Water Quality Prediction

Using Machine Learning

Group Members:

Aritra Nandy
Asansol Engineering College
MCA
201080571010051

Bikram Roy
Asansol Engineering College
MCA
201080571010002

Deep Mondal
Asansol Engineering College
MCA
201080571010045

Sanjib Mondal
Asansol Engineering College
MCA
201080571010012

Sudip Chatterjee
Asansol Engineering College
MCA
201080571010016

Department of Computer Applications

Asansol Engineering College

Kalyanpur, Sen Raleigh Road, Asansol – 713304

CERTIFICATE

This is to certify that the project work entitled ***"Water Quality Prediction"*** *is a bonafide record of work carried out in the* ***Department of Computer Application,*** *Asansol Engineering College, Asansol.*

Name	*Roll Number*	*Registration Number*
Bikram Roy	***10871020004***	***201080571010002***
Sanjib Mondal	***10871020014***	***201080571010012***
Sudip Chatterjee	***10871020016***	***201080571010014***
Deep Mondal	***10871020047***	***201080571010045***
Aritra Nandy	***10871020053***	***201080571010051***

The students of 4th Semester MCA 2020-22 under my / our supervision in requirement of partial fulfillment of the Award of Degree of MCA from Maulana Abul Kalam Azad University of Technology, West Bengal.

Signature of
The Project Guide
Name: Mr. Arnab Chakraborty

Recommendation & Signature of
The Principal
Name: Dr. P. P. Bhattacharya

Recommendation &Signature of
The Head of Department
Name: Dr. D. K. PAL

Recommendation &Signature of
Internal / External Examiners

ABSTRACT

Water is one of the most important natural resources for all living organisms on earth. The monitoring of treated wastewater discharge quality is vitally important for the stability and protection of the ecosystem. Collecting and analyzing water samples in the laboratory consumes much time and resources. In the last decade, many machine learning techniques, like multivariate linear regression REGRESION and artificial neural network DECISION TREE model, have been proposed to address the problem. However, simple linear regression analysis cKNNot accurately forecast water quality because of complicated linear and nonlinear relationships in the water quality dataset. The DECISION TREE model also has shortcomings though it can accurately predict water quality in some scenarios. For example, DECISION TREE models are unable to formulate the non-linear relationship hidden in the dataset when the input parameters are ambiguous, which is common in water quality dataset.

The adaptive neuro-fuzzy inference system has been proven to be an effective tool in formulating the complicated linear and non-linear relationship hidden in datasets. Although the KNN model can achieve good performance in the water quality prediction, it has some limitations. Firstly, the size of the training dataset should not be less than the number of training parameters required in the model. Secondly, when the data distribution in the testing dataset is not reflected in the training dataset, the KNN model may generate out-of-range errors. Lastly, a strong correlation is required between input and target parameters. If the correlation is weak, the KNN model cKNNot accurately formulate the hidden relationship.

In this dissertation, several methods have been proposed to improve the performance of water quality prediction models. Stratified sampling is employed to cover different kinds of data distribution in the training and testing datasets. The wavelet denoising technique is used to remove the noise hidden in the dataset. A deep prediction performance comparison between KNN, DECISION TREE and REGRESSION model is presented after stratified sampling and wavelet denoising techniques are applied. Because water quality data can be thought as a time series dataset, a time series analysis method is integrated with the KNN model to improve prediction performance. Lastly, intelligence algorithms are used to optimize the parameters of membership functions in the KNN model to promote the prediction accuracy. Experiments based on water quality datasets collected from Las Vegas Wash since 2007 and Boulder Basin of Lake Mead, Nevada, between 2011 and 2016 are used to evaluate the proposed models.

ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to my faculty, Prof. Arnab Chakraborty for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

Aritra Nandy

Bikram Roy

Deep Mondal

Sanjib Mondal

Sudip Chatterjee

Table of contents

Introduction

Background

To protect the environment and human health, treated wastewater discharge must be sampled and monitored in most developed countries to assure discharge permits are met [1]. In the past, scientists had to collect and analyze a large number of wastewater samples to understand how wastewater discharges components impacted the environment. The collection and analysis of treated wastewater effluents is time-consuming and costly. Machine learning methods are proposed to address the problem. The usage of machine learning methods would result in a reduction in sampling frequency and minimization of costs associated with analysis. At first, deterministic models and multivariate linear regression (REGRESION) analysis were used to speed up the process of evaluating the quality of wastewater effluent discharges [2] [3]. As a water quality dataset can be considered as a time series dataset, which is likely to have a complicated nonlinear relationship, the performance of deterministic and REGRESION models is expected to be poor.

In the past decade, many machine learning techniques have been proposed to address the problem. Artificial neural networks (KNN) are adopted to explore the non-linear relationships residing in water quality datasets [3][4]. Various KNN models have been designed to predict water and wastewater discharge quality based on previous existing datasets. A comprehensive comparison between KNN and REGRESION models for oxygen demand prediction has been performed

The experimental results show that a three-layer neural network model outperforms an REGRESION model. In [4], neural network models are used to predict four parameters in the Qiantang River and the proposed model has higher accuracy and better stability in the experiment. Although KNN models can effectively improve the prediction accuracy of water quality parameters, shortcomings still exist. Especially in some scenarios where the input parameters are ambiguous, neural networks struggle to formulate a non-linear relationship. Many studies have proven that an adaptive neuro-fuzzy inference system (DECISION TREE), which can integrate linear and non-linear relationships hidden in the dataset, is a better option in this scenario [5]. The experimental results in [6] show that an DECISION TREE model worked much better than an KNN model in predicting dissolved oxygen, even though there were only 45 data samples available. The experimental results confirm that the proposed method works. The DECISION TREE model has also been applied in effluent quality prediction, and an experiment with a dataset of around 150 data samples has proven that the DECISION TREE model is better than the KNN model [7].

Although the DECISION TREE model can achieve good performance in the water quality prediction, it has some limitations. Firstly, the size of training dataset should not be less than the number of training parameters required in the model. Secondly, when the data distribution in the testing dataset is not reflected in the training dataset, the DECISION TREE model may generate out-of-range errors. Lastly, a strong correlation is required between input and target parameters. If the correlation is weak, the DECISION TREE model cKNNot accurately formulate the hidden relationship.

On the other hand, the fuzzy time series (FTS) model is an accurate and reliable model to predict time series data. It has been widely used to solve time series dataset prediction

problem [8][9]. As a water quality data is a kind of time series dataset, the FTS model is applicable in this scenario.

Motivation

Among the three types of prediction models introduced in Section 1.1, both the REGRESION and KNN models have no requirement on the size of data samples and correlation level between parameters. When training the DECISION TREE model, researchers need to follow the rules proposed by the model creator, such as the number of data samples should not be less than the number of parameters in the model. In real water quality monitoring systems, the size of the water quality dataset spans between a few hundred to a few thousand [3][4][6][7][8][10][11]. This motivates us to study given different dataset settings, how to choose the right prediction model which can provides good prediction performance for the target parameters.

In this dissertation, the water quality prediction problem is classified into five categories based on the size of a water quality dataset. The following assumptions are used in our classification. First, it assumes that each water quality monitoring station samples the water every half month. Thus, one monitoring station collects 24 data samples in each year. Second, it assumes that a water quality monitoring system has one or more water quality monitoring stations. For a system with five or more water quality monitoring stations and has been running for 20 years, the water quality dataset collected in the system should have more than 2400 water quality samples. In this dissertation, the dataset with 2400 or more data samples is called a large size dataset. A medium-size dataset would be one from a monitoring system with more than two monitoring stations

which has been running for 10 more years and has 480 but less than 2400 data samples. The dataset will be treated as a small dataset if the size of the dataset is less than 480.

Five scenarios are envisioned:

Scenario 1: Large size dataset $(\geq 2400 \text{samples})$ and strong correlation between parameters.

In this scenario, as there are sufficient data samples to cover different water quality conditions, REGRESION, KNN, and DECISION TREE models can be directly employed to predict water quality. The most reliable and accurate model can be selected out based on the prediction performance of each model in the testing stage.

Scenario 2: Large size dataset and weak correlation between parameters.

As a water quality data is a kind of time series dataset, timing impact should be taken into consideration. The DECISION TREE model is not applicable for predicting parameters with a weak correlation. Is it possible to expand the input parameters in the time domain to obtain a strong correlation among them? Another option is to apply DECISION TREE to predict parameters in the timely expanded dataset.

Scenario 3: Medium size dataset (480~2400 sample) and strong correlation between parameters For this scenario, one important issue is how to select the membership functions and network structure to obtain optimum configuration for the DECISION TREE model. Another issue is that the DECISION TREE model may have the risk of falling into out-of-range error trap when the size of the dataset is limited.

Scenario 4: Medium size dataset and weak correlation between parameters

Intelligence algorithms have been proven to be a reliable way to find the near-optimum solution in a search problem. Is it possible to integrate intelligence algorithms with the DECISION TREE model to find the optimum configuration of parameters in the DECISION TREE model so that the prediction accuracy can be greatly improved by the hybrid DECISION TREE model? Scenario 5: Small dataset (<480 samples)
Not applicable, the model built upon small dataset is not reliable or portable.

Contribution

In this dissertation, to address the water quality prediction problems described in Scenarios 2 to 4, different solutions are proposed and evaluated. Different methods including stratified sampling, wavelet de-noising, time series analysis, and intelligence algorithms are selected to integrate with the DECISION TREE model to predict water quality according to the scenario that the collected water quality dataset fits in. The major contributions of our work are listed below:

- For issues in Scenario 2:

1) Input parameters are expanded in the time domain to build correlation between parameters so that the DECISION TREE model can be applied. The KNN model is applied to predict the water quality parameters which have small fluctuation. As the historical data of the prediction target, it is more suitable to apply this model to predict the water quality parameter which has a very long record in one single water quality monitoring station.

1) Stratified sampling strategy is used to mitigate the uneven distribution of training and testing dataset and thus eliminate out-of-range errors of the DECISION TREE model in the testing stage.

2) A general framework of water quality prediction system based on wavelet de-noised DECISION TREE model using stratified sampling is proposed.

3) The general input parameter selection method is proposed to identify the most correlated input parameters.

4) The network structure selection method is proposed to increase the robustness and scalability of the DECISION TREE-based prediction system.

-

1) Integrate intelligence algorithms combined with the DECISION TREE model are proposed to improve the water quality prediction performance. In this model can be used to predict many water quality parameters with no limitation on the number of water quality monitoring stations.

