Dissertation on

WATER POTABILITY PREDICTION USING MACHINE LEARNING ALGORITHMS

Thesis submitted toward partial fulfillment of the requirements for the degree of

Master of Technology

in

IT (Courseware Engineering)

Submitted by

SK BASIR UDDIN

EXAMINATION ROLL NO.: M4CWE24013

UNIVERSITY REGISTRATION NO.: 143922 of 2018-2019

Under the supervision of

Mr. Joydeep Mukherjee

School of Education Technology
Jadavpur University

Course affiliated to

Faculty of Engineering and Technology

Jadavpur University

Kolkata – 700032

INDIA

2024

M. TECH. IT (COURSEWARE ENGINEERING) Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled "**Water Potability Prediction Using Machine Learning Algorithms**" is a bonafide work carried out by **SK BASIR UDDIN** under our supervision and guidance for partial fulfillment of the requirement for Post Graduate Degree of Master of Technology IT (Courseware Engineering) in the School of Education Technology during the academic session 2023 – 2024.

SUPERVISOR

School of Education Technology
Jadavpur University
Kolkata-700032

DIRECTOR

School of Education Technology
Jadavpur University
Kolkata-700032

DEAN

FISLM
Jadavpur University
Kolkata-700032

M. TECH. IT (COURSEWARE ENGINEERING) Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF APPROVAL **

This foregoing thesis is hereby approved as a creditable study of an Engineering Subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it has been submitted.

Committee of final examination
for evaluation of the Thesis.

_____

_____

_____

_____

** Only in case the thesis is approved.

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains a Literature survey and original research work by the undersigned candidate, as part of my Master of Technology in **IT (Courseware Engineering) in the Department of School of Education Technology**.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that I have thoroughly cited and referenced all material and findings that are not original to this research, as provided by these rules and conduct.

Name: **SK BASIR UDDIN**

REGISTRATION NO.: **143922 of 2018-2019**
Examination Roll No.: **M4CWE24013**
Thesis Title: **Water Potability Prediction Using Machine Learning Algorithms.**

Date: **Signature of Candidate**

ACKNOWLEDGEMENT

I would like to take this opportunity to express my heartiest gratitude to my respected thesis advisor, **Mr. Joydeep Mukherjee**, Asst. Professor, School of Educational Technology, Jadavpur University for giving me an opportunity to work under his guidance as well as for establishing chances for me to engage in an exciting area of research. He has always been a steady source of encouragement, and his belief in my ability has always inspired me to work more. Without his guidance, encouragement, cooperation and support, I would not have been able to complete this thesis.

I would like to express my sincere gratitude to our respected Director, **Prof. (Dr.) Matangini Chattopadhyay**, Professor, School of Educational Technology, Jadavpur University for her unwavering support throughout the journey of my thesis.

I am grateful to **Dr. Saswati Mukherjee**, Asst. Professor, School of Educational Technology, Jadavpur University for her constant support throughout the tenure of my master's degree program at the School of Educational Technology, Jadavpur University.

I would like to convey my gratitude and love to all of my batch mates, without whom life would be a very boring place. They were always there to lift my spirits.

I would also like to thank all my seniors, nonteaching staff, and juniors for their thoughtful cooperation.

Last, but not least, I want to express my heartfelt gratitude and my love for my family who have been the constant support and strength for me to go on with my higher education.

Date: **Sk Basir Uddin**

CONTENTS

Title **Page**

LIST OF FIGURES

CHAPTER 3:

CHAPTER 4:

CHAPTER 5:

LIST OF TABLES

EXECUTIVE SUMMARY

Safe drinking water is vital to human health. Laboratory testing is the foundation of traditional methods for determining the potability of water, but this process can be time and resource consuming. A viable substitute for estimating water quality based on different physicochemical factors is machine learning. The efficacy of machine learning techniques in improving the prediction of water potability is examined in this thesis.

To determine whether water samples were potable or not, A drinking water dataset has been used that included characteristics including pH, total dissolved solids (TDS) and mineral content (sodium, magnesium, calcium, etc.). Pretreatment methods and exploratory data analysis (EDA) were used to get the data ready for modeling. The Decision Tree Classifier, Random Forest Classifier, AdaBoost Classifier and Gradient Boost Classifier are the four machine learning techniques that were assessed. Each model's performance was maximized by hyperparameter optimization.

The outcomes of the prediction of water potability were quite accurate. In order to evaluate the machine learning algorithms used, this thesis compares and contrasts them, emphasizing their advantages and disadvantages for this particular use. The results show how machine learning may be used to quickly and effectively analyses the potability of water, improving public health outcomes.

CHAPTER 1

1. INTRODUCTION

1.1. Overview

One of the most important aspects of sustainable development and a fundamental human right is having access to clean and safe drinking water. However, the need for effective and trustworthy techniques to evaluate water quality has increased due to the worldwide water problem, which has been made worse by elements including urbanization, population expansion and climate change. Precise estimation of water potability is necessary for both the efficient management of water resources and the detection and mitigation of health hazards associated with water.

Conventional techniques for evaluating the quality of water frequently depend on laboratory-based investigations, which may be expensive, time-consuming and labour-intensive. These techniques usually entail gathering water samples, bringing them to a lab and running a number of tests on them to ascertain the water's physicochemical and microbiological characteristics. Even though these techniques yield precise findings, they are not appropriate for quick decision-making or real-time monitoring in the water management sector.

The prediction and optimization of water quality have gained new opportunities in recent years due to the development of sophisticated data analytics and machine learning approaches. The utilization of machine learning algorithms may facilitate the analysis of intricate data on water quality, detect trends and generate precise forecasts, thus augmenting the efficacy and efficiency of water management tactics. These methods may be used for a number of water quality assessment tasks, including identifying contaminants in the water, forecasting potability and simplifying water treatment procedures.

This thesis, "Water Potability Prediction Using Machine Learning Algorithms," aims to explore the application of various machine learning models in predicting water potability based on a comprehensive dataset of physicochemical parameters. The study investigates the performance of four machine learning algorithms: Decision Tree Classifier, Random Forest Classifier, Ada Boost Classifier and Gradient Boost Classifier. Through a rigorous process of data preprocessing, model training, and hyperparameter tuning, the research seeks to identify the most effective approach for accurately predicting water potability.

The dataset used in this study includes features such as pH, Sodium, Magnesium, Calcium, Chloride, Potassium, Carbonate, Sulphate, TDS, EC and TH, which are known to be

influential in determining water quality. These parameters are commonly used in water quality assessment and are indicative of various aspects of water chemistry, such as acidity, salinity and mineral content.

1.2. The significance of accurately predicting water potability for ensuring safety

For communities all over the world to be safe and healthy, accurate water potability prediction is essential. Water contamination can cause a number of health problems, including skin infections, gastrointestinal disorders and potentially fatal conditions like cholera and typhoid fever. Children and vulnerable groups, including the elderly and those with compromised immune systems, are especially vulnerable to these health risks.

Accurately predicting the potability of water can aid in locating hazardous water sources and setting priorities for actions to enhance water quality. Early detection of possible contamination allows for the implementation of preventive measures aimed at reducing the transmission of waterborne illnesses and safeguarding public health. Furthermore, precise forecasts can direct the processes involved in water treatment, guaranteeing that the water is adequately sterilized and satisfies safety regulations.

Accurately forecasting water potability is essential for sustainable water resource management, aside from the health implications. Policymakers and water authorities are better able to prioritize infrastructure development projects and allocate resources by identifying areas where access to safe drinking water is limited. The planning and execution of water conservation strategies can benefit from this information as well, ensuring that the limited water resources are used fairly and effectively.

Predicting water potability accurately can also help achieve the Sustainable Development Goals (SDGs) of the United Nations, especially Goal 6: "Ensure availability and sustainable management of water and sanitation for all." Communities can become more resilient to water-related problems and advance general development and well-being by expanding access to clean drinking water and encouraging sustainable water management techniques.

1.3. The limitations of traditional methods and emphasize the potential of machine learning in improving predictions

Traditional techniques for determining the potability of water frequently rely on laboratory-based analyses, which can be expensive, time-consuming and labour-intensive. These techniques usually entail gathering water samples, bringing them to a lab, and running a number of tests on them to ascertain the water's physicochemical and microbiological characteristics. Even though these techniques yield accurate results, they are not appropriate for quick decision-making or real-time monitoring in the water management sector.

Furthermore, traditional approaches frequently call for trained workers and specialized tools, which might not be easily accessible everywhere, especially in developing nations or isolated locations. This restriction may cause assessments of the quality of the water to be delayed and may prevent prompt action to resolve problems related to water.

Another drawback of traditional methods is their inability to handle large volumes of data and identify complex patterns in water quality data. As the amount of available water quality data continues to grow due to advancements in sensor technologies and data collection methods, traditional methods may struggle to process and analyse this information efficiently.

On the other hand, machine learning approaches present a viable way around the drawbacks of conventional approaches. In a timely and economical manner, machine learning algorithms can analyse intricate data on water quality, spot trends and generate precise forecasts. To provide real-time insights into water quality, these algorithms can process massive amounts of data from multiple sources, such as sensor networks, satellite imagery, and historical records.

Additionally, machine learning models allow for continuous improvement in water potability predictions by adapting to changing environmental conditions and incorporating new data as it becomes available. This flexibility is especially crucial in the face of new threats like climate change and the introduction of new pollutants, which have the potential to gradually change the dynamics of water quality.

Water authorities and policymakers can make better decisions about public health interventions, water treatment procedures, and resource management by utilizing machine learning. Predictions based on machine learning can be used to more effectively identify

possible sources of contamination, optimize water treatment procedures and prioritize regions for monitoring water quality.

1.4. Problem Statement

Water scarcity is a serious problem that affects millions of people around the world. Making sure that drinking water is safe is crucial and being able to predict whether water is drinkable can help with this. Traditional methods for testing water quality take a lot of time and need experts to carry them out. Machine learning offers a faster and more accessible solution. This thesis will focus on building and testing machine learning models to predict if water is safe to drink. By looking at different methods, the research aims to find the best ways to classify water samples. The findings from this study could improve water quality monitoring and help in making better decisions to tackle the global water crisis.

1.5. Motivation and Objectives

The goal of this thesis is to advance the development of effective and trustworthy machine-learning techniques for predicting the potability of water. The study aims to determine the most effective method for accurately predicting water potability by comparing the performance of four machine learning algorithms (Decision Tree Classifier, Random Forest Classifier, Ada Boost Classifier and Gradient Boost Classifier) on an extensive dataset of physicochemical parameters.

The primary objectives of this thesis are:

- To preprocess and clean the water quality dataset, ensuring data quality and consistency for model training.
- To train and evaluate the performance of Decision Tree Classifier, Random Forest Classifier, Ada Boost Classifier and Gradient Boost Classifier on the water quality dataset.
- To optimize the performance of the machine learning models through hyperparameter tuning and feature engineering.
- To compare the performance of the four machine learning algorithms and identify the most effective approach for predicting water potability.

By fulfilling these goals, the thesis hopes to address the global water crisis, advance sustainable development and support continued efforts to guarantee that everyone has access to clean, drinkable water.

1.6. Scope of the Work

The scope of this thesis is limited to predicting water potability using machine learning techniques based on a dataset of physicochemical parameters. The study focuses on four specific machine learning algorithms: Decision Tree Classifier, Random Forest Classifier, Ada Boost Classifier and Gradient Boost Classifier.

The dataset [6] used in this study includes features such as pH, Sodium, Magnesium, Calcium, Chloride, Potassium, Carbonate, Sulphate, TDS, EC and TH, which are known to be influential in determining water quality. The dataset is assumed to be representative of the water quality conditions in the study area and is used to train and evaluate the machine learning models.

Primary data collection and analysis related to water quality are not included in the thesis. Rather, it develops and evaluates the model using the dataset that is provided. Furthermore, the study does not take into account the effects of microbiological parameters or other variables, such as environmental circumstances or human activity, that may have an impact on the potability of water.

The purpose of this thesis is to offer policymakers and water authorities insights and suggestions regarding the use of machine learning in the assessment and management of water quality. However, this study is not intended to address the specific deployment and implementation of the developed models in real-world scenarios.

1.7. Organization of Thesis

This thesis is organized into a total of 5 chapters, as follows:

Chapter 1: Introduction

- General.
- The significance of accurately predicting water potability to ensure safety.
- The limitations of traditional methods and emphasize the potential of machine learning in improving predictions.
- Problem Statement.
- Motivation and Objectives.
- Scope of the work.

Chapter 2: Literature Survey

- Comprehensive literature review of Water potability predictions using machine learning algorithms.

Chapter 3: Proposed Approach

- Overview of the dataset with statistical calculation and target distribution.
- Experimental environment in which the work has been done. This gives a brief concept of machine learning, python (the language which has been used), and its different libraries.
- Discuss about Machine learning algorithms that have been used in this work.
- Briefly discuss about the selection of performance metrics.
- Explanation of feature selection.
- Brief discussion on data correlation method.
- Importance of features.
- Description of data preprocessing steps applied to the datasets.
- Hyperparameter tunning and its different types and the application of hyperparameter tunning on machine learning algorithms used in this thesis.
- Explanation of K-fold validation.
- Specification of performance metrics used for evaluation.

Chapter 4: Experimental Results

- Discussion about the results with the performance metrics table before applying the hyperparameter tuning of the models – Decision Tree, Random Forest, Ada Boost, Gradient Boost.
- Discussion about the results with the ROC AUC score curve and confusion matrix diagram after applying the hyperparameter tuning of the models – Decision Tree, Random Forest, Ada Boost, Gradient Boost.
- Explanation of Evaluation results with different hyperparameter tuning processes with table.

Chapter 5: Analysis and Discussion

- Explanation of all the performance of the four algorithms used in this thesis.
- Comparative analysis of Performance Metrics with the help of a line chart.
- Comparison of outcomes among different models.

Chapter 6: Conclusion and Future Scope

- Conclusions.
- Discussion about the future scope of the thesis.

CHAPTER 2

2. LITERATURE SURVEY

S. Patel, K. Shah, S. Vaghela, M. Aglodiya and R. Bhattad [1] explored the use of machine learning algorithms for forecasting water potability in order to meet the urgent necessity of guaranteeing safe drinking water. In order to identify the most accurate predictive model, researchers compared the performance of many algorithms they used to analyze water quality characteristics. Their study, which shows increased forecast accuracy and dependability above conventional methods, emphasizes the need to apply advanced data analytics in environmental monitoring. Through the integration of extensive datasets and machine learning, this research offers significant insights into the efficient management of water quality and the preservation of public health.

N. Nasir et al. [2] investigated the classification of water quality using machine learning algorithms to improve the precision and effectiveness of water quality assessments. In order to classify water samples according to different quality parameters, the authors assessed a number of machine learning models and determined which algorithms worked best for the job. Their results highlight how machine learning has the potential to completely transform environmental monitoring by offering automated, dependable, and quick evaluations of water quality. This study makes a substantial contribution to the field by illustrating how cutting-edge computational techniques can be used in practice to guarantee clean and safe water sources.

T. Deng, K. W. Chau and H. F. Duan [3] discussed the use of machine learning to predict marine water quality for coastal hydro-environment management. Water quality parameters were predicted using models, which improved prediction timeliness and accuracy. The study, which demonstrates notable advancements over conventional techniques, emphasizes the significance of sophisticated computational approaches in the management of coastal water quality. By utilizing extensive datasets and machine learning, this study offers insightful information about practical methods for monitoring and protecting marine environments.

S. Singha, S. Pasupuleti, S. S. Singha, R. Singh and S. Kumar [4] explored the use of machine learning to predict groundwater quality and emphasize how this technique could enhance the management of water resources. The authors assessed models using physicochemical parameters and showed that machine learning greatly improves prediction efficiency and accuracy compared to traditional techniques. By incorporating strong datasets and cutting-edge methodologies, the research supports the preservation and sustainable management of groundwater resources.

A. El Bilali, A. Taleb and Y. Brouziyne [5] investigated the application of machine learning algorithms to forecast groundwater quality for irrigation in an effort to maximize the use of water resources in agriculture. The authors used a variety of machine learning models to forecast groundwater quality, focusing on elements essential for irrigation. Their results demonstrate how well these models work at producing precise and timely forecasts, which can be very helpful when making decisions about sustainable farming practices. This study emphasizes how machine learning has the potential to revolutionize irrigation management by improving the accuracy and consistency of groundwater quality assessments.

O. O. Ajayi, A. B. Bagula, H. C. Maluleke, Z. Gaffoor, N. Jovanovic and K. C. Pietersen [6] presented a dataset for machine learning-based drinking and irrigation water quality assessment. Through the application and comparison of multiple models, the authors assembled an extensive dataset of water quality parameters. Their study highlighted the importance of solid data and cutting-edge approaches in the management of water resources and demonstrates notable gains in accuracy and efficiency over conventional approaches.

Kathleen Joslyn and John Lipor [7] presented a supervised learning method for fault detection in water systems and water quality parameter prediction. To increase prediction accuracy and timely fault detection, they analyzed water quality data using a variety of supervised machine learning models. Their research demonstrates how well these models work to spot potential problems and ensure water safety. By incorporating sophisticated computational techniques and resilient datasets, this study offers significant perspectives for augmenting the dependability and effectiveness of water quality monitoring systems.

K. S. Prathibha, R. K. N. Kumar, R. S. Joseph and S. Subramani [8] explored the use of deep learning to predict water quality parameters and compute the Water Quality Index (WQI) for Ulsoor Lake in Bangalore. The study offered insights into efficient water quality management by demonstrating the superior accuracy of deep learning models over conventional techniques.

S. Babu, B. B. Nagaleela, C. G. Karthik and L. N. Yepuri [9] explored the use of neural networks for water quality prediction, aiming to enhance the accuracy and efficiency of water quality assessments. The authors implemented various neural network models to predict key water quality parameters. Their research highlights the potential of neural networks to outperform traditional prediction methods, providing more reliable and timely water quality assessments. By utilizing advanced neural network architectures and robust datasets, this study contributes significantly to the field of water resource management and environmental monitoring.

S. Kolli, M. Ranjani, P. Kavitha, D. A. P. Daniel and A. Chandramauli [10] investigated predicting water quality by integrating IoT technology with machine learning. They used IoT devices for real-time data collection and applied machine learning models to enhance monitoring accuracy and efficiency. This approach contributed significantly to smart water management and environmental sustainability.

Nikhil M. Ragi, Ravishankar Holla and Manju G. [11] explored the application of machine learning techniques to predict water quality parameters in an effort to improve the precision and consistency of water quality evaluations. The authors from R V College of Engineering used a variety of machine learning models to evaluate data on water quality and forecast important indicators. Their research demonstrated how machine learning algorithms can outperform conventional techniques, providing more accurate and timely predictions. This study makes a substantial contribution to the fields of environmental monitoring and water resource management by employing sophisticated analytical techniques and robust datasets.

S. K. Gandhimathi, G. K. Muppagowni, K. Sandhya, R. R. Arava, K. Vindhya Anuragh and G. Kotapati [12] looked into the creation of an effective machine learning algorithm prediction mechanism for water quality analysis. The study aimed to use machine learning techniques to overcome the shortcomings of conventional methods for assessing water quality. The authors investigated various machine learning algorithms, such as Support Vector Machines (SVM), Random Forest (RF) and Artificial Neural Networks (ANN), to precisely predict water quality parameters. The intended mechanism was designed to improve the efficacy and dependability of water quality analysis, ultimately aiding in the efficient management and preservation of water resources.

E. Kuruvilla and S. Kundapura [13] focused on the prediction of water quality using machine learning techniques. Their paper explores various machine learning algorithms, including Neural Networks (NN), Random Forest (RF), Support Vector Machine (SVM), BTM and MLR, to classify a water quality dataset from different parts of India. The study highlights the importance of variables affecting water quality, such as pH, nitrate, total coliform (TC), dissolved oxygen (DO), and electrical conductivity (EC). The high classification accuracy in water quality achieved in the study is attributed to the use of feature correlation, data preprocessing techniques, and machine learning categorization. This underscores the significance of machine learning in enhancing water quality assessment and management.

S. Suma, R. Moon, M. Umer, K. S. Raju, N. Bhaskar and R. Okali [14] discussed machine learning's role in accurately predicting and classifying water quality parameters such as dissolved oxygen, pH, and nitrates. By utilizing models like Neural Networks, Random Forest, and SVM, the research enhances water quality assessment, which aids in improved management and conservation.

R. Akshay, G. Tarun, P. U. Kiran, K. D. Devi and M. Vidhyalakshmi [15] conducted a thorough investigation into how machine learning algorithms, including Random Forest, Support Vector Machine, and Neural Networks, affect the precision of water quality predictions. The study likely examines the impact of key water quality parameters such as dissolved oxygen, pH, conductivity, biological oxygen demand, nitrate, and coliforms on the prediction process. By utilizing machine learning methodologies, the research aims to enhance the efficacy and reliability of water quality evaluation, thereby supporting improved water management strategies and environmental preservation efforts.

P. Rawat, M. Bajaj, V. Sharma and S. Vats [16] thoroughly examined the usefulness of machine learning algorithms in forecasting water quality. The study investigated how different machine learning models such as Support Vector Machine, Random Forest, and Neural Networks affect the precision of water quality predictions. Key water quality parameters like dissolved oxygen, pH, conductivity, biological oxygen demand, nitrate, and coliforms are analyzed to determine their importance in the prediction process. The study aims to enhance the accuracy and efficiency of water quality assessment through machine learning techniques, ultimately leading to improved water management plans and environmental preservation initiatives. The conclusions offer valuable insights into how machine learning can address water quality issues and advance sustainable water resource management.

S. A. Ansari, C. Sharma and T. Agarwal [17] introduced a novel mean and prediction imputation-based approach for predicting water potability using machine learning techniques. Their method combined mean and prediction imputation to enhance the accuracy of water potability prediction models. By leveraging advanced machine learning techniques, this research contributes to improving the reliability of water quality assessments, which is crucial for ensuring safe drinking water and environmental sustainability.

F. Akhter, H. R. Siddiquei, M. E. E. Alahi, K. P. Jayasundera and S. C. Mukhopadhyay [18] presented a portable water quality monitoring system with IoT capabilities and a multifunctional sensor utilizing MWCNT/PDMS technology. Focusing on agricultural applications, the study aims to provide farmers with real-time monitoring of crucial water quality parameters for their operations. The integration of multifunctional sensors enhances the accuracy and reliability of the system, offering valuable insights into water quality management for improved crop yields and sustainable agriculture.

R. Alnaqeb, F. Alrashdi, K. Alketbi and H. Ismail [19] introduced a machine learning-based approach for predicting water potability. The study explored advanced computational methods to provide accurate predictions regarding water quality. The research contributede to enhancing water quality assessment methodologies, which is crucial for ensuring access to safe drinking water and maintaining public health standards.

B. Sakaa et al. [20] focused on modeling the water quality index (WQI) using random forest and an improved Sequential Minimal Optimization (SMO) algorithm for Support Vector Machine (SVM). The research aims to enhance the accuracy of WQI prediction in the Saf-Saf river basin, contributing to effective river basin management and environmental conservation efforts.

Md. M. Hassan et al. [21] explored the application of various machine learning algorithms, including Random Forest, Neural Network, Multinomial Logistic Regression, Support Vector Machine and Bagged Tree Model, to predict the water quality index (WQI). The study uses a comprehensive dataset of water quality parameters, such as dissolved oxygen, total coliform, biological oxygen demand, nitrate, pH, and electrical conductivity, collected from various locations in India. This research demonstrated the potential of machine learning techniques in improving water quality assessment and prediction, contributing to sustainable water resource management and public health.

A. Juna et al. [22] introduce a novel method that combines the strong learning capabilities of the Multilayer Perceptron with the flexibility of the KNN imputer to handle missing data. The study aims to provide more precise estimates of water quality parameters using these methods. This approach could enhanced decision-making in water resource management and ensure the sustainability and safety of water supplies.

K. P. Rasheed Abdul Haq and V. P. Harigovindan [23] investigated the use of hybrid deep learning models for predicting water quality in smart aquaculture systems. Their study aims to improve prediction efficiency and accuracy by exploring deep learning approaches, aquaculture management, and water quality prediction methods. The research provides insights into enhancing water quality prediction for smart aquaculture through advanced deep learning techniques.

A. O. Alnahit, A. K. Mishra and A. A. Khan [24] investigated stream water quality prediction using boosted regression tree and random forest models. This study highlighted the application of these machine learning models to forecast water quality characteristics in a semi-arid environment, aiming to address the high costs associated with traditional irrigation water quality assessments in developing countries. The use of these models may improve the accuracy and reliability of water quality management for agriculture.

J. R. Vilupuru, D. C. Amuluru and K. Ghousiya Begum [25] proposed a method for forecasting the water quality index (WQI) using advanced artificial intelligence algorithms, including NARNET and LSTM models. The study also employs SVM, KNN, and Naive Bayes techniques for classifying WQI data. Statistical evaluation is used to assess the performance of these AI models in predicting water quality, demonstrating their effectiveness in handling complex data for accurate water quality predictions.

S. Wang, H. Peng and S. Liang [26] suggested a method for predicting estuarine water quality using interpretable machine learning approaches. The study focuseed on integrating IGRA and LSTM algorithms to account for temporal sequences and multivariate correlations in water quality data. This approach aims to enhance the prediction accuracy of water quality and contribute to the protection of aquatic ecosystems.

CHAPTER 3

3. PROPOSED APPROACH

This chapter dives into the details of the water potability prediction algorithm. First, the chapter explores the dataset used to train the model. Then, it examines some key concepts of machine learning, providing the foundation for the chosen algorithm. Finally, the chapter walks through the detailed procedure used to implement the algorithm, offering a step-by-step breakdown of its inner workings. Fig 3.1 shows the flow diagram illustrating the whole process.

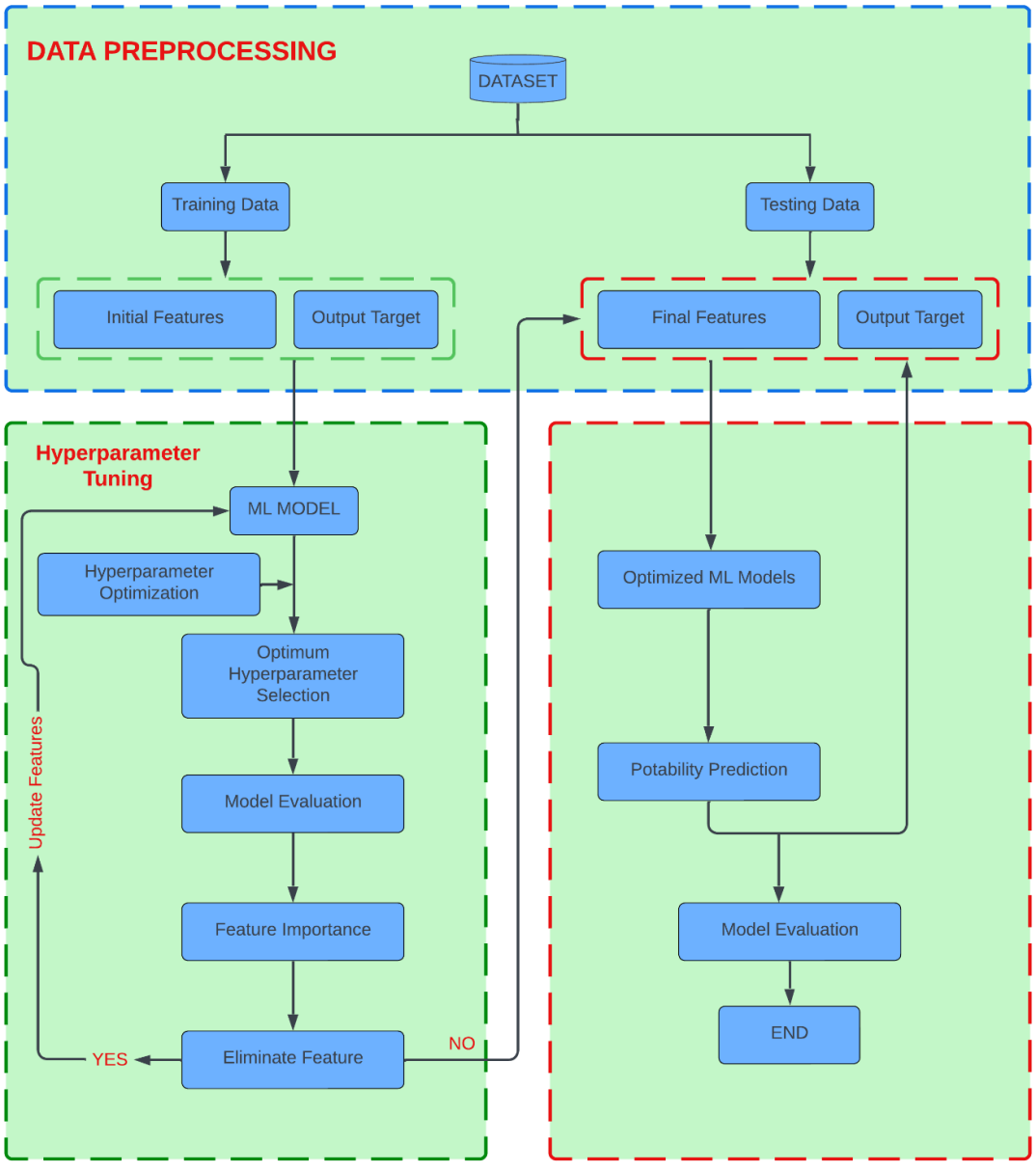

Figure 3. 1 Flow chart of all Machine Learning Processes

(Self-made)

3.1. Dataset

Physical attributes like temperature and clarity, chemical constituents like the percentage of sodium present, the amount of dissolved oxygen and solids, pH level, etc. and microbial attributes like the presence of microorganisms are the three categories into which water parameters are commonly divided. The datasets [6] offered in this study are limited to the physio-chemical parameters that is, the physical and chemical components of drinking and irrigation water. In the paper, two datasets—one for irrigation water and the other for drinking water are given. Smaller datasets that were taken from the Internet were combined to generate both databases.

pH, Na, Mg, Ca, Cl, K, CO3, HCO3, SO4, Turbidity, TDS, EC and TH were taken into consideration for the drinking water dataset, whereas RSC, PI, KR, MH, Na%, SAR and SSP were taken into consideration for the irrigation water dataset. An extra "label" column was added to both datasets. The numbers in the final column, "1" or "0," respectively, indicated whether or not the sample was "fit for use." This extra column was called "Usable" for the irrigation dataset and "Potability" for the drinking water dataset[6].

The drinking water dataset is utilized in this study. There are 14 characteristics and 718 outcomes in the dataset, as shown in Figure 3.2.

```
# Read the Dataset
df = pd.read_csv('DrinkingWater_Final_Dataset.csv')
# Print the shape of dataset
df.shape
```

```
(718, 14)
```

Figure 3. 2 Rows and columns Count of the Dataset

The head of the dataset looks like this:

```
df.head(5)
```

	id	pH	Sodium	Magnesium	Calcium	Chloride	Potassium	Carbonate	Sulphate	TDS	EC	TH	WQI	Potability
0	L1	7.26	77.51	32.55	81.4	63.55	1.95	419.68	68.16	640.0	1045.0	338.0	70.51	0
1	L2	7.54	36.11	31.10	42.4	19.53	1.56	273.28	57.12	400.0	645.0	234.0	48.31	1
2	L3	7.66	119.37	19.00	58.2	60.71	1.17	195.20	220.80	640.0	998.0	224.0	55.70	0
3	L4	5.98	117.07	29.65	145.4	37.28	2.73	580.72	190.08	950.0	1516.0	486.0	97.02	0
4	L5	7.02	45.08	23.35	49.6	23.43	1.17	248.88	74.88	410.0	658.0	220.0	46.45	1

Figure 3. 3 Top Five Rows of the Dataset

3.1.1. Statistical calculation of the features

```
# Statistical Calculation of the dataset
df.describe()
```

	pH	Sodium	Magnesium	Calcium	Chloride	Potassium	Carbonate	Sulphate	TDS	EC	TH	WQI	Potability
count	718.000000	718.000000	718.000000	718.000000	718.000000	718.000000	718.000000	718.000000	718.000000	718.000000	718.000000	718.000000	718.000000
mean	7.677284	84.948579	56.268189	70.926671	119.842312	12.113496	117.960822	50.705153	499.612535	1012.396281	197.817298	64.169847	0.614206
std	0.688892	147.306800	113.798579	141.330367	344.770005	17.270676	93.635222	78.488945	978.478140	1465.507731	154.203773	71.379101	0.487122
min	2.410000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	2.100000	0.000000	8.300000	0.000000
25%	7.270000	19.052500	14.000000	18.075000	17.750000	3.800000	54.612500	14.400000	0.000000	434.250000	116.247500	33.835000	0.000000
50%	7.640000	38.875000	19.000000	28.850000	36.240000	6.935000	116.160000	35.040000	195.750000	543.400000	162.000000	40.165000	1.000000
75%	8.110000	70.910000	36.825000	62.000000	76.600000	11.000000	144.980000	53.887500	696.500000	1020.000000	271.500000	63.812500	1.000000
max	10.010000	1157.800000	990.000000	1158.000000	4700.000000	110.000000	580.720000	1159.000000	11619.000000	16598.000000	1400.000000	722.690000	1.000000

Figure 3. 4 Statistical calculation of the dataset

3.1.2. Summary of Dataset

Table 3. 1 Dataset Summary

Specific subject area	Water Quality Prediction, Machine Learning
How data was acquired	Data was curated through the aggregation of several smaller datasets. These smaller datasets were compared and combined based on common features (parameters) after converting all parameters to the same units. Python script was used to calculate the "fitness of use" of each water sample using regional or global accepted range. Finally, two datasets were obtained, the first for drinking water with about 718 samples, and the second for irrigation water with 360 data samples.
Description of data	Physio-chemical parameters of water samples.
Data parameters	Potential of Hydrogen (pH), Sodium (Na), Magnesium (Mg), Calcium (Ca), Chloride (Cl), Potassium (K), Carbonate (CO3), Bicarbonate (HCO3), Turbidity, Total Dissolved Solids (TDS), Electrical conductivity (EC), Total Hardness (TH), Residual Sodium Carbonate (RSC), Permeability Index (PI), Kelly's Ratio (KR), Magnesium Hazard (MH), Sodium Percentage (Na%), Sodium Adsorption Ratio (SAR), Soluble Sodium Percentage (SSP)
Data source	Data on drinking and irrigation water scrapped from repositories on the Internet.

3.1.3. Target Distribution

Researchers can uncover hidden patterns and correlations between data variables with the use of data visualization. Presenting facts or information to readers in an effective and succinct manner is the aim of data visualization. To visually represent data, infographics, maps and charts are commonly utilized. Data visualization highlights trends and anomalies while assisting in the transformation of data in a more comprehensible manner. There are two classes in the dataset: 38.6% of the data are in the "potable" class and 61.4% of the data are in the "not potable" class, as shown in Figure 3.5. The potable class denotes the cleanliness and suitability of the water for human consumption.

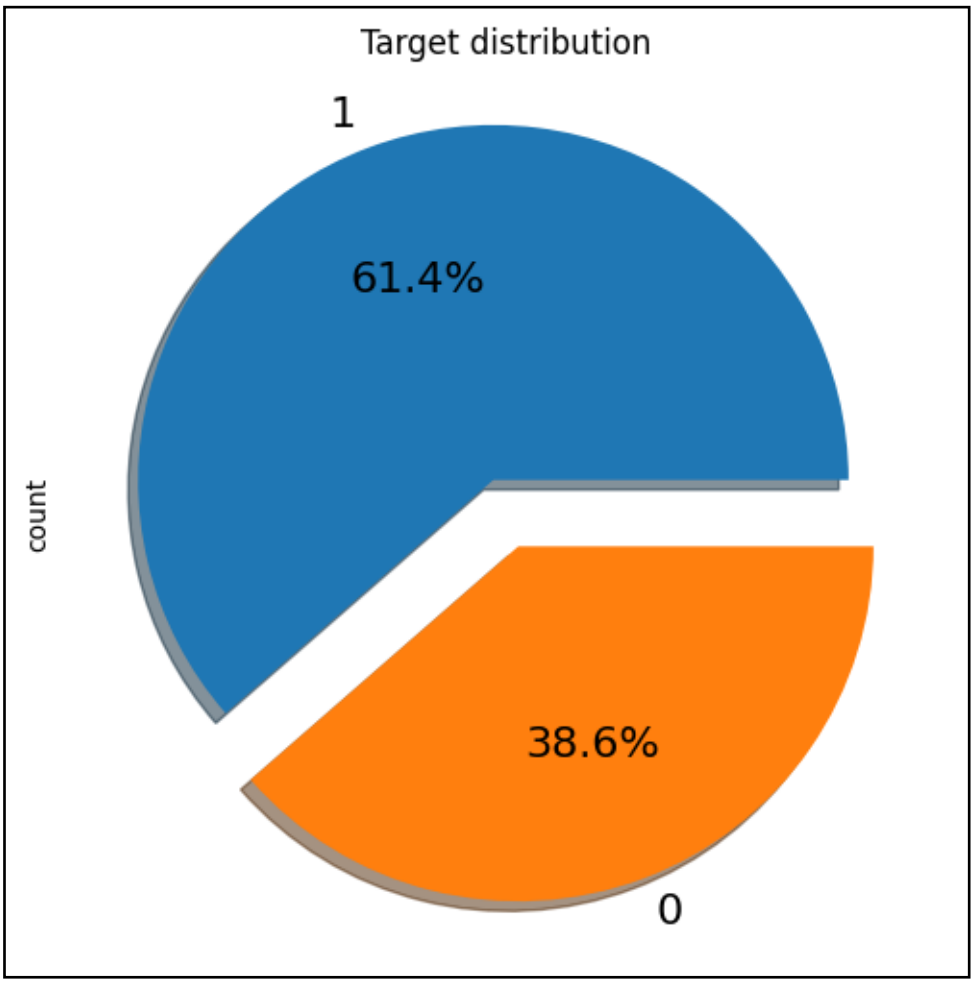

Figure 3. 5 Distribution of Target Features

3.1.4. Features Distribution

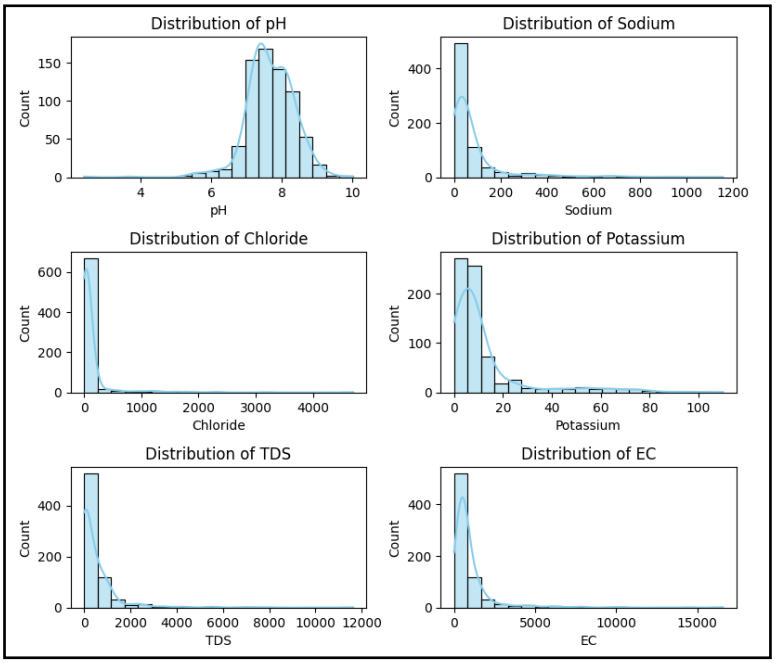

Figure 3. 6 Features Distribution (1-6)

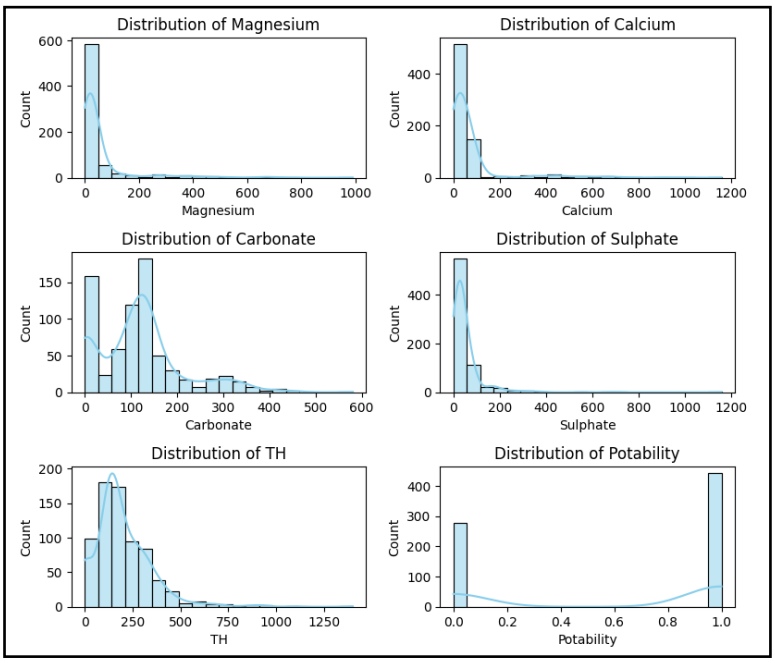

Figure 3. 7 Features Distribution (7-12)

3.2. Experimentation Environment

3.2.1. Google Colab

Google Colab, also known as Colaboratory, is a cloud-based platform that provides a Jupyter Notebook environment for free. This eliminates the need for local setup and allows researchers to leverage Google's powerful computing resources, including GPUs and TPUs, directly from their web browser. This makes Colab a valuable tool for working with the water potability dataset, as it facilitates machine learning model training and experimentation without requiring significant hardware investment.

3.2.2. Python

Python is a commonly used high-level programming language, it was designed by Guido van Rossum which can be easy to interpret and read. Python has specific functionality and is convenient to be used for both quantitative and analytical computational purposes. Data Science Python is popularly used and, as well as being a dynamic and open-source language, is a top choice. Its massive libraries are also used to manipulate the data however for a beginner data analyst they are really simple to learn. The Python libraries used in this thesis are briefly described as follows:

NumPy: NumPy is a library that consists of multidimensional array objects and a set of array processing routines. NumPy is used along with SciPy and Matplotlib packages. This combination is used for technical computing. Mathematical and logical operations are performed with the help of NumPy.

Pandas: Pandas is a software library that is designed for manipulating the data and analysis in a python programming language. It is open-source which is released under the BSD license of three clauses. It is based on the NumPy package and the Data Frame is its main data structure.

Matplotlib: Matplotlib is a module of Python used to plot attractive graphs. Visual representation in data science is a significant step. One can quickly understand how data is split by using visual representation. There are many libraries to represent the data, but the matplotlib is very widely known and easier to visualize.

SKlearn: Scikit-learn is a free Python library. It features multiple clustering classification and regression algorithms including random forests, DBSCAN, k-means,

gradient boosting, support vector machines and gradient boosting which is programmed to interface with the NumPy and SciPy libraries.

Seaborn: Seaborn is an open-source Python library that is used for statistical graphics. It offers a data set-oriented API to analyse relationships among different variables, as well as resources to select colour palettes that are truly in the data.

3.3. Machine Learning

The field of study known as machine learning enables machines to learn without being explicitly programmed. When a computer program's performance at tasks in a class of tasks T, as assessed by a performance measure P, improves with experience E, that program is said to have learned from experience E related to that class of tasks T. Machine learning is a broad term for a program that uses data analysis and data exploration to manage a variety of tasks [27].

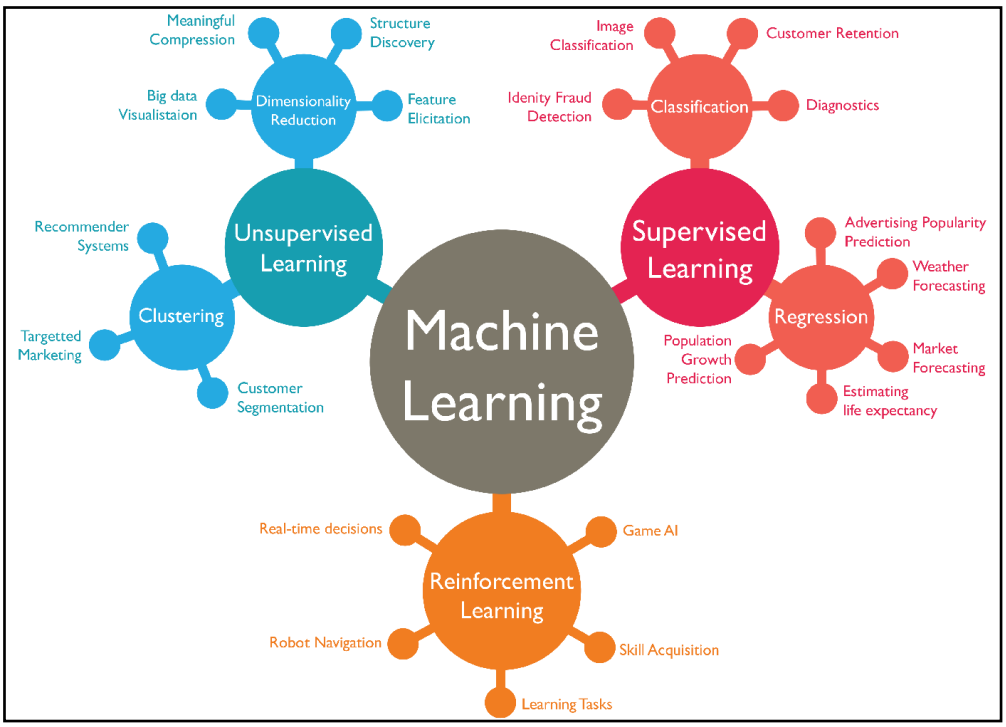

Figure 3. 8 Types of Machine Learning

Source: https://scikit-learn.org/1.3/tutorial/machine_learning_map/index.html

Applications of machine learning that are widely used include the identification of email spam, the theft of credit cards, stock predictions, personal assistants, product suggestions, self-driving cars, sentiment analysis etc.

3.3.1. Supervised Learning

Supervised learning is the model that is used the most often for machine learning operations. When the mapping between input-output data is reliable, it is frequently employed for such types of data. Learning from labelled test data is known as supervised learning, which is a branch of machine learning that focuses on developing models for regression or classification.

3.3.2. Unsupervised Learning

In the situation of unsupervised learning, where all the data is unlabelled, the classes are not explicitly labelled on the data. The model can learn from the data by spotting implicit patterns. Based on the data, unsupervised learning classifies the densities, structures, connected segments and other comparable attributes.

3.3.3. Reinforcement Learning

A branch of machine learning is reinforcement learning. It involves acting appropriately to maximize reward in a certain scenario. To identify the optimal course of action or direction it will take in a certain event, many algorithms and computers are used. The difference between supervised learning and reinforcement learning is that in supervised learning, the answer key is included in the training data, allowing the model to be trained with the correct response from the start, whereas in reinforcement learning, there won't be a response and the reinforcement agent will decide how to carry out the task. In the absence of training data, it is necessary for it to learn from its experience.

3.4. Machine Learning Algorithms

Forecasting refers to making future predictions, usually using historical data. For making forecasts for a very long time, statistical models were frequently used. The function of generalization in machine learning has been taken into account. In this thesis, we have used supervised learning algorithms such as Random Forest Classifier, Decision Tree Classifier, Gradient Boost Classifier and Adaptive Boost Classifier. These can make it easier to find better outcomes compared to traditional analytical techniques.

3.4.1. Random Forest Classifier

Similar to groups of decision trees are Random Forest Classifiers. In order to forecast a class, each tree poses a number of queries on the data. Every tree casts a majority vote to make the ultimate decision. Accuracy is increased and overfitting is prevented using this

method. Each tree selects the best question at each stage based on a notion known as Gini impurity, even though there isn't a single equation for the whole forest. This impurity essentially quantifies the degree of data fusion present at a given time. The forest often produces more accurate forecasts when it poses the most instructive queries. The Gini impurity for a class k at a node t is:

$$Gini(t) = 1 - \Sigma\left(pk(t)\right)^2 - - - - - - - - (1)$$

Here, pk(t) represents the proportion of samples at node t belonging to class k. Minimizing the Gini impurity ensures you choose the split that best separates the data based on class labels. By combining multiple, diverse decision trees through random feature selection and bagging, Random Forest achieves high accuracy and robustness in classification tasks.

3.4.2. Decision Tree Classifier

One effective tool for classification is a decision tree classifier. It creates a tree structure with questions about data features at each node. To partition the data, the algorithm selects the most informative question by measuring the class label mix (Gini impurity, Gini(t) = 1-Σ(pk(t)) ^2). Until it reaches a leaf node with the anticipated class label, this procedure iterates at each node, branching out based on the response. Decision trees are comprehensible and helpful for comprehending how factors affect the target variable because of this question-based methodology.

$$(x, X) = (x_1 + x_2 + x_3 + \ldots\ldots + x_k, X) - - - - - - - - (2)$$

The dependent variable, Y, is the target variable that we are trying to understand, classify or generalize. The vector **x** is composed of the features **x1, x2, x3** etc.

3.4.3. Gradient Boost Classifier

Gradient Boosting Classifiers are masters of ensemble learning; they can solve challenging classification problems by combining the strength of several models. They work in phases, integrating weaker learners to gradually create a strong model. These poor learners, which are frequently shallow decision trees, might not be the most precise instruments taken apart.

Gradient Boosting, on the other hand, systematically adds them together while following the path of a loss function, L (y, f(x)), which quantifies the difference between the true label (y) and the model's prediction (f(x)). Logarithmic loss is a popular loss function used

in classification. The ensemble gets stronger over time as each new tree concentrates on fixing the mistakes made by the ones before it. Adaptive learning further improves this repeated refining.

Trees with better performance in rectifying past errors are assigned a higher weight (α), ensuring the model prioritizes the most challenging aspects of the data. This combination of iterative learning, focus on correcting errors, and adaptive weighting, is captured in the minimization of the loss function with each new tree ($min \sum i\ L(y * i, Fm(x * i))$), is what makes Gradient Boosting Classifiers such a powerful tool.

3.4.4. AdaBoost Classifier

For classification problems, AdaBoost, which stands for Adaptive Boosting, is the best ensemble learning method available. It combines several poor learners often shallow decision trees—to create a strong ensemble iteratively. AdaBoost employs a more calculated strategy than a simple average. It gives the weak learners and the data points themselves weights (w*i). Weights of data points that prior learners misclassified are increased, which compels the algorithm in later iterations ($w * i\ =\ D(i)\ *\ exp(-\alpha\ *\ y * i\ *\ h\_t(x * i))$ to concentrate on these difficult cases. Here, α is a learning rate parameter that controls the impact of each new weak learner, y*i is the true label, and h_t(x*i) represents the prediction of the t-th weak learner for a given data point (x*i). Additionally, more accurate weak learners receive a higher weight, ensuring their contributions hold more significance in the final ensemble. This adaptive approach, captured mathematically in the weight update equation, continuously adjusts the focus to improve on past errors, progressively strengthening the overall ensemble and making AdaBoost a powerful tool for conquering complex classification problems.

3.5. Selection of Machine Learning Algorithms

For every problem, choosing an algorithm is not a trivial decision. There is no proper algorithm that works for any problem, but few algorithms are widely recognized for performing the algorithms better than others in some cases. One cannot assume more accuracy from the algorithms for all types of data, accuracy will differ from data to data. In this thesis Machine Learning Algorithms such as Decision Tree Classifier, Random Forest Classifier, Gradient Boost Classifier and Ada Boost Classifier were considered which they expected to perform well on the issues.

3.6. Selection of Performance Metrics

It is essential to use the best performance criteria when assessing a water potability prediction model. The project's primary focus should be on metrics intended for classification tasks since it deals with the crucial categorization of safe from harmful drinking water. Among them is accuracy (total correctness), which serves as a benchmark. But accuracy (preventing false positives, which would classify contaminated water as safe and pose a serious public health risk) and memory (preventing false negatives, which would mean missing samples of healthy water and maybe missing sources of potable water) is also crucial. Furthermore, for a more nuanced assessment, the F1-score which balances accuracy and recall might be taken into consideration based on the balance of the dataset (equal or unequal safe/unsafe samples). Evaluating these metrics together provides a comprehensive picture of the model's effectiveness in accurately distinguishing safe and unsafe water samples, ensuring the robustness of the water potability prediction system.

3.7. Feature Selection

There are various types of factors that can make the model of Machine Learning more effective on any given task. One of the methods of feature selection is data correlation**.** which will have a major impact on the model’s performance. This will reduce a lot of strain on the Machine Learning model during the preprocessing and cleansing of the data. The data attributes chosen for training the Machine Learning model would have a major impact on the efficiency of the model. Because of the irrelevant features that are presented, the model output will be reduced. The feature selection method provides an efficient way to remove data redundancy and irrelevant data that helps to reduce computation time, improve accuracy and also enhance understanding of the model. The selection of features plays a crucial role in classification and involves selecting a subset of features that reflect the complete attributes that currently exist. Feature selection techniques are intended to improve classification efficiency by selecting the essential features from the data sets according to particular algorithms.

3.8. Data Correlation Method

Data correlation is a method that helps to predict one attribute from another attribute and is used as a basic quantity in many modelling techniques. If one feature increases, the correlation will be positive, so the other feature increases as well and negative if one feature increases there will be a reduction in another. If there is no relation between any two attributes then it is said to be no correlation. If there is a linear relationship between the constant variables then the Pearson correlation coefficient is used. If there is a non-linear relation between the constant variables then the Spearman correlation coefficient is used. Since the considered data set is linear so the Pearson correlation coefficient is used for the selection of features in this study. This correlation for all the attributes is shown in Figure 3.8. To improve the efficiency of the Machine Learning model, the attributes that have negative correlations were removed. It is a statistic measuring the linear correlation of two variables X and Y. It has a value between +1 and -1, where 1 is a linear positive correlation, 0 is not a linear correlation and 1 is a linear negative correlation.

The motivation for considering the correlation is when people know a score on one measure, they can make a prediction of another measure that is highly related to it more accurately. The more accurate the prediction, the stronger the relationship between the variables.

Correlation Values of the Dataset

	pH	Sodium	Magnesium	Calcium	Chloride	Potassium	Carbonate	Sulphate	TDS	EC	TH	WQI	Potability
pH	1.000000	-0.125903	-0.120612	-0.144589	-0.100019	-0.107411	0.124256	0.006166	-0.206245	-0.133295	-0.063449	-0.139432	0.212620
Sodium	-0.125903	1.000000	0.775433	0.773288	0.640255	0.749988	-0.353335	0.055338	0.442001	0.356534	0.553279	0.835522	-0.474703
Magnesium	-0.120612	0.775433	1.000000	0.782078	0.765669	0.751535	-0.348562	-0.010458	0.555141	0.470986	0.520343	0.921479	-0.424179
Calcium	-0.144589	0.773288	0.782078	1.000000	0.655552	0.728845	-0.253341	-0.013507	0.463089	0.367746	0.494101	0.872919	-0.432351
Chloride	-0.100019	0.640255	0.765669	0.655552	1.000000	0.676211	-0.258651	0.116935	0.626584	0.555888	0.568802	0.859006	-0.317176
Potassium	-0.107411	0.749988	0.751535	0.728845	0.676211	1.000000	-0.383850	0.030706	0.442644	0.355177	0.561791	0.843895	-0.473013
Carbonate	0.124256	-0.353335	-0.348562	-0.253341	-0.258651	-0.383850	1.000000	0.046015	-0.300418	-0.314305	0.063900	-0.303052	0.084923
Sulphate	0.006166	0.055338	-0.010458	-0.013507	0.116935	0.030706	0.046015	1.000000	0.043543	0.026006	0.228441	0.066788	-0.221667
TDS	-0.206245	0.442001	0.555141	0.463089	0.626584	0.442644	-0.300418	0.043543	1.000000	0.978443	0.362390	0.702463	-0.481486
EC	-0.133295	0.356534	0.470986	0.367746	0.555888	0.355177	-0.314305	0.026006	0.978443	1.000000	0.253454	0.613267	-0.386902
TH	-0.063449	0.553279	0.520343	0.494101	0.568802	0.561791	0.063900	0.228441	0.362390	0.253454	1.000000	0.646277	-0.546986
WQI	-0.139432	0.835522	0.921479	0.872919	0.859006	0.843895	-0.303052	0.066788	0.702463	0.613267	0.646277	1.000000	-0.530114
Potability	0.212620	-0.474703	-0.424179	-0.432351	-0.317176	-0.473013	0.084923	-0.221667	-0.481486	-0.386902	-0.546986	-0.530114	1.000000

Fig. 3.8: Correlation Values

The heat map for correlation between non-numerical attributes is plotted as follows

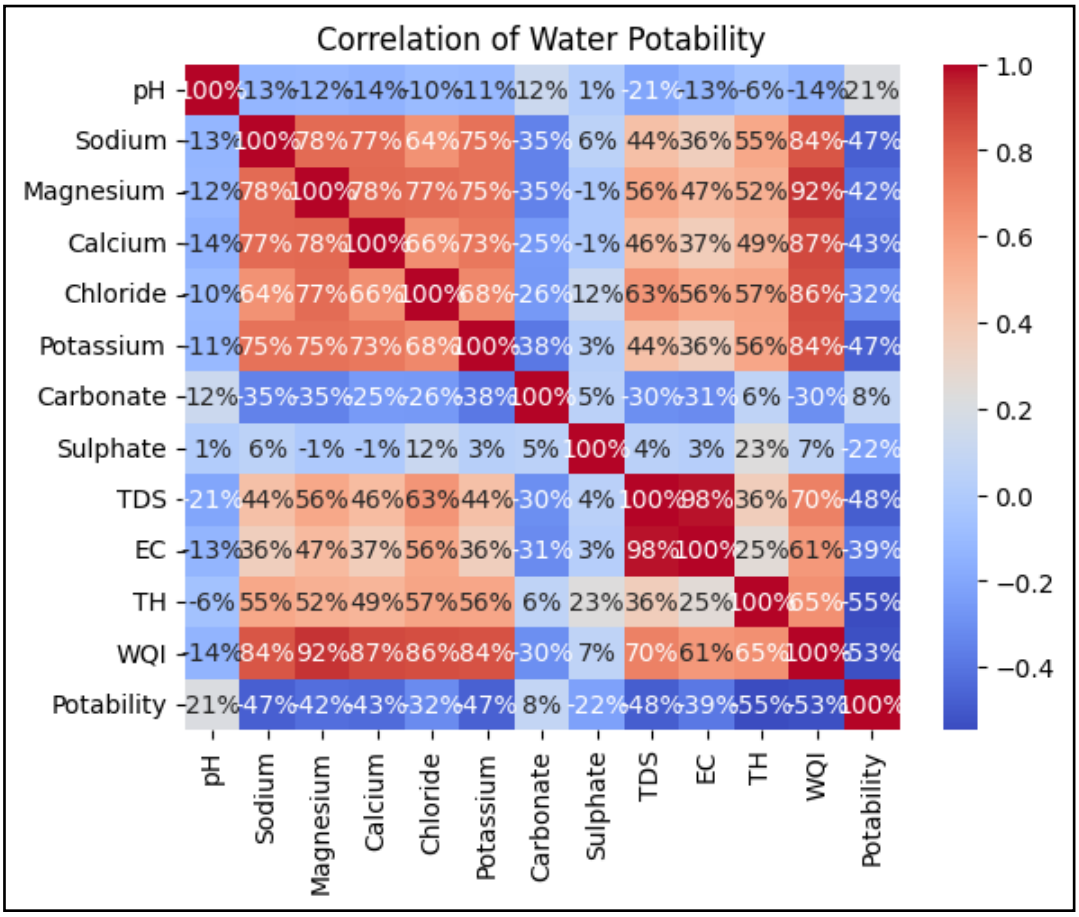

Figure 3. 9 Heat Map

3.9. Feature Importance

Feature Importance refers to a class of approaches for assigning values to input features to a predictive model which determines the relative significance of each factor while forecasting.

Feature importance scores provide an overview of the model. Most significant scores are determined using a prediction approach that was fitted to the dataset. Inspecting the score of importance gives insight into that particular model and what features are the most essential and least important to the model while making a prediction. This is a type of interpretation of the model that can be carried out for those models that encourage it.

Feature Importance can be used to enhance a predictive model. This can be accomplished by selecting those features to remove (lowest scores) or those features to retain, using the importance scores. This is a type of selection of features and can simplify the

modelling problem, accelerate the modelling process and in certain cases improve model performance.

3.10. Data Preprocessing

To make sure that the dataset is clear, consistent and appropriate for analysis, data preparation is a crucial step. First, the Python packages NumPy and Pandas were used to search the dataset for any null values. It was discovered to be devoid of null values, meaning that no imputation or missing data removal was required. After that, statistical techniques were applied to the dataset to check for outliers, and the results showed that no significant outliers existed that would have distorted the analysis's findings. Furthermore, the dataset had a column that was redundant because it was the same as the index. In order to simplify the dataset and prevent any needless repetition, this column was eliminated.

The preparation technique was made simpler by the absence of category encoding because the dataset is solely composed of numerical values. Techniques for normalization and standardization were used to get the data ready for modelling. By scaling the data to a range of [0, 1], normalization ensured that the magnitudes of the various attributes were consistent. In order to meet the requirements of algorithms that presume normally distributed data, standardization was applied to further alter the data to have a mean of 0 and a standard deviation of 1. By ensuring that the dataset was suitably scaled, these preprocessing activities improved the performance and dependability of tasks involving machine learning and subsequent analysis. These crucial preprocessing procedures were addressed in order to lay a strong basis for reliable and accurate modelling.

3.10.1. Acceptable Range for Values for Water Parameters

Table 3. 2 Acceptable range of the parameters.

Parameter	Unit/Formula	Range
ph	-	5 – 9.7
Na	mg/L	< 200
Mg	mg/L	< 50
Ca	mg/L	< 75
Cl	mg/L	< 300
K	mg/L	< 12
SO4	mg/L	< 500
CO3	mg/L	1.2 – 2
HCO3	mg/L	120 – 200
Turbidity	NTU	< 5
TDS	mg/L	< 1200
EC	mg/L	< 170
TH	mg/L	100 - 300

3.11. Hyperparameter Tuning

Hyperparameter tuning is a crucial step in the process of developing machine learning models. Hyperparameters are settings that are not learned directly from the data during the training process but rather are set by the user before training begins. They have a significant impact on the performance and generalization ability of the model.

The purpose of hyperparameter optimization is to find the global optimal value x of the objective function f(x) that can be evaluated for any arbitrary [$x \in X, x* = arg\ min\ x \in X\ f(xx)$], and X is a hyperparameter space that can contain categorical, discrete, and continuous variables. In order to construct the design of different machine learning models, the

application of effective hyperparameter optimization techniques can simplify the process of identifying the best hyperparameters for the models. HPO contains four major components: First, an estimator that could be a regressor or any classifier with one or more objective functions. Second, a search space. Third, an optimization method to find the best combinations, and Fourth, a function to make a comparison between the effectiveness of various hyperparameter configurations. Some of the common hyperparameter techniques which have been used in this study are discussed below

3.11.1. Grid Search

Grid search is a method that thoroughly examines a manually defined portion of the target algorithm's hyperparameter space. A conventional method for determining the optimal value involves doing a grid search, which involves executing experiments or procedures under various conditions. For instance, if three components are present, a 15 × 15 × 15 grid would need to conduct 3375 tests under various situations. Grid search is more useful in the following situations: (1) the model's total number of parameters is small, say M <10. The number of test solutions is proportional to LM, where L is the number of test solutions along each grid dimension because the grid is M-dimensional. (2) The grid's bounds can be defined by knowing that the answer falls inside a particular range of values. (3) It is not prohibitively long to compute LM from the direct problem d=g(m) since it can be done rapidly enough. (4) To ensure that the minimum is not lost due to an excessively coarse grid spacing, the error function E(m) is uniform on the scale of the grid spacing, Δm.

The grid search approach is fraught with issues. First, if there are several components, the number of studies may be prohibitive. Choosing the optimal location on the grid might be deceptive, especially if the optimum is quite flat because there can be substantial experimental error. This implies that even if the trials are repeated under the same conditions, different results may be produced. The third is that it could identify a false (local) optimum or lose features around the optimum if the original grid is too tiny to support the number of tests.

3.11.2. Random Search

Grid search may be fundamentally improved with random search. It denotes a haphazard exploration of hyper-parameters from certain distributions over potential parameter values. The search is carried out until the targeted accuracy is attained or the predefined budget is depleted. These techniques, which are the most basic forms of stochastic optimization, come

in quite handy when dealing with issues like simulations that run quickly and tiny search spaces. For every hyperparameter, RS determines a value before the probability distribution function. Based on the generated hyperparameter sets, the GS and RS both calculate the cost measure. Despite its simplicity, RS has shown itself to be more successful than Grid search in several situations.

It has been demonstrated that random search yields superior results for a number of reasons. Firstly, because the budget may be individually determined based on the search space distribution, random search performs better, particularly when there are several hyper-parameters that are not distributed uniformly. Second: Resource allocation and parallelization are simple due to the independence of each evaluation. Instead, then investing a lot of time in a tiny, underperforming area, RS minimizes system efficiency by sampling several parameter combinations from a predetermined distribution. This is in contrast to GS. Furthermore, if provided with an enough budget, this approach can identify global optimal values or values that are nearly global. Third, longer search periods cannot provide better results in Grid searches, but more time spent searching will increase the chance of discovering the ideal hyperparameter configuration, even when obtaining optimal results via random search is not promising.

In the early phases of HPO, it is advised to employ random search to rapidly restrict the search space before utilizing guided algorithms to improve outcomes. The primary disadvantage of RS and GS is that they spend time analysing underperforming regions of the search space since each assessment in an iteration does not rely on earlier evaluations.

3.12. Applying Hyperparameter Tuning in ML Models

In order to put the theory into practice, several experiments have been performed on an industrial-based synthetic polymer model. This section describes experiments with two different HPO techniques on four general and representative ML algorithms. In the first part of the section, we discussed the experimental setup and the main HPO process. In the second part, we compare and analyse the results of the application of different HPO methods.

An overview of common ML models used in this work, and their hyper-parameters are listed below:

Table 3. 3 Hyperparameters of different Models.

ML MODELS	HYPERPARAMETER
1. Decision Tree Classifier	ccp_alpha class_weight, criterion max_depth max_features max_leaf_nodes min_impurity_decrease min_samples_leaf min_samples_split min_weight_fraction_leaf random_state splitter
2. Random Forest Classifier	n_estimators max_features max_depth min_samples_split
3. Gradient Boost Classifier	max_iter learning_rate max_depth max_leaf_nodes
4. AdaBoost Classifier	algorithm base_estimator estimator learning_rate n_estimators random_state

3.13. K-fold Cross-Validation

Cross-validation (CV) is a procedure of statistical analysis used to assess the effectiveness of a Machine Learning technique, as well as a re-sampling method used to validate an algorithm if there is insufficient data. Stratification is the process of rearranging the data to ensure each fold is a good representative of the whole. Data splitting into folds may be controlled by criteria such as ensuring that each fold has the same ratio of outcomes with a given categorical value, such as the class outcome value. This process is called stratified k-fold cross-validation. Common techniques of cross-validation include K-fold cross-validation, Stratified K-fold cross-validation, and cross-validation leave-one-out. The motivation behind

the 5-fold stratified cross-validation is that the estimator has a lower variance than a single hold-out set estimation method which could be very essential if there is a limited amount of data. There will be plenty of variance in the results estimate for various data samples, or for specific data partitions to create training and test sets. The 5-fold stratified cross-validation removes this variance by comparing more than 5 separate partitions, thereby making the performance estimate less sensitive to data partitioning.

3.14. Performance Metrics

Selecting the best performance measures is essential to assess how well the machine learning model predicts the potability of water. Given that the primary focus of the thesis is on the crucial duty of classifying safe drinking water, the metrics that have been chosen should make it evident how successfully the model separates water samples that are suitable for human consumption from those that are not.

3.14.1. Accuracy

When assessing a machine learning model's effectiveness in predicting water potability, accuracy is a crucial parameter. It shows the total percentage of water samples that the model properly categorized. In mathematics, correctness is written as:

Accuracy = (True Positives + True Negatives) / Total Samples --------(3)

In this case, samples of harmful water are reliably categorized as unsafe (true positives), while samples of safe water are accurately labelled as safe (true negatives).

A high accuracy number (preferably around 1) means that the model performs well in differentiating between water samples that are safe and those that are not. Ensuring the public's health depends on accurately identifying water that is safe and dangerous.

But it's critical to recognize accuracy limitations, especially when working with unbalanced datasets. A high accuracy might be deceptive if the dataset is substantially biased toward one class (for example, largely safe water samples). If the model is just successful in forecasting the majority class, it may fail to recognize the less common but still significant dangerous water samples.

Consequently, even while accuracy offers a useful foundation for assessment in a thesis on water potability prediction, other criteria should be taken into account, particularly when working with unbalanced datasets.

3.14.2. Precision

Regarding the prediction of water potability, accuracy provides a general overview of a model's performance. But it's important to go deeper and see how well the model recognizes safe drinking water for public health concerns. This is the point at which accuracy is important.

The percentage of real positives among all anticipated positives is the subject of precision. In terms of math, it is stated as:

Precision = True Positives / (True Positives + False Positives) --------(4)

In this case, false positives are samples of safe water that were inadvertently misclassified as hazardous, whereas true positives are still unsafe water that was properly classified as unsafe. A precision rating that is high suggests that the model is very good at preventing false positives. This is crucial in relation to the potability of the water. False positives, or the labelling of safe water as harmful, can waste resources and cause unwarranted public fear.

It guarantees that the model correctly finds sources of clean water by giving high precision priority, hence reducing the possibility of it being missed.

For a thorough analysis, precision must be taken into account in addition to other measures like recall (which are covered later). A very precise model may be too cautious, giving up some real positives (safe water properly categorized as safe) in order to reduce false positives.

Therefore, in order to guarantee that a water potability prediction model properly distinguishes between safe and hazardous water samples while limiting misclassifications that might have an impact on public health, an evaluation of the model must take a balanced approach, taking into account both accuracy and precision.

3.14.3. Recall

Though they don't provide a complete picture, accuracy, and precision are crucial criteria for assessing a water potability prediction model. Another important parameter is recall, which measures how well the model can identify all of the water samples that are actually safe. Mathematically, recall is expressed as:

Recall = True Positives / (True Positives + False Negatives) --------(5)

In this case, false negatives are samples of safe water that were mistakenly categorized as unsafe, while true positives of unsafe water that were accurately identified as unsafe remain the same.

A high recall score means the model performs well in terms of reducing false negatives. Even a small number of missing clean water samples might have serious repercussions. Individuals may unintentionally drink tainted water, posing health hazards. It guarantees that the model efficiently detects safe water sources by giving high recall priority, hence reducing the possibility of its being missed.

However, recall should be taken into account in addition to other measures, much like accuracy. In order to prevent missing any safe water samples, an excessively aggressive model with an unusually high recall rate can categorize some harmful ones as safe. This may be dangerous for everyone's health.

Consequently, a well-rounded strategy is required. Assessing the model's performance using accuracy, precision, and recall offers a thorough knowledge of its advantages and disadvantages. This enables us to minimize misclassifications that could have an effect on public health while guaranteeing the model correctly distinguishes between safe and harmful water samples. By going over these indicators collectively, it will be able to show how well the model protects clean drinking water.

3.14.4. F1-Score

Though each provides a unique viewpoint, accuracy, precision, and recall are all useful criteria for assessing a water potability prediction model. A model may perform well in one domain (such as strong recall) but poorly in another (such as low accuracy). The F1-score can help in this situation by offering a more impartial assessment of the model's performance.
The F1-score is the harmonic mean of precision and recall, calculated as:

F1-score = 2 * (Precision * Recall) / (Precision + Recall) --------(6)

Models that prioritize one statistic over another are effectively penalized by this metric. A high F1-score suggests that the model strikes a good compromise between avoiding false positives (safe water mistakenly categorized as harmful) and false negatives (safe water overlooked) and finding genuine positives (unsafe water).

CHAPTER 4

4. EXPERIMENTAL RESULTS

The performance evaluation of four machine learning models the AdaBoost Classifier, Decision Tree Classifier, Random Forest Classifier and Gradient Boost Classifier for predicting water potability is covered in detail in this chapter. There are two separate stages to the review process.

The first stage looks at the models' initial performance without any hyperparameter adjustments, setting a baseline. This preliminary analysis evaluates the capacity of each model, in its default configuration, to distinguish between potable and non-potable water samples. Metrics like as F1-score, recall, accuracy and precision has been employed to assess this capacity.

The effects of hyperparameter adjustment are investigated in the second stage. Hyperparameters are configurations that govern a model's learning process. The objective is to increase the model's prediction accuracy and capacity to identify underlying patterns in the water quality data by methodically modifying these parameters. The best hyperparameter combinations for each model have been found using strategies such as grid search and randomized search. In order to determine if hyperparameter tuning is beneficial in this particular application, a comparison between the performance of the adjusted models and the baseline findings has been done at the conclusion.

4.1. Results Before Hyperparameter Tuning

This section examines the models' original performance prior to any modifications are made to their internal setups. Here, researchers assess their capacity to forecast potability of water using their default configurations.

4.1.1. AdaBoost Classifier

The AdaBoost Classifier achieved an initial accuracy of 0.9814, demonstrating a strong capability to correctly classify water samples. This suggests that the model can effectively learn the patterns within the data and distinguish between potable and non-potable water. Precision, recall, and F1-score, which provide more nuanced insights into the model's performance, were also high at 0.9921, 0.9767 and 0.9843 respectively, as shown in Figure 4.1. Precision indicates a low probability of misclassifying a potable water sample as non-potable (e.g., out of 100 positive predictions, 99 were truly potable). Recall reflects the model's ability to identify all

true positive cases (e.g. Out of 100 actual potable samples, the model correctly identified 97). The F1-score combines these metrics, providing a balanced view of the model's performance.

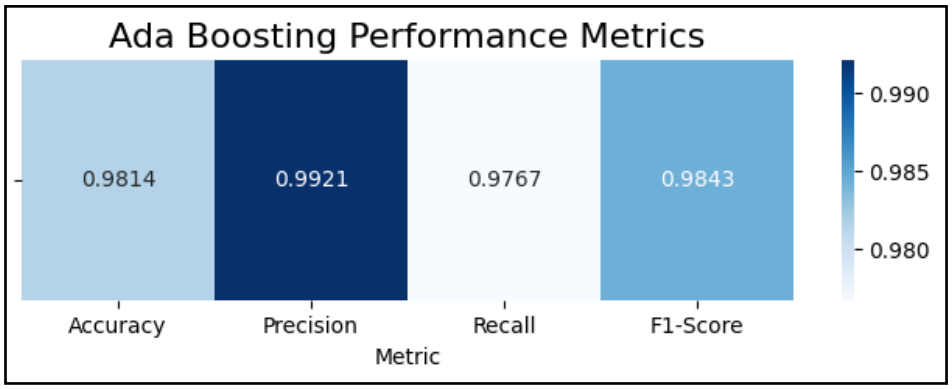

Figure 4. 1 Performance Metrics (Ada Boost)

4.1.2. Decision Tree Classifier

Although the Decision Tree Classifier's accuracy of 0.9629 is a little less than that of the AdaBoost Classifier, it still shows promise for predicting water potability. The accuracy, recall, and F1-score values (0.9689, 0.9639 and 0.9769) indicate that the model can successfully learn the patterns in the data and produce precise classifications, much like the AdaBoost Classifier, as shown in Figure 4.2. It may be required to take a deeper look at the confusion matrix in order to pinpoint possible locations that might benefit from hyperparameter adjustment. For example, the decision tree may be having trouble with samples that are on the verge of potability or misclassifying a certain kind of contaminant. By modifying the tree's depth, splitting criteria or minimum samples per leaf to enhance its decision-making process for water potability prediction, hyperparameter tuning might solve these problems.

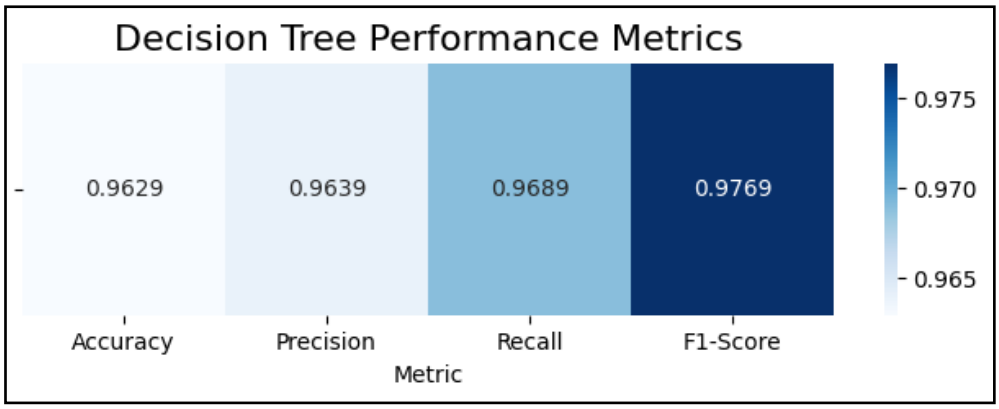

Figure 4. 2 Performance Metrics (Decision Tree)

4.1.3. Random Forest Classifier

While less accurate than the previous two models, the Random Forest Classifier's 0.9630 accuracy is still within a reasonable range for predicting water potability. The F1-score, precision, and recall values (0.9618, 0.9767 and 0.9692) shown in Figure 4.3 that this model performs similarly to the other models. But in this case, it's critical to take into account the trade-off between recall and accuracy.

Predicting water potability requires a good recall rate in particular. False negative classification of a tainted water sample as safe might have grave repercussions. Researchers may ascertain if the Random Forest gives priority to memory or accuracy (accurately recognizing clean water) by examining the confusion matrix. Hyperparameter tweaking may be utilized to boost recall at the possible cost of a modest drop in precision if the model exhibits a significant number of false negatives. The acceptable risk tolerance for incorrectly categorized samples must be carefully considered before making this decision.

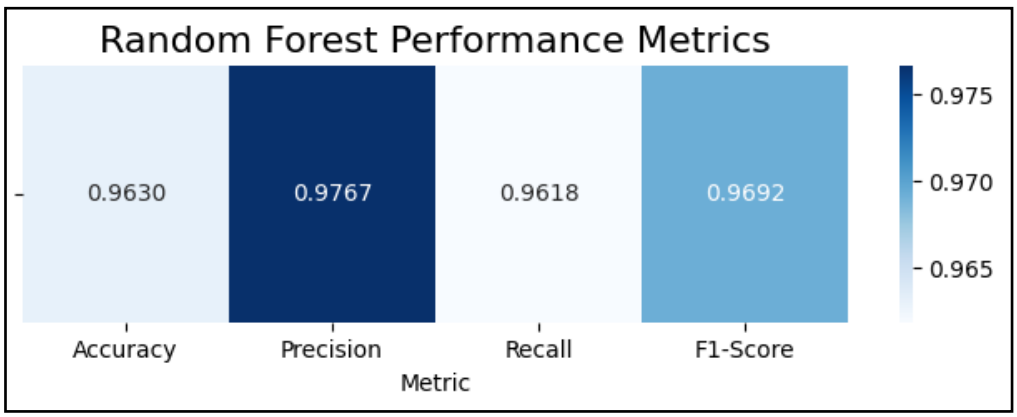

Figure 4. 3 Performance Metrics (Random Forest)

4.1.4. Gradient Boost Classifier

With an astounding accuracy of 0.9861, the Gradient Boost Classifier outperformed the AdaBoost Classifier. The strong recall (0.9844), precision (0.9921) and F1-score (0.9883) numbers (Shown in Figure 4.4) all lend more credence to this. Even though these findings point to high overall skills, further investigation of the confusion matrix is nevertheless advised in order to spot any potential biases or areas in need of development.

The Gradient Boost Classifier in this instance seems to have struck a good balance between precision and recall, in contrast to the Random Forest, which could give preference to recall. The high numbers for these measures make this clear. This balance guarantees a large number of accurately categorized samples, both safe (true negatives) and polluted (true positives), in the context of water potability prediction. This is especially important since it might lead to serious repercussions if contaminated water samples are missed (false negatives).

Although the emphasis may change, hyperparameter adjustment for the Gradient Boost Classifier may still be useful. Given the model's current high performance, tweaking might focus on improving it even more or investigating its applicability to new data. Methods such as grid search or randomized search might be used to find hyperparameter settings that could increase the resilience of the model.

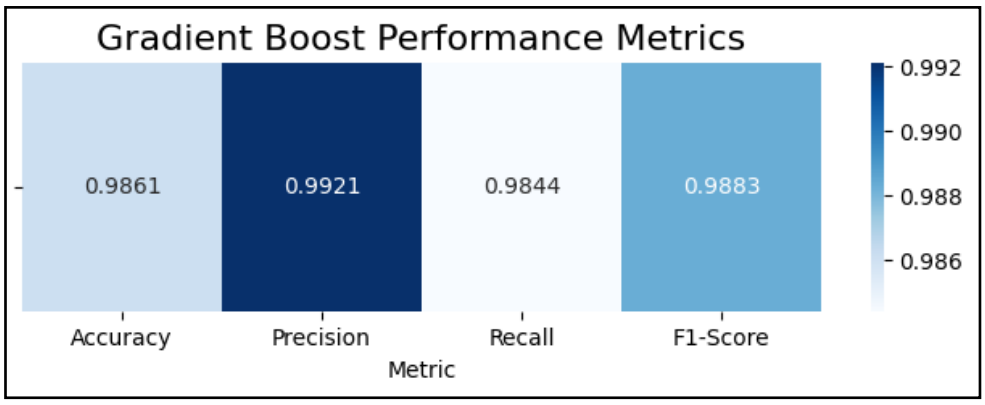

Figure 4. 4 Performance Metrics (Gradient Boost)

4.1.5. Result of all the models in One Table

Table 4. 1 Result of all models.

Model	Accuracy Score	Precision Score	Recall Score	F-1 Score
Ada Boosting	0.9814	0.9921	0.9767	0.9843
Random Forest	0.9630	0.9767	0.9692	0.9618
Decision Tree	0.9629	0.9618	0.9889	0.9769
Gradient Boost	0.9861	0.9923	0.9845	0.9886

4.2. Results After Hyperparameter Tuning

In order to maximize the performance of a model by modifying its internal settings, hyperparameter tuning is an essential step in machine learning. The way in which the model learns from the data and produces precise predictions may be greatly impacted by these configurations.

4.2.1. Ada Boost Classifier

For the AdaBoost Classifier, random search cross-validation a method that effectively investigates a random selection of hyperparameter values proved to be quite successful. The model performed much better with this method. From a baseline accuracy of 0.980 (as previously mentioned), accuracy increased to an astounding 0.986. Other important measures, such as accuracy (0.992), recall (0.984), F1-score (0.988) and ROC AUC score (0.986), also showed improvement. In particular, the ROC AUC score (shown in Figure 4.5) shows a good capacity to distinguish between water samples that are drinkable and those that are not.

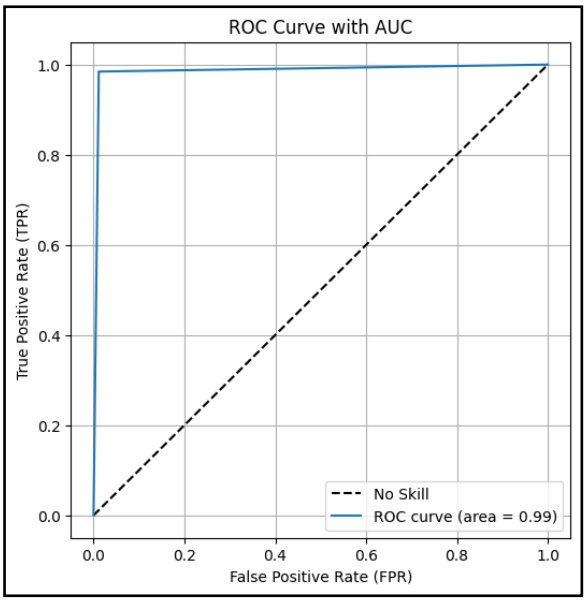

Figure 4. 5 ROC curve with AUC (Ada Boosting)

By looking at the confusion matrix (Figure 4.6), the benefits of hyperparameter adjustment are further demonstrated. This indicates a very low rate of incorrect categorization. The model has remarkable accuracy in recognizing safe drinking water, with 86 true positives (potable samples properly categorized) and just 1 false negative (potable sample missed). There were just two false positives (erroneously categorized) and 130 true negatives (properly identified) on the non-potable side. For water potability prediction, this balanced performance across both classes is essential.

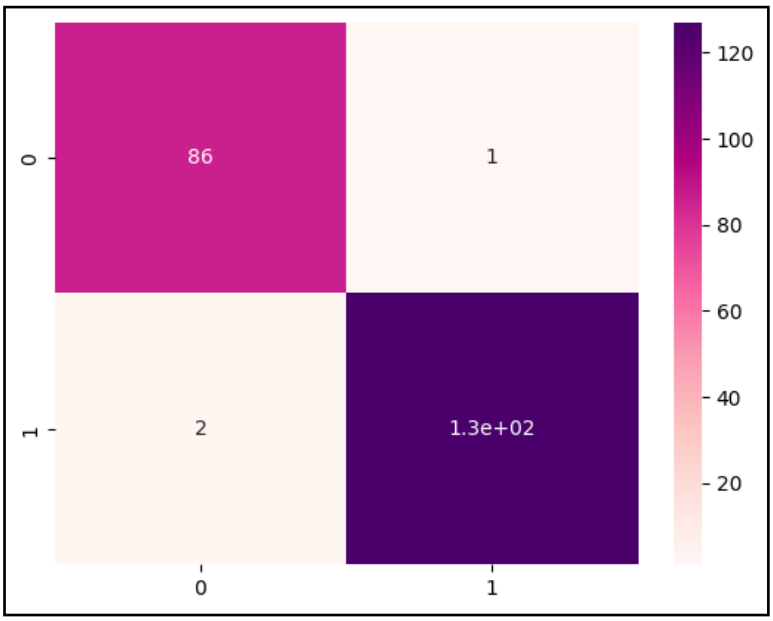

Figure 4. 6 Heatmap of Confusion Matrix (Ada Boosting)

4.2.2. Decision Tree Classifier

Once more, random search cross-validation proved to be the most successful technique for Decision Tree Classifier hyperparameter tweaking. The impact was not the same as the AdaBoost Classifier, though. The accuracy decreased a little to 0.953, while the F1-score was steady at 0.981. This change raises the possibility that, after tuning, recall may become more important than accuracy. Even if this results in a few more false positives (non-potable water that is mistakenly categorized as safe), the Decision Tree may be concentrating on identifying all actual positive cases (contaminated water samples).

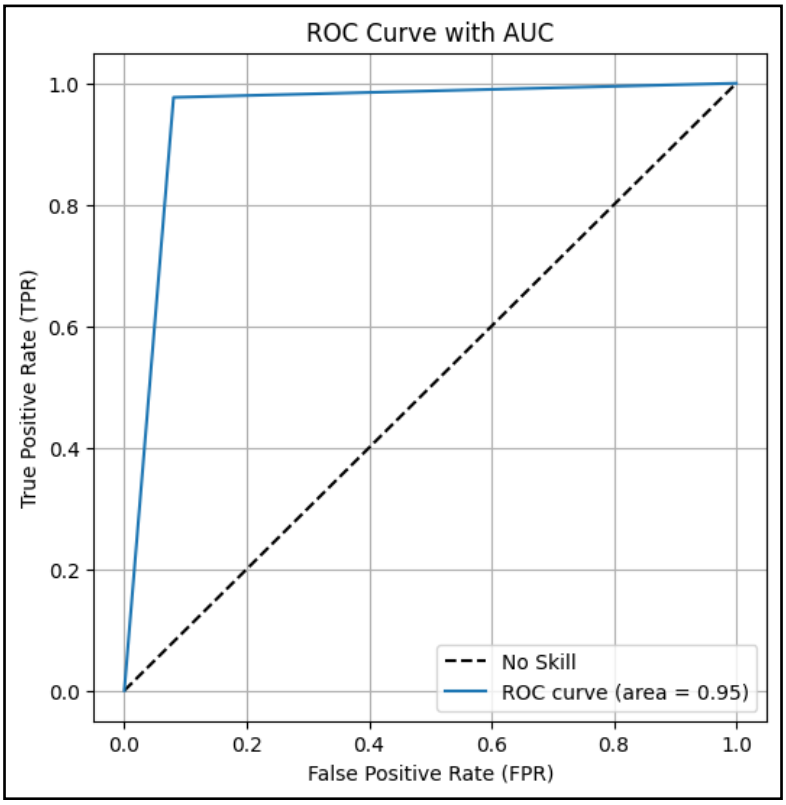

Figure 4. 7 ROC curve with AUC (Decision Tree)

To verify this idea, a more thorough examination of the confusion matrix is necessary. There may be a small rise in false positives to offset any potential reduction in false negatives (missed potable water samples). A decent capacity to discriminate between classes is indicated by the ROC AUC score of 0.948, as shown in Figure 4.7, which is not quite as great as the tweaked AdaBoost Classifier. With 80 true positives, 7 false negatives, 3 false positives, and 130 true negatives, the Decision Tree Classifier performs well overall, as shown in Figure 4.8.

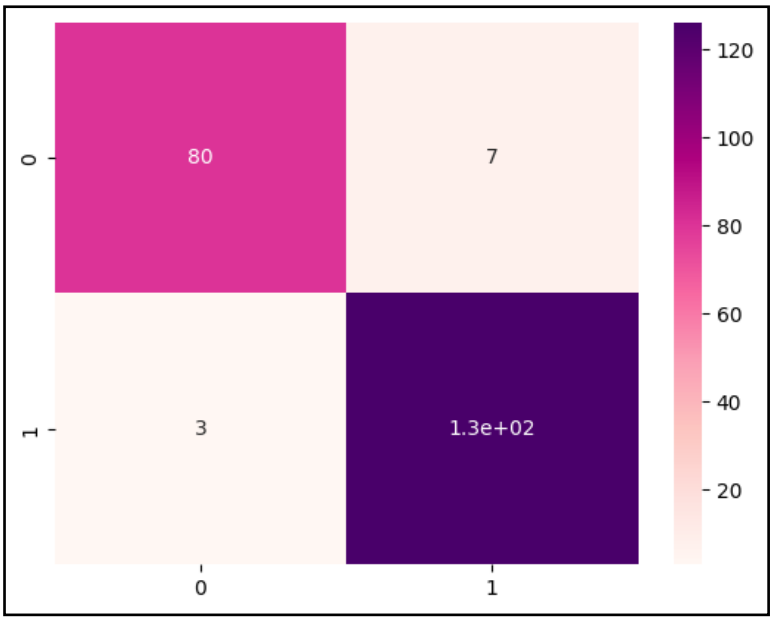

Figure 4. 8 Heatmap of Confusion Matrix (Decision Tree)

4.2.3. Random Forest Classifier

Random search cross-validation maintained its position as the most successful hyperparameter tuning method, this time for the Random Forest classifier. The accuracy increased significantly to 0.976, representing a positive improvement over the original test. This upward trend is further supported by the high F1-score of 0.981, which indicates a well-balanced performance in terms of precision and recall. The algorithm appears to be capable of accurately recognizing both safe and hazardous water samples.

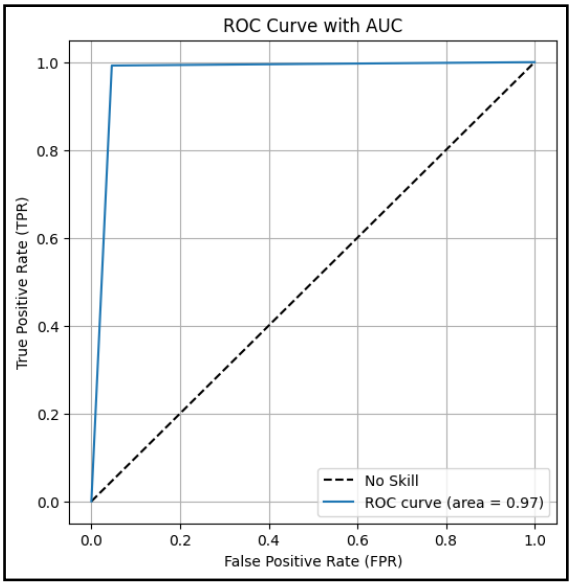

Figure 4. 9 ROC curve with AUC (Random Forest)

To validate this finding, the ROC AUC score is utilized. This measure shows a good capacity to differentiate between potable and non-potable water samples, reaching 0.973 after adjustment shown in Figure 4.9. In an ideal scenario, the confusion matrix (Figure 4.10) would show a decrease from the pre-tuning findings in both false positives (erroneously categorized non-potable samples) and false negatives (missed potable samples). This favourable trend has been confirmed by the given confusion matrix, which shows strong overall performance with 130 true negatives, 4 false positives, 1 false positive and 83 genuine positives.

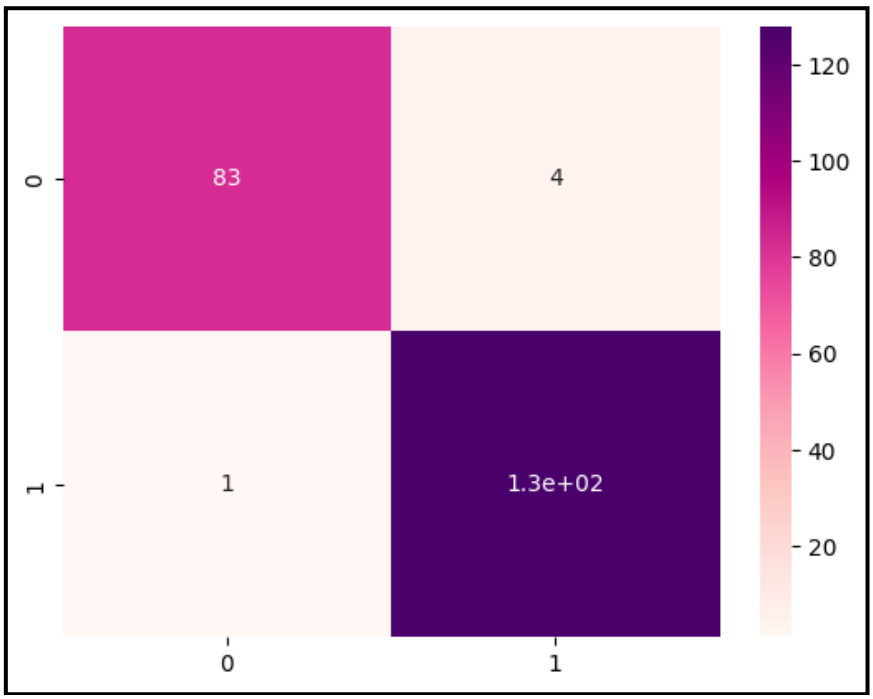

Figure 4. 10 Heatmap of Confusion Matrix (Random Forest)

4.2.4. Gradient Boost Classifier

Using random search cross-validation, the Gradient Boost Classifier replicated the AdaBoost Classifier's improved performance. The remarkable 0.986 accuracy was attained by both models, which also had the same precision (0.992), recall (0.984), F1-score (0.988) and ROC AUC score (0.986). This shows (Figure 4.11) that the Gradient Boost Classifier was able to perform on par with the AdaBoost Classifier following hyperparameter adjustment.

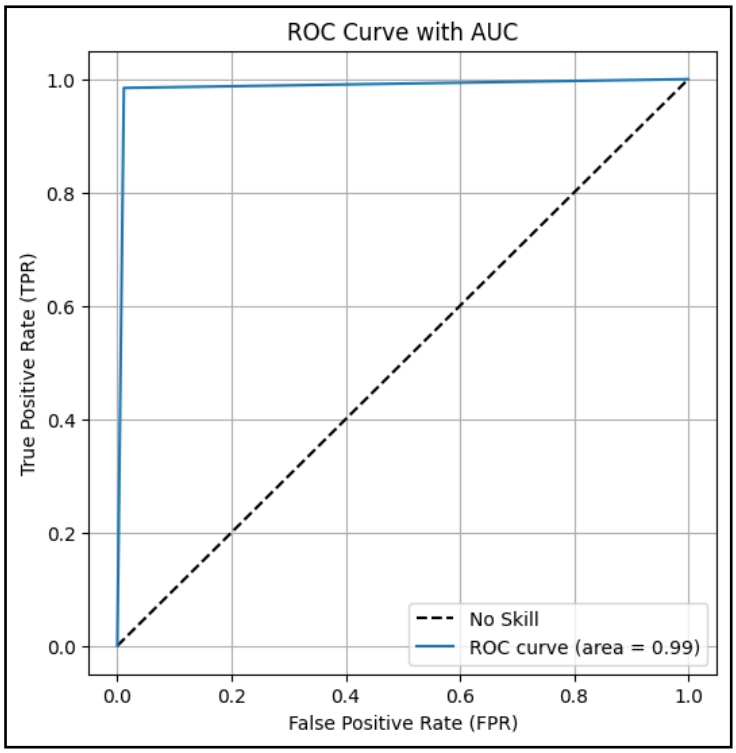

Figure 4. 11 ROC curve with AUC (Gradient Boost)

Examining more closely, the Gradient Boost Classifier's confusion matrix shows excellent outcomes. The model shows exceptional capability in accurately classifying both safe and contaminated water samples, with 86 true positives (correctly classified potable samples), 1 false negative (missed potable sample), 2 false positives (incorrectly classified non-potable samples) and 130 true negatives (correctly classified non-potable samples), as shown in Figure 4.12. This low percentage of misclassification in both classes demonstrates how well hyperparameter adjustment works to optimize the Gradient Boost Classifier for this particular use case.

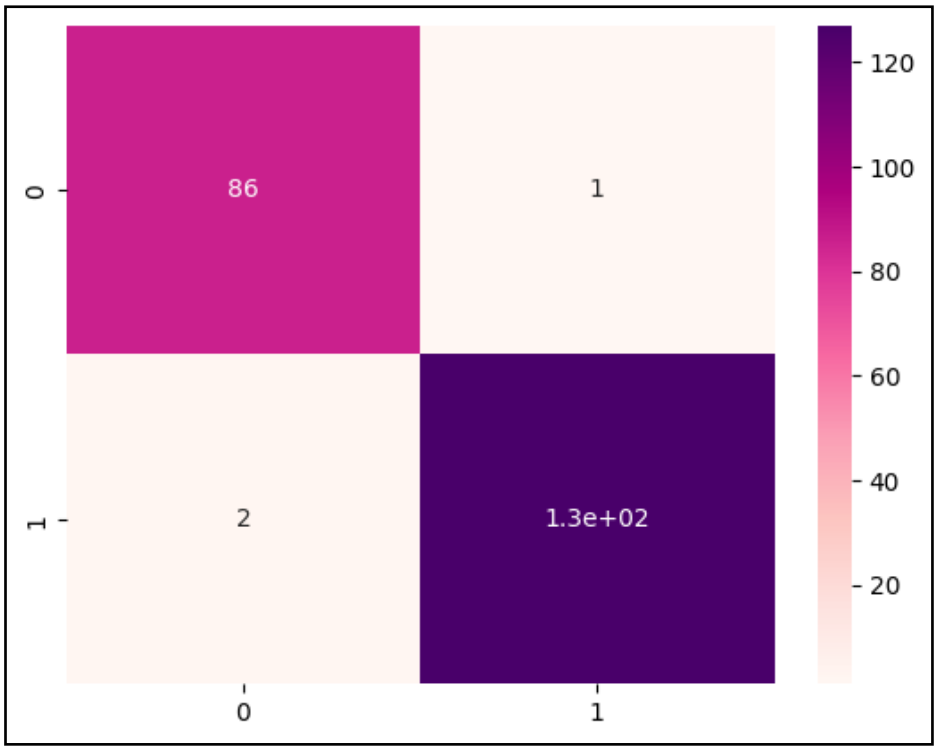

Figure 4. 12 Heatmap of Confusion Matrix (Gradient Boost)

It's crucial to recognize that more research is necessary given the almost comparable performance of the Gradient Boost and AdaBoost classifiers upon adjustment. There may be minor benefits to one model over the other, depending on the particular requirements and goals of the water potability prediction job. For example, the AdaBoost Classifier may be a better option because of its more straightforward structure if computing economy is a priority.

4.3. Summary of Evaluation Results with Different Hyperparameter Tuning Processes

The four machine learning models demonstrated encouraging results in terms of predicting the potability of water. The AdaBoost and Gradient Boost classifiers performed much better when their hyperparameters were tuned; both achieved an accuracy of 0.986 and nearly comparable precision, recall, F1-score and ROC AUC score. Following adjustment, the Random Forest Classifier also performed well, with a high F1-score and an accuracy of 0.976. After adjusting, the Decision Tree Classifier showed a little drop in accuracy but kept a strong F1-score. The results of all models with different hyperparameter tuning processes are shown in Table 4.2.

The AdaBoost Classifier and Gradient Boost Classifier seem to be the best models for predicting water potability in this study when taking into account the overall performance measures. Particularly after hyperparameter adjustment, their remarkable accuracy, precision, recall, F1-score and ROC AUC score demonstrate their potent capacity to discriminate between potable and non-potable water samples. The decision between these two models may be influenced by variables such as the decision tree classifier's interpretability (it may be simpler to comprehend than gradient boosting) or the particular cost of misclassification (such as false negatives for the AdaBoost Classifier). The best model for this application may be determined with the aid of further study or domain expertise.

Table 4. 2 Result of all models with different Hyperparameter Tuning Processes

Score	Ada Boost		Decision Tree		Random Forest		Gradient Boost	
	Grid Search	Random Search	Grid Search	Random Search	Grid Search	Random Search	Grid Search	Random Search
Accuracy	0.979	0.986	0.946	0.953	0.975	0.976	0.983	0.986
Precision	0.983	0.992	0.949	0.947	0.965	0.969	0.979	0.982
Recall	0.993	0.984	0.981	0.976	0.988	0.992	0.988	0.989
F-1	0.986	0.988	0.979	0.981	0.982	0.981	0.989	0.981
Roc-Auc	0.978	0.986	0.948	0.948	0.982	0.973	0.984	0.986

CHAPTER 5

5. ANALYSIS AND DISCUSSION

The four machine learning models used to predict the potability of water the AdaBoost Classifier, Decision Tree Classifier, Random Forest Classifier, and Gradient Boost Classifier are thoroughly analyzed and discussed in this chapter. In the end, the research has established the models' overall efficacy for predicting water potability by comparing different assessment metrics across the models and analysing performance based on actual and anticipated values.

5.1. Performance of ML Models

This chapter examines each model's performance using a sample of real and anticipated water potability values. This study highlights possible areas for development and offers insights into how well each model matches the actual data.

5.1.1. AdaBoost

Based on the comparison between the AdaBoost model's predictions and the actual water potability labels (Predicted: 0: 88, 1: 128; Actual: 1: 129, 0: 87), the analysis has explored the advantages and disadvantages of the model.

The AdaBoost model was successful in detecting safe drinking water, as evidenced by its high prediction accuracy (128 out of 129) for potable water samples. A deeper examination, however, identifies a non-potable sample that was overlooked (one sample that was deemed potable but was indeed infected). Depending on the application's risk tolerance, this can need further study into the precise aspects that may have contributed to the misclassification. These counts of actual and predicted values are shown in Figure 5.1.

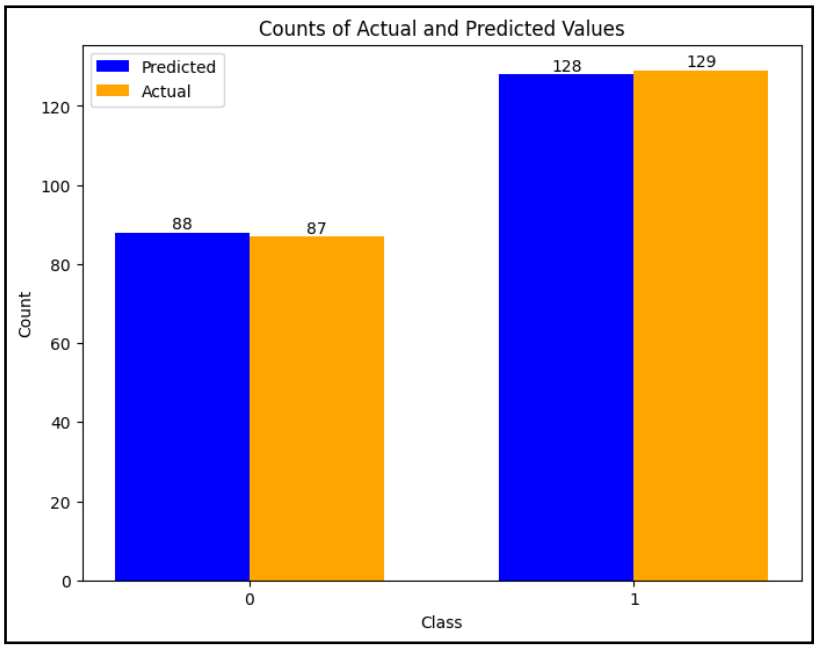

Figure 5. 1 Counts of Actual and Predicted Values (Ada Boost)

5.1.2. Decision Tree

By comparing the predicted values of the Decision Tree with the actual labels (Predicted: 0: 82, 1: 134; Actual: 1: 129, 0: 87), the study analyzed the Decision Tree's performance.

A bias exists in the Decision Tree model's classification of samples as drinkable (134 projected positive). Even if this appears advantageous, it's crucial to take the situation into account. A false negative (missing a polluted sample) might have more serious repercussions than a false positive (non-potable water incorrectly declared OK) in places where access to clean water is scarce. To fully comprehend the Decision Tree model's trade-off between recall and precision, more examination of the confusion matrix would be helpful. These counts of actual and predicted values are shown in Figure 5.2.

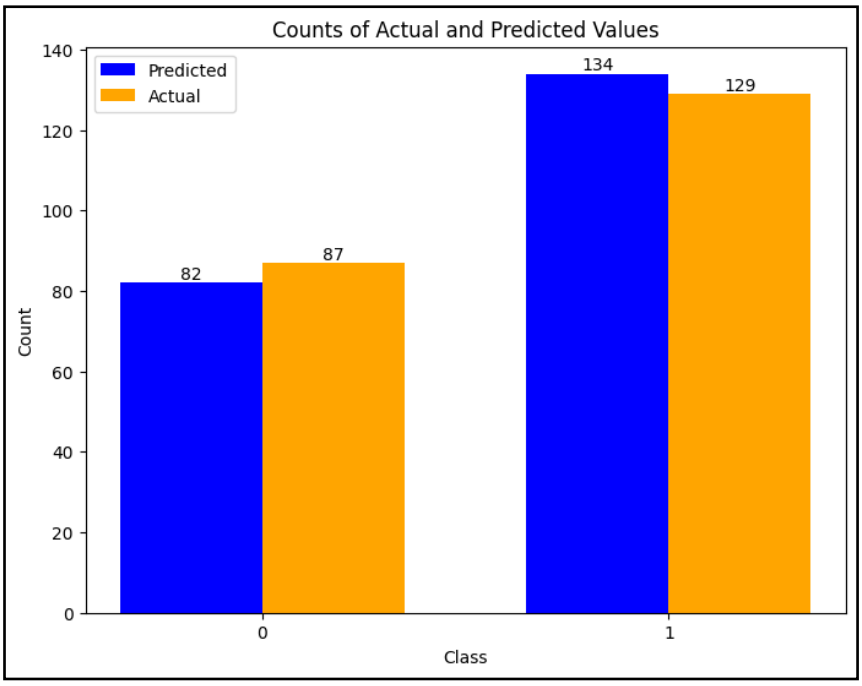

Figure 5. 2 Counts of Actual and Predicted Values (Decision Tree)

5.1.3. Random Forest

Based on the given data, the Random Forest's prediction accuracy is examined in the discussion (Predicted: 0: 84, 1: 132; Actual: 1: 129, 0: 87). Although this first impression points to a sensible balance between classifying water samples as potable or non-potable, a more thorough investigation is required.

It was necessary to compute performance measures like accuracy, precision, and recall on a bigger dataset in order to reach firm conclusions on the Random Forest's generalizability. This thorough examination has been offered a solid grasp of the model's performance in practical water potability prediction tasks. Additionally, it has been demonstrated how well the model manages the trade-off that, depending on the risk tolerance of the application, must be made between false positives, or safe water that is incorrectly categorized as polluted, and false negatives, or contaminated water that is identified as safe. These counts of actual and predicted values are shown in Figure 5.3.

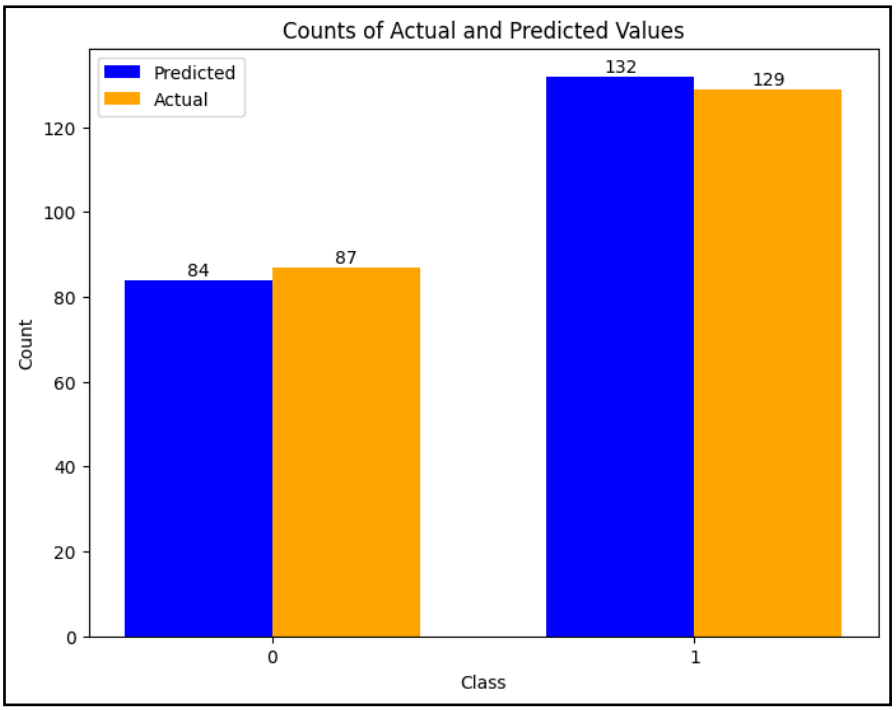

Figure 5. 3 Counts of Actual and Predicted Values (Random Forest)

5.1.4. Gradient Boost

Based on the predicted water potability classifications and the actual labels (Predicted: 0: 85, 1: 131; Actual: 1: 129, 0: 87), the study looks at how well the Gradient Boost model performed. Like AdaBoost, the Gradient Boost model predicts potable water samples with excellent accuracy, indicating that it may be used to detect safe drinking water.

The model's accuracy and capacity to accurately categorize both potable and non-potable water samples have been shown by the confusion matrix. Potential flaws, such as a propensity to label non-potable water as potable (false negatives) or vice versa (false positives), can be found by analysing the confusion matrix. This data is essential for assessing the model's applicability in real-world scenarios, where the context-specific repercussions of misclassification might change. These counts of actual and predicted values are shown in Figure 5.4.

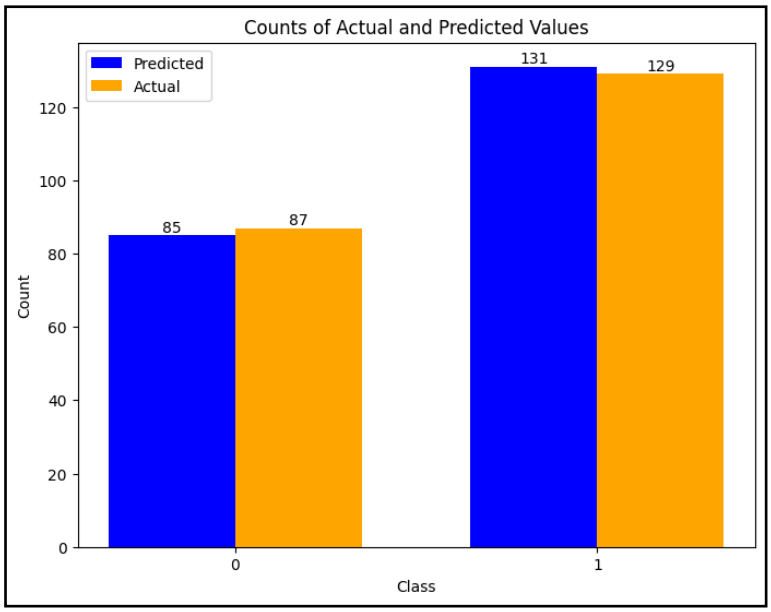

Figure 5. 4 Counts of Actual and Predicted Values (Gradient Boost)

5.2. Comparative Analysis of Performance Metrics

This section will compare the performance metrics of all four models, providing a comprehensive overview of their strengths and limitations:

5.2.1. Accuracy Score

The accuracy ratings of each model (AdaBoost: 0.986, Decision Tree: 0.953, Random Forest: 0.976, Gradient Boost: 0.986) are compared in the analysis, as shown in Figure 5.5. The best accuracy was obtained by AdaBoost and Gradient Boost, demonstrating their prowess in the classification of water samples.

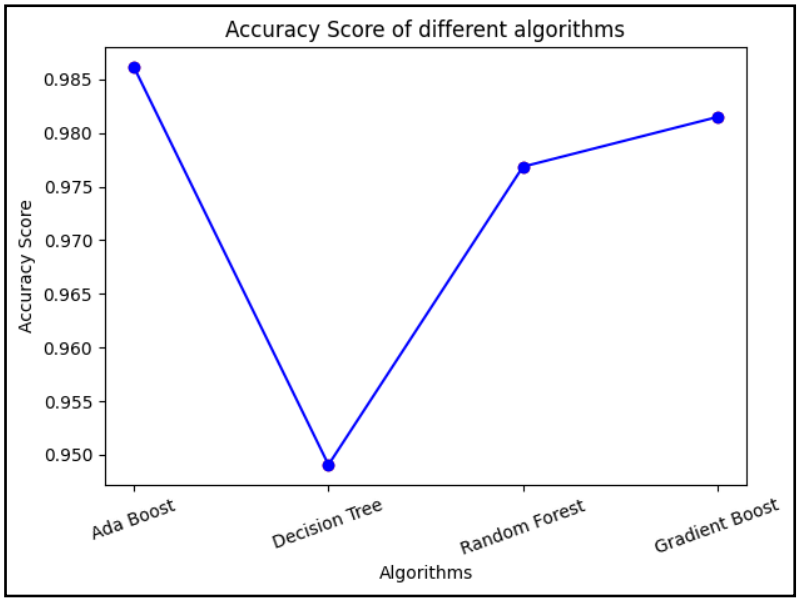

Figure 5. 5 Accuracy Score of different algorithms

But accuracy isn't the only consideration. Certain other metrics may be important, depending on the application. For example, a high recall model would be desirable if reducing false negatives of contaminated water that is classed as safe is important.

5.2.2. Precision Score

Each model's precision score (AdaBoost: 0.992, Decision Tree: 0.947, Random Forest: 0.969, Gradient Boost: 0.982) is examined in detail, as shown in Figure 5.6. Here, AdaBoost excels with the highest precision score, demonstrating its remarkable capacity to correctly identify samples of potable water and reduce false positives (safe water that is inadvertently categorized as polluted).

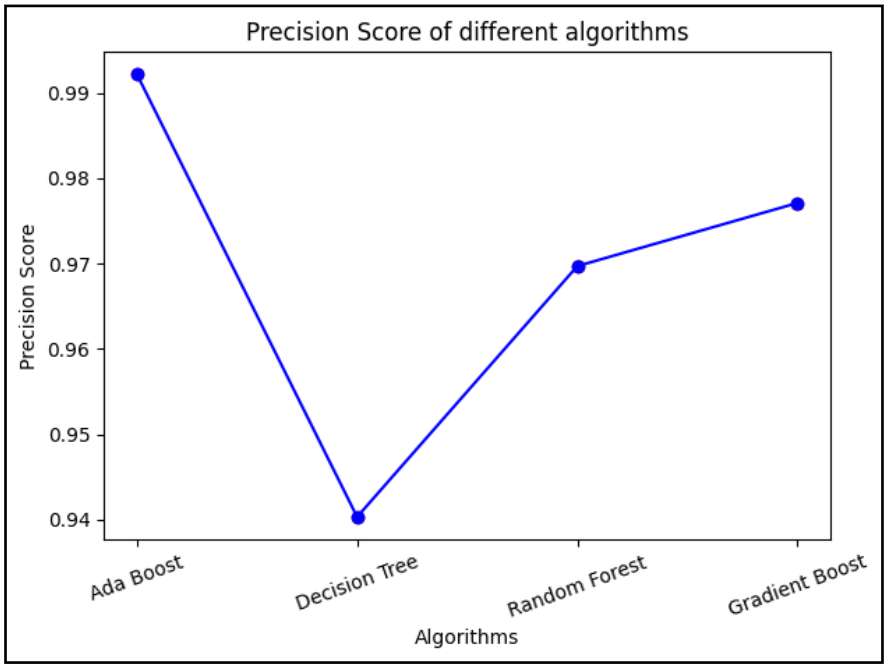

Figure 5. 6 Precision Score of different algorithms

In situations when there is a significant chance of misclassifying clean drinking water, this statistic is very crucial. For example, a false positive might cause people to drink polluted water, which could have serious implications, in places where access to clean water is scarce.

5.2.3. Recall Score

The recall scores of each model AdaBoost: 0.984, Decision Tree: 0.976, Random Forest: 0.992, Gradient Boost: 0.989 are examined in the analysis, as shown in Figure 5.7. In this instance, the Random Forest comes out as having the highest recall, demonstrating its ability to accurately classify potable water samples and detect all true positive cases.

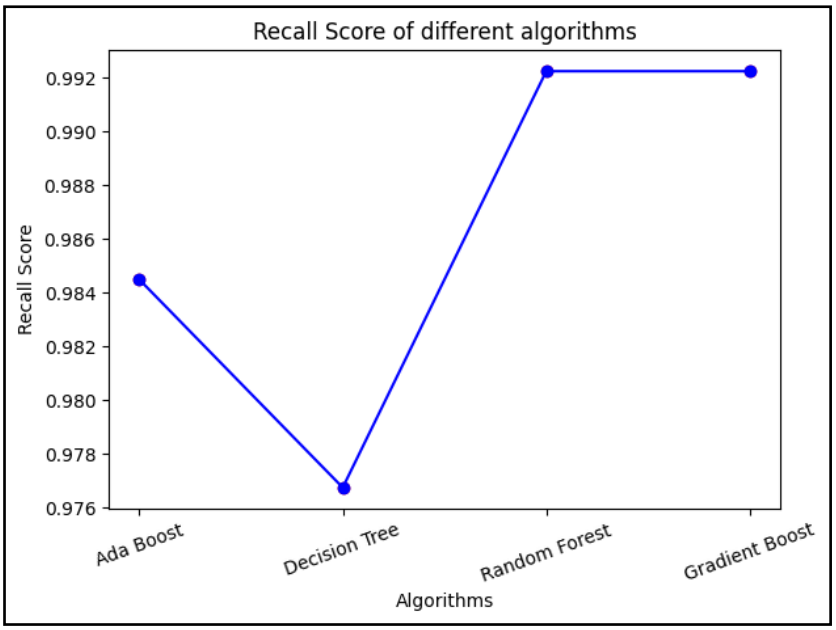

Figure 5. 7 Recall Score of different algorithms

With the best recall score, the Random Forest algorithm is most effective at reducing false negatives, or the mistaken classification of polluted water samples as safe. This might be especially helpful in situations when it would be costly to overlook a tainted sample.

5.2.4. F-1 Score

The F1-score comparison (AdaBoost: 0.988, Decision Tree: 0.981, Random Forest: 0.981, Gradient Boost: 0.986) rounds up the analysis. This statistic provides a fair assessment of a model's efficacy by taking into account both accuracy and recall. The highest F1-scores are here attained by AdaBoost and Gradient Boost, indicating that they successfully detect true positives (properly categorizing potable water samples) while avoiding false positives (safe water that is inadvertently labelled polluted). Values of F-1 Score of different algorithms are shown in Figure 5.8.

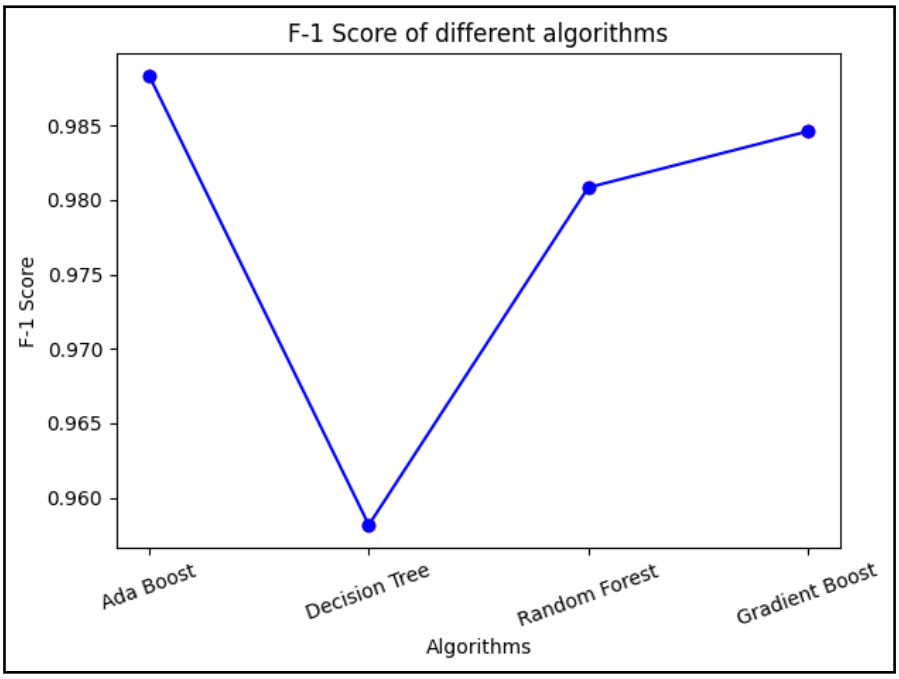

Figure 5. 8 F-1 Score of different algorithms

This is especially important in practical applications because there might be negative outcomes from both over-prediction (designating clean water as dangerous) and under-prediction (missing polluted water). Selecting a model that minimizes both kinds of mistakes is made easier by the F1-score.

5.2.5. ROC AUC Score

A review of the ROC AUC values (AdaBoost: 0.986, Decision Tree: 0.948, Random Forest: 0.973, Gradient Boost: 0.984) brings the investigation to a close. This statistic (in Figure 5.9) assesses how well a model can distinguish between two kinds of data potable and non-potable water in this example. AdaBoost and Gradient Boost have the greatest ROC AUC ratings, much like accuracy. This implies that they have extraordinary capacity to discriminate between the two kinds of water.

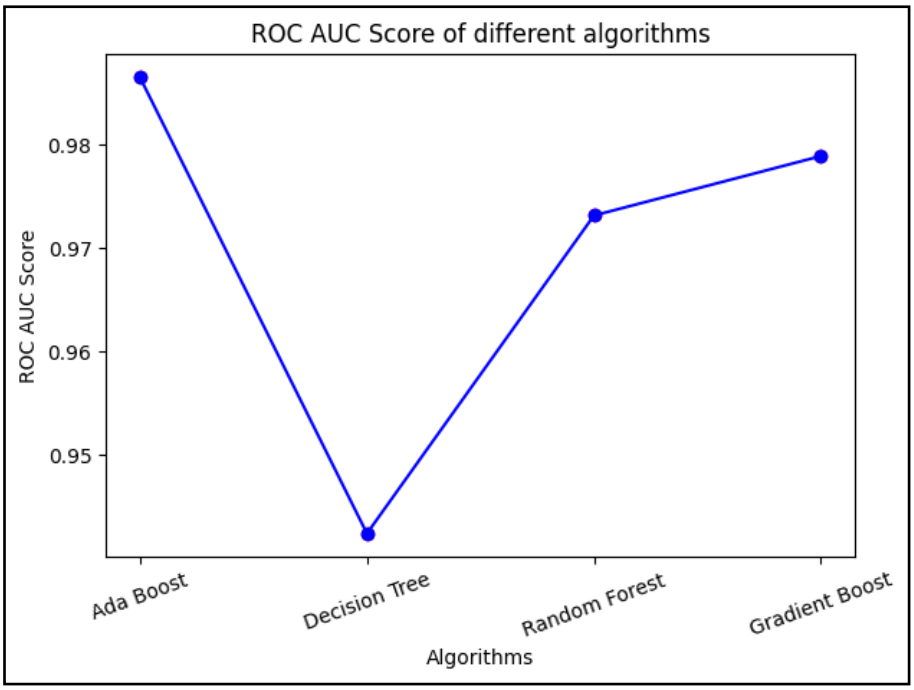

Figure 5. 9 ROC AUC Score of different algorithms

On the other hand, the trade-off between false positives and false negatives is not explained by ROC AUC. Other assessment measures, such as the F1-score, which takes both precision and recall into account, maybe more instructive depending on the application's requirements.

5.3. Comparison of ML Models

AdaBoost, Decision Tree, Random Forest, and Gradient Boost are the four models that are thoroughly compared in this section. The examination takes into account their interpretability, computational efficiency, and performance on a number of measures. Through an analysis of the advantages and disadvantages of each model, the best option for predicting water potability in a given application may be identified. The values of all performance metrics for all algorithms are shown in Table 5.1.

Table 5. 1 Summarizing the key points for comparison

Model	Accuracy	Precision	Recall	F1-Score	ROC AUC Score	Interpretability	Computational Efficiency
AdaBoost	0.986 (Highest)	0.992 (Highest)	0.984	0.988 (Highest)	0.986 (Highest)	Best	Best
Decision Tree	0.953	0.947	0.976	0.981	0.948	Better	Good
Random Forest	0.976	0.969	0.992 (Highest)	0.981	0.973	Good	Better
Gradient Boost	0.986 (Highest)	0.982	0.989	0.986 (Highest)	0.984	Best	Best

Based on the analysis, both AdaBoost and Gradient Boost emerged as the top performers in terms of accuracy, F1-score and ROC AUC score. AdaBoost holds an edge in precision, making it ideal for applications were minimizing false positives (safe water misclassified as contaminated) is crucial. Gradient Boost offers a slight advantage in recall, potentially beneficial when the focus is on identifying all true positives (contaminated water samples).

Decision Trees, while boasting high interpretability, exhibited a lower overall performance compared to AdaBoost and Gradient Boost. This might be a suitable choice if understanding the model's decision-making process is a priority, even if it comes at a slight cost to accuracy.

Random Forest achieves a good balance between precision and recall but falls behind AdaBoost and Gradient Boost in terms of accuracy and ROC AUC score. However, its strength lies in high computational efficiency, making it a viable option for resource-constrained environments.

CHAPTER 6

6. CONCLUSION AND FUTURE SCOPE

6.1. Conclusion

The usefulness of machine learning models for forecasting water potability was examined in this thesis. The AdaBoost, Decision Tree, Random Forest, and Gradient Boost Classifiers were the four well-known models used. During the evaluation, metrics like accuracy, precision, recall, F1-score and ROC AUC score analyzed their performance on a water potability dataset.

The models were slightly overfitted before hyperparameter tuning but performed better after. The best-performing models were the Gradient Boost and AdaBoost Classifiers. AdaBoost achieved an F1-score of 0.988, an accuracy of 0.986, a precision of 0.992 and a recall of 0.984. Similarly, the Gradient Boost Classifier excelled with an accuracy of 0.986, a precision of 0.992, a recall of 0.984 and an F1-score of 0.988.

Based on the analysis, the best algorithms were AdaBoost and Gradient Boost, achieving remarkable accuracy, F1-score, and ROC AUC scores. AdaBoost stood out with the highest precision, making it ideal for applications where reducing false positives for safe water misidentified as contaminated is crucial. Gradient Boost had a slight edge in Recall, beneficial for identifying all true positives (contaminated water samples).

Decision Trees performed worse overall than AdaBoost and Gradient Boost, despite their high interpretability. This might be a good option if understanding the model's decision-making is crucial, even if there is a small accuracy loss. While Random Forest's precision and recall were well-balanced, its accuracy and ROC AUC scores lagged behind the best. Its strong computational efficiency makes it suitable for resource-limited environments.

In summary, AdaBoost and Gradient Boost Classifiers are the best models for predicting water potability, with AdaBoost performing slightly better. Though not as effective, Random Forest and Decision Tree Classifiers still perform admirably and might be useful in specific scenarios. The results of this study offer valuable insights for managing and monitoring water quality, guiding the creation of more accurate and efficient techniques for estimating water potability.

6.2. Future Scope

This thesis provides impressive fresh ideas for research. To improve the generalizability of the model, training datasets should be expanded to include a greater range of water samples from different locations and types of contamination. Combining the advantages of several models into an ensemble approach shows promise for increased reliability and accuracy. Additional performance improvements may be possible through feature engineering, which investigates novel features or selection strategies unique to water potability prediction.

The deployment in the real world is the ultimate goal. User interfaces that are easy to use for data collection, prediction, and result visualization are necessary for the integration of these models into surveillance systems. For implementation to be successful, issues like detector testing, security of data, and real-time decision-making must be resolved.

REFERENCES

[1] S. Patel, K. Shah, S. Vaghela, M. Aglodiya, and R. Bhattad, "Water Potability Prediction Using Machine Learning," 2023, doi: 10.21203/rs.3.rs-2965961/v1.

[2] N. Nasir *et al.*, "Water quality classification using machine learning algorithms," *Journal of Water Process Engineering*, vol. 48, Aug. 2022, doi: 10.1016/j.jwpe.2022.102920.

[3] T. Deng, K. W. Chau, and H. F. Duan, "Machine learning based marine water quality prediction for coastal hydro-environment management," *J Environ Manage*, vol. 284, Apr. 2021, doi: 10.1016/j.jenvman.2021.112051.

[4] S. Singha, S. Pasupuleti, S. S. Singha, R. Singh, and S. Kumar, "Prediction of groundwater quality using efficient machine learning technique," *Chemosphere*, vol. 276, Aug. 2021, doi: 10.1016/j.chemosphere.2021.130265.

[5] A. El Bilali, A. Taleb, and Y. Brouziyne, "Groundwater quality forecasting using machine learning algorithms for irrigation purposes," *Agric Water Manag*, vol. 245, Feb. 2021, doi: 10.1016/j.agwat.2020.106625.

[6] O. O. Ajayi, A. B. Bagula, H. C. Maluleke, Z. Gaffoor, N. Jovanovic, and K. C. Pietersen, "WaterNet: A Network for Monitoring and Assessing Water Quality for Drinking and Irrigation Purposes," *IEEE Access*, vol. 10, pp. 48318–48337, 2022, doi: 10.1109/ACCESS.2022.3172274.

[7] Naoki. Abe, *2018 IEEE International Conference on Big Data : proceedings : Dec 10 - Dec 13, 2018, Seattle, WA, USA*. IEEE, 2018.

[8] K. S. Prathibha, R. K. N. Kumar, R. S. Joseph, and S. Subramani, "Predicting the parameters of water quality and calculating the Water Quality Index of Ulsoor Lake, Bangalore, India using Deep Learning Techniques," in *Proceedings - IEEE International Conference on Advances in Computing, Communication and Applied Informatics, ACCAI 2022*, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ACCAI53970.2022.9752559.

[9] S. Babu, B. B. Nagaleela, C. G. Karthik, and L. N. Yepuri, "Water Quality Prediction using Neural Networks," in *Proceedings of the International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering, ICECONF 2023*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ICECONF57129.2023.10084120.

[10] S. Kolli, M. Ranjani, P. Kavitha, D. A. P. Daniel, and A. Chandramauli, "Prediction of water quality parameters by IoT and machine learning," in *2023 International Conference on Computer Communication and Informatics, ICCCI 2023*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ICCCI56745.2023.10128475.

[11] *RTEICT 2019 : 2019 4th IEEE International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT) : 2019 proceedings : Bengaluru, Karnataka, India, 17-18 May, 2019*. IEEE, 2019.

[12] S. K. Gandhimathi, G. K. Muppagowni, K. Sandhya, R. R. Arava, K. Vindhya Anuragh, and G. Kotapati, "A Proficient Prediction Mechanism for Analyzing Water Quality Using Machine Learning Algorithms," in *International Conference on Self Sustainable Artificial Intelligence Systems, ICSSAS 2023 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 552–559. doi: 10.1109/ICSSAS57918.2023.10331622.

[13] E. Kuruvilla and S. Kundapura, "Performance Comparison of Machine Learning Algorithms in Groundwater Potability Prediction," in *7th IEEE International Conference on Recent Advances and Innovations in Engineering, ICRAIE 2022 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 53–58. doi: 10.1109/ICRAIE56454.2022.10054298.

[14] S. Suma, R. Moon, M. Umer, K. S. Raju, N. Bhaskar, and R. Okali, "A Prediction of Water Quality Analysis Using Machine Learning," in *2nd IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics, ICDCECE 2023*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ICDCECE57866.2023.10150940.

[15] R. Akshay, G. Tarun, P. U. Kiran, K. D. Devi, and M. Vidhyalakshmi, "Water-Quality-Analysis using Machine Learning," in *Proceedings of the 2022 11th International Conference on System Modeling and Advancement in Research Trends, SMART 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 13–18. doi: 10.1109/SMART55829.2022.10047533.

[16] P. Rawat, M. Bajaj, V. Sharma, and S. Vats, "A Comprehensive Analysis of the Effectiveness of Machine Learning Algorithms for Predicting Water Quality," in *International Conference on Innovative Data Communication Technologies and Application, ICIDCA 2023 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 1108–1114. doi: 10.1109/ICIDCA56705.2023.10099968.

[17] S. A. Ansari, C. Sharma, and T. Agarwal, "Mean and Prediction Imputation-Based Approach for Predicting Water Potability Using Machine Learning," in *2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions), ICRITO 2022*, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ICRITO56286.2022.9964809.

[18] F. Akhter, H. R. Siddiquei, M. E. E. Alahi, K. P. Jayasundera, and S. C. Mukhopadhyay, "An IoT-Enabled Portable Water Quality Monitoring System With MWCNT/PDMS Multifunctional Sensor for Agricultural Applications," *IEEE Internet Things J*, vol. 9, no. 16, pp. 14307–14316, Aug. 2022, doi: 10.1109/JIOT.2021.3069894.

[19] R. Alnaqeb, F. Alrashdi, K. Alketbi, and H. Ismail, "Machine Learning-based Water Potability Prediction," in *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA*, IEEE Computer Society, 2022. doi: 10.1109/AICCSA56895.2022.10017579.

[20] B. Sakaa *et al.*, "Water quality index modeling using random forest and improved SMO algorithm for support vector machine in Saf-Saf river basin," *Environmental Science and Pollution Research*, vol. 29, no. 32, pp. 48491–48508, Jul. 2022, doi: 10.1007/s11356-022-18644-x.

[21] Md. M. Hassan *et al.*, "Efficient Prediction of Water Quality Index (WQI) Using Machine Learning Algorithms," *Human-Centric Intelligent Systems*, vol. 1, no. 3–4, p. 86, 2021, doi: 10.2991/hcis.k.211203.001.

[22] A. Juna *et al.*, "Water Quality Prediction Using KNN Imputer and Multilayer Perceptron," *Water (Switzerland)*, vol. 14, no. 17, Sep. 2022, doi: 10.3390/w14172592.

[23] K. P. Rasheed Abdul Haq and V. P. Harigovindan, "Water Quality Prediction for Smart Aquaculture Using Hybrid Deep Learning Models," *IEEE Access*, vol. 10, pp. 60078–60098, 2022, doi: 10.1109/ACCESS.2022.3180482.

[24] A. O. Alnahit, A. K. Mishra, and A. A. Khan, "Stream water quality prediction using boosted regression tree and random forest models," *Stochastic Environmental Research and Risk Assessment*, vol. 36, no. 9, pp. 2661–2680, Sep. 2022, doi: 10.1007/s00477-021-02152-4.

[25] J. R. Vilupuru, D. C. Amuluru, and K. Ghousiya Begum, "Water Quality Analysis using Artificial Intelligence Algorithms," in *4th International Conference on Inventive Research in Computing Applications, ICIRCA 2022 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 1193–1199. doi: 10.1109/ICIRCA54612.2022.9985650.

[26] S. Wang, H. Peng, and S. Liang, "Prediction of estuarine water quality using interpretable machine learning approach," *J Hydrol (Amst)*, vol. 605, Feb. 2022, doi: 10.1016/j.jhydrol.2021.127320.

[27] Géron, A., Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 3rd ed. Sebastopol, CA: O'Reilly Media, 2022.

APPENDIX

The following codes have been written in Python and executed using Google Colab for this thesis work.

Importing Libraries

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

Load the Dataset

```
# Read the Dataset
df = pd.read_csv('DrinkingWater_Final_Dataset.csv')
# Print the shape of dataset
df.shape
df
df.head(5)
```

Feature Engineering

1. **Checking null values**
2. **Statistical Description**
3. **Data-Correlation**
4. **Dropping unnecessary column(Id)**

```
df.isnull().sum()
df.describe()
df = df.drop(['id'], axis = 1)
#Correlation of Water Potability using sns
sns.heatmap(df.corr(), cmap = 'coolwarm', annot = True, fmt = '.0%')
plt.title("Correlation of Water Potability")
plt.show()
df.corr()
df = df.drop(['WQI'], axis = 1)
df
```

Data Visualization

```
# Check distribution of each feature with Hist Plot
plt.figure(figsize=(16, 9))
plt.subplots_adjust(wspace=0.3, hspace=0.4)
# Loop through each column in the dataset
o = 1
for i, col in enumerate(df.columns):
    plt.subplot(4, 4, o)
    sns.histplot(data=df, x=col, kde=True, color='skyblue', edgecolor='black', bins=20)
    plt.title(f'Distribution of {col}')
    o += 1
# Display the plots
plt.tight_layout()
plt.show()

sns.pairplot(df,hue = 'Potability',palette = 'Set1')


# Target distribution
plt.figure(figsize=(6,6))
# Pie plot
df['Potability'].value_counts().plot.pie(explode=[0.1,0.1],
                autopct='%1.1f%%', shadow=True,
                textprops={'fontsize':16}).set_title("Target distribution");
```

Splitting the data into train and test

```
from sklearn.model_selection import train_test_split
# Define the feature and target variable names
features = df.columns.tolist()  # Get all features except the last column
target = features.pop()  # Remove the last column (target) from the feature list
# Split the data into features (X) and target variable (y)
X = df[features]
y = df[target]
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# Print the shapes of the resulting data splits
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

Scaling the data

```
# Adjust for Normalization if needed
normalization = MinMaxScaler()
normalization.fit(X_train)
# Transform training data
X_train = normalization.transform(X_train)
# Transform testing data (using the same fitted scaler)
X_test = normalization.transform(X_test)
```

Model import

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier,
AdaBoostClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

Defining Models

```
# models/ Algorithms
def models(X_train_scaled,Y_train):
        abc = AdaBoostClassifier()
        abc.fit(X_train, y_train)
        y_abc = abc.predict(X_test)
        dtc = DecisionTreeClassifier()
        dtc.fit(X_train_scaled, y_train)
        y_dtc = dtc.predict(X_test)
        rfc = RandomForestClassifier()
        rfc.fit(X_train_scaled, y_train)
        y_rfc = rfc.predict(X_test)
        gbc = GradientBoostingClassifier()
        gbc.fit(X_train_scaled, y_train)
        y_gbc = gbc.predict(X_test)
        return abc,dtc,rfc,gbc
```

```
def evaluate_models(X_train, X_test, y_train, y_test):
  models = {
     "Ada Boosting": AdaBoostClassifier(),
     "Random Forest": RandomForestClassifier(),
     "Decision Tree": DecisionTreeClassifier(),
     "Gradient Boosting": GradientBoostingClassifier()
  }
  results = []
  for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    results.append({
       "Model": model_name,
       "Accuracy": accuracy_score(y_test, y_pred),
       "Precision": precision_score(y_test, y_pred),
       "Recall": recall_score(y_test, y_pred),
       "F1-Score": f1_score(y_test, y_pred),
    })
  return pd.DataFrame(results)
results_df = evaluate_models(X_train, X_test, y_train, y_test)
print("Performance Metrics:")
print(results_df.to_string())


# Performance metrics
accuracy = 0.9814
recall = 0.9767
precision = 0.9921
f1 = 0.9843
# Create a DataFrame with the metrics
metrics_df = pd.DataFrame({
   'Metric': ['Accuracy', 'Precision', 'Recall', 'F1-Score'],
   'Score': [accuracy, precision, recall, f1]
})
# Set the index to the Metric column
metrics_df.set_index('Metric', inplace=True)
# Create a heatmap
plt.figure(figsize=(8, 2))  # Adjust the size as needed
ax = sns.heatmap(metrics_df.T, annot=True, cmap='Blues', cbar=True, fmt=".4f")
# Title
ax.set_title('Ada Boosting Performance Metrics', fontsize=16)
ax.set_yticklabels([])  # Hide y-axis labels
# Display the heatmap
plt.show()
import matplotlib.pyplot as plt
```

```
import seaborn as sns
import pandas as pd
# Performance metrics
accuracy = 0.9629
recall = 0.9689
precision = 0.9639
f1 = 0.9769
# Create a DataFrame with the metrics
metrics_df = pd.DataFrame({
   'Metric': ['Accuracy', 'Precision', 'Recall', 'F1-Score'],
   'Score': [accuracy, precision, recall, f1]
})
# Set the index to the Metric column
metrics_df.set_index('Metric', inplace=True)
# Create a heatmap
plt.figure(figsize=(8, 2))  # Adjust the size as needed
ax = sns.heatmap(metrics_df.T, annot=True, cmap='Blues', cbar=True, fmt=".4f")
# Title
ax.set_title('Decision Tree Performance Metrics', fontsize=16)
ax.set_yticklabels([])  # Hide y-axis labels
# Display the heatmap
plt.show()


import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Performance metrics
accuracy = 0.9630
recall = 0.9618
precision = 0.9767
f1 = 0.9692
# Create a DataFrame with the metrics
metrics_df = pd.DataFrame({
   'Metric': ['Accuracy', 'Precision', 'Recall', 'F1-Score'],
   'Score': [accuracy, precision, recall, f1]
})
# Set the index to the Metric column
metrics_df.set_index('Metric', inplace=True)
# Create a heatmap
plt.figure(figsize=(8, 2))  # Adjust the size as needed
ax = sns.heatmap(metrics_df.T, annot=True, cmap='Blues', cbar=True, fmt=".4f")
ax.set_title('Random Forest Performance Metrics', fontsize=16)
ax.set_yticklabels([])  # Hide y-axis labels
# Display the heatmap
plt.show()
```

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Performance metrics
accuracy = 0.9861
recall = 0.9844
precision = 0.9921
f1 = 0.9883
# Create a DataFrame with the metrics
metrics_df = pd.DataFrame({
   'Metric': ['Accuracy', 'Precision', 'Recall', 'F1-Score'],
   'Score': [accuracy, precision, recall, f1]
})
# Set the index to the Metric column
metrics_df.set_index('Metric', inplace=True
# Create a heatmap
plt.figure(figsize=(8, 2))  # Adjust the size as needed
ax = sns.heatmap(metrics_df.T, annot=True, cmap='Blues', cbar=True, fmt=".4f")
# Title
ax.set_title('Gradient Boost Performance Metrics', fontsize=16)
ax.set_yticklabels([])  # Hide y-axis labels
# Display the heatmap
plt.show()
```

Accuracy Scores

```
# Plotting the accuracy scores
bar_width = 0.35
plt.figure(figsize=(10, 6))
x = [i - bar_width/3 for i, _ in enumerate(results_df['Model'])]  # Reduce gap between bars
bars = plt.bar(x, results_df['Accuracy'], color=['blue', 'green', 'red', 'magenta'],
width=bar_width)
plt.xlabel('Models')
plt.ylabel('Accuracy Score')
plt.title('Accuracy Scores of Different Models')
plt.ylim(0, 1)
# Adjust x-axis ticks to match bar positions
plt.xticks(x, results_df['Model'], rotation=0)
for bar in bars:
   yval = bar.get_height()
   plt.text(bar.get_x() + bar_width/2, yval - 0.05, f'{yval:.3f}', ha='center', va='top',
color='white')
plt.tight_layout()  # Adjust spacing to avoid clipping labels
plt.show()
```

Precision Scores

```
# Plotting the Precision scores
bar_width = 0.35
plt.figure(figsize=(10, 6))
x = [i - bar_width/3 for i, _ in enumerate(results_df['Model'])]  # Reduce gap between bars
bars = plt.bar(x, results_df['Precision'], color=['blue', 'green', 'red', 'magenta'],
width=bar_width)
plt.xlabel('Models')
plt.ylabel('Precision Score')
plt.title('Precision Scores of Different Models')
plt.ylim(0, 1)
# Adjust x-axis ticks to match bar positions
plt.xticks(x, results_df['Model'], rotation=0)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar_width/2, yval - 0.05, f'{yval:.3f}', ha='center', va='top',
color='white')
plt.tight_layout()  # Adjust spacing to avoid clipping labels
plt.show()
```

Recall Scores

```
# Plotting the Recall scores
bar_width = 0.35
plt.figure(figsize=(10, 6))
x = [i - bar_width/3 for i, _ in enumerate(results_df['Model'])]  # Reduce gap between bars
bars = plt.bar(x, results_df['Recall'], color=['blue', 'green', 'red', 'magenta'], width=bar_width)
plt.xlabel('Models')
plt.ylabel('Recall Score')
plt.title('Recall Scores of Different Models')
plt.ylim(0, 1)
# Adjust x-axis ticks to match bar positions
plt.xticks(x, results_df['Model'], rotation=0)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar_width/2, yval - 0.05, f'{yval:.3f}', ha='center', va='top',
color='white')
plt.tight_layout()  # Adjust spacing to avoid clipping labels
plt.show()
```

F1-Score

```
# Plotting the F1-Score
bar_width = 0.35
plt.figure(figsize=(10, 6))
x = [i - bar_width/3 for i, _ in enumerate(results_df['Model'])]  # Reduce gap between bars
bars = plt.bar(x, results_df['F1-Score'], color=['blue', 'green', 'red', 'magenta'],
width=bar_width)
plt.xlabel('Models')
plt.ylabel('F1-Score Score')
plt.title('F1-Score Scores of Different Models')
plt.ylim(0, 1)
# Adjust x-axis ticks to match bar positions
plt.xticks(x, results_df['Model'], rotation=0)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar_width/2, yval - 0.05, f'{yval:.3f}', ha='center', va='top',
color='white')
plt.tight_layout()  # Adjust spacing to avoid clipping labels
plt.show()
```

Hyperparameter tuning

1. Ada Boost Classifier

Using Grid Search CV Technique

```
from sklearn import metrics
from sklearn.model_selection import GridSearchCV


abc = AdaBoostClassifier()

abc.get_params()

scaler = StandardScaler()  # Adjust for Standardization if needed
scaler.fit(X_train)
# Transform training data
X_train = scaler.transform(X_train)
# Transform testing data (using the same fitted scaler)
X_test = scaler.transform(X_test)

from sklearn.model_selection import GridSearchCV
param_grid = {
  'n_estimators': [50, 100, 200, 500]  # Max depth of decision trees (weak learners)
}
abc_grid = GridSearchCV(abc, param_grid, cv=5)
# fitting the model for grid search
abc_grid.fit(X_train, y_train)
```

```
best_params = abc_grid.best_params_  # Get the best hyperparameter combination found by GridSearchCV
best_estimator = abc_grid.best_estimator_  # Get the SVC model instance with the best hyperparameters

best_params
best_estimator

print("Accuracy: ",accuracy_score(y_test,abc_grid.predict(X_test)))
print("Precision: ",precision_score(y_test,abc_grid.predict(X_test)))
print("Recall: ",recall_score(y_test,abc_grid.predict(X_test)))
print("F1-Score: ",f1_score(y_test,abc_grid.predict(X_test)))

# Use the best model for prediction
y_pred = best_estimator.predict(X_test)
# Evaluate the performance of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

from sklearn.metrics import roc_auc_score
auc_score = roc_auc_score(y_test, y_pred)
print("AUC Score:", auc_score)

import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve
plt.figure(figsize = (6, 6))
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr, label = "")
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.show()

from sklearn.metrics import confusion_matrix
sns.heatmap(confusion_matrix(y_test, y_pred), cmap = 'RdPu', annot = True)
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
plt.figure(figsize=(6, 6))  # Maintain figure size
# Assuming you have y_test (true labels) and y_pred (predicted probabilities)
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
# Calculate AUC
roc_auc = auc(fpr, tpr)
# Plot ROC curve
plt.plot([0, 1], [0, 1], 'k--', label='No Skill')  # Diagonal line for reference
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % roc_auc)  # Plot with AUC label
# Add labels and title
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('ROC Curve with AUC')
# Add legend
plt.legend(loc="lower right")
# Add grid lines for better readability
plt.grid(True)
plt.show()

from sklearn.metrics import precision_recall_curve
import matplotlib.pyplot as plt
# Calculate precision, recall at different thresholds
precision, recall, thresholds = precision_recall_curve(y_test, y_pred)
# Plot precision-recall curve
plt.figure()
plt.step(recall, precision, where='post')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
plt.title('Precision-Recall Curve')
plt.grid(True)
plt.show()
```

Using Random Search Technique

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import uniform, randint

param_dist = {
  'n_estimators': range(50, 501, 50)  # Max depth of trees (uniform distribution)
}
# Create the AdaBoostClassifier model
abc = AdaBoostClassifier()
abc_random_search = RandomizedSearchCV(abc, param_dist, cv=5, n_iter=100)
```

```
abc_random_search.fit(X_train, y_train)  # Fit the RandomizedSearchCV to your training data (X_train and y_train)

best_params = abc_random_search.best_params_  # Get the best hyperparameter combination found by RandomizedSearchCV
best_estimator = abc_random_search.best_estimator_  # Get the SVC model instance with the best hyperparameters

best_params
best_estimator

print("Accuracy: ",accuracy_score(y_test,abc_random_search.predict(X_test)))
print("Precision: ",precision_score(y_test,abc_random_search.predict(X_test)))
print("Recall: ",recall_score(y_test,abc_random_search.predict(X_test)))
print("F1-Score: ",f1_score(y_test,abc_random_search.predict(X_test)))

# Use the best model for prediction
y_pred = best_estimator.predict(X_test)
# Evaluate the performance of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

from sklearn.metrics import roc_auc_score
auc_score = roc_auc_score(y_test, y_pred)
print("AUC Score:", auc_score)

import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
plt.figure(figsize=(6, 6))  # Maintain figure size
# Assuming you have y_test (true labels) and y_pred (predicted probabilities)
fpr, tpr, thresholds = roc_curve(y_test, y_pred
# Calculate AUC
roc_auc = auc(fpr, tpr)
# Plot ROC curve
plt.plot([0, 1], [0, 1], 'k--', label='No Skill')  # Diagonal line for reference
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % roc_auc)  # Plot with AUC label
# Add labels and title
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('ROC Curve with AUC')
# Add legend
plt.legend(loc="lower right"
# Add grid lines for better readability
plt.grid(True)
plt.show()
```

```
from sklearn.metrics import confusion_matrix
sns.heatmap(confusion_matrix(y_test, y_pred), cmap = 'RdPu', annot = True)

from sklearn.metrics import precision_recall_curve
import matplotlib.pyplot as plt
# Calculate precision, recall at different thresholds
precision, recall, thresholds = precision_recall_curve(y_test, y_pred)
# Plot precision-recall curve
plt.figure()
plt.step(recall, precision, where='post')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
plt.title('Precision-Recall Curve')
plt.grid(True)
plt.show()
```

Comparision of different accuracy score obtained from different algorithms after Hyperparameter Tuning

Accuracy Score

```
Algorithms=['Ada Boost', 'Decision Tree','Random Forest','Gradient Boost']
Accuracy_Score=[accuracy_score(y_test,abc_random_search.predict(X_test)),
        accuracy_score(y_test,dtc_random_search.predict(X_test)),
        accuracy_score(y_test,rfc_random_search.predict(X_test)),
        accuracy_score(y_test,gbc_random_search.predict(X_test))]
plt.figure()
# Line chart for overall trend
plt.plot(Algorithms, Accuracy_Score, marker='o', linestyle='-', color='blue', label='ROC AUC
Score')
# Scatter plot for individual data points
plt.scatter(Algorithms, Accuracy_Score, color='red', alpha=1.0)
plt.xlabel('Algorithms')
plt.ylabel('Accuracy Score')
plt.title('Accuracy Score of different algorithms')
plt.xticks(rotation=20)
plt.tight_layout()
plt.show()
```

Precision Score

```
Algorithms=['Ada Boost', 'Decision Tree','Random Forest','Gradient Boost']
Precision_Score=[precision_score(y_test,abc_random_search.predict(X_test)),
                 precision_score(y_test,dtc_random_search.predict(X_test)),
                 precision_score(y_test,rfc_random_search.predict(X_test)),
                 precision_score(y_test,gbc_random_search.predict(X_test))]
plt.figure()
# Line chart for overall trend
plt.plot(Algorithms, Precision_Score, marker='o', linestyle='-', color='blue', label='ROC AUC
Score')
# Scatter plot for individual data points
plt.scatter(Algorithms, Precision_Score, color='red', alpha=1.0)
plt.xlabel('Algorithms')
plt.ylabel('Precision Score')
plt.title('Precision Score of different algorithms')
plt.xticks(rotation=20)
plt.tight_layout()
plt.show()
Recall Score
Algorithms=['Ada Boost', 'Decision Tree','Random Forest','Gradient Boost']
Recall_Score=[recall_score(y_test,abc_random_search.predict(X_test)),
              recall_score(y_test,dtc_random_search.predict(X_test)),
              recall_score(y_test,rfc_random_search.predict(X_test)),
              recall_score(y_test,gbc_random_search.predict(X_test))]
plt.figure()
plt.plot(Algorithms, Recall_Score, marker='o', linestyle='-', color='blue', label='ROC AUC
Score')
# Scatter plot for individual data points
plt.scatter(Algorithms, Recall_Score, color='red', alpha=1.0)
plt.xlabel('Algorithms')
plt.ylabel('Recall Score')
plt.title('Recall Score of different algorithms')
plt.xticks(rotation=20)
plt.tight_layout()
plt.show()
```

F-1 Score

```
Algorithms=['Ada Boost', 'Decision Tree','Random Forest','Gradient Boost']
F1_Score=[f1_score(y_test,abc_random_search.predict(X_test)),
          f1_score(y_test,dtc_random_search.predict(X_test)),
          f1_score(y_test,rfc_random_search.predict(X_test)),
          f1_score(y_test,gbc_random_search.predict(X_test))]
plt.figure()
plt.plot(Algorithms, F1_Score, marker='o', linestyle='-', color='blue', label='ROC AUC
Score')
# Scatter plot for individual data points
plt.scatter(Algorithms, F1_Score, color='red', alpha=1.0)
plt.xlabel('Algorithms')
plt.ylabel('F-1 Score')
plt.title('F-1 Score of different algorithms')
plt.xticks(rotation=20)
plt.tight_layout()
plt.show()
ROC AUC Score
Algorithms=['Ada Boost', 'Decision Tree','Random Forest','Gradient Boost']
ROC_AUC_Score=[roc_auc_score(y_test,abc_random_search.predict(X_test)),
          roc_auc_score(y_test,dtc_random_search.predict(X_test)),
          roc_auc_score(y_test,rfc_random_search.predict(X_test)),
          roc_auc_score(y_test,gbc_random_search.predict(X_test))]
plt.figure()
plt.plot(Algorithms, ROC_AUC_Score, marker='o', linestyle='-', color='blue', label='ROC
AUC Score')
# Scatter plot for individual data points
plt.scatter(Algorithms, ROC_AUC_Score, color='red', alpha=1.0)
plt.xlabel('Algorithms')
plt.ylabel('ROC AUC Score')
plt.title('ROC AUC Score of different algorithms')
plt.xticks(rotation=20)
plt.tight_layout()
plt.show()
```

Load the Models and Showing Actual vs Predicted Values

import pickle

Ada Boost

```
with open('abc_random_search.pkl', 'wb') as file:
    pickle.dump(abc_random_search, file)

with open('abc_random_search.pkl', 'rb') as file:
    loaded_abc_model = pickle.load(file)

# prediction of ABC
pred_abc=abc_random_search.predict(X_test)
value_counts = {}
for element in pred_abc:
  if element in value_counts:
    value_counts[element] += 1
  else:
    value_counts[element] =
# Print the value counts
print("Value Counts:")
for element, count in value_counts.items():
  print(f"{element}: {count}")
y_test.value_counts()
```

```
import matplotlib.pyplot as plt
# Predicted values
pred_counts = [value_counts.get(0, 0), value_counts.get(1, 0)]
# Actual values
actual_counts = [87, 129]  # These values are from your y_test.value_counts()
labels = ['0', '1']
# Plotting
plt.figure(figsize=(8, 6))
bar_width = 0.35
index = range(len(labels)
# Plot for predicted values
bars_pred = plt.bar(index, pred_counts, bar_width, label='Predicted', color='blue')
# Plot for actual values
bars_actual = plt.bar([i + bar_width for i in index], actual_counts, bar_width, label='Actual',
color='orange')
# Annotate bars with counts
for bars in [bars_pred, bars_actual]:
    for bar in bars:
        height = bar.get_height()
        plt.text(bar.get_x() + bar.get_width() / 2, height, f'{height}', ha='center', va='bottom')
plt.xlabel('Class')
plt.ylabel('Count')
plt.title('Counts of Actual and Predicted Values')
plt.xticks([i + bar_width / 2 for i in index], labels)
plt.legend()
plt.show()
```

Prediction Using Input Data

```
# Define your input data (replace with your values)
pH = 7.3
Sodium = 76.51
Magnesium = 31.51
Calcium = 81.9
Chloride = 62.53
Potassium = 1.23
Carbonate = 423.68
Sulphate = 69.13
TDS = 635
EC = 1053
TH = 336
# Prepare the input data (might involve scaling or formatting)
input_data = np.array([[pH, Sodium, Magnesium, Calcium, Chloride, Potassium, Carbonate,
Sulphate, TDS, EC, TH]])
# Make a prediction using the model
prediction = loaded_abc_model.predict(input_data)
# Interpret the prediction (e.g., probability of being potable water)
print("Model Prediction:", prediction)
if prediction == 0:
  print("Result: Water is Unsafe")
elif prediction == 1:
  print("Result: Water is Safe")
else:
  print("Invalid prediction value. Expected 0 or 1.")
```