warnings.filterwarnings('ignore')

water\_data = pd.read\_csv(r'C:\Users\muppi\Downloads\water\_potability.csv') water\_data.head() #importing the datafile

water\_data.columns #listing the columns present water\_data.shape #noofrowsandcoloums present in dataset water\_data.dtypes #typeofdata

water\_data.info() #any missing data water\_data.describe() #accessing the statistics of data water\_data.duplicated().any() #checks any duplicate data

water\_data.isnull().sum() #checking the no of null values present

null\_df = water\_data.isnull().sum().reset\_index() #creating a null dataset for missing values null\_df.columns = ['Column','Null\_count']

null\_df['%miss\_value'] = round(null\_df['Null\_count']/len(water\_data),2)\*100 null\_df

import matplotlib.pyplot as plt #visualizing null values sns.heatmap(water\_data.isnull(), yticklabels=False, cbar=False, cmap='viridis') plt.show() #visualizing null values

water\_data['ph'].plot(kind = 'hist') #checking the distributiion of ph import matplotlib.pyplot as plt #checking the distribution of sulfate water\_data['Sulfate'].plot(kind = 'hist'

plt.show()

import matplotlib.pyplot as plt #checking the distribution of trihalomethanes water\_data['Trihalomethanes'].plot(kind='hist')

plt.show()

import matplotlib.pyplot as plt #checking the distribution of trihalomethanes water\_data['Trihalomethanes'].plot(kind='hist')

plt.show()

import matplotlib.pyplot as plt #checking kde plot for trihalomethanes # KDE Plot for the 'Trihalomethanes' column

fig = plt.figure()

ax = fig.add\_subplot(111)

# Ensure there are no NaN values in the column (KDE cannot handle NaNs) water\_data['Trihalomethanes'].dropna().plot(kind='kde', ax=ax)

# Optional: Add labels and title ax.set\_title('KDE of Trihalomethanes') ax.set\_xlabel('Trihalomethanes') plt.show()

water\_data['ph'] = water\_data['ph'].fillna(water\_data['ph'].mean()) #checking for null values after removing null values

water\_data['Trihalomethanes'] = water\_data['Trihalomethanes'].fillna(water\_data['Trihalomethanes'].mean())

water\_data['Sulfate'] = water\_data['Sulfate'].fillna(water\_data['Sulfate'].mean()) corr\_matrix = water\_data.corr() #checking for corelations

corr\_matrix

plt.figure(figsize=(18,16)) #visualizing correlations sns.heatmap(corr\_matrix,annot=True,cmap='coolwarm') plt.show()

import numpy as np corr\_matrix1 = corr\_matrix.abs()

upper\_tri = corr\_matrix1.where(np.triu(np.ones(corr\_matrix1.shape),k=1).astype(np.bool\_)) upper\_tri

matrix = np.triu(corr\_matrix)

sns.heatmap(water\_data.corr(),annot=True,linewidths=.8,mask=matrix,cmap="rocket",cbar= False)

data\_hist\_plot =water\_data.hist(figsize=(20,20),color = "#5F9EA0") for col in water\_data.columns:

sns.histplot(data=water\_data,x=col,kde=True,hue='Potability')

plt.show() water\_data.groupby('Potability').mean().T for col in water\_data.columns:

sns.boxenplot(data=water\_data,x=col)

plt.show()

sns.countplot(water\_data['Potability']) X = water\_data.drop('Potability',axis=1) Y = water\_data['Potability']

X.head()

Y.head()

from sklearn.preprocessing import StandardScaler std\_scaler = StandardScaler()

X\_Scaled =std\_scaler.fit\_transform(X)

X\_Scaled

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, # 20% test data random\_state=42, # reproducibility stratify=Y # maintain class balance

)

X\_train.shape,X\_test.shape

from sklearn.linear\_model import LogisticRegression from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier from sklearn.ensemble import ExtraTreesClassifier from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier from sklearn.ensemble import GradientBoostingClassifier from sklearn.naive\_bayes import GaussianNB

from sklearn.ensemble import AdaBoostClassifier LR = LogisticRegression()

DT = DecisionTreeClassifier() RF =

ExtraTreesClassifier() SVM = SVC()

KNN = KNeighborsClassifier() GBC = AdaBoostClassifier() ABC = AdaBoostClassifier() NB = GaussianNB()

from sklearn.model\_selection import cross\_val\_score from sklearn.model\_selection import cross\_val\_score

models = [LR, DT, RF, ETC, SVM, KNN, GBC, ABC, NB]

features = X\_Scaled labels = Y

CV = 5

accu\_list = [] ModelName = []

for model in models:

model\_name = model. class . name # fixed line

accuracies = cross\_val\_score(model, features, labels, scoring='accuracy', cv= CV) accu\_list.append(accuracies.mean() \* 100)

ModelName.append(model\_name)

model\_acc\_df = pd.DataFrame({"Model": ModelName,"Cross\_val\_Accuracy":accu\_list}) model\_acc\_df

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import StandardScaler # Step 1: Separate features and labels

X = water\_data.drop('Potability', axis=1) Y = water\_data['Potability']

# Step 2: Impute missing values in features

imputer = SimpleImputer(strategy='mean') # or 'median' X\_imputed = imputer.fit\_transform(X)

# Step 3: Scale the features scaler = StandardScaler()

X\_Scaled = scaler.fit\_transform(X\_imputed)

from sklearn.model\_selection import cross\_val\_score models =

CV = 5

accu\_list = [] ModelName = []

for model in models:

model\_name = model. class . name

accuracies = cross\_val\_score(model, features, labels, scoring='accuracy', cv=CV) accu\_list.append(accuracies.mean() \* 100)

ModelName.append(model\_name)

model\_acc\_df = pd.DataFrame({"Model": ModelName, "Cross\_val\_Accuracy": accu\_list}) model\_acc\_df

from sklearn.metrics import classification\_report from sklearn.impute import SimpleImputer

# Create the imputer and apply it to the feature data

imputer = SimpleImputer(strategy='mean') # or 'median', 'most\_frequent' X\_imputed = imputer.fit\_transform(X) # Impute original X before train-test split # Proceed with train-test split again

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X\_imputed, Y, test\_size=0.2, random\_state=42)

# Scale the imputed data

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train) X\_test = scaler.transform(X\_test)

from sklearn.metrics import classification\_report # Train models

SVM.fit(X\_train, Y\_train) ETC.fit(X\_train, Y\_train) RF.fit(X\_train, Y\_train)

# Predict

Y\_pred\_svm = SVM.predict(X\_test) Y\_pred\_etc = ETC.predict(X\_test) Y\_pred\_rf = RF.predict(X\_test)

# Evaluate

print("Support Vector Machine Classification Report:") print(classification\_report(Y\_test, Y\_pred\_svm)) print("\nExtra Trees Classifier Classification Report:") print(classification\_report(Y\_test, Y\_pred\_etc)) print("\nRandom Forest Classification Report:") print(classification\_report(Y\_test, Y\_pred\_rf))

from sklearn.metrics import roc\_curve, auc import matplotlib.pyplot as plt

# Predict probabilities for ROC

Y\_score = ETC.predict\_proba(X\_test)[:, 1] # Compute ROC curve and AUC

fpr, tpr, threshold = roc\_curve(Y\_test, Y\_score) roc\_auc = auc(fpr, tpr)

# Plot ROC curve plt.figure(figsize=(7, 5))

plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (area = {roc\_auc:.2f})') plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05]) plt.xlabel('False Positive Rate') plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC)') plt.legend(loc="lower right")

plt.show()

from sklearn.model\_selection import GridSearchCV, RandomizedSearchCV, StratifiedKFold

from sklearn.ensemble import RandomForestClassifier

# Define hyperparameters params\_RF = {

"min\_samples\_split": [2, 6],

"min\_samples\_leaf": [1, 4],

"n\_estimators": [100, 200, 300], "criterion": ["gini", "entropy"]

}

# Cross-validation method

cv\_method = StratifiedKFold(n\_splits=3) # Grid Search

GridSearchCV\_RF = GridSearchCV( estimator=RandomForestClassifier(), param\_grid=params\_RF, cv=cv\_method,

verbose=1, n\_jobs=2, scoring="accuracy",

return\_train\_score=True

)

# Fit model GridSearchCV\_RF.fit(X\_train, Y\_train)

# Best parameters

best\_params\_RF = GridSearchCV\_RF.best\_params\_

print("Best Hyperparameters for Random Forest are =", best\_params\_RF)

best\_estimator = GridSearchCV\_RF.best\_estimator\_ best\_estimator.fit(X\_train,Y\_train)

Y\_pred\_best = best\_estimator.predict(X\_test) print(classification\_report(Y\_test,Y\_pred\_best)) from sklearn.metrics import accuracy\_score

# Get the best estimator from GridSearchCV best\_rf\_model = GridSearchCV\_RF.best\_estimator\_ # Predict on the test seta

Y\_pred\_best = best\_rf\_model.predict(X\_test) # Print accuracy

print(f"Accuracy of Random Forest Model = {round(accuracy\_score(Y\_test, Y\_pred\_best)\*100, 2)}%")

water\_data.columns

list1 = water\_data.iloc[2:3,0:9].values.flatten().tolist() list1

ph = float(input('Enter the Ph Value = '))

Hardness = float(input('Enter the Hardness Value = ')) Solids = float(input('Enter the Solids Value = '))

Chloramines = float(input('Enter the Chloramines Value = ')) Sulfate = float(input('Enter the Sulfate Value = ')) Conductivity = float(input('Enter the Conductivity Value = '))

Organic\_Carbon = float(input('Enter Organic Carbon Value = ')) Trihalomethanes = float(input('Enter the Trihalomethanes Value = ')) Turbidity = float(input('Enter the Turbidity Value = '))

input\_data = [ph,Hardness,Solids,Chloramines,Sulfate,Conductivity,Organic\_Carbon,Trihalomethanes,Tur bidity]

water\_data\_input = std\_scaler.transform([[ph,Hardness,Solids,Chloramines,Sulfate,Conductivity,Organic\_Carbon

,Trihalomethanes,Turbidity]])

water\_data\_input

model\_prediction = best\_estimator.predict(water\_data\_input) model\_prediction

if model\_prediction[0] == 0:

print("Water is Not SAFE for Consumption") else:

print("Water is SAFE for Consumption")

# 5.3 OUTPUT/SCREENSHOTS

Fig 5.1 : Dataset represents the Rows and Columns of the data set and the information in it.

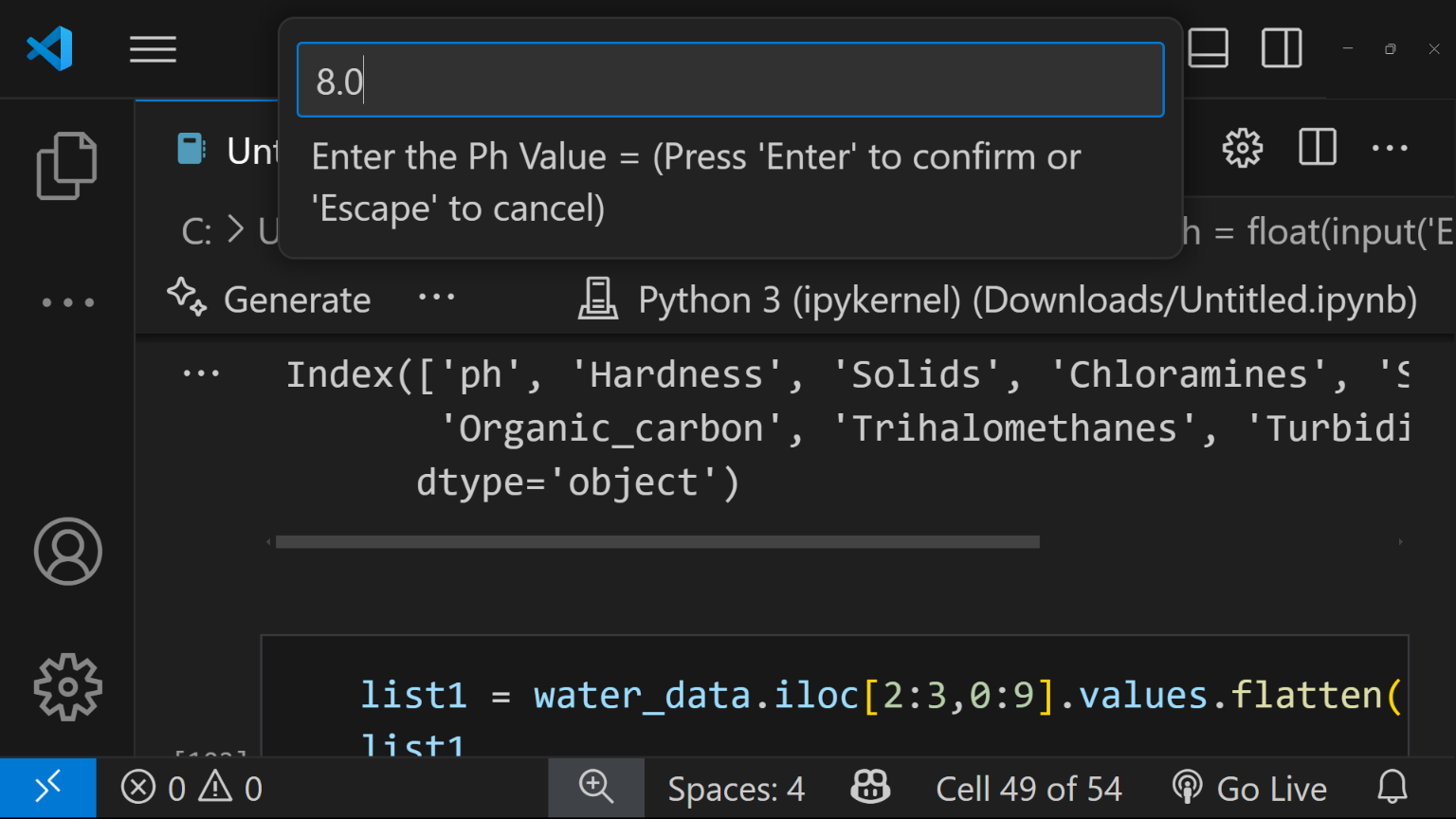


Fig 5.2 : Entering the Ph Value.

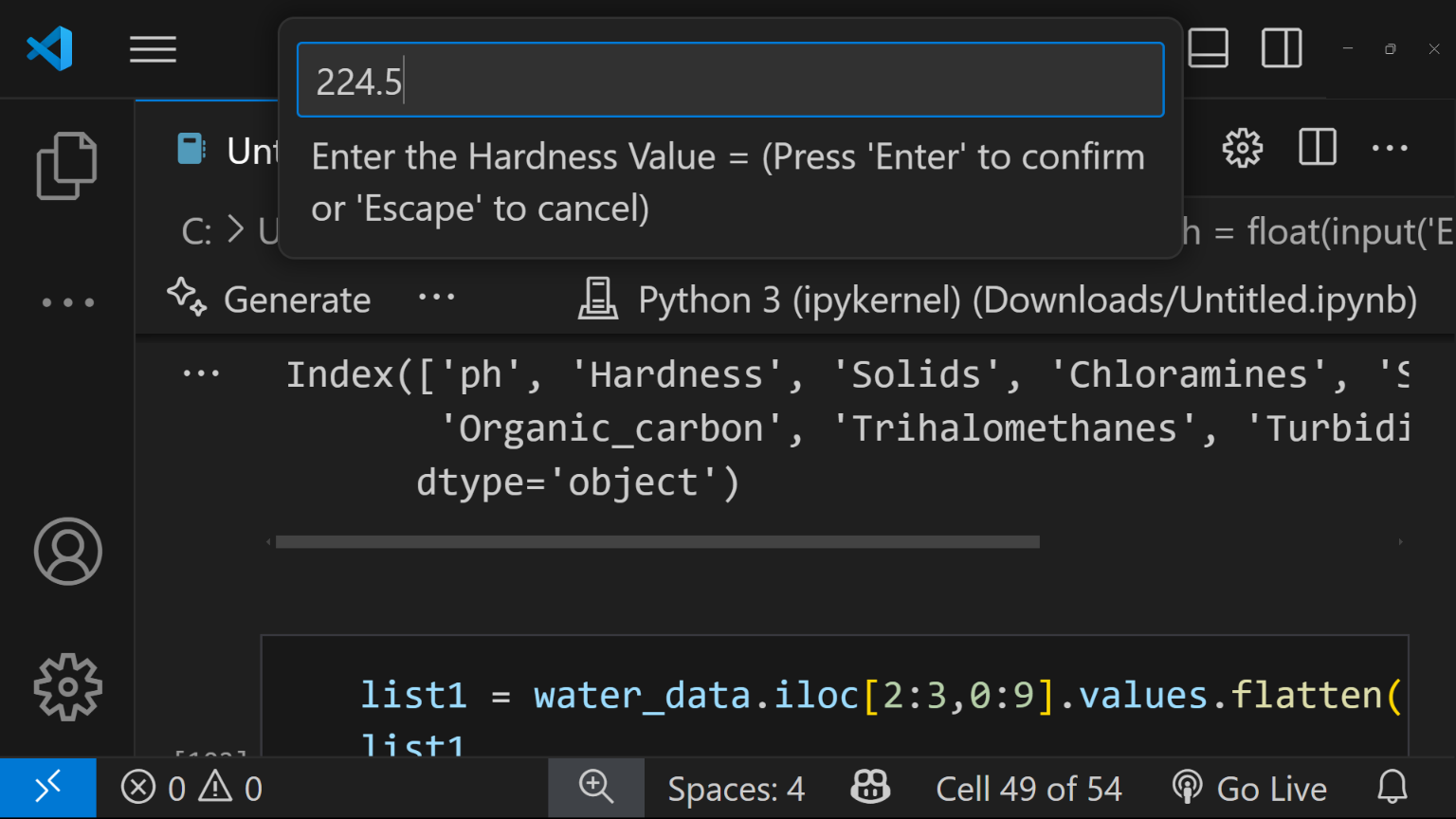


Fig 5.3 : Entering the Hardness Value.

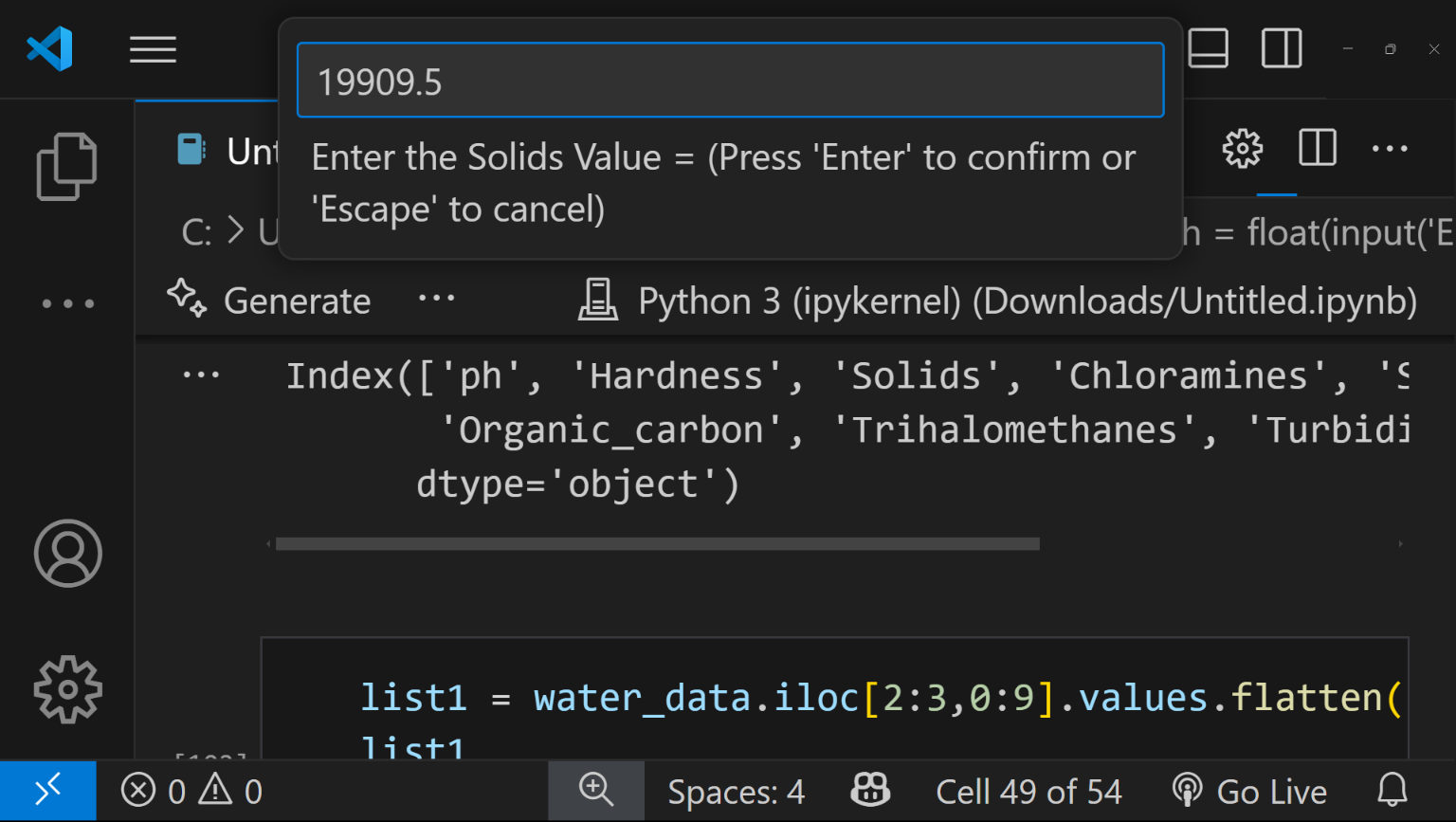


Fig 5.4 : Entering the Solids Value.

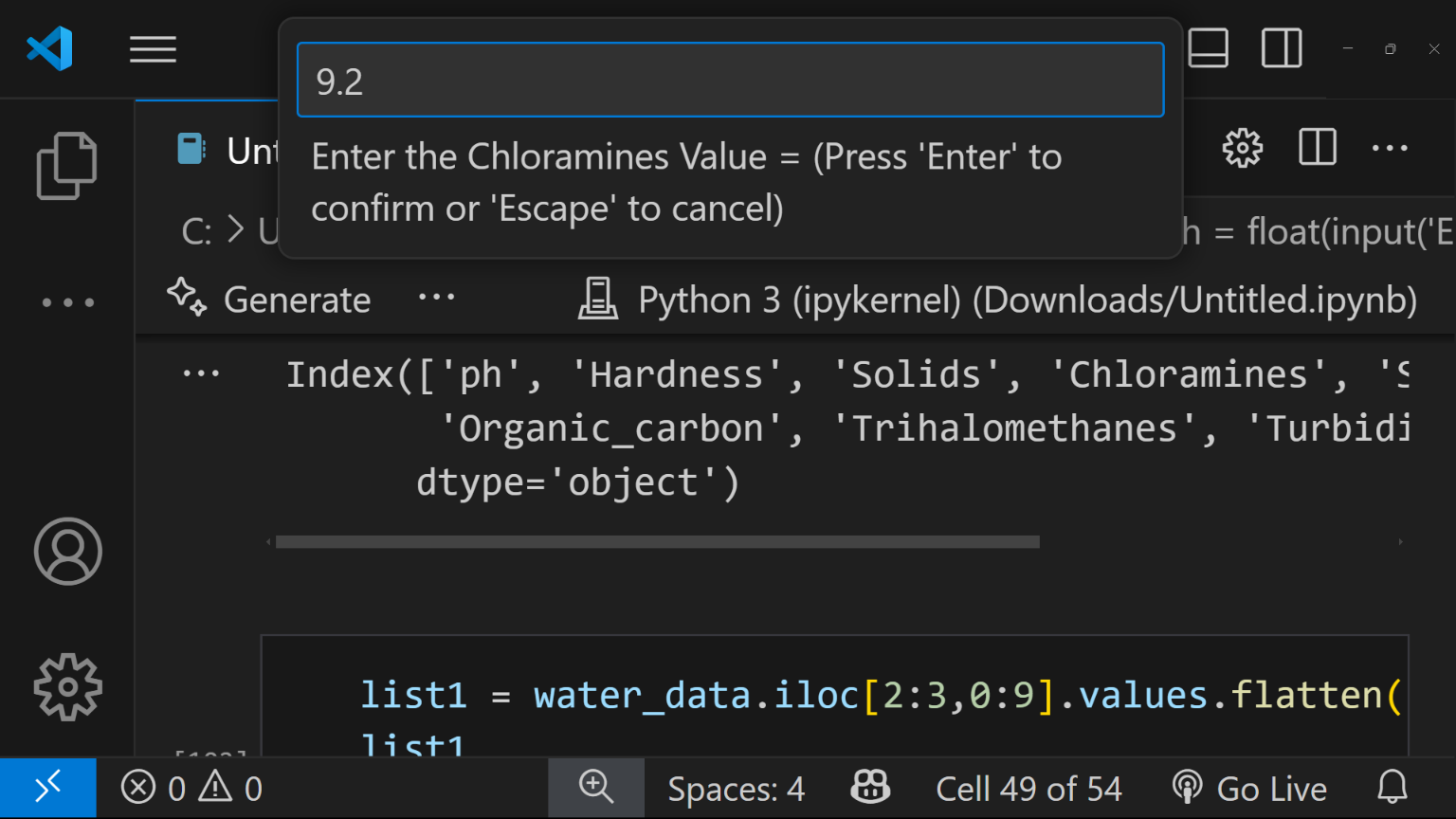


Fig 5.5 : Entering the Chloramines Value.

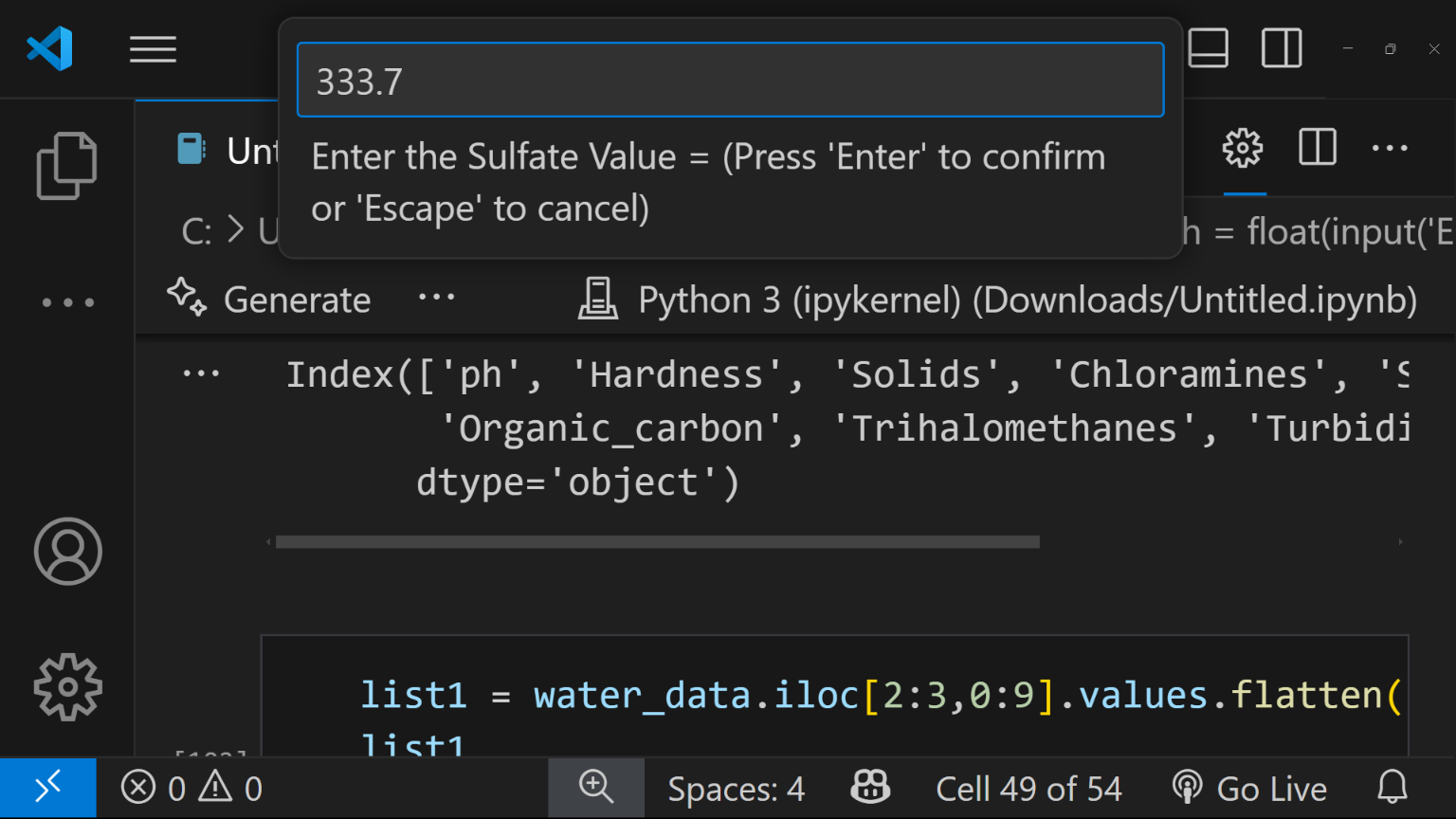


Fig 5.6 : Entering the Sulfates Value.

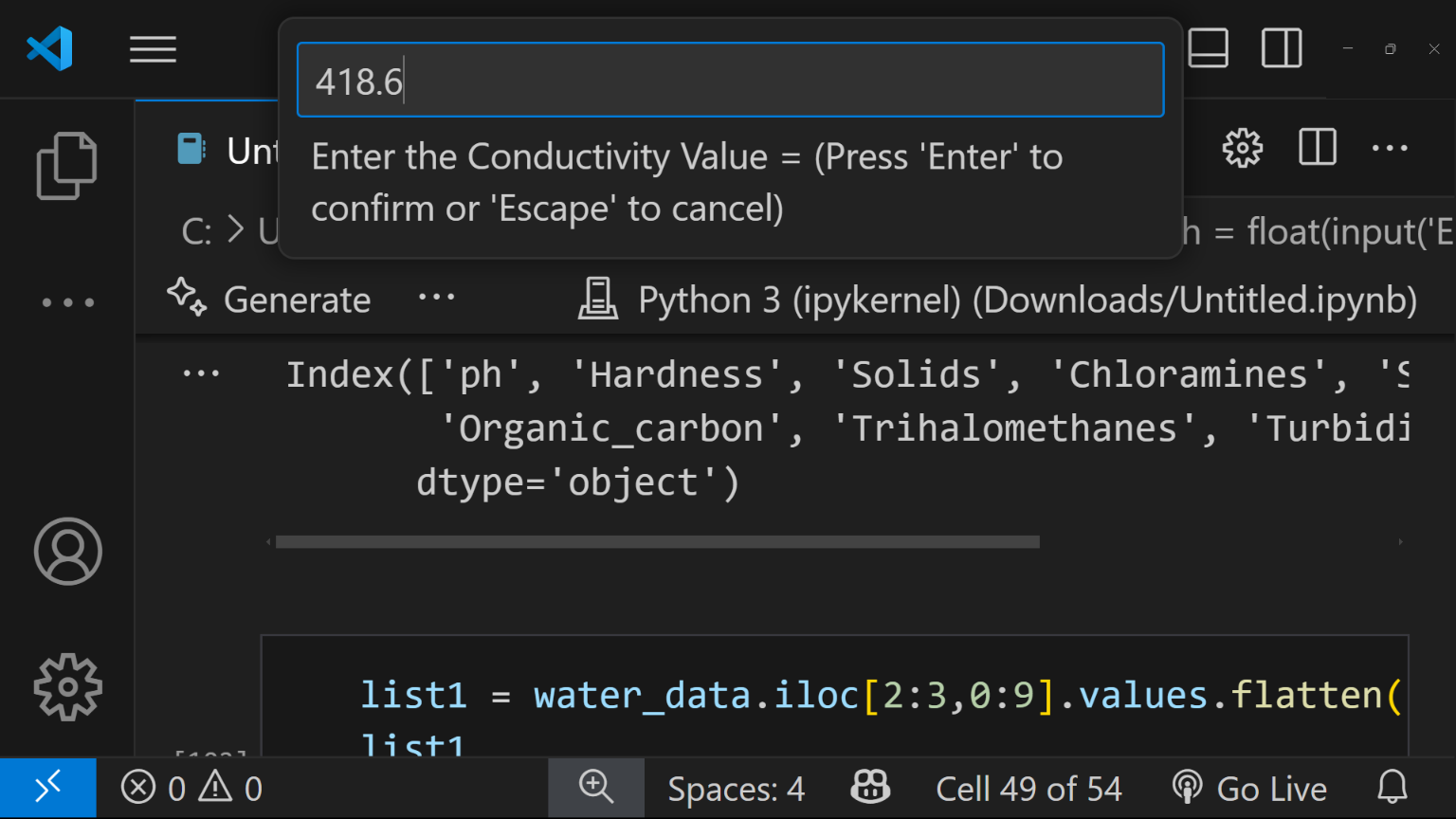


Fig 5.7 : Entering the Conductivity Value.

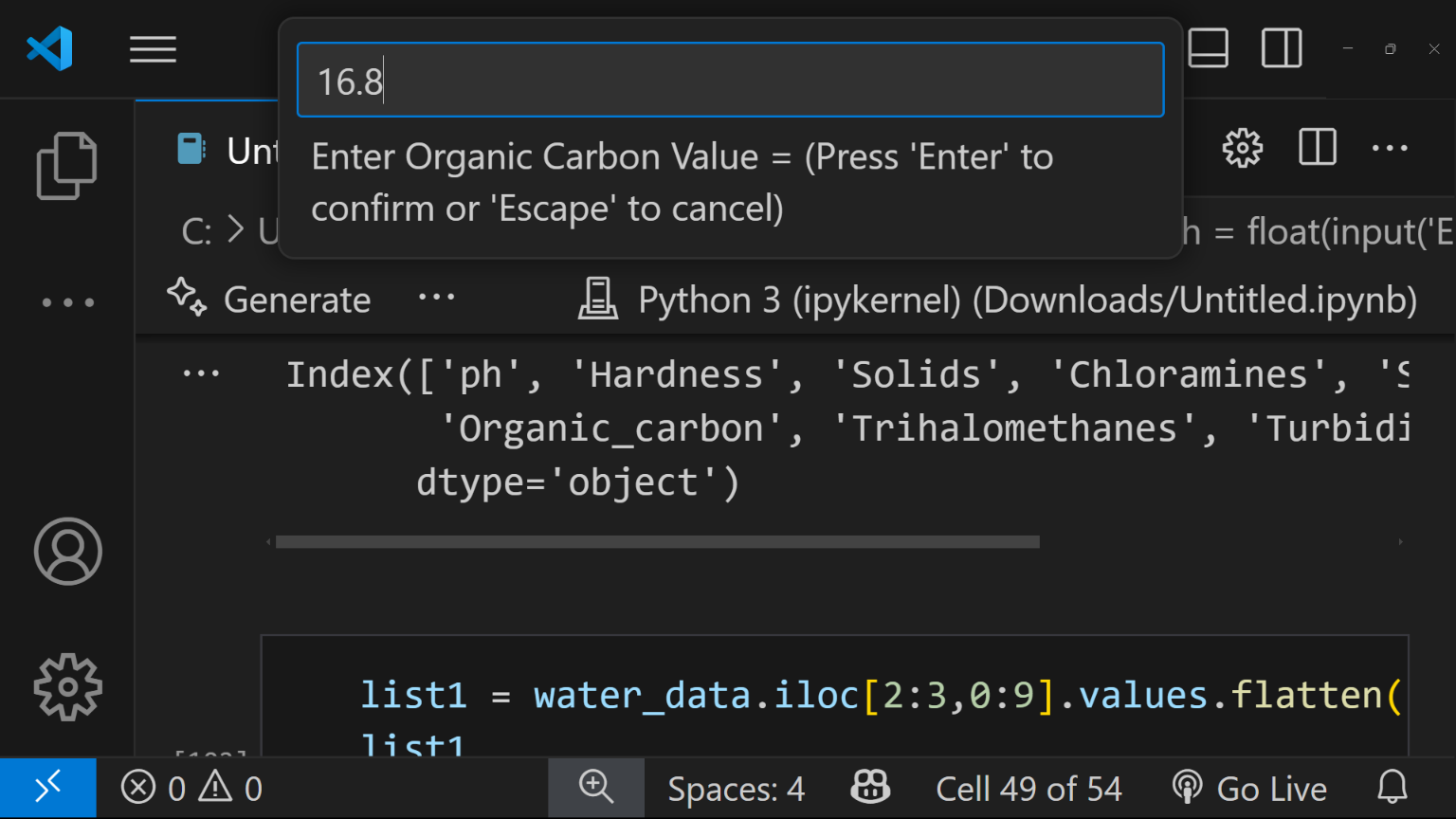


Fig 5.8 : Entering the Organic Carbon Value.

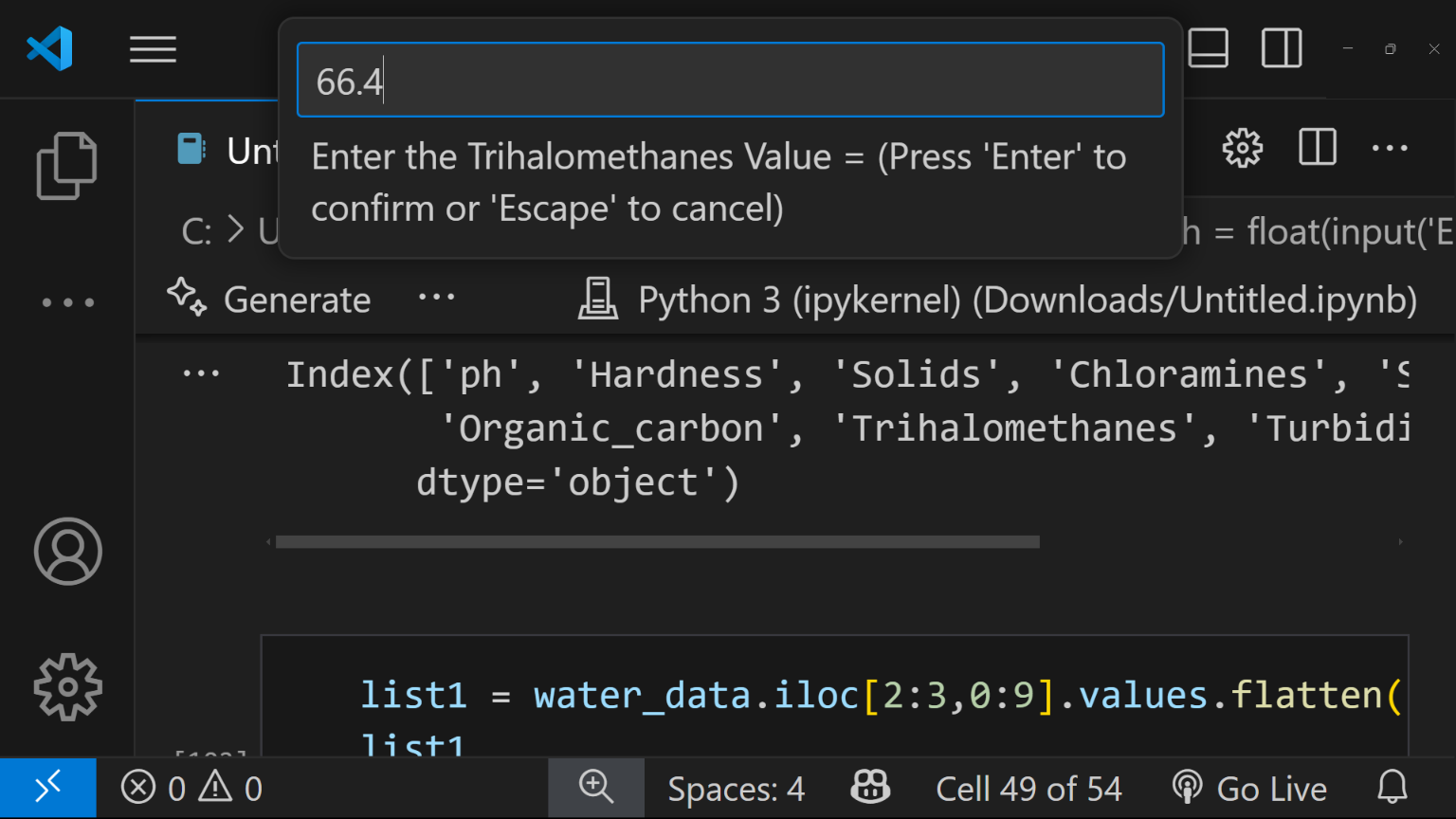


Fig 5.9 : Entering the Trihalomethanes Value.

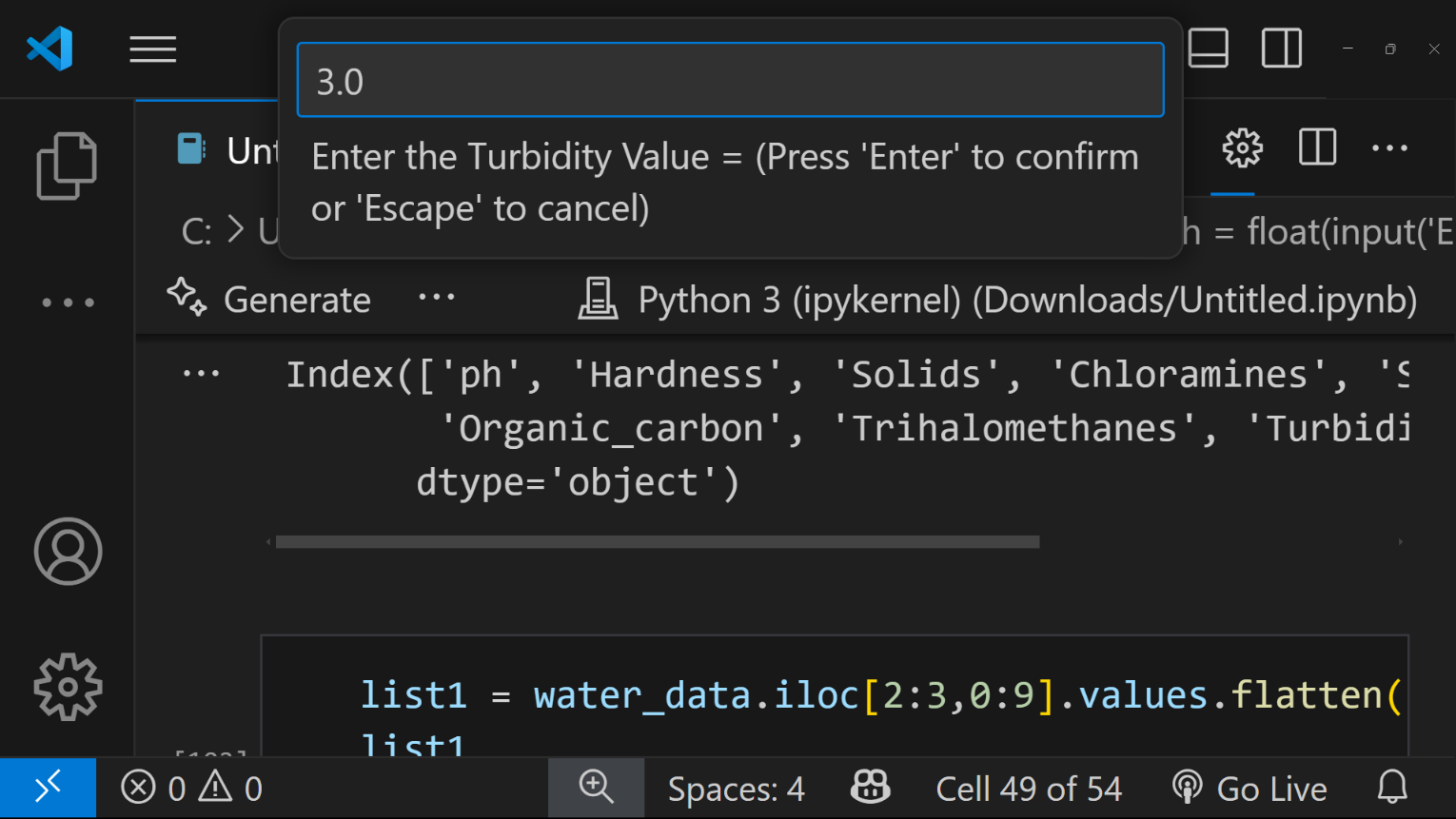


Fig 5.10 : Entering the turbidity value

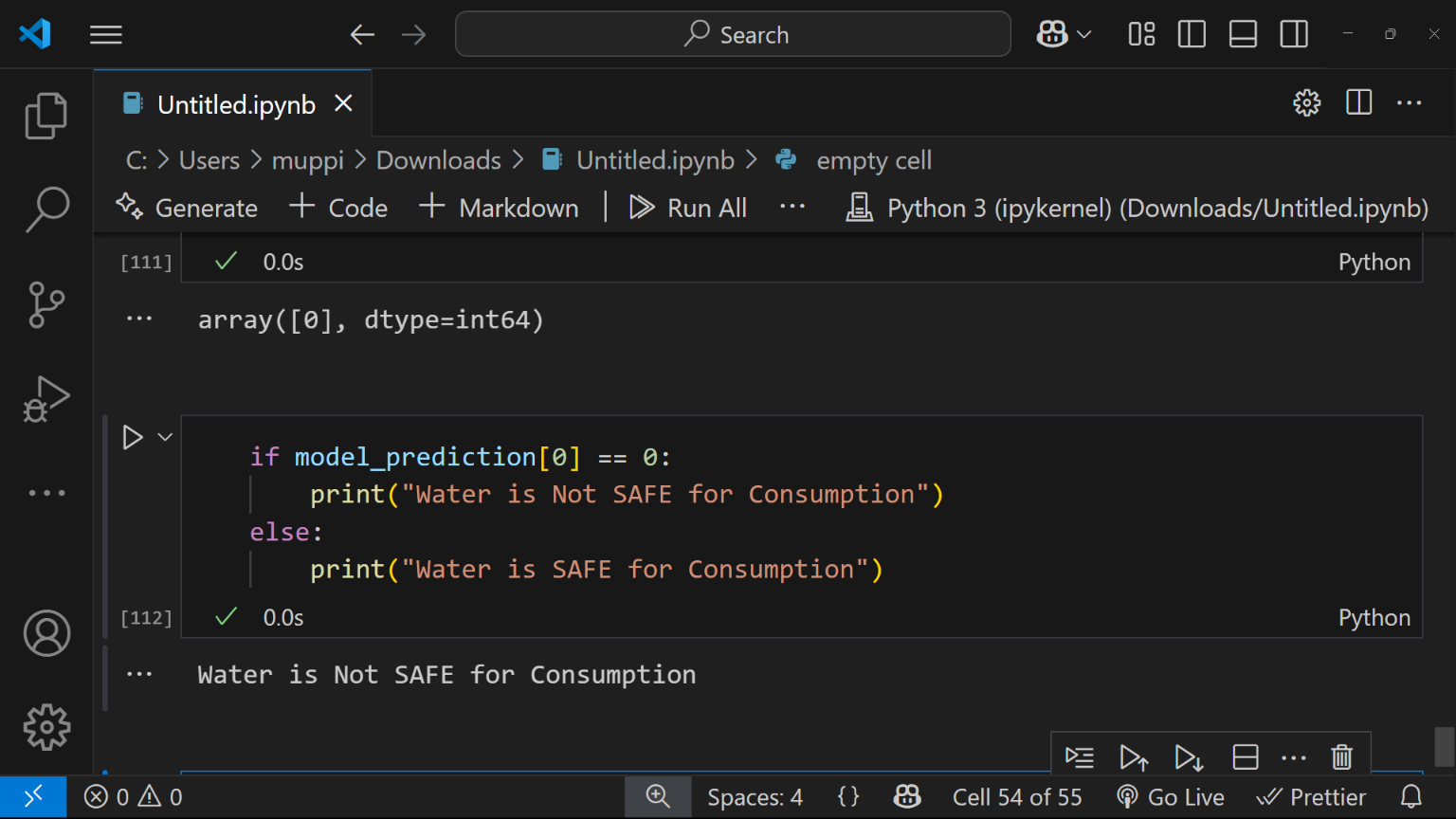


Fig 5.11 Final Output – Water is Not SAFE for Consumption.

**CHAPTER-06**

**TESTING AND VALIDATION**

**6.1 Introduction**

To ensure the correctness, reliability, and practical applicability of the water quality prediction model, a systematic approach was adopted by designing and executing a series of diverse test cases. These test cases simulate real-world water quality conditions and are intended to verify how well the model generalizes beyond the training data. The primary objective of this testing phase is to ensure that the model can correctly classify new, unseen inputs as either potable (safe for consumption) or non-potable (unsafe for consumption). Each test case consists of a complete set of input values representing all required physicochemical parameters that influence water quality, such as pH, Hardness, Solids, Chloramines, Sulfate, Conductivity, Organic Carbon, Trihalomethanes, and Turbidity.

The model’s predictive process begins with accepting user-provided or test-defined values for these parameters. Before prediction, the input data is first standardized using the same StandardScaler instance that was applied during training, ensuring that the feature scales remain consistent. The output is then interpreted and translated into user-friendly messages: *"Water is SAFE for Consumption"* or *"Water is NOT SAFE for Consumption"*, respectively.

Beyond ideal cases, boundary and edge test cases were also explored to test the model’s response to extreme inputs. In one case, an input sample had a very high sulfate concentration, approaching the upper environmental safety limit. Similarly, a test case was conducted with borderline safe levels of all parameters—values just within the acceptable thresholds. The model predicted this sample as potable, showcasing its ability to handle nuanced situations where the distinction between safe and unsafe is less clear.

Another important category of testing involved incomplete or artificially constructed cases that are unlikely but possible in real-world scenarios—for instance, extremely low values across all features, or a sample with high levels of organic carbon but low chloramine. These test cases aimed to identify any unusual biases or erratic behavior in the model’s prediction logic. In all cases, the model’s responses aligned with domain expectations or known safe limits, increasing confidence in its real-world applicability.

Overall, the use of such varied and strategically designed test cases provides strong validation of the model's performance, stability, and generalization ability. By simulating realistic and edge-case scenarios, the model was tested under different conditions that it might encounter in actual deployment. This rigorous testing phase ensures that the prediction system is not just theoretically accurate, but also practical and trustworthy for end-users, including environmental analysts, water supply authorities, and the general public. The clear and intuitive prediction messages further enhance its usability, allowing individuals with no technical.

**6.2 Test Case Scenarios**

Table 6.1 Test Case Scenarios

|  |  |  |
| --- | --- | --- |
| **Test Case ID** | **Input Parameters (Sample)** | **Expected Output** |
| TC-01 | pH: 7.0, Hardness: 180, Solids: 15000, Chloramines: 7.2, Sulfate: 333, Conductivity: 410, Organic Carbon: 10, Trihalomethanes: 80, Turbidity: 3.5 | Water is SAFE for Consumption |
| TC-02 | pH: 3.2, Hardness: 230, Solids: 20000, Chloramines: 6.0, Sulfate: 200, Conductivity: 300, Organic Carbon: 15, Trihalomethanes: 90, Turbidity: 6.5 | Water is NOT SAFE for Consumption |
| TC-03 | pH: 6.5, Hardness: 100, Solids: 10000, Chloramines: 4.0, Sulfate: 250, Conductivity: 500, Organic Carbon: 8, Trihalomethanes: 60, Turbidity: 2.5 | Water is SAFE for Consumption |
| TC-04 | pH: 8.5, Hardness: 310, Solids: 22000, Chloramines: 8.0, Sulfate: 180, Conductivity: 250, Organic Carbon: 12, Trihalomethanes: 110, Turbidity: 7.0 | Water is NOT SAFE for Consumption |
| TC-05 | pH: 6.8, Hardness: 140, Solids: 12000, Chloramines: 6.5, Sulfate: 250, Conductivity: 370, Organic Carbon: 9, Trihalomethanes: 70, Turbidity: 3.0 | Water is SAFE for Consumption |
| TC-06 | |  | | --- | |  |  |  | | --- | | pH: 5.0, Hardness: 320, Solids: 28000, Chloramines: 9.0, Sulfate: 400, Conductivity: 600, Organic Carbon: 18, Trihalomethanes: 120, Turbidity: 8.0 | | Water is NOT SAFE for Consumption |

**CHAPTER-07**

**CONCLUSION & FUTURE ENHANCEMENT**

**CONCLUSION**

In our comprehensive Water Quality Analysis Using Machine Learning project, we employed a diverse set of machine learning algorithms, including Support Vector Machine (SVM), Linear Regression, Random Forest, k-Nearest Neighbors (KNN), and Decision Tree, to predict and assess critical water quality parameters. Through rigorous experimentation and evaluation, we have gained valuable insights into their respective capabilities and performance. The results of our analysis have revealed a clear ranking of these algorithms in terms of predictive accuracy and generalization: Support Vector Machine (SVM) emerged as the top-performing algorithm in our study, consistently delivering the most accurate predictions for water quality parameters. SVM's ability to handle complex decision boundaries and its robust generalization make it an excellent choice for this task. Linear Regression, while more straightforward in its approach, demonstrated strong performance, securing the second position in our ranking. Linear models like this one can provide valuable insights into linear relationships between variables in water quality data. Random Forest proved to be a competitive choice for water quality analysis, showcasing its ensemble learning capabilities and ability to capture complex interactions within the data. k-Nearest Neighbors (KNN), although not at the top of our ranking, still displayed respectable performance. Fine-tuning the number of neighbors and exploring distance-weighted voting may further enhance its predictive accuracy. Decision Tree, while a valuable starting point, showed limitations in predictive accuracy compared to the other algorithms. However, its interpretability can be advantageous for understanding decision rules within the data.

**FUTURE ENHANCEMENT**

In the next phase of our water quality analysis project, we will focus on three key areas for future enhancements.

* Firstly, we will integrate deep learning techniques, specifically convolutional neural networks (CNNs) and recurrent neural networks (RNNs), to extract complex spatial and temporal patterns from water quality data, aiming to enhance predictive accuracy.
* Secondly, we plan to expand our real-time monitoring network by deploying additional sensors in strategic locations, enabling more extensive data coverage and quicker responses to emerging water quality issues.
* Thirdly, we will develop a robust anomaly detection system to automatically identify and alert stakeholders to abnormal water quality events, serving as a valuable early warning mechanism.

**CHAPTER-08**

**BIBLIOGRAPHY**

[1] Bhumika Nandwana, Satyanarayan Tazi, Sheifalee Trivedi, Dinesh Kumar, and Santosh Kumar Vipparthi. “A survey paper on hand gesture recognition”. In: 2017 7th International Conference on Communication Systems and Network Technologies (CSNT). 2017, pp. 147–152. DOI: 10.1109/CSNT.2017.8418527.

[2] Harpreet Kaur and Jyoti Rani. “A review: Study of various techniques of Hand gesture recognition”. In: 2016 IEEE 1st International Conference onPower Electronics, Intelligent Control and Energy Systems (ICPEICES). 2016, pp. 1–5. DOI: 10.1109/ICPEICES.2016.7853514.

[3] Jayesh S. Sonkusare, Nilkanth B. Chopade, Ravindra Sor, and Sunil L. Tade. “A Review on Hand Gesture Recognition System”. In: 2015 Interna tional Conference on Computing Communication Control and Automation. 2015, pp. 790–794. DOI: 10.1109/ICCUBEA.2015.158.

[4] Hamid A. Jalab and Herman. K. Omer. “Human computer interface using hand gesture recognition based on neural network”. In: 2015 5th National Symposium on Information Technology: Towards New Smart World (NSITNSW). 2015, pp. 1–6. DOI: 10.1109/NSITNSW.2015.7176405.

[5] Hung-Yuan Chung, Yao-Liang Chung, and Wei-Feng Tsai. “An Efficient Hand Gesture Recognition System Based on Deep CNN”. In: 2019 IEEE International Conference on Industrial Technology (ICIT). 2019, pp. 853– 858. DOI: 10.1109/ICIT.2019.8755038.

[6] Zoltan Vamossy, Andras Toth, and Balazs Benedek. “Virtual Hand - Hand Gesture Recognition System”. In: 2007 5th International Symposium on Intelligent Systems and Informatics. 2007, pp. 97–102. DOI: 10.1109/ SISY.2007.4342632.