Outline of the Dissertation

The rest of the dissertation is organized as follows reviews different kinds of water quality prediction models. The study area, basic methods and evaluation metrics that are commonly used in the study of both scenarios are introduced. A wastewater quality prediction system based on stratified sampling and wavelet denoising DECISION TREE model is presented to solve the Scenario 3 problem is considered and time series

analysis are presented to solve it. The integration of intelligence algorithms and the DECISION TREE model is presented to solve Scenario 4 problem concludes this dissertation.

LITERATURE REVIEW

Modeling the quality of water resources is vitally important for water scheduling and management. In the past, scientists regularly sampled the water in water quality monitoring stations and assessed the components in the water sample in a lab. However, this process takes a long time, and thus, the detected results are not timely. With the emergence of artificial intelligence (AI) techniques since the last decade, researchers have begun to adopt multivariate linear regression KNN model, DECISION TREE model, REGRESION model to predict water quality by exploring the linear and non-linear elationships residing in water quality datasets. In addition, the wavelet denoising method and intelligent algorithms are also proposed to combine with machine learning techniques to enhance the prediction accuracy. In the following, we will review these related work in three categories of machine learning methods.

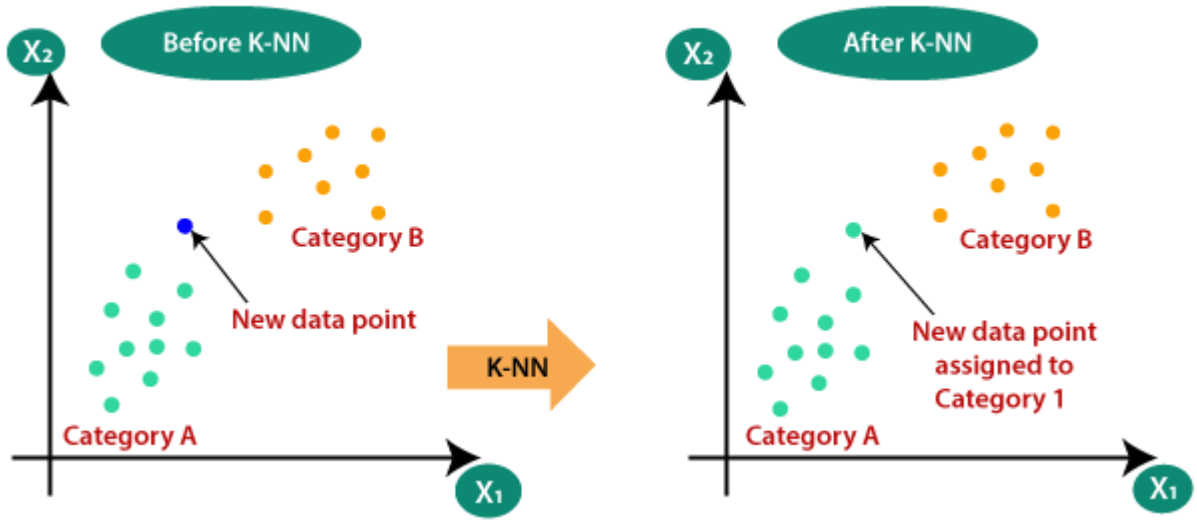
KNN: -

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



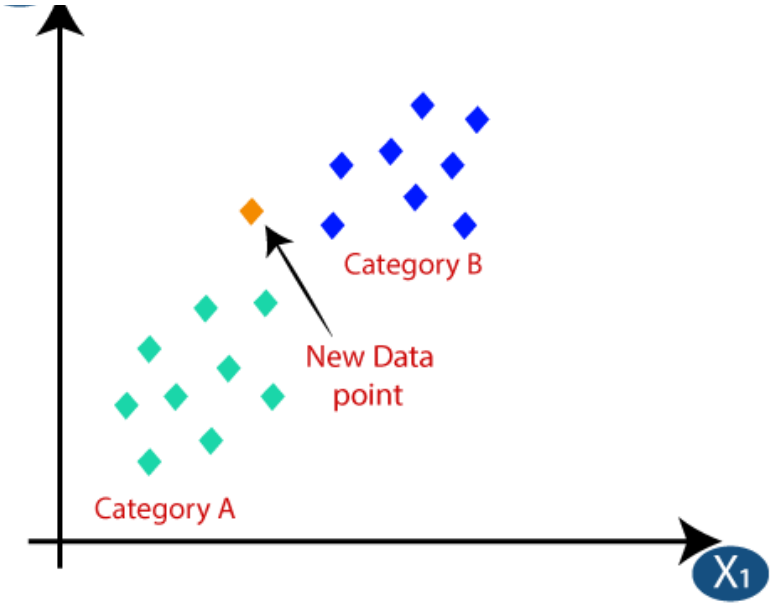
How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

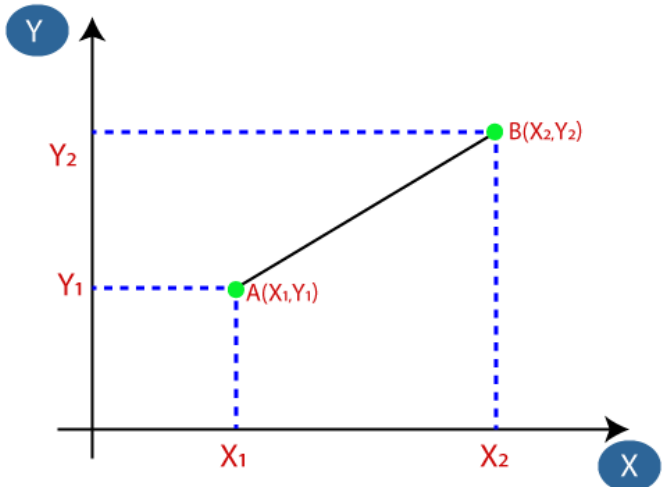
- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category.

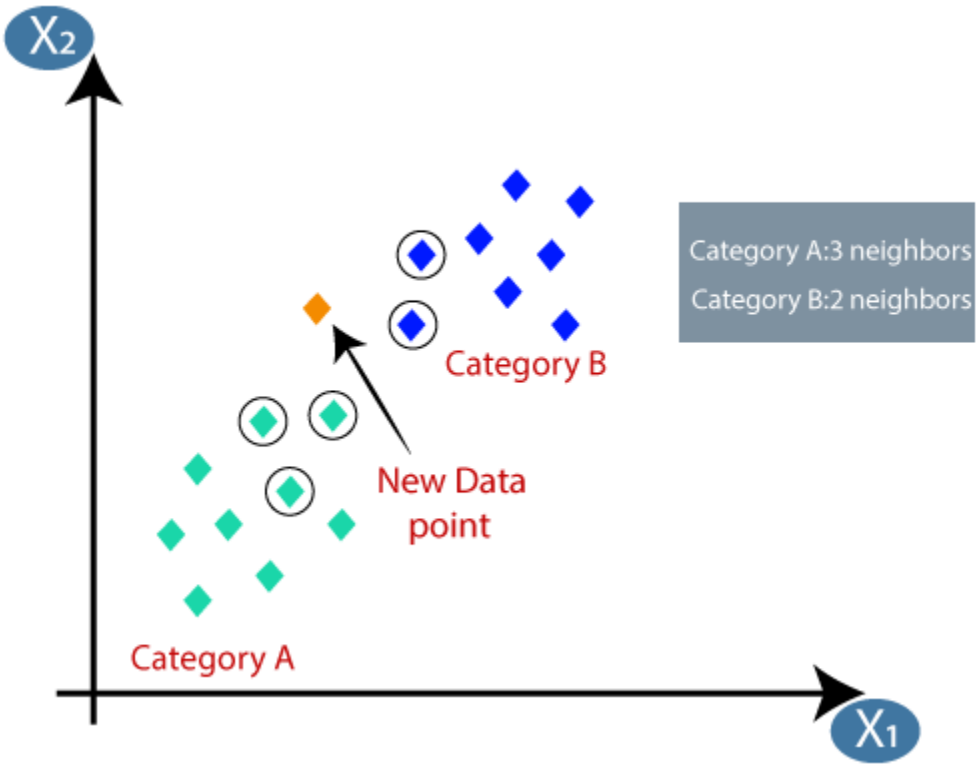
Consider the below image



- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

Advantages of KNN Algorithm:

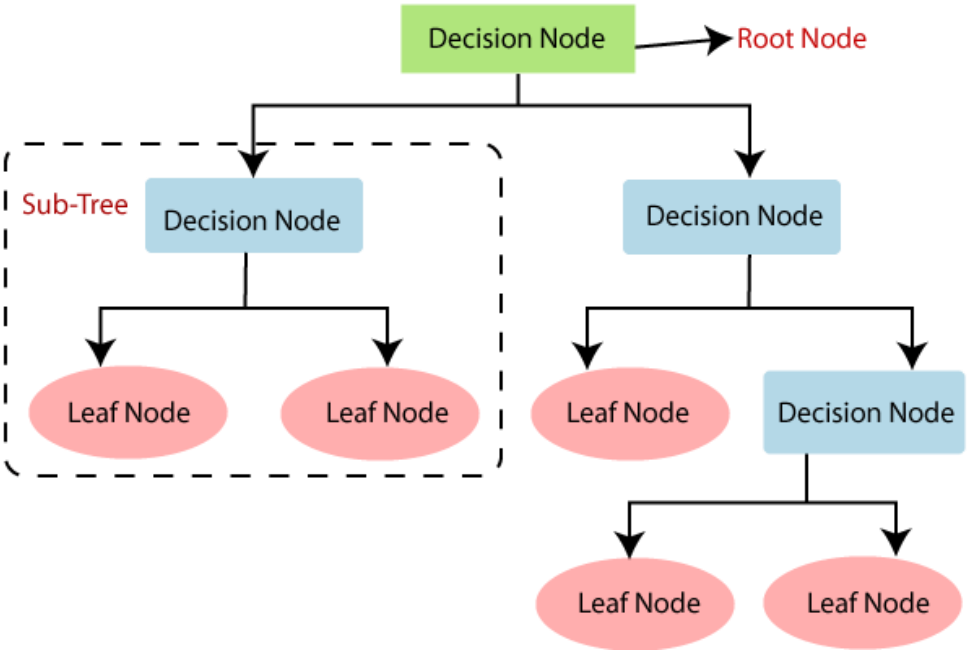
- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

Decision Tree: -

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**
- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- ***It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.***
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
- Below diagram explains the general structure of a decision tree:



Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

Decision Tree Terminologies

 Root Node: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.

 Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.

 Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.

 Branch/Sub Tree: A tree formed by splitting the tree.

 Pruning: Pruning is the process of removing the unwanted branches from the tree.

 Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes.

How does the Decision Tree algorithm Work?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.

- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Advantages of the Decision Tree

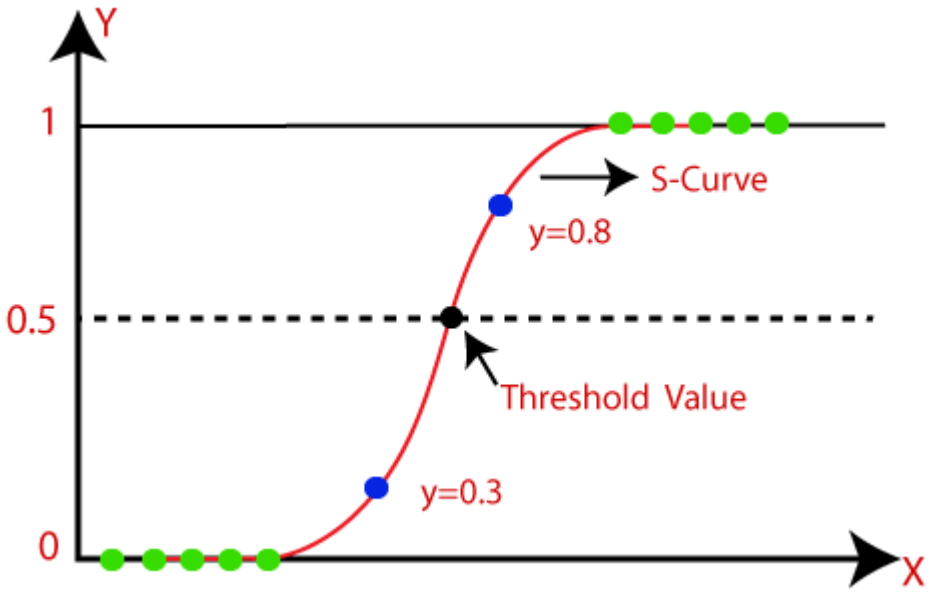
- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

Disadvantages of the Decision Tree

- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the **Random Forest algorithm.**
- For more class labels, the computational complexity of the decision tree may increase.

Logistic Regression: -

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



Logistic Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Assumptions for Logistic Regression:

- The dependent variable must be categorical in nature.
- The independent variable should not have multi-collinearity.

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

- We know the equation of the straight line can be written as:

$$y= b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}$$; 0 for y= 0, and infinity for y=1

- But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

The above equation is the final equation for Logistic Regression.

Type of Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

- **Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.

- **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
- **Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

Study Area

The Las Vegas Wash (LVW) is the primary channel carrying the Las Vegas Valley's treated wastewater discharges and runoff to Lake Mead, a man-made lake on the Colorado River [37]. With an average flow of 200 million gallons per day, the LVW contributes approximately 1.5% of the total water flow to the lake. The flow in the Wash consists of highly treated wastewater, urban runoff, shallow groundwater, and stormwater. Lake Mead is the largest reservoir in the United States, located in the Colorado River, which is the drinking water source for 30 million people in Nevada, Arizona, and California. Salinity control is a huge issue for the Colorado River, and part of an international treaty with Mexico. Agricultural damage due to salinity costs the US over 454 million per year [38].

Along the LVW, several government organizations, including the Southern Nevada Water Authority (SNWA) and United States Geological Survey (USGS), have built multiple water quality monitoring stations to monitor and track the wastewater quality. Six locations have been selected as the places to build monitoring stations, by nearly all organizations, because of their geographical advantages. These six water quality monitoring stations are labeled in Figure 1 [37], and their geographical distribution is given.

There are three basins occupied by the Lake Mead Reservoir, with the Boulder Basin (BB) as the most western one. It lies within the boundaries of Clark County, Nevada and Mohave County, Arizona. The BB provides drinking water resources for the people living there. The water in the BB finally joins the Lake Mead. The geographical distribution of the BB in Lake Mead is depicted in Figure 2.

Figure 1. The geographical distribution of the six water quality monitoring stations in the Las Vegas Wash

Figure 2. Location of Boulder Basin water quality monitoring station

Stratified Sampling

proportionally select data points from different strata for different purposes [39]. This method is widely used in classification problems to verify proposed models, in which the training and testing datasets of each fold contain roughly the same proportion of each class label. Compared with the traditional sampling method, which generates n random partitions from the original dataset, stratified sampling makes sure the training and testing dataset can evenly cover all of the different categories.

A stratified sampling method works better in scenarios where the problem has data sparsity restrictions. When sampling from a large dataset, randomly sampling data from the collected dataset can cover all scenarios. However, when it comes to a medium or even a small order of magnitude dataset, some types of scenarios probably fall out of the sampled dataset. Since most of the water quality monitoring stations have only been built in the past several decades, and a large proportion of the water quality parameters have only been recordable in the past 20 years, the water quality dataset belongs to this case, which has limited number of data samples.

3.3 Adaptive Neuro-Fuzzy Inference System

The DECISION TREE is a hybrid learning model, which integrates the neural network and fuzzy logic into an integrated system. The system can achieve high performance in formulating nonlinear relationships and forecasting chaotic time series. It can construct a reliable and accurate input-output mapping relationship based on the fuzzy if-then rules. The DECISION TREE model used in this study is generated based on the fuzzy model proposed in [40]. Given two input parameters, x and y , and one output function, f , the rule set built upon the model can be expressed as follows:

$$\text{Rule 1 : if } x \text{ is } A_1 \text{ and } y \text{ is } B_1 \text{ then } f_1 = a_1 x + b_1 y + c_1 \quad (1)$$

Rule 2 : if x is A_2 and y is B_2 then $f_2 = a_2 x + b_2 y + c_2$ (2)

where A_1 , A_2 , B_1 , and B_2 represent four input membership functions for input parameters x and y . In this example, each input has two membership functions. The value of the consequent parameters a_i , b_i , and c_i are calculated by the least square error method.

The network structure of an DECISION TREE model consists of five layers. The data flow in Figure 3 illustrates the process of deriving the output from two inputs. The function of each layer is fuzzification, normalization, merge, and summarization. Assuming function μ_{Ai} and μ_{Bi} are the membership functions of fuzzy sets A_i and B_i , parameters d_i and $_i$ are the premise parameters σ that define the structure of the membership functions in each node. To each layer, i , the output of the jth node is denoted as $O_{i\,j}$. A brief introduction of the corresponding function in each layer is as follows:

Layer 1: Generates membership weights for the fuzzy sets based on membership function;

$$O_1^i = \mu_{Ai}(x) = \exp\left(\frac{-(x-d_i)^2}{\sigma_i^2}\right) \quad (3)$$

Layer 2: Generates rules by executing, AND operator to the two incoming membership weights;

$$O_2^i = w_i = \mu_{Ai}(x) \times \mu_{Bi}(y) \quad (4)$$

Layer 3: Normalizes the weight for the merge process;

$$O_3^i = \overline{w}_i = \frac{w_i}{\sum_{i=1}^{4} w_i} \quad (5)$$

Layer 4: Computes the output by multiplying the incoming normalized weight with the result of the linear regression model in the current node;

$$O_4^i = \overline{w}_i z_i = \overline{w}_i (a_i x + b_i y + c_i) \quad (6)$$

Layer 5: Summarizes all of the products from Layer 4.

$$0_{5}^{i} = \sum_{i\,1}^{4} \overline{w}_{i}\, y_{i} = \frac{\sum_{i\,1} w_i\, y_i}{\sum_{i=1}^{4} w_i} \tag{7}$$

3.4 Wavelet Transform

Wavelet Transform (WT) is widely used in the analysis of time series signals due to its effectiveness in smoothing the irregularity of non-stationary data [30]. According to the way the scale parameter is discretized, it is classified into continuous or discrete WT. As continuous WT requires a large number of data samples, discrete WT is selected as the de-noising technique in this study. Discrete WT decomposes the input signal into a mutually orthogonal set of wavelets by using a discrete set of wavelet scales and translations. Compared to continuous WT, it requires much less computation time and is simpler to develop. Given the limited number of the highest coefficients of the discrete WT spectrum, an inverse transform can be performed with the

same wavelet basis to remove the noise hidden in the true signal. The corresponding wavelet transformation can be defined as:

$$WT(a, b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{+\infty} f(t)\, \psi^{*}\left(\frac{t-b}{a}\right) dt \quad (8)$$

where the variables n and m are integers that control the wavelet dilation and translation, a is the scale index parameter, and b is the time-shifting parameter. All of the points that can be represented as (a_m, na_mb) are included in the subset of the wavelet scales and translations. $\psi(t)$ is a continuous function in both time and frequency domains, called mother wavelet, to get a stably invertible transform; and f (t) is the input signal, or time series.

3.5 Evaluation Metrics

There are many evaluation metrics available to examine the performance of the proposed model. In this study, the mean average percentage error (MAPE), mean square error (MSE), RMSE, and coefficient of determination (R^2) are adopted to compare the performance of different models. MAPE is used to represent the difference between the predicted value and true value, in percentage form. RMSE is the value calculated by rooting the square of the mean of the residuals between the true value and predicted value. R^2 is an indicator to show how close the data are to the fitted regression line. The mathematical definitions of the evaluation metrics are defined as:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n} \left| \frac{y_{true_i} - y_{pred_i}}{y_{true_i}} \right| \times 100\% \quad (9)$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_{pred_i} - y_{true_i})^2 \quad (10)$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_{pred_i} - y_{true_i})^2} \quad (11)$$

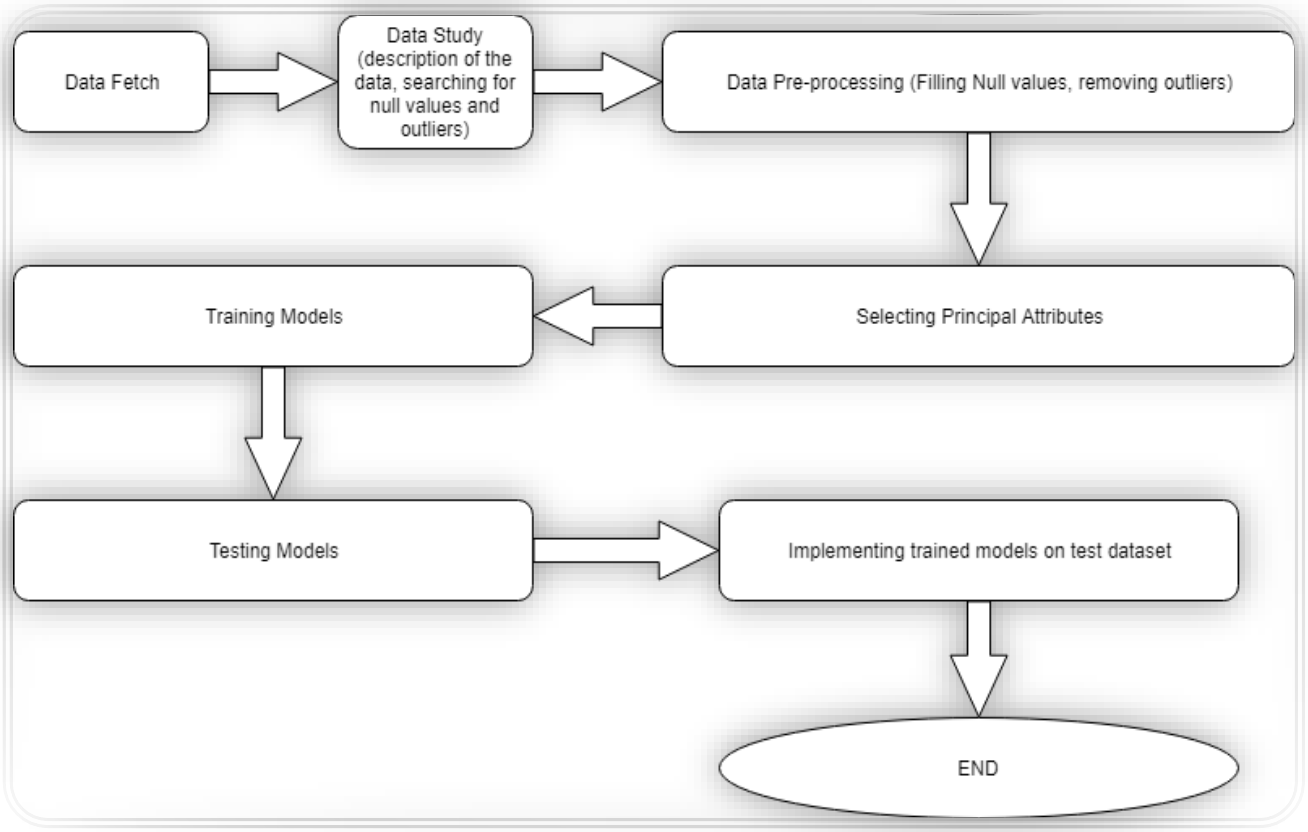
$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_{pred_i} - y_{true_i})^2}{\sum_{i=1}^{n} (y_{true_i} - y_{mean})^2} \quad (12)$$

where index i represents the position of the element in the vector, y_{true} is a vector holding all of the observed value, y_{mean} stands for the average value of vector y_{true}, y_{pred} is a vector storing all the forecasting value, and n is the size of the dataset.

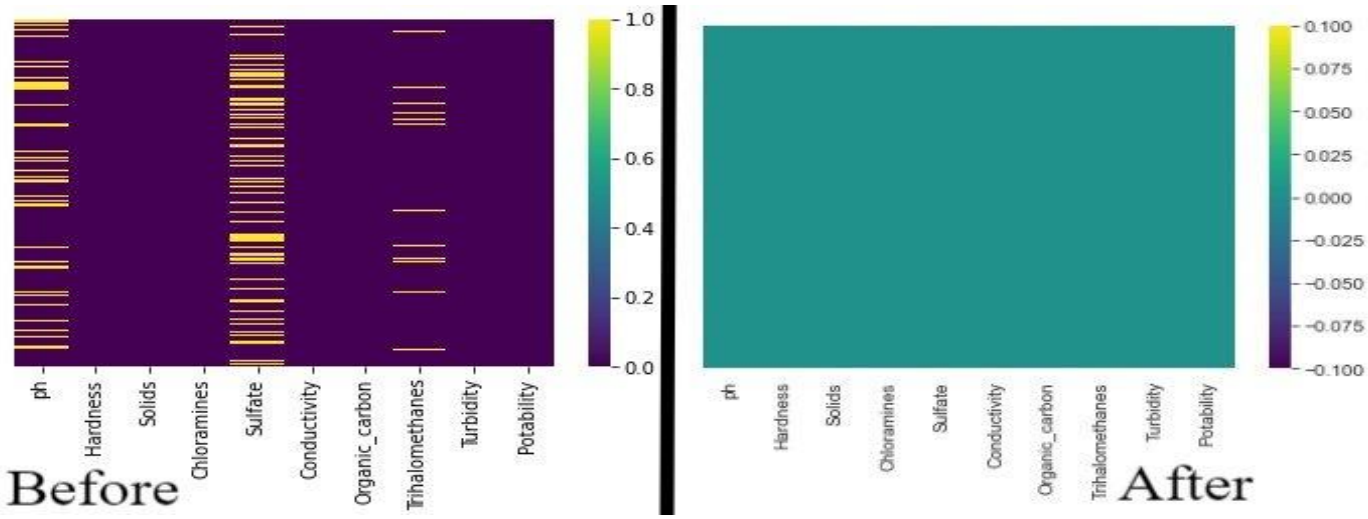
Dataset

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
1		204.8904555	20791.31898	7.300211873	368.5164413	564.3086542	10.37978308	86.99097046	2.963135381	0
2	3.716080075	129.4229205	18630.05786	6.635245884		592.8853591	15.18001312	56.32907628	4.500656275	0
3	8.099124189	224.2362594	19909.54173	9.275883603		418.6062131	16.86863693	66.42009251	3.05593375	0
4	8.316765884	214.3733941	22018.41744	8.059332377	356.8861356	363.2665162	18.4365245	100.3416744	4.628770537	0
5	9.092223456	181.1015092	17978.98634	6.546599974	310.1357375	398.4108134	11.55827944	31.99799273	4.075075425	0
6	5.584086638	188.3133238	28748.68774	7.544868789	326.6783629	280.4679159	8.39973464	54.91786184	2.559708228	0
7	10.22386216	248.0717353	28749.71654	7.513408466	393.6633955	283.6516335	13.78969532	84.60355617	2.672988737	0
8	8.635848719	203.3615226	13672.09176	4.563008686	303.3097712	474.6076449	12.3638167	62.79830896	4.401424715	0
9		118.9885791	14285.58385	7.804173553	268.6469407	389.3755659	12.70604897	53.92884577	3.595017181	0
10	11.18028447	227.2314692	25484.50849	9.077200017	404.0416347	563.8854815	17.92780641	71.97660103	4.370561937	0
11	7.360640106	165.5207973	32452.61441	7.550700907	326.6243535	425.3834195	15.58681044	78.74001566	3.662291783	0
12	7.974521649	218.6933005	18767.65668	8.110384501		364.0982305	14.5257457	76.48591118	4.011718108	0
13	7.119824384	156.7049933	18730.81365	3.606036091	282.3440505	347.7150273	15.92953591	79.50077834	3.445756223	0
14		150.1749234	27331.36196	6.838223471	299.4157813	379.7618348	19.37080718	76.50999553	4.413974183	0
15	7.496232208	205.3449822	28388.00489	5.072557774		444.6453523	13.2283111	70.30021265	4.777382337	0
16	6.347271761	186.7328807	41065.23476	9.629596276	364.4876872	516.7432819	11.53978119	75.07161729	4.376348291	0
17	7.0517858	211.0494061	30980.60079	10.09479601		315.1412672	20.39702184	56.65160379	4.268428858	0
18	9.181560007	273.8138067	24041.32628	6.904989726	398.3505168	477.9746419	13.38734078	71.45736221	4.503660796	0
19	8.975464348	279.3571668	19460.39813	6.204320859		431.44399	12.88875905	63.8212371	2.43608559	0
20	7.371050302	214.4966105	25630.32004	4.43266929	335.7544386	469.9145515	12.50916394	62.79727715	2.560299148	0
21		227.4350484	22305.56741	10.33391789		554.8200865	16.33169328	45.38281518	4.133422644	0
22	6.660212026	168.2837469	30944.36359	5.858769131	310.9308583	523.6712975	17.88423519	77.04231805	3.749701241	0
23		215.9778587	17107.22423	5.607060453	326.9439777	436.256194	14.18906221	59.85547583	5.459250956	0
24	3.902475686	196.9032467	21167.5001	6.996311586		444.4788825	16.60903316	90.18167588	4.528522696	0
25	5.400301781	140.7390623	17266.59342	10.05685248	328.3582407	472.8740733	11.25638117	56.93190645	4.82478639	0
26	6.514415093	198.7673513	21218.70287	8.67093692	323.596349	413.2904501	14.89999957	79.84784281	5.200885077	0
27	3.445061864	207.9262602	33424.76868	8.782147481	384.0070058	441.7858757	13.80590221	30.2845972	4.184396969	0
28		145.7681806	13224.93564	7.906444721	304.0019928	298.9906665	12.72952472	49.5368488	4.004871128	0
29		266.4210181	26362.96501	7.70006347	395.3894903	364.4801067	10.34895076	53.00838135	3.991564248	0
30		148.1530614	15193.41347	9.046832707	307.0117926	563.8047433	16.56865557	52.67618503	6.038184953	0
31	7.181448581	209.6256005	15196.22999	5.994678646	338.3364311	342.1112863	7.922598333	71.53795326	5.088859989	0
32	9.825489908	190.7566183	19677.89247	6.757540731		452.8362349	16.8990378	47.08197119	2.857472426	0
33	10.43329098	117.7912296	22326.89205	8.161504811	307.7075086	412.9868338	12.89070856	65.73347839	5.057310517	0
34	7.414148196	235.0445345	32555.85254	6.845952024	387.1753165	411.9833642	10.24481507	44.48929725	3.160624468	0
35		232.280452	14787.20626	5.474914832		383.981723	12.16693715	86.08072748	5.029166724	0
36	5.115817064	191.952743	19620.54533	6.060712995	323.8363839	441.7483793	10.96648615	49.23823132	3.902088768	0
37	3.641629777	183.9087223	24752.07246	5.538314133	286.059556	456.8600959	9.034066753	73.59465676	3.464353035	0
38	5.618064406	304.2359121	17281.97517	6.101083505		399.4715663	12.26500158	81.58899246	2.896546941	0

Methodology

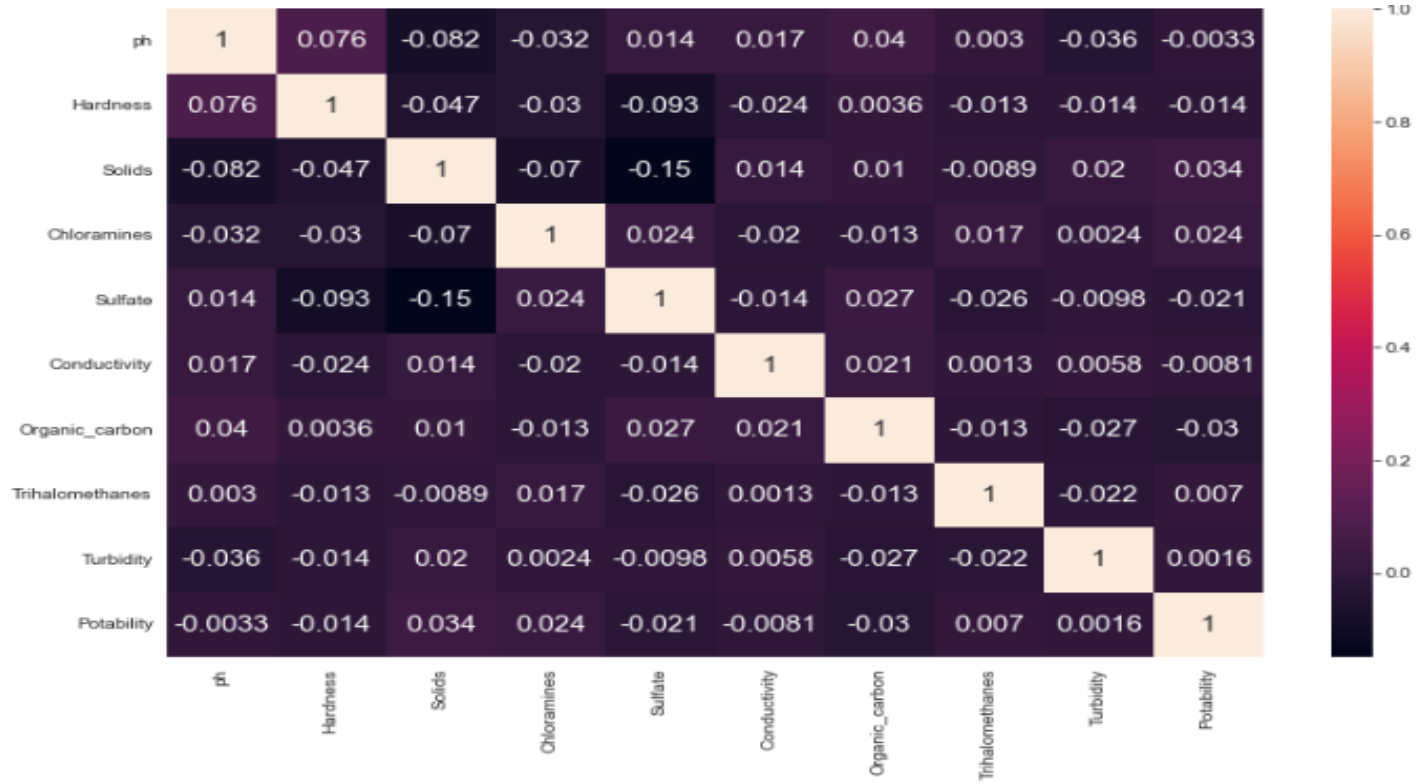


Data Pre-processing



We handled null values and NAN (not a number) values.

Heat Map Matrix



Codes

Data Gathering

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('water_potability.csv')
df.head()
```

Out[2]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	NaN	204.890456	20791.31898	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135	0
1	3.716080	129.422921	18630.05786	6.635246	NaN	592.885359	15.180013	56.329076	4.500656	0
2	8.099124	224.236259	19909.54173	9.275884	NaN	418.606213	16.868637	66.420093	3.055934	0
3	8.316766	214.373394	22018.41744	8.059332	356.886136	363.266516	18.436525	100.341674	4.628771	0
4	9.092223	181.101509	17978.98634	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075	0

Exploratory Data Analysis

In [3]:
```
df.shape
```

Out[3]: (3276, 10)

In [4]:
```
df.isnull().head()
```

Out[4]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	True	False	False	False	False	False	False	False	False	False
1	False	False	False	False	True	False	False	False	False	False
2	False	False	False	False	True	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False

In [5]:
```
df.isnull().sum()
```

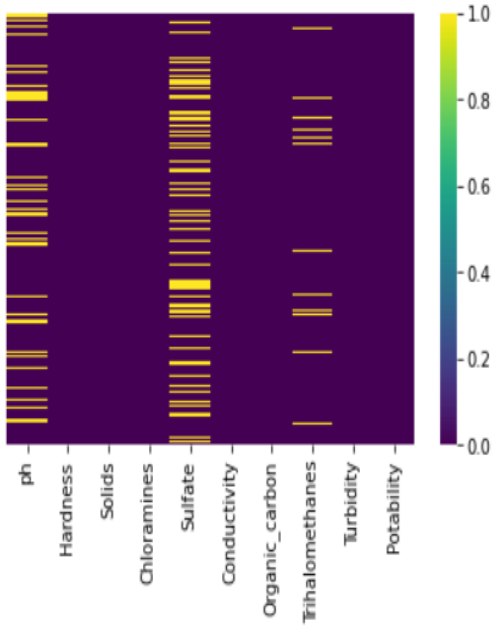
Out[5]:
```
ph                491
Hardness            0
Solids              0
Chloramines         0
Sulfate           781
Conductivity        0
Organic_carbon      0
Trihalomethanes   162
Turbidity           0
Potability          0
dtype: int64
```

In [5]:
```
1 df.isnull().sum()
```

Out[5]:
```
ph                491
Hardness            0
Solids              0
Chloramines         0
Sulfate           781
Conductivity        0
Organic_carbon      0
Trihalomethanes   162
Turbidity           0
Potability          0
dtype: int64
```

In [6]:
```
1 sns.heatmap(df.isnull(),yticklabels=False,cbar=True,cmap='viridis')
```

Out[6]: <AxesSubplot:>



In [7]:

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3276 entries, 0 to 3275
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ph               2785 non-null   float64
 1   Hardness         3276 non-null   float64
 2   Solids           3276 non-null   float64
 3   Chloramines      3276 non-null   float64
 4   Sulfate          2495 non-null   float64
 5   Conductivity     3276 non-null   float64
 6   Organic_carbon   3276 non-null   float64
 7   Trihalomethanes  3114 non-null   float64
 8   Turbidity        3276 non-null   float64
 9   Potability       3276 non-null   int64
dtypes: float64(9), int64(1)
memory usage: 256.1 KB
```

In [8]:

```
1 df.describe(include = "all")
```

Out[8]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organi
count	2785.000000	3276.000000	3276.000000	3276.000000	2495.000000	3276.000000	327
mean	7.080795	196.369496	22014.092526	7.122277	333.775777	426.205111	
std	1.594320	32.879761	8768.570828	1.583085	41.416840	80.824064	
min	0.000000	47.432000	320.942611	0.352000	129.000000	181.483754	
25%	6.093092	176.850538	15666.690300	6.127421	307.699498	365.734414	
50%	7.036752	196.967627	20927.833605	7.130299	333.073546	421.884968	
75%	8.062066	216.667456	27332.762125	8.114887	359.950170	481.792305	
max	14.000000	323.124000	61227.196010	13.127000	481.030642	753.342620	

In [9]:

```
df.fillna(df.mean(), inplace=True)
df.isnull().sum()
```

Out[9]:

```
ph                0
Hardness          0
Solids            0
Chloramines       0
Sulfate           0
Conductivity      0
Organic_carbon    0
Trihalomethanes   0
Turbidity         0
Potability        0
dtype: int64
```

In [10]:

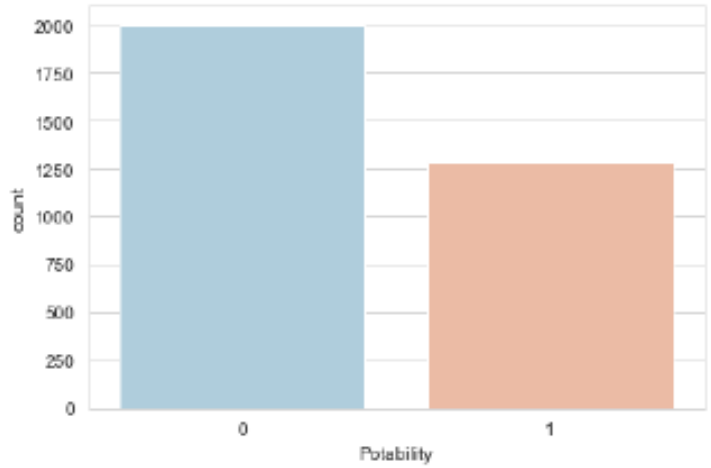
```
df.Potability.value_counts()
```

Out[10]:

```
0    1998
1    1278
Name: Potability, dtype: int64
```

In [11]:

```
sns.set_style('whitegrid')
sns.countplot(x = 'Potability', data = df, palette = 'RdBu_r')
plt.show()
```



In [12]:

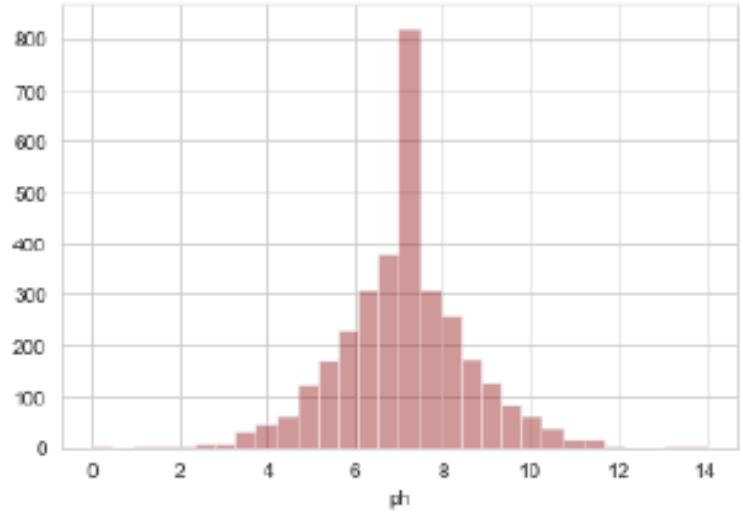
```
print ("0 - Count =", len(df[df.Potability == 0]), end= ", ")
print ("1 - Count =", len(df[df.Potability == 1]), end= ", ")
print ("Total Count =", len(df))
```

```
0 - Count = 1998, 1 - Count = 1278, Total Count = 3276
```

In [13]:

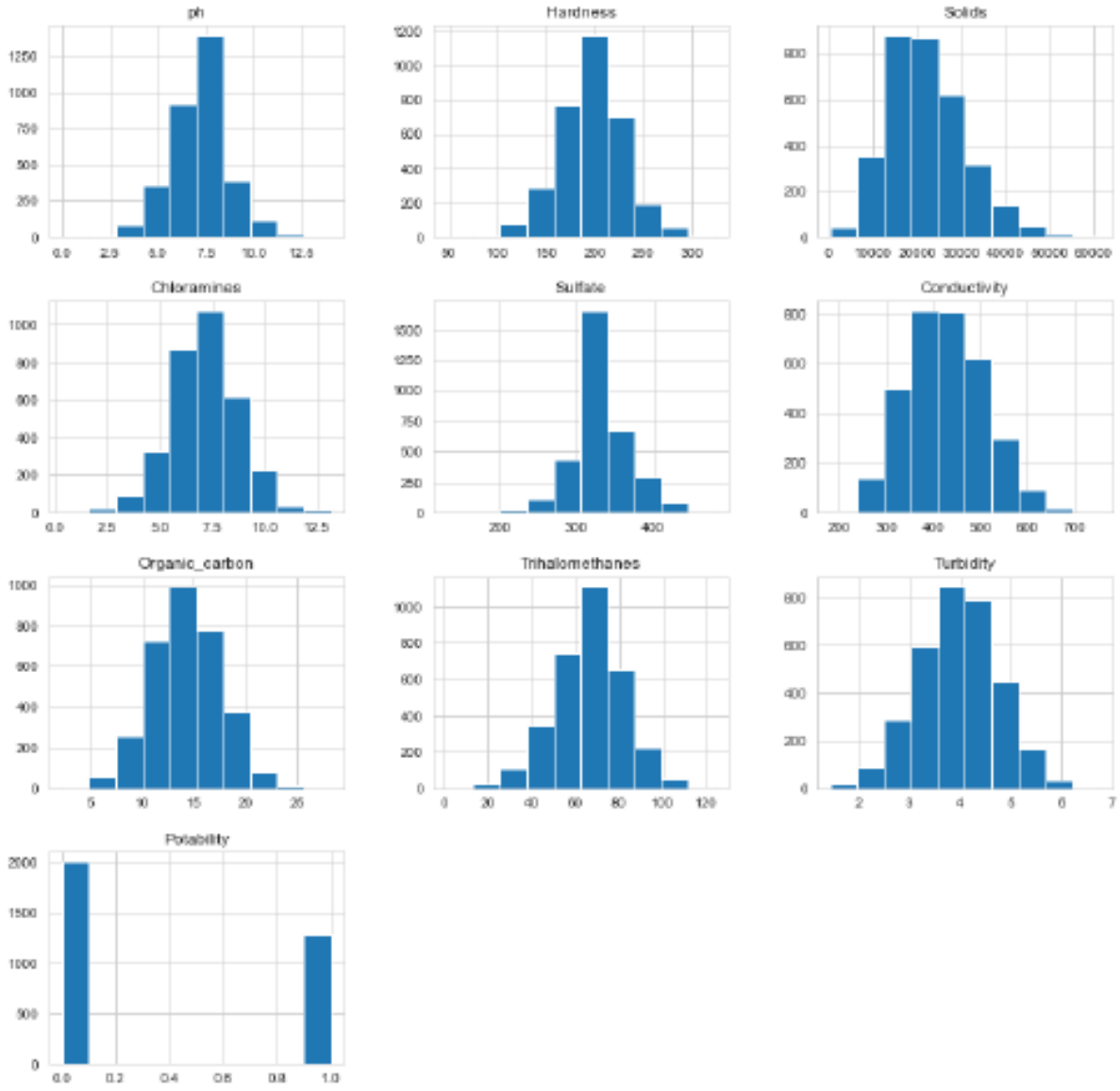
```
sns.distplot(df['ph'].dropna(), kde = False, color = 'darkred', bins = 30)
plt.show()
```

```
C:\Users\sudip\anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```



In [14]:

```
1 df.hist(figsize=(14,14))
2 plt.show()
```



In [15]:

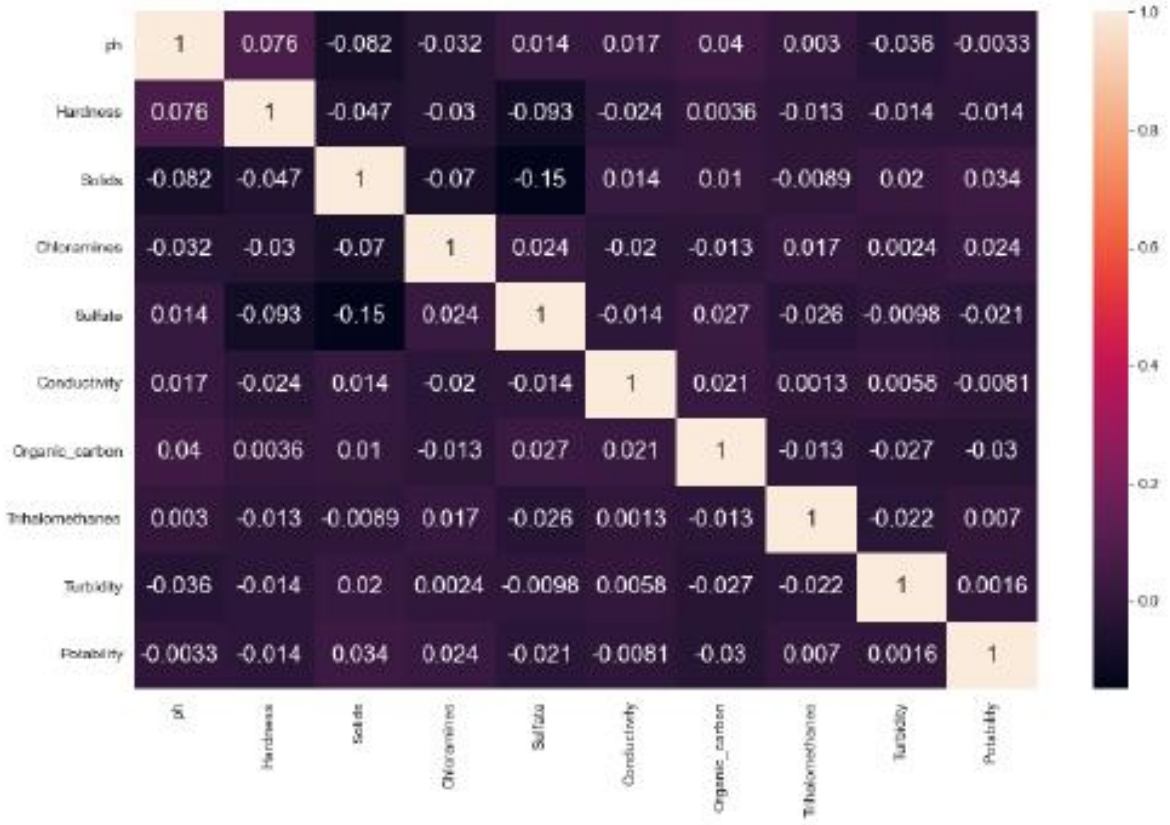
```
1 df.corr()
```

Out[15]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic
ph	1.000000	0.075833	-0.081884	-0.031811	0.014403	0.017192	0
Hardness	0.075833	1.000000	-0.046899	-0.030054	-0.092766	-0.023915	0
Solids	-0.081884	-0.046899	1.000000	-0.070148	-0.149840	0.013831	0
Chloramines	-0.031811	-0.030054	-0.070148	1.000000	0.023791	-0.020486	-0
Sulfate	0.014403	-0.092766	-0.149840	0.023791	1.000000	-0.014059	0
Conductivity	0.017192	-0.023915	0.013831	-0.020486	-0.014059	1.000000	0
Organic_carbon	0.040061	0.003610	0.010242	-0.012653	0.026909	0.020966	1
Trihalomethanes	0.002994	-0.012690	-0.008875	0.016627	-0.025605	0.001255	-0
Turbidity	-0.036222	-0.014449	0.019546	0.002363	-0.009790	0.005798	-0
Potability	-0.003287	-0.013837	0.033743	0.023779	-0.020619	-0.008128	-0

In [16]:

```
plt.figure(figsize = (13,8))
sns.heatmap(df.corr(), annot = True, annot_kws = {"size":15})
plt.show()
```

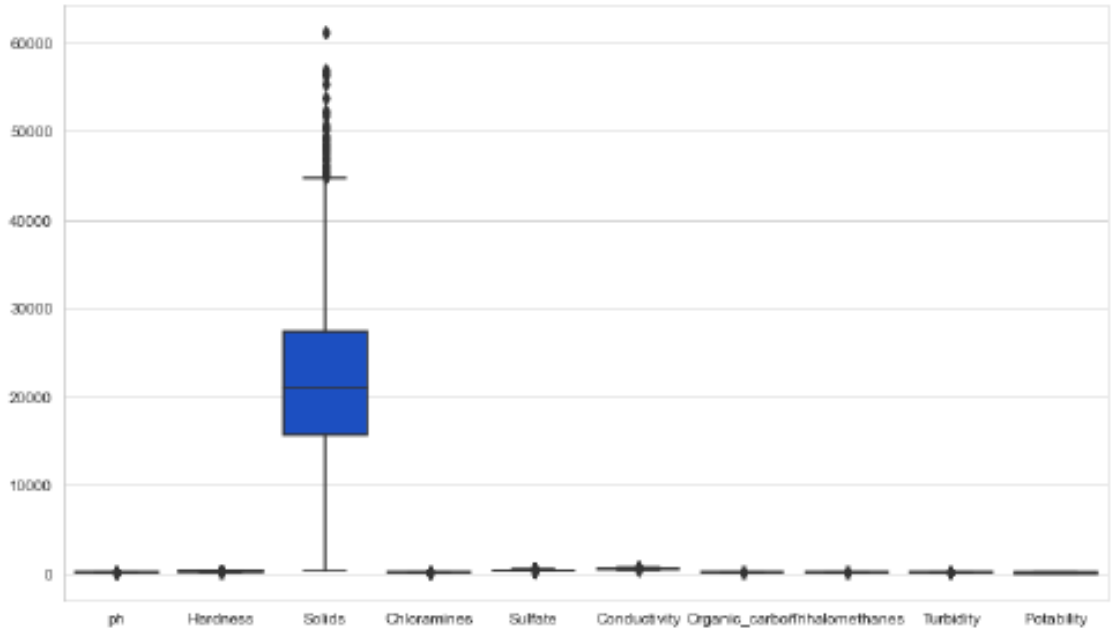


In [17]:

```
1  plt.figure(figsize=(12,7))
2  sns.boxplot(data = df, palette = 'winter')
```

Out[17]:

<AxesSubplot:>



In [18]:

```
X = df.drop('Potability',axis=1)
Y= df['Potability']
```

In [19]:

```
##### ....
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size= 0.2, random_state=1
```

Logistic Regression

In [20]:

```
df.fillna(df.mean(), inplace=True)
df.isnull().sum()
```

Out[20]:

```
ph                0
Hardness          0
Solids            0
Chloramines       0
Sulfate           0
Conductivity      0
Organic_carbon    0
Trihalomethanes   0
Turbidity         0
Potability        0
dtype: int64
```

In [21]:

```
from sklearn.model_selection import train_test_split
```

In [ ]:

```

```

In [22]:

```
x_train,x_test,y_train,y_test=train_test_split(df.drop('Potability',axis=1),df['Potabi
x_test.fillna(x_train.mean(), inplace=True)
x_test = x_test.fillna(x_train.mean())
y_test.fillna(y_train.mean(), inplace=True)
y_test = y_test.fillna(y_train.mean())
print("No. of Train rows -> ",y_train.shape, x_train.shape)
print("No. of Test rows -> ", y_test.shape, x_test.shape)
```

```
No. of Train rows ->  (2293,) (2293, 9)
No. of Test rows ->  (983,) (983, 9)
```

In [23]:

```
from sklearn.linear_model import LogisticRegression
```

In [24]:

```
logmodel = LogisticRegression(max_iter = 500)
logmodel
```

Out[24]:

```
LogisticRegression(max_iter=500)
```

In [25]:

```
1 df.head()
```

Out[25]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon
0	7.080795	204.890456	20791.31898	7.300212	368.516441	564.308654	10.379783
1	3.716080	129.422921	18630.05786	6.635246	333.775777	592.885359	15.180013
2	8.099124	224.236259	19909.54173	9.275884	333.775777	418.606213	16.868637
3	8.316766	214.373394	22018.41744	8.059332	356.886136	363.266516	18.436525
4	9.092223	181.101509	17978.98634	6.546600	310.135738	398.410813	11.558279

In [26]:

```
1 logmodel.fit(x_train, y_train)
```

Out[26]:

```
LogisticRegression(max_iter=500)
```

In [27]:

```
predictions = logmodel.predict(x_test)
print (predictions.shape)
predictions
```

```
(983,)
```

Out[27]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [28]:

```
from sklearn.metrics import classification_report
```

In [29]:

```
score = logmodel.score(x_test, y_test)
print('Test Accuracy score', score * 100,'%')
```

```
Test Accuracy score 61.34282807731435 %
```

In [30]:

```
from sklearn.metrics import confusion_matrix
```

In [31]:

```
print(confusion_matrix(y_test, predictions))
confusion_df = pd.DataFrame(confusion_matrix(y_test, predictions),
             columns = ["Predicted Class " + str(class_name) for class_name in [0, 1]],
             index = ["Class " + str(class_name) for class_name in [0, 1]])

confusion_df
```

```
[[603   0]
 [380   0]]
```

Out[31]:

	Predicted Class 0	Predicted Class 1
Class 0	603	0
Class 1	380	0

In [32]:

```python
# Precision = TP/(TP + FP)
# Recall = TP/(TP + FN)
# F1-score = 2 x (precision x recall)/(precision + recall)
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.61      1.00      0.76       603
           1       0.00      0.00      0.00       380

    accuracy                           0.61       983
   macro avg       0.31      0.50      0.38       983
weighted avg       0.38      0.61      0.47       983


C:\Users\sudip\anaconda3\lib\site-packages\sklearn\metrics\_classification.p
y:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and be
ing set to 0.0 in labels with no predicted samples. Use `zero_division` para
meter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\sudip\anaconda3\lib\site-packages\sklearn\metrics\_classification.p
y:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and be
ing set to 0.0 in labels with no predicted samples. Use `zero_division` para
meter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\sudip\anaconda3\lib\site-packages\sklearn\metrics\_classification.p
y:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and be
ing set to 0.0 in labels with no predicted samples. Use `zero_division` para
meter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

KNN Model

In [43]:

```
1 from sklearn.neighbors import KNeighborsClassifier
```

In [44]:

```
1 from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [45]:

```
1 knn=KNeighborsClassifier (metric='manhattan', n_neighbors=3)
2 knn.fit(X_train, Y_train)
```

Out[45]:

```
KNeighborsClassifier(metric='manhattan', n_neighbors=3)
```

In [46]:

```
prediction_knn=knn.predict(X_test)
accuracy_knn=accuracy_score(Y_test, prediction_knn)*100
print('accuracy_score score : ', accuracy_score(Y_test, prediction_knn)*100,'%')
```

```
accuracy_score score :  55.33536585365854 %
```

In [48]:

```
confusion_matrix(prediction_knn, Y_test)
```

Out[48]:

```
array([[282, 173],
       [120,  81]], dtype=int64)
```

In [ ]:

```
print(f"Classification Report =\n {classification_report(Y_test,prediction_knn)}")
```

Train Decision Tree Classifier and check accuracy

In [24]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
dt=DecisionTreeClassifier(criterion= 'gini', min_samples_split= 10, splitter= 'best')
dt.fit(X_train,Y_train)
```

Out[24]:

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=Non
e,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=10,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
prediction=dt.predict(X_test)
print(f"Accuracy Score = {accuracy_score(Y_test,prediction)*100}")
print(f"Confusion Matrix =\n {confusion_matrix(Y_test,prediction)}")
print(f"Classification Report =\n {classification_report(Y_test,prediction)}")

```

```
Accuracy Score = 59.29878048780488
Confusion Matrix =
 [[274 128]
 [139 115]]
Classification Report =
               precision    recall  f1-score   support

           0       0.66      0.68      0.67       402
           1       0.47      0.45      0.46       254

    accuracy                           0.59       656
   macro avg       0.57      0.57      0.57       656
weighted avg       0.59      0.59      0.59       656
```

In [26]:

```
res = dt.predict([[5.735724, 158.318741,25363.016594,7.728601,377.543291,568.304671,13.
res
```

Out[26]:

```
1
```

KNN Clasification (Using Euclidian Distance)

In [50]:

```
import pandas as pd
from sklearn import neighbors
import numpy as np

import seaborn
```

In [68]:

```
df
```

Out[68]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbo
0	7.080795	204.890456	20791.31898	7.300212	368.516441	564.308654	10.37978
1	3.716080	129.422921	18630.05786	6.635246	333.775777	592.885359	15.18001
2	8.099124	224.236259	19909.54173	9.275884	333.775777	418.606213	16.86863
3	8.316766	214.373394	22018.41744	8.059332	356.886136	363.266516	18.43652
4	9.092223	181.101509	17978.98634	6.546600	310.135738	398.410813	11.55827
...	...	...	...	...	...	...	.
3271	4.668102	193.681736	47580.99160	7.166639	359.948574	526.424171	13.89441
3272	7.808856	193.553212	17329.80216	8.061362	333.775777	392.449580	19.90322
3273	9.419510	175.762646	33155.57822	7.350233	333.775777	432.044783	11.03907
3274	5.126763	230.603758	11983.86938	6.303357	333.775777	402.883113	11.16894
3275	7.874671	195.102299	17404.17706	7.509306	333.775777	327.459761	16.14036

3276 rows × 10 columns

In [63]:

```
#df['ph'].head()
X = df[['ph', 'Hardness','Solids','Chloramines','Sulfate','Conductivity','Organic_carbo
y = np.array(df[['Potability']])

```

Out[63]:

```
array([[0],
       [0],
       [0],
       ...,
       [1],
       [1],
       [1]], dtype=int64)
```

In [94]:

```
clf = neighbors.KNeighborsClassifier(5, weights = 'uniform')
trained_model = clf.fit(X, y)
print ("trained_model: ",trained_model)
```

```
trained_model:  KNeighborsClassifier()

C:\Users\sudip\anaconda3\lib\site-packages\sklearn\neighbors\_classificatio
n.py:179: DataConversionWarning: A column-vector y was passed when a 1d arra
y was expected. Please change the shape of y to (n_samples,), for example us
ing ravel().
  return self._fit(X, y)
```

In [95]:

```
trained_model.score(X, y)
```

Out[95]:

```
0.7112332112332113
```

In [89]:

```
x_test = np.array([[7.080795,204.890456,20791.31898,7.300212,368.516441,564.308654,10.3
x2_test = np.array([[7.874671,195.102299,17404.1770,7.509306,333.775777,327.459761,16.1
```

In [88]:

```
if trained_model.predict(x_test) == 0:
    print("Not Potable")
else:
    print("Potable")

```

```
Not Potable
```

In [91]:

```
trained_model.predict_proba(x2_test)
```

Out[91]:

```
array([[0., 1.]])
```

Thank You

Conclusions

In this dissertation, the water quality prediction problems are classified into four categories according to the dataset size. Several machine learning methods based on the DECISION TREE model are proposed to improve the prediction performance for problems falling into Scenarios 2 and 3. First, to eliminate out-of-range errors of the DECISION TREE model in the testing stage, the stratified sampling strategy is used for mitigating the uneven distribution of training and testing datasets. A general framework of water quality prediction system based on wavelet de-noised DECISION TREE model using stratified sampling is proposed.

For problems in Scenario 2, i.e., medium size dataset and strong correlation between parameters, wavelet transform is integrated with DECISION TREE model to predict wastewater discharge quality parameters related to salinity levels in Chapter 4. The initial dataset is preprocessed with a statistical stratified sampling strategy. Experimental results show that the proposed model with stratified sampling outperforms MLR, ANNs, DECISION TREE, EDECISION TREE, and WT-DECISION TREE models. This demonstrates that the proposed model, with stratified sampling and a general-purpose input parameter selection method, is reliable to model the quality of parameters related to salinity. This model can be applied to reduce the number of parameters monitored to lower the cost associated with monitoring the quality of wastewater discharges.

For problems in Scenario 3, i.e., medium size dataset and weak correlation between parameters, two types of solutions are proposed for datasets with long history monitored at one single station and datasets monitored at multiple stations, respectively. In Chapter 5, for the first type of dataset, the time series analysis method is used to pre-process the water quality dataset to figure out appropriate input parameters for the DECISION TREE model. Besides, the FTS model is also applied to this scenario. The experimental results on two water quality datasets show that the FTS model could accurately predict the value of a target parameter when the target parameter has a strong correlation with its historical record, such as the parameter DO from the LVW, and the parameters DO and EC from the BB. On the other hand, when the target parameter has a strong correlation with other parameters, except itself, like parameters EC and TDS from the LVW, the DECISION TREE-TS model achieved better prediction accuracy over other models. This demonstrates that using the FTS and DECISION TREE models, integrated with time-series analysis, is an effective and reliable tool to model water quality, even when the correlation between the original parameters is weak.

In Chapter 6, for the second type of dataset, the GA and PSO are used to optimize the parameter in membership function of the DECISION TREE model. It can be seen from the experimental results that DECISION TREE-GA and DECISION TREE-PSO outperform the pure DECISION TREE model in predicting parameters DO and TDS when the correlation between the input parameters and output parameters is weak even when the time series impact is considered. DECISION TREE-PSO achieves the best performance in both training and testing data. This proves the DECISION TREE-PSO is a reliable and robust model to predict water quality.

Future Work

It can be seen from the experimental results from Chapter 6, both GA and PSO can improve the performance of the DECISION TREE model. The two algorithms are the most widely used evolutionary algorithms. In the future, more evolution algorithms, like differential evolution and ant colony optimization, will be explored to forecast the water quality. On the other hand, ideas from other data processing methods, like boost learning and weighted timing analysis will be investigated.

More data sources are required to verify the reliability and robustness of the proposed models. So far, the water quality dataset from the LVW collected by Southern Nevada Water Authority and Las Vegas Wash Coordination Committee, and dataset collected from Boulder Basin have been used as the experimental dataset. In the future, more efforts will be made to find more datasets to build a more reliable water quality prediction model

References

[1] L. Metcalf and H. P. Eddy, Wastewater Engineering: Treatment and Resource Recovery, 5th Edition. McGraw Hill, Chapter 3, 2014.

[2] Z. Xu, J. Xu, "A Deterministic Model for Predicting Hourly Dissolved Oxygen Change: Development and Application to a Shallow Eutrophic Lake," Journal of Water, vol. 8, no. 2, pp. 1-15, Jan. 2016

[3] A. H. Zare, "Evaluation of multivariate linear regression and artificial neural networks in prediction of water quality parameters," Journal of Environmental Health Science & Engineering, vol. 12, no. 1, pp. 1-8, Jan. 2014.

[4] L. Li, P. Jiang, H. Guang, L. Dong, G. Wu, and H. Wu " Water quality prediction based on recurrent neural network and improved evidence theory: a case study of Qiantang River, China,"

Environmental Science and Pollution Research, vol. 26, no. 19, pp. 19879-19896, Mar. 2019.

[3]. J.-S. R. Jang, "DECISION TREE: Adaptive-Network-Based Fuzzy Inference Systems," IEEE Transactions on Systems, Man, and Cybernetics, vol. 23, no. 3, pp. 665-685, May. 1993.

[4]. A. Najah, A. El-Shafie, O. A. Karim, and A. H. El-Shafie, "Performance of DECISION TREE versus MLP-NN dissolved oxygen prediction models in water quality monitoring," Environmental Science and Pollution Research, vol. 21, no. 3, pp. 1658 1670, Aug. 2013.

[5]. J. Wan, M. Huang, Y. Ma, W. Guo, Y. Wang, H. Zhang, W. Li, and X. Sun, "Prediction of effluent quality of a paper mill wastewater treatment using an adaptive network-based

fuzzy inference system," Applied Soft Computing, vol. 11, no. 3, pp. 3238-3246, Apr. 2011.

[6]. W. Deng, G. Wang, and X. Zhang, "A novel hybrid water quality time series prediction method based on cloud model and fuzzy forecasting," Chemometrics and Intelligent Laboratory Systems, vol. 149, pp. 39-49, Dec. 2015.

[7]. P. Singh, "Rainfall and financial forecasting using fuzzy time series and neural networks based model," International Journal of Machine Learning and Cybernetics,vol. 9, no. 3, pp. 491-506, May. 2018.

[8]. A. K. Kadam, V. M. Wagh, A. A. Muley, B. N. Umrikar, and R. N. Sankhua, "Prediction of water quality index using artificial neural network and multiple linear regression modelling approach in Shivganga River basin, India," Modeling Earth Systems and Environment, vol. 5, no. 3, pp. 951-96, Mar. 2019.

[9]. A. Sarkar and P. Pandey, "River Water Quality Modelling Using Artificial Neural Network Technique," Aquatic Procedia, vol. 4, pp. 1070-1077, 2015.

[10]. A. P. Piotrowski, M. J. Napiorkowski, J. J. Napiorkowski, and M. Osuch, "Comparing various artificial neural network types for water temperature prediction in rivers," Journal of Hydrology, vol. 529, no. 1, pp. 302-315, Oct. 2015.

[11]. A. Najah, A. El-Shafie, O. A. Karim, and A. H. El-Shafie, "Application of artificial neural networks for water quality prediction," Neural Computing and Applications, vol. 22, no. 1, pp. 187-201, Apr. 2012.

[12]. L. Zhang, G. X. Zhang, and R. R. Li, "Water quality analysis and prediction using hybrid time series and neural network models," Journal of Agricultural Science and Technology, vol. 18, no. 4, pp. 975-983, Dec. 2015.

[13]. M. J. Alizadeh and M. R. Kavianpour, "Development of wavelet-ANN p models to predict water quality parameters in Hilo Bay, Pacific Ocean," Marine Pollution Bulletin, vol. 98, no. 1, pp. 171-178, 2015.

A. G. Mohammad, K. Reza, C. Kwok-Wing, S. Shahaboddin, and T. G. Pezhman, "Forecasting pan evaporation with an integrated artificial neural network quantum-behaved article swarm optimization model: a case study in Talesh northern Iran," Engineering Applications of Computational Fluid Mechanics, vol. 12, no. 1, pp. 724-737, Sep. 2018.

[14]. A. A. Nadiri, S. Shokri, F. T.Tsai, and A. A. Moghaddam, "Prediction of effluent quality parameters of a wastewater treatment plant using a supervised committee fuzzy logic model,"

[15]. M. Khadr and M. Elshemy, "Data-driven modeling for water quality prediction case study: The drains system associated with Manzala Lake, Egypt," Ain Shams Engineering Journal, pp. 1-9, Sep. 2016.

[16]. A. A. M. Ahmed and S. M. A. Shah, "Application of adaptive neuro-fuzzy inference system (DECISION TREE) to estimate the biochemical oxygen demand (BOD) of Surma River," Journal of King Saud University - Engineering Sciences, vol. 29, no. 3, pp. 237-243, Jul. 2017.

[17]. S. Tiwari, R. Babbar, and G. Kaur, "Performance evaluation of two DECISION TREE models for predicting water quality Index of River Satluj (India)," Advances in Civil Engineering, vol. 2018, pp. 1-10, Mar. 2018.

[18]. S. Shahaboddin, J. N. Ehsan, E. A. Jason, A. M. Azizah, M. Amir, and C. Kwok-wing, "Ensemble models with uncertainty analysis for multi-day ahead forecasting of chlorophyll-a

[19].concentration in coastal waters.," Engineering Applications of Computational Fluid Mechanics, vol. 13, no. 1, pp. 91-101, Dec. 2018. E. Tipton, L. Hedges, M. Vaden-Kiernan, G. Borman, K. Sullivan, and S. Caverly, "Sample selection in randomized experiments: a new method using propensity score stratified sampling," Journal of Research on Educational Effectiveness, vol. 7, no. 1, pp. 114-135, Jan. 2014.

[20].M. Sugeno and G. Kang, "Structure identification of fuzzy model," Fuzzy Sets and Systems, vol. 28, no. 1, pp. 15-33, Oct. 1988.

[21]. "Secondary Drinking Water Standards: Guidance for Nuisance Chemicals," [Online]. Available: https://www.epa.gov/dwstandardsregulations/secondary-drinking-water-standards-guidance-nuisance-chemicals.

[22]. A. K.Venkatesan, S. Ahmad, W. Johnson, J. R. Batista, "Systems dynamic model to forecast salinity load to the Colorado River due to urbanization within the Las Vegas Valley," Science of the Total Environment of Research on Educational Effectiveness, vol. 469, no. 13, pp. 2616-2625, Apr. 2011.

[23]. S. Maiti and R. K. Tiwari, "A comparative study of artificial neural networks, Bayesian neural networks and adaptive neuro-fuzzy inference system in groundwater level prediction,"

[24].C. Cheng, T. Chen and C. Chiang, "Trend-Weighted Fuzzy Time-Series Model for TAIEX Forecasting," International Conference on Neural Information Processing, vol. 3, pp. 469-477, Mar. 2005.

[25].J. H. Holland, Adaptation in Natural and Artificial Systems, MA, Cambridge: MIT Press, 1992.

[46] R. Hinterding, H. Gielewski, and T. C. Peachey. "The Nature of Mutation in Genetic Algorithms,, in Proc. 6th International Conference on Genetic Algorithms, pp. 65 72, 1995.

Certificate

This is to certify that Mr. Aritra Nandy, of Asansol Engineering College, registration number: 201080571010051, has successfully completed a project on Water Quality Prