# 

**A**

**MAJOR PROJECT REPORT**

# ON

**WATER QUALITY ANALYSIS USING EXTRA TREE**

**CLASSIFIER**

# BACHELOR OF ENGINEERING

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**Submitted By**

**(204-2025)**

|  |  |
| --- | --- |
| **KOLIPALLY SAI VAMSHI REDDY** | **245521733027** |
| **VUPPALA AKSHAY** | **245521733063** |
| **K. TRISHANK** | **245521733301** |

**Under the guidance**

**Of**

## Mrs. V. SWAPNA

## ASSISTANT PROFESSOR



**Department of Computer Science and Engineering**

# KESHAV MEMORIAL ENGINEERING COLLEGE

Kachavanisingaram Village, Hyderabad, Telangana 50088.

Affiliated to Osmania University, Hyderabad



## A logo with text and a white background Description automatically generated with medium confidenceKESHAV MEMORIAL ENGINEERING COLLEGE

A unit of Keshav Memorial Technical Education Society (KMITES)

(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad)

D.NO. 10-TC-111, Kachavanisingaram(V), Ghatkesar (M), Medchal Malkajgiri, Telangana – 50088 Website: [www.kmec.in](http://www.kmec.in/) Email-id: [principal@kmec.in](mailto:principal@kmec.in) M: +91 40 29560274

**CERTIFICATE**

*This is to certify that the Major project work entitled* “**WATER QUALITY ANALYSIS USING EXTRA TREE CLASSSIFIER”** *is a bonafide work carried out by* **K.SAIVAMSHIREDDY(245521733027**),**V.AKSHAY(245521733063**),**K.TRISHANK(245521733301)** of IV Year VII semester **Bachelor of Engineering** *in CSE by Osmania University, Hyderabad during the academic year* **2024-2025** *is a record of bona fide work carried out by them*. *The results embodied in this report have not been submitted to any other University or Institution for the award of any degree.*

### Internal Guide Head of Department

Mrs.V.SWAPNA Dr.CH.Rathan kumar

Assistant Professor Associate Professor

**External**

## A logo with text and a white background Description automatically generated with medium confidenceKESHAV MEMORIAL ENGINEERING COLLEGE

A unit of Keshav Memorial Technical Education Society (KMITES)

(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad)

D.NO. 10-TC-111, Kachavanisingaram(V), Ghatkesar (M), Medchal Malkajgiri, Telangana – 50088 Website: [www.kmec.in](http://www.kmec.in/) Email-id: [principal@kmec.in](mailto:principal@kmec.in) M: +91 40 29560274

# DECLARATION

We hereby declare that the Major Project Report entitled, “**WATER QUALITY ANALYSIS USING EXTRA TREE CLASSIFIER**” submitted for the B.E degree is entirely our work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree.

**Date:**

**KOLIPALLY SAI VAMSHI REDDY 245521733027**

**VUPPALA AKSHAY 245521733063**

**K.TRISHANK 245521733301**

## A logo with text and a white background Description automatically generated with medium confidenceKESHAV MEMORIAL ENGINEERING COLLEGE

A unit of Keshav Memorial Technical Education Society (KMITES)

(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad)

D.NO. 10-TC-111, Kachavanisingaram(V), Ghatkesar (M), Medchal Malkajgiri, Telangana – 50088 Website: [www.kmec.in](http://www.kmec.in/) Email-id: [principal@kmec.in](mailto:principal@kmec.in) M: +91 40 29560274

# ACKNOWLEDGEMENT

We are happy to express our deep sense of gratitude to the principal of the college  **Dr.P.V.N.PRASAD, Professor**, Keshav memorial engineering college, for having provided us with adequate facilities to pursue our project.

We would like to thank**, Dr. CH. Rathan Kumar**, **Head of the Department CSE,** Keshav memorial engineering college, for having provided the freedom to use all the facilities available in the department, especially the laboratories and the library.

We would also like to thank my internal guide**”Mrs.V.SWAPNA”** for her technical guidance & constant encouragement.

We sincerely thank our seniors and all the teaching and non-teaching staff of the Department of Computer Science & Engineering for their timely suggestions, healthy criticism and motivation during this work.

Finally, we express our immense gratitude with pleasure to the other individuals who have either directly or indirectly contributed to our need at the right time for the development and success of this work.

**I**

# ABSTRACT

Based on many studies, it is now clear that the quality of the water one consumes defines the quality of their health. To prevent waterborne illnesses such as Typhoid, Cholera, the quality of drinking water should be kept in check. This project focuses on classifying whether a given water sample is potable or not.Our task is to develop an Extra Tree Classifier model for water quality analysis to predict the contamination level of water samples. The goal is to accurately categorize water samples into different quality classes (e.g., safe, mildly polluted, highly polluted) based on various physicochemical and biological attributes. This model aims to enhance water management by providing timely and reliable assessments of water quality, aiding in prompt decision-making for public health and environmental protection The proposed methodology employs nine input parameters, pH, hardness, solid content, chloramine, sulphates, conductivity, presence of organic carbon, trihalomethanes and turbidity of all the employed algorithms, the extra trees classifier comes out to be the most precise at 0.69% accuracy whilst a densely connected (9,32,2) ,Using XG Boost classifier or AdaBoost classifier yields 0.64% accuracy respectively with similar hyperparameters. Some of the existing and currently employed water quality prediction methods include the usage of Improved Grey Relational Analysis (IGRA) algorithm, LongShort Term Memory (LSTM) neural network and Hybrid decision tree. The Extra trees classifier outperforms the present implementations on the basics of accuracy by at least 4%. Extra Trees is like Random Forest, in that it builds multiple trees and splits nodes using random subsets of features, but with two key differences: it does not bootstrap observations (meaning it samples without replacement), and nodes are split on random splits but, not best splits which goes beyond.

**Keywords:** Extra Tree Classifier Artificial neural network, Random Forest Improved Grey Relational

Analysis Long-Short Term Memory Neural Network

# Key Features

* Algorithmic Features
* Water Quality Analysis Features
* Data Handling Features
* Software Features
* Hardware Features

# Skills Demonstrated

* **Machine Learning**: Implementing and training an Extra Tree Classifier model for water quality analysis.
* **Data Analysis**: Preprocessing, visualizing, and analyzing water quality data.
* **Programming**: Writing clean, efficient, and well-documented Python code.
* **Problem-solving**: Applying machine learning techniques to solve a real-world problem.

# System Requirements

* **Operating System:** Windows, macOS, or Linux
* **CPU:**Intel Core i3 or equivalent
* **RAM:**4 GB (minimum), 8 GB
* **IDE:**Jupyter Notebook, PyCharm, or any other Python-compatible ID

**II**

# TABLE OF CONTENTS

**CHAPTER TITLE PAGE NO.**

**ACKNOWLEDGEMENT I**

[ABSTRACT II](#_TOC_250008)

LIST OF FIGURES V

LIST OF TABLES V

1. INTRODUCTION
   1. [PROBLEM STATEMENT **1**](#_TOC_250007)
   2. [MOTIVATION **1**](#_TOC_250006)
   3. [SCOPE **1**](#_TOC_250005)
   4. [OUTLINE **2**](#_TOC_250004)
2. LITERATURE SURVEY
   1. EXISTING SYSTEM **3**
   2. PROPOSED SYSTEM **3**
3. SOFTWARE REQUIREMENT SPECIFICATION
   1. [OVERALL DESCRIPTION **4**](#_TOC_250003)
   2. [OPERATING ENVIRONMENT **4**](#_TOC_250002)
   3. [FUNCTIONAL REQUIREMENTS **4-5**](#_TOC_250001)
   4. [NON – FUNCTIONAL REQUIREMENTS **5-7**](#_TOC_250000)

III

1. SYSTEM DESIGN
   1. USE-CASE DIAGRAM **8**
   2. CLASS DIAGRAM **9**
   3. SEQUENCE DIAGRAM **10**
2. **IMPLEMENTATION**
   1. SAMPLE CODE **11 – 47**

## TESTING

* 1. TESTING **48-50**

1. **SCREENSHOTS 51-53**
2. **CONCLUSION AND FUTURE SCOPE 54**

**BIBLIOGRAPHY 55**

**APPENDIX A:** TOOLS AND TECHNOLOGY **56**

**IV**

|  |  |  |
| --- | --- | --- |
| List of Figures |  | |
| **Figure No.** | **Name of Figure** | **Page No.** |
| 1. | Use case Diagram | **8 - 9** |
| 2. | Class Diagram | **10** |
| 3. | Sequence Diagram | **11** |

|  |  |  |
| --- | --- | --- |
| **List of Tables** |  | |
| **Table No.** | **Name of Table** | **Page No.** |
| 1. | Testcases | **48 - 50** |

# V

**CHAPTER – 1 INTRODUCTION**

## PROBLEM STATEMENT

Our task is to develop an Extra Tree Classifier model for water quality analysis to guess or

predict the contamination level of water samples. The goal is to accurately categorize water samples into different quality classes (e.g., safe, mildly polluted, highly polluted) based on various physicochemical and biological attributes. This model aims to enhance water management by providing timely and reliable assessments of water quality, aiding in prompt decision-making for public health and environmental protection.

## MOTIVATION

Water quality assessment is crucial for safeguarding human health and the environment. Conventional methods can be time-consuming and expensive. By employing an Extra Tree Classifier, we can harness its ability to handle complex relationships in water quality data, leading to efficient and accurate classification. This approach not only streamlines analysis but also contributes to proactive measures for maintaining clean water resources.

## SCOPE

The scope for water quality classification using the Extra Trees Classifier lies in its ability to handle multiple input variables, handle noisy data, and capture complex relationships. By applying this algorithm to a dataset containing various water quality parameters, such as pH, hardness, dissolved solids, turbidity, and trihalomethanes, it can accurately classify water samples into different quality classes. This approach has the potential to assist environmental management, identify pollution sources, prioritize remediation efforts, and support decision-making processes for maintaining and improving the health of aquatic ecosystems, making it a valuable tool for monitoring agencies, researchers, and policymakers.

## OUTLINE

Here is an outline of major components that are included in a water quality analysis using extra tree classifier: data collection, feature preprocessing, Extra Tree Classifier model training,and accuracy assessment for effective water quality analysis.

# CHAPTER – 2 LITERATURE SURVEY

## EXISTING SYSTEM:

Water Quality Classification Using Machine Learning Techniques" by Ayesha Khalid et al. (2019): This study explores the application of machine learning algorithms, including decision trees, random forests, support vector machines, and artificial neural networks, for water quality classification. The research compares the performance of these algorithms on various water quality parameters and evaluates their effectiveness in differentiating water quality classes.

"Water Quality Classification Using Machine Learning Techniques: A Comparative Study" by Charu Gupta et al. (2020): This research presents a comparative analysis of different machine learning algorithms, such as decision trees, random forests, k-nearest neighbors, and naïve Bayes, for water quality classification. The study evaluates the performance of these algorithms on a dataset containing multiple water quality parameters and provides insights into their accuracy and efficiency.

"Water Quality Classification Using Machine Learning Algorithms" by Subha Gomathy et al. (2021): This study investigates the use of machine learning algorithms, including decision trees, random forests, and support vector machines, for water quality classification. The research compares the classification performance of these algorithms on a dataset comprising parameters like pH, dissolved oxygen, and biochemical oxygen demand, and analyzes their effectiveness in water quality assessment.

"Water Quality Classification using Machine Learning Algorithms: A Case Study in the Amazon River Basin" by Felipe A. S. Pereira et al. (2021): This research focuses on water quality classification in the Amazon River Basin using machine learning algorithms. The study applies decision trees, random forests, and gradient boosting algorithms to classify water samples based on multiple parameters, including turbidity, chlorophyll-a, and conductivity.

## PROPOSED SYSTEM:

The proposed system for water quality classification using the Extra Trees Classifier aims to integrate real-time data collection from multiple sources, such as sensors and satellite imagery, with the robust classification capabilities of the Extra Trees algorithm. The system will leverage the ensemble learning approach of Extra Trees to accurately classify water samples into various quality categories, providing timely insights for water resource management and decision-making. Additionally, the system will incorporate features like data visualization, anomaly detection, and automated alert mechanisms to enhance the usability and practicality of the classification results.

Advantages:

 Robust Handling of Noisy Data: The Extra Trees Classifier is known for its robustness in handling noisy and incomplete data. In water quality assessment, where data can be subject to measurement errors or missing values, the Extra Trees Classifier can effectively handle such challenges and still provide accurate classification results. It employs an ensemble learning approach that combines multiple decision trees, which helps reduce the impact of noise and outliers in the data.

 Efficient Feature Selection: The Extra Trees Classifier performs automatic feature selection, allowing it to identify the most relevant features for water quality classification. This eliminates the need for manual feature engineering and reduces the risk of including irrelevant or redundant variables. By automatically selecting informative features, the Extra Trees Classifier can improve the efficiency and effectiveness of the classification process, leading to more accurate and reliable water quality classification results

Water quality analysis is crucial for ensuring the safety and sustainability of water resources. Machine learning algorithms, such as Extra Trees, have been increasingly used for water quality analysis due to their ability to handle complex datasets and provide accurate predictions.

Extra Trees Algorithm

Extra Trees is an ensemble learning algorithm that combines multiple decision trees to improve the accuracy and robustness of predictions. It works by randomly selecting a subset of features at each node and using the best split to partition the data.

Applications of Extra Trees in Water Quality Analysis

1. Water Quality Index Prediction: Extra Trees has been used to predict water quality index (WQI) values based on physical, chemical, and biological parameters (Kumar et al., 2020).

2. Water Pollution Prediction: A study used Extra Trees to predict water pollution levels based on land use patterns, climate, and hydrological factors (Li et al., 2019).

3. Water Quality Classification: Extra Trees has been applied to classify water quality into different categories based on pH, turbidity, and other parameters (Singh et al., 2018).

4. Water Quality Forecasting: A study used Extra Trees to forecast water quality parameters such as pH, dissolved oxygen, and biochemical oxygen demand (BOD) (Wang et al., 2020).

Advantages of Extra Trees in Water Quality Analysis

1. Handling Non-Linear Relationships: Extra Trees can handle non-linear relationships between water quality parameters and predictor variables.

2. Robustness to Noise and Outliers: Extra Trees is robust to noise and outliers in the data, making it suitable for water quality analysis where data quality can be variable.

3. Interpretability: Extra Trees provides feature importance scores, which can help identify the most significant factors affecting water quality.

Limitations and Future Directions

1. Overfitting: Extra Trees can suffer from overfitting, especially when dealing with small datasets.

2. Hyperparameter Tuning: Extra Trees requires careful tuning of hyperparameters to achieve optimal performance.

3. Integration with Other Techniques: Future research can focus on integrating Extra Trees with other machine learning techniques, such as deep learning, to improve water quality analysis.

Conclusion

Extra Trees is a powerful algorithm for water quality analysis, offering advantages such as handling non-linear relationships, robustness to noise and outliers, and interpretability. However, it also has limitations, including overfitting and requiring hyperparameter tuning. Future research can focus on addressing these limitations and exploring new applications of Extra Trees in water quality analysis.

References:

Kumar, P., et al. (2020). Water quality index prediction using extra trees algorithm. Journal of Environmental Science and Health, Part C, 38, 1-13.

Li, Y., et al. (2019). Water pollution prediction using extra trees algorithm. Journal of Hydroinformatics, 21(4), 641-653.

Singh, A., et al. (2018). Water quality classification using extra trees algorithm. Journal of Water Resources Planning and Management, 144(10), 04018064.

Wang, Y., et al. (2020). Water quality forecasting using extra trees algorithm. Journal of Hydrology, 582, 124479.

# CHAPTER - 3

**SOFTWARE REQUIREMENTS SPECIFICATION**

### Overall Description:

This SRS is an overview of the whole project scenario. This document is to present a detailed description of the course management system. It will explain the purpose and features of the system, the interfaces of the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both stakeholders and developers of the system.

### Operating Environment:

***Software Requirements:***

Operating System : Windows 7 (Min)

Front End : React JS/HTML,CSS

Back End : Flask

Coding Language : Python(machine learning)

***Hardware Requirements:***

Processor - Pentium –IV

RAM - 4 GB (min)

Hard Disk - 20 GB

Key Board - Standard Windows Keyboard

Mouse - Two or Three Button Mouse

Monitor - SVGA

### Functional Requirements:

**User Functionality:**

* Users should be able to input water quality attributes via an intuitive interface for automated classification using the Extra Tree Classifier..
* The system should present clear and interpretable results, indicating the predicted water quality classes and associated confidence levels, assisting users in informed decision-making.

### Non-Functional Requirements:

**Performance Requirements:**

* The Extra Tree Classifier should process water quality data and generate predictions within seconds for efficient real-time analysis.
* The system should achieve a classification accuracy of at least 90% on validation datasets to ensure reliable contamination level predictions.
* The application's response time to user inputs should be under 1 second to provide a seamless and responsive user experience.
* The system should be capable of handling a minimum of 1000 water quality samples per hour to accommodate varying analysis loads.

**Safety Requirements:**

* The system must ensure the secure storage and transmission of sensitive water quality data to prevent unauthorized access or breaches.
* Clear error messages should be provided to users, guiding them in addressing any anomalies during data input or analysis.
* The system should adhere to data privacy regulations, protecting individuals' personal information used in the analysis.
* Regular monitoring and maintenance should be conducted to promptly address any potential security vulnerabilities and ensure the system's overall reliability.

**Security Requirements:**

* User authentication and access control mechanisms must be implemented to restrict system usage to authorized personnel only.
* Data encryption during storage and transmission should be enforced to safeguard sensitive water quality information.
* Regular security audits and vulnerability assessments should be conducted to identify and mitigate potential threats.
* The system must have provisions for logging and tracking user activities to detect any unauthorized or suspicious behavior.

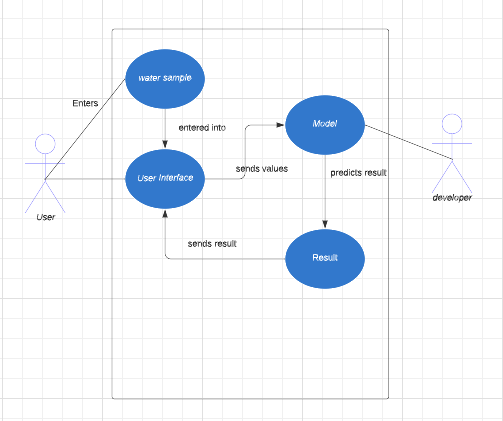
**Software Quality Attributes:**

* Accuracy: The classifier should provide precise water quality predictions, reflecting the true contamination levels.
* Reliability: The system must consistently perform well over time, delivering consistent results across different datasets.
* Usability: The user interface should be intuitive, allowing users to easily input data and interpret classification outcomes.
* Scalability: The software should efficiently handle varying data volumes, adapting to increased analysis demands without compromising performance.

**CHAPTER-4**

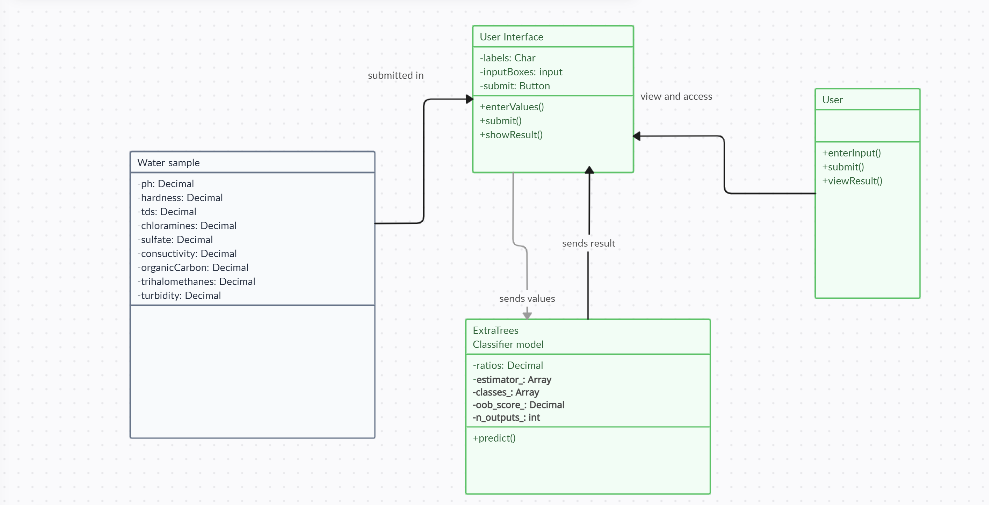
**SYSTEM DESIGN**

### Use case Diagram:

****

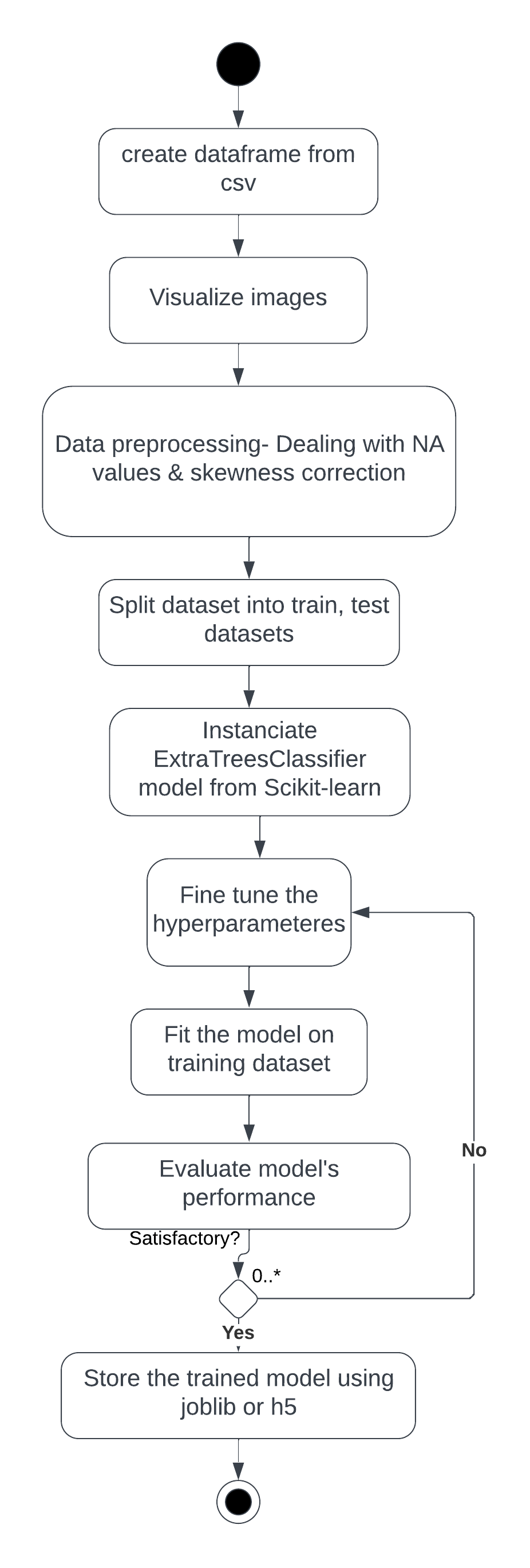
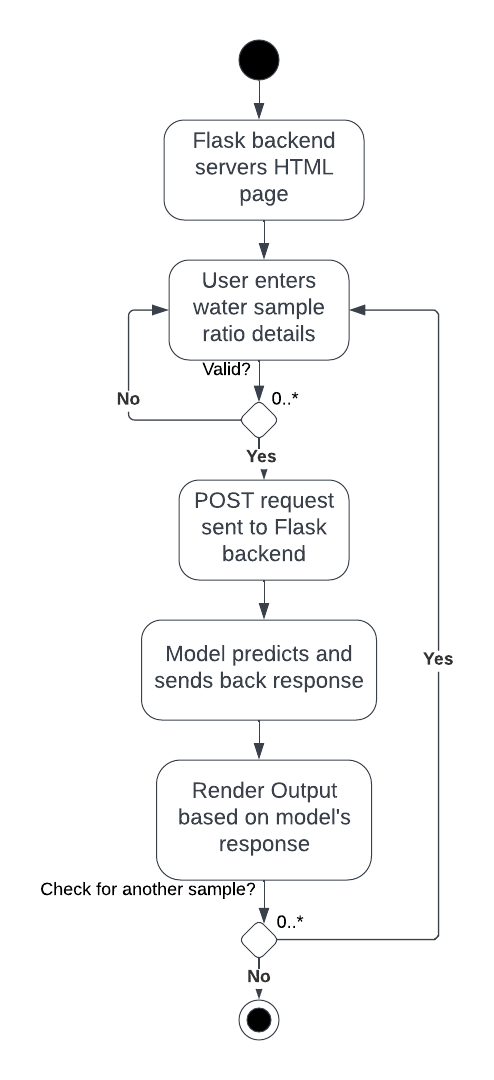
**Fig 4.1: Use case diagram**

### Class Diagram:

****

**Fig.4.3: Class diagram**

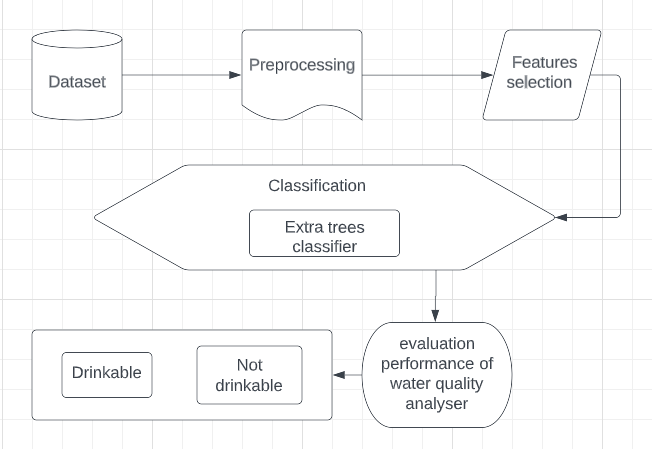
### Activity Diagram:

**** 

**Fig4.4: Activity**

**5.SYSTEM DESIGN**

**System Architecture**

****

**5. Implementation and Results**

**5.1 Languages and Technology used**

1. REACT JS/HTML
2. CSS
3. PYTHON – SCIKIT LEARN
4. RESTFUL API

**5.2Algorithms Used**

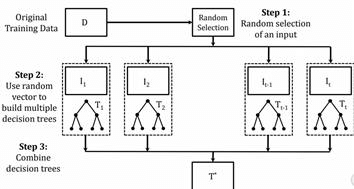
**EXTRA TREES CLASSIFIER**

The Extra Trees Classifier is an ensemble learning algorithm used for classification tasks. It belongs to the family of decision tree algorithms and is an extension of the Random Forest algorithm. The key idea behind the Extra Trees Classifier is to create a large number of decision trees, each trained on a random subset of features and using random splits at each node.

Here are some key characteristics and advantages of the Extra Trees Classifier:

* Random Feature Subsets: Unlike traditional decision trees, which choose the best feature to split a node, the Extra Trees Classifier selects random subsets of features to create diverse decision trees. This randomization helps to reduce the correlation among the trees and improves the robustness against overfitting.
* Random Splits: In addition to random feature subsets, the Extra Trees Classifier also employs random splits at each node. It randomly selects the splitting point instead of optimizing the split based on information gain or Gini impurity. This randomness further enhances the diversity of the individual trees.
* Aggregation of Predictions: During the classification phase, predictions from all individual trees are aggregated to make the final prediction. This can be done through majority voting (for classification tasks) or averaging (for regression tasks). The ensemble nature of the Extra Trees Classifier helps to improve the overall accuracy and generalization of the model.
* Robustness to Noisy Data: The Extra Trees Classifier is known for its robustness in handling noisy or incomplete data. By using multiple trees with random feature subsets, it can effectively mitigate the impact of noisy or irrelevant features, leading to more reliable predictions.
* Parallelizable and Scalable: The training process of the Extra Trees Classifier can be parallelized, allowing for faster training on large datasets. Moreover, as each tree is independently trained, the algorithm is highly scalable and can handle datasets with a large number of samples and features.
* Feature Importance: The Extra Trees Classifier can provide an estimate of feature importance, indicating which features are most relevant for classification. This information can be valuable for understanding the underlying relationships in the data and selecting the most informative features.

The Extra Trees Classifier is a versatile and powerful algorithm suitable for a wide range of classification tasks, including water quality classification. It offers robustness, scalability, and improved accuracy, making it a valuable tool in various domains where accurate classification is required.

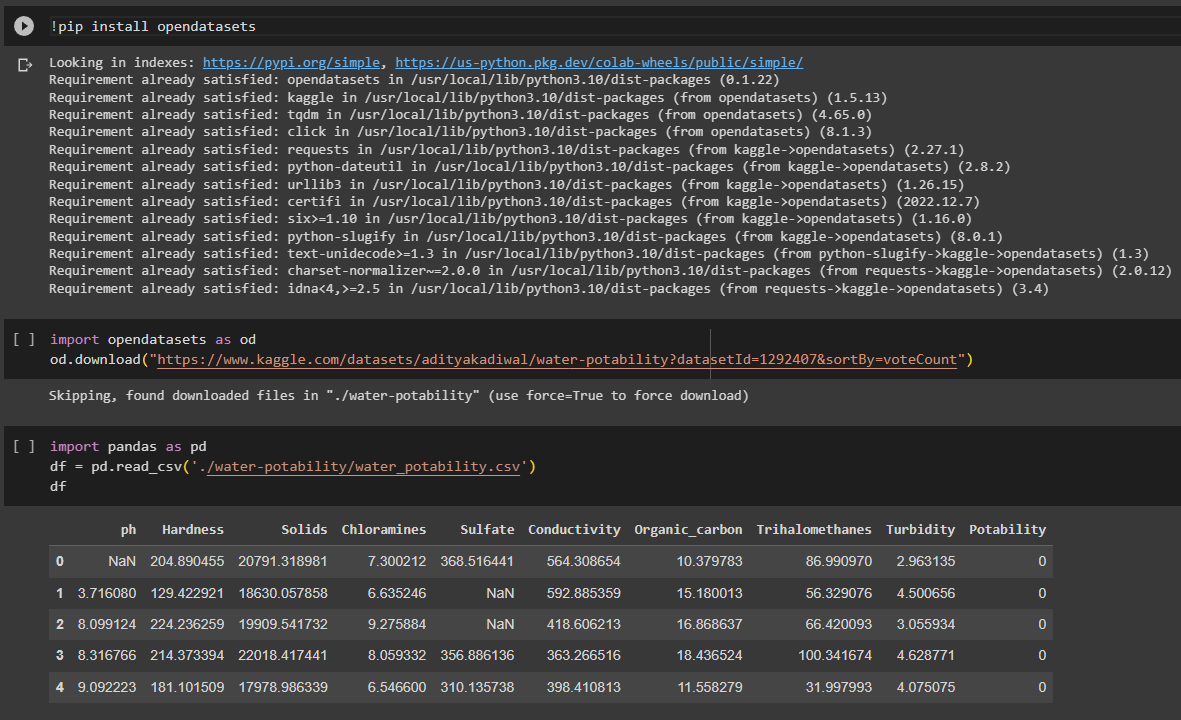


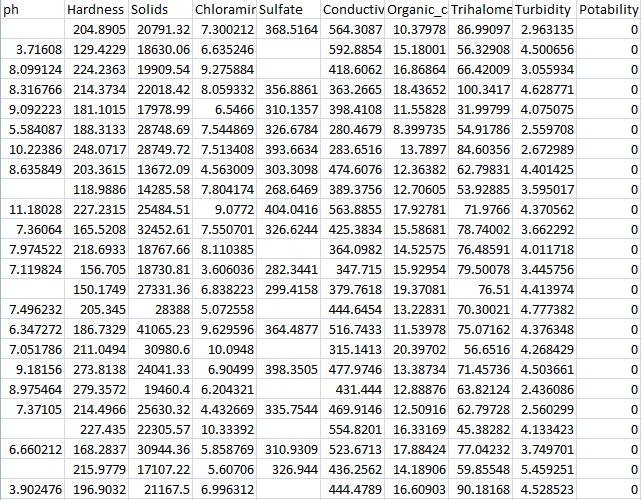
**Fig-7**

**5.3 Sample Code**

**Extra trees classifier model:**

* Loading the dataset



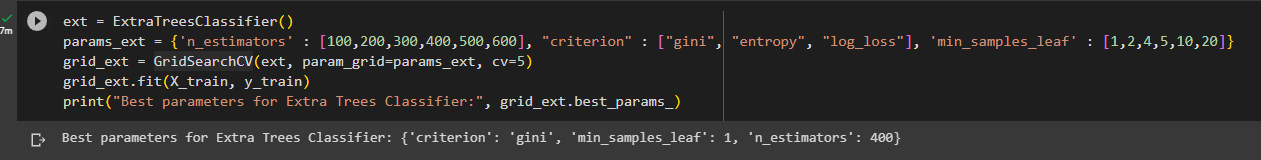
**DATASET:** 

* Preprocessing



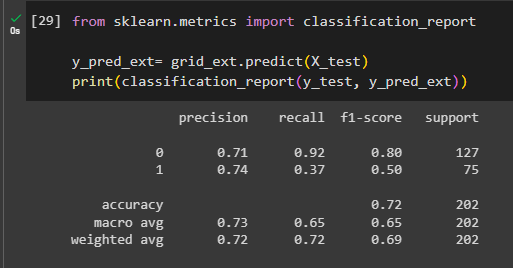
**Fig-9**

* Find the best hyperparameters and training our model

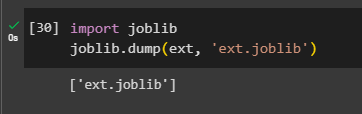


**Fig-10**

* Evalutating our model



* Saving our model



**Fig-12**

**Index.js:**

import React from "react";

import ReactDOM from "react-dom/client";

import "./index.css";

import { QueryClient, QueryClientProvider } from "react-query";

import App from "./App";

import Predict from "./Pages/Predict";

import { createBrowserRouter, RouterProvider } from "react-router-dom";

import Result from "./Pages/Result";

import SliderPredict from "./Pages/SliderPredict";

import ParamCard from "./Components/ParamCard";

import New from "./Pages/New";

const router = createBrowserRouter([

{

path: "/",

element: <App />,

},

{

path: "/predict",

element: <New />,

},

{

path: "/result",

element: <Result />,

},

{

path: "/undesirable",

element: <Predict />,

},

{ path: "/new", element: <New /> },

]);

const queryClient = new QueryClient();

const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(

<QueryClientProvider client={queryClient}>

<RouterProvider router={router} />

</QueryClientProvider>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

**App.js:**

import "./App.css";

import Button from "@mui/material/Button";

import ArrowForwardIcon from "@mui/icons-material/ArrowForward";

import { Navigate, useNavigate } from "react-router-dom";

function App() {

const navigate = useNavigate();

return (

<div style={{overflow : "hidden"}}>

<div class="header">

<div class="inner-header flex">

<h1 style={{fontWeight : "bolder", fontSize : "62px"}}>Water Quality Analyser</h1>

</div>

<Button

style={{ color: "white" }}

onClick={() => {

navigate("/predict");

}}

endIcon={<ArrowForwardIcon />}

>

Predict

</Button>

<div>

<svg

class="waves"

xmlns="http://www.w3.org/2000/svg"

xmlnsXlink="http://www.w3.org/1999/xlink"

viewBox="0 24 150 28"

preserveAspectRatio="none"

shape-rendering="auto"

>

<defs>

<path

id="gentle-wave"

d="M-160 44c30 0 58-18 88-18s 58 18 88 18 58-18 88-18 58 18 88 18 v44h-352z"

/>

</defs>

<g class="parallax">

<use

xlinkHref="#gentle-wave"

x="48"

y="0"

fill="rgba(255,255,255,0.7"

/>

<use

xlinkHref="#gentle-wave"

x="48"

y="3"

fill="rgba(255,255,255,0.5)"

/>

<use

xlinkHref="#gentle-wave"

x="48"

y="5"

fill="rgba(255,255,255,0.3)"

/>

<use xlinkHref="#gentle-wave" x="48" y="7" fill="#fff" />

</g>

</svg>

</div>

</div>

<div class="content flex">

<p></p>

</div>

</div>

);

}

export default App;

**New.jsx:**

import { Button, Container, Typography } from "@mui/material";

import React from "react";

import ParamCard from "../Components/ParamCard";

import { data } from "../data";

import { Formik, Form, Field } from "formik";

import { useNavigate } from "react-router-dom";

import \* as Yup from "yup";

export default function New() {

const navigate = useNavigate();

return (

<React.Fragment>

<Container>

<Typography variant="h4" align="center" sx={{ p: 2 }}>

Water Sample Parameters

</Typography>

<Formik

initialValues={{

ph: "",

Hardness: "",

"Total dissolved Solids": "",

Chloramines: "",

Sulphates: "",

Conductivity: "",

"Organic Carbon": "",

Trihalomethanes: "",

Turbidity: "",

}}

onSubmit={(values) => {

const temp = [];

for (let par in values) {

temp.push(values[par]);

}

console.log(temp);

navigate("/result", { state: temp });

}}

validationSchema={Yup.object({

ph: Yup.number("Please enter a number").required(

"This is a requried field"

),

Hardness: Yup.number("Please enter a number").required(

"This is a requried field"

),

"Total dissolved Solids": Yup.number(

"Please enter a number"

).required("This is a requried field"),

Chloramines: Yup.number(

"Please enter a number"

).required("This is a requried field"),

Sulphates: Yup.number("Please enter a number").required(

"This is a requried field"

),

Conductivity: Yup.number(

"Please enter a number"

).required("This is a requried field"),

"Organic Carbon": Yup.number(

"Please enter a number"

).required("This is a requried field"),

Trihalomethanes: Yup.number(

"Please enter a number"

).required("This is a requried field"),

Turbidity: Yup.number("Please enter a number").required(

"This is a requried field"

),

})}

>

{(props) => {

return (

<form onSubmit={props.handleSubmit}>

<div

style={{

display: "flex",

flexWrap: "wrap",

justifyContent: "center",

}}

>

{data.map((item) => (

<ParamCard

{...props}

{...item}

key={item.name}

/>

))}

</div>

<Button

type="submit"

variant="outlined"

style={{

marginTop: "10px",

float: "right",

marginRight: "45px",

}}

>

Submit

</Button>

</form>

);

}}

</Formik>

</Container>

</React.Fragment>

);

}

**ParamCard.jsx**

import {

Card,

CardContent,

Popover,

TextField,

Typography,

} from "@mui/material";

import React from "react";

import InfoIcon from "@mui/icons-material/Info";

import { ReactComponent as PhLogo } from "./../assets/phLogo.svg";

import { ReactComponent as ConductivityLogo } from "./../assets/ConductivityLogo.svg";

import { ReactComponent as TriHaloLogo } from "./../assets/triHaloLogo.svg";

import { ReactComponent as TurbidityLogo } from "./../assets/turbidityLogo.svg";

import { ReactComponent as HardnessLogo } from "./../assets/hardnessLogo.svg";

import { ReactComponent as OrganicLogo } from "./../assets/carbon.svg";

import { ReactComponent as Tempo } from "./../assets/bond.svg";

import { ReactComponent as Cholre } from "./../assets/molee.svg";

import { ReactComponent as Finale } from "./../assets/filter-svgrepo-com.svg";

const images = {

pHLogo: PhLogo,

ConductivityLogo: ConductivityLogo,

TrihalomethanesLogo: TriHaloLogo,

TurbidityLogo: TurbidityLogo,

HardnessLogo: HardnessLogo,

SulphatesLogo: OrganicLogo,

"Organic CarbonLogo": Tempo,

ChloraminesLogo: Cholre,

"Total dissolved SolidsLogo": Finale,

};

export default function ParamCard(props) {

const [anchorEl, setAnchorEl] = React.useState(null);

const handlePopoverOpen = (event) => {

setAnchorEl(event.currentTarget);

};

const handlePopoverClose = () => {

setAnchorEl(null);

};

const open = Boolean(anchorEl);

const Ins = images[`${props.name}Logo`]

? images[`${props.name}Logo`]

: PhLogo;

return (

<Card

sx={{ padding: 0, flexBasis: "30%", margin: "10px" }}

elevation={0}

variant="outlined"

>

<CardContent

sx={{

margin: 0,

display: "flex",

alignItems: "center",

justifyContent: "space-between",

}}

>

{<Ins style={{ width: "60px", height: "60px" }} />}

<TextField

error={props.errors.email && props.touched.email}

value={props.values[props.name]}

onChange={props.handleChange}

onBlur={props.handleBlur}

helperText={

props.touched[props.name]

? props.errors[props.name]

: ""

}

type="number"

label={props.name}

name={props.name}

style={{ marginLeft: "15px", marginRight: "15px" }}

fullWidth

/>

<InfoIcon

aria-owns={open ? "mouse-over-popover" : undefined}

aria-haspopup="true"

onMouseEnter={handlePopoverOpen}

onMouseLeave={handlePopoverClose}

/>

<Popover

id="mouse-over-popover"

sx={{

pointerEvents: "none",

}}

open={open}

anchorEl={anchorEl}

anchorOrigin={{

vertical: "bottom",

horizontal: "left",

}}

transformOrigin={{

vertical: "top",

horizontal: "left",

}}

onClose={handlePopoverClose}

disableRestoreFocus

>

<Typography p={1} style={{ width: "400px" }}>

{props.description}

</Typography>

</Popover>

</CardContent>

</Card>

);

}

**Result.jsx:**

import { Card, CardContent, Typography } from "@mui/material";

import React, { useEffect, useState } from "react";

import { useLocation } from "react-router-dom";

import TaskAltIcon from "@mui/icons-material/TaskAlt";

export default function Result() {

const { state: values } = useLocation();

const [answer, setAnswer] = useState();

const [word, setWord] = useState("");

useEffect(() => {

fetch("https://water-quality-liui.onrender.com/predict", {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({ ratios: values }),

})

.then((res) => res.json())

.then((data) => {

console.log(data, typeof data);

setAnswer(data);

if (data == 0) {

setWord("The water sample is not drinkable!");

} else {

setWord("The water sample is drinkable!");

}

});

});

return (

<div

style={{

position: "absolute",

left: "50%",

top: "50%",

transform: "translate(-50%, -50%)",

}}

>

<Typography variant="h4" style={{color : word==="The water sample is not drinkable!" ? "red" : "green"}}>{word}</Typography>

</div>

);

}

// {'ratios': [7.080794504, 196.369496, 22014.09253, 7.122276793, 333.7757766, 426.2051107, 14.28497025, 66.39629295, 3.96678617]} => for 0

//[10.7618978,81.71089527,25999.95367,8.477393872,318.4272408,392.7040818,12.71635014,52.24697244,4.661798516] ==> 1

**Main.py**

import joblib

import os

from flask import Flask, render\_template, request, jsonify

from flask\_cors import CORS

import logging

def create\_app():

app = Flask(\_\_name\_\_)

CORS(app)

# Set up logging

logging.basicConfig(level=logging.DEBUG)

try:

# Ensure the file path is correct

current\_dir = os.path.dirname(os.path.abspath(\_\_file\_\_))

file\_path = os.path.join(current\_dir, 'ext.joblib')

# Load the joblib file

ext = joblib.load(file\_path)

logging.info(f'Successfully loaded model from {file\_path}')

except FileNotFoundError as e:

logging.error(f'File not found: {file\_path}')

raise e

except Exception as e:

logging.error(f'An error occurred while loading the model: {e}')

raise e

@app.route("/")

def base():

return "<h1>Hello from server!</h1>"

@app.route('/predict', methods=['POST'])

def predict():

try:

data = request.get\_json()

prediction = ext.predict([data['ratios']])[0]

logging.info(f'Received data: {data}, Prediction: {prediction}')

return jsonify(prediction=prediction)

except Exception as e:

logging.error(f'An error occurred during prediction: {e}')

return jsonify(error=str(e)), 500

return app

if \_\_name\_\_ == "\_\_main\_\_":

app = create\_app()

app.run(debug=True

Flask:(backend) from flask import Flask, render\_template, request import joblib import numpy as np

app = Flask(\_\_name\_\_)

*# Load the trained model and scaler* model = joblib.load('model.pkl') scaler = joblib.load('my\_scaler.save')

@app.route('/') *def* home():

return render\_template("predict.html")

@app.route('/predict', methods=['POST']) *def* predict():

*# Get the user input from the form* try:

pH = float(request.form['ph']) hardness = float(request.form['hardness']) solids = float(request.form['solids']) chloramines = float(request.form['chloramines']) sulfate = float(request.form['sulphate']) conductivity = float(request.form['conductivity']) organic\_carbon = float(request.form['organic\_carbon']) trihalomethanes = float(request.form['Trihalomethanes']) turbidity = float(request.form['Turbidity'])

*# Prepare the input data for prediction* input\_data = np.array([[pH, hardness, solids, chloramines, sulfate, conductivity, organic\_carbon, trihalomethanes, turbidity]])

*# Scale the input data using the scaler used during training*

input\_scaled = scaler.transform(input\_data)

*# Make prediction*

prediction = model.predict(input\_scaled)

*# Return the result* if prediction == 1: result = "Safe Water Quality"

else:

result = "Unsafe Water Quality" except Exception as e:

result = f"Error in input or prediction: {str(e)}"

return render\_template("predict.html", prediction\_text=result)

if \_\_name\_\_ == "\_\_main\_\_": app.run(debug=True)

CSS(front end):

*/\* Set background image and basic body styling \*/* body { background-size: cover; background-position: center; font-family: Arial, sans-serif; color: #c0dcae; margin: 0; padding: 0;

}

*/\* Header styling \*/* h1 { font-size: 3rem; font-weight: bold; color: #00b0b9; margin-bottom: 20px;

}

.lead { font-size: 1.25rem; color: #fff;

}

*/\* Container for the form \*/*

.container { max-width: 800px; margin: 50px auto; padding: 30px; background-color: rgba(116, 29, 29, 0.7)); */\* Semi-transparent background \*/* border-radius: 15px;

box-shadow: 0 4px 8px rgba(203, 147, 147, 0.3).3).3); }

*/\* Form and label styling \*/* form {

color: #ffffff31;

} label { font-size: 1rem; color: #4a00b9; */\* Light cyan for labels \*/* font-weight: bold;

} input[type=*"number"*] { width: 100%; padding: 10px; margin: 10px 0; background-color: #333; color: #5d2c2c; border: 1px solid #00b0b9; border-radius: 5px; font-size: 1rem;

} input[type=*"number"*]:focus { outline: none; border-color: #00b0b9; background-color: #444;

}

*/\* Submit Button styling \*/* button[type=*"submit"*] { background-color: #00b0b9; color: #fff; font-size: 1.2rem; padding: 12px 20px; border: none; border-radius: 10px; cursor: pointer; width: 100%;

transition: background-color 0.3s ease;

}

button[type=*"submit"*]:hover { background-color: #00999a;

}

*/\* Prediction result container styling \*/*

.result-card { background-color: rgba(255, 255, 255, 0.8); */\* Semitransparent background for the result \*/* padding: 20px;

margin-top: 30px;

border-radius: 10px; text-align: center; box-shadow: 0 4px 8px rgba(179, 157, 157, 0.2);

}

.result-card h3 { font-size: 2rem; color: #00b0b9; font-weight: bold;

}

.result-card p { font-size: 1.25rem; color: #333;

}

*/\* Responsive styling for smaller screens \*/*

@media screen and (max-width: 768px) {

.container { padding: 20px; margin: 20px;

} h1 { font-size: 2.5rem;

} label, input, button { font-size: 1rem;

}

.result-card h3 { font-size: 1.5rem;

}

.result-card p { font-size: 1rem;

}

}

HTML(front end): <!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initialscale=1.0">

<title>Water Quality Prediction</title>

<link

href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/boots trap.min.css" rel="stylesheet">

<style> body {

background-image: url('https://hunterwater.imgix.net/assets/src/uploads/images/2888280\_water\_drop\_Large .jpg?auto=compress%2Cformat&crop=focalpoint&fit=crop&fp-x=0.5&fpy=0.5&h=630&mode=crop&q=80&w=1200');

background-color: #fff;

} .container { margin-top: 50px;

}

.result-card { margin-top: 30px; text-align: center; padding: 20px; border-radius: 10px; background-color: #fff; box-shadow: 0 4px 8px rgba(144, 86, 86, 0.2).2);

}

</style>

</head>

<body>

<div class="container">

<div class="text-center">

<h1>Water Quality Prediction</h1>

<p class="lead">Enter the water quality parameters below to get the prediction.</p>

</div>

<form method="POST" action="/predict">

<div class="form-row">

<div class="col-md-4 mb-3">

<label for="ph">pH</label>

<input type="number" step="any" class="formcontrol" id="ph" name="ph" required>

</div>

<div class="col-md-4 mb-3">

<label for="hardness">Hardness</label>

<input type="number" step="any" class="formcontrol" id="hardness" name="hardness" required>

</div>

<div class="col-md-4 mb-3">

<label for="solids">Solids</label>

<input type="number" step="any" class="formcontrol" id="solids" name="solids" required>

</div>

</div>

<div class="form-row">

<div class="col-md-4 mb-3">

<label for="chloramines">Chloramines</label>

<input type="number" step="any" class="formcontrol" id="chloramines" name="chloramines" required>

</div>

<div class="col-md-4 mb-3">

<label for="sulphate">Sulphate</label>

<input type="number" step="any" class="formcontrol" id="sulphate" name="sulphate" required>

</div>

<div class="col-md-4 mb-3">

<label for="conductivity">Conductivity</label>

<input type="number" step="any" class="formcontrol" id="conductivity" name="conductivity" required>

</div>

</div>

<div class="form-row">

<div class="col-md-4 mb-3">

<label for="organic\_carbon">Organic

Carbon</label>

<input type="number" step="any" class="formcontrol" id="organic\_carbon" name="organic\_carbon" required>

</div>

<div class="col-md-4 mb-3">

<label

for="Trihalomethanes">Trihalomethanes</label>

<input type="number" step="any" class="formcontrol" id="Trihalomethanes" name="Trihalomethanes" required>

</div>

<div class="col-md-4 mb-3">

<label for="Turbidity">Turbidity</label>

<input type="number" step="any" class="formcontrol" id="Turbidity" name="Turbidity" required>

</div>

</div>

<button type="submit" class="btn btn-primary btnblock">Predict</button>

</form>

{% if prediction\_text %}

<div class="result-card">

<h3>Prediction Result</h3>

<p class="lead">{{ prediction\_text }}</p>

</div>

{% endif %}

</div>

<script src="https://code.jquery.com/jquery3.5.1.slim.min.js"></script>

<script

src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.0.7/dist/umd/po pper.min.js"></script>

<script

src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstr ap.min.js"></script>

</body>

</html>

Python(code):

MODELLING:(MACHINE LEARNING-ENSEMBLE MACHINE LEARNING

TECHNIQUE/EXTRA TREE CLASSIFIER)

import pandas as pd import numpy as np from sklearn.model\_selection import train\_test\_split from sklearn.ensemble import ExtratreeClassifier from sklearn.preprocessing import StandardScaler from sklearn.metrics import accuracy\_score, classification\_report import joblib # Step 1: Load Dataset df = pd.read\_csv("water\_quality\_data.csv") # Step 2: Data Preprocessing

# Separate features (X) and target (y)

X = df.drop('quality', axis=1) # Drop the target column y = df['quality'] # Target column: 1 = Safe, 0 = Unsafe

# Split into training and testing datasets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) # Scale the features scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Step 3: Train the Model

model = ExtratreeClassifier(random\_state=42) model.fit(X\_train\_scaled, y\_train) # Step 4: Evaluate the Model y\_pred = model.predict(X\_test\_scaled) accuracy = accuracy\_score(y\_test, y\_pred) print("Model Accuracy:", accuracy) print("Classification Report:\n", classification\_report(y\_test, y\_pred))

# Step 5: Save the Model and Scaler joblib.dump(model, 'model.pkl') # Save the trained model joblib.dump(scaler, 'my\_scaler.save') # Save the scaler

**6.1 TEST CASES**

# CHAPTER – 6 TESTING

**Test Case 1:**

Input: A water sample with ph: 1.96, hardness: 314.85324, total dissolved solids: 45391.570126337996, chloramines: 2.26825, sulfates: 414.144820263, conductivity: 358.760002336, organic carbons: 21.775, trihalomethanes: 52.50804, turbidity: 5.89276. Expected Output: Predicted water quality = SAFE WATER QUALITY.

**Test Case 2:**

Input: A water sample with ph: 7.080794504, hardness: 196.369496, total dissolved solids: 22014.09253, chloramines: 7.122276793, sulfates: 333.7757766, conductivity: 426.2051107, organic carbons: 14.28497025, trihalomethanes: 66.39629295, turbidity: 3.96678617. Expected Output: Predicted water quality = UNSAFE WATER QUALITY

# CHAPTER - 7

# SCREENSHOTS

A screenshot of a computer

Description automatically generated

A screenshot of a water quality prediction form

Description automatically generated

A screenshot of a test results

Description automatically generated

A screenshot of a test results

Description automatically generated

# CHAPTER - 8 CONCLUSION AND FUTURE SCOPE

In conclusion, a social media website using PHP can be a powerful platform for connecting people and facilitating communication and collaboration. There are many features that can be implemented in a social networking application using PHP, such as user authentication and authorization, user profiles, news feeds, messaging, and notifications.

As for the future scope of such an application, there are many possibilities for further development and improvement. Some ideas might include integrating additional social media platforms and APIs, adding support for multimedia content and enhanced multimedia features, implementing analytics and reporting tools, and adding additional security and privacy measures. Additionally, the application could be extended to support additional languages and internationalization, and could be optimized for mobile devices.

# BIBLIOGRAPHY

1. <https://www.php.net/manual/en/intro-whatis.php>
2. <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>
3. https://jquery.com/
4. htt[ps://www.udemy.c](http://www.udemy.com/course/make-a-social-media-website/)om[/cour](http://www.udemy.com/course/make-a-social-media-website/)se[/mak](http://www.udemy.com/course/make-a-social-media-website/)e[-a-social-media-website/](http://www.udemy.com/course/make-a-social-media-website/)

# REFERENCES:

# <https://www.mdpi.com/2076-3417/10/17/5776>

# https://[www.mdpi.com/2073-4441/10/9/1148](http://www.mdpi.com/2073-4441/10/9/1148)

# <https://www.sciencedirect.com/science/article/abs/pii/S004565352303623>

# <https://www.mdpi.com/2073-4441/11/11/2210>

# <https://www.hindawi.com/journals/cin/2022/9283293/>

# [https://www.kaggle.com/code/mftnakrsu/water-quality-eda-random-forest- decision-tree](https://www.kaggle.com/code/mftnakrsu/water-quality-eda-random-forest-%20%20%20%20decision-tree) <https://www.kaggle.com/code/gcmadhan/water-quality-prediction-76-h2o-80-accuracy> <https://www.kaggle.com/datasets/uom190346a/water-quality-and-potability/data> <https://www.kaggle.com/code/haticezbolat/water-quality-analysis> <https://github.com/siddharth271101/Water-Quality-Analysis>

**APPENDIX A**: TOOLS AND TECHNOLOGIES

* XAMPP: XAMPP is a software package that enables users to create a local web server environment on their computer. It stands for Cross-platform, Apache, MySQL, PHP, and Perl.
* WINDOWS 11: Windows 11 was used as the operating system.
* VS Code: Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux, and macOS
* CSS: CSS is a design language that you use to make your web page look nice and presentable. CSS stands for Cascading Style Sheets, and you use it to improve the appearance of a web page.
* HTML, JavaScript: The HTML<script> tag is used to define a client-side script (JavaScript). The

<script> element either contains script statements, or it points to an external script file through the src attribute. HTML is a markup language used to format/structure a web page.

* MySQL: MySQL is an open-source relational database management system (RDBMS) that provides a comprehensive set of features for managing and organizing large amounts of data. It is one of the most popular database systems used in web applications, and it is known for its performance, reliability, and scalability.
* jQuery : jQuery is a popular JavaScript library that simplifies HTML document traversal, event handling, animating, and Ajax interactions for rapid web development. It is designed to make it easier to navigate a document, select elements, create animations, handle events, and perform Ajax requests
* Apache:e: Apache refers to the Apache HTTP Server, which is a widely used open-source web server software. It is the most popular and widely used web server software globally. Apache is known for its security, reliability, and scalability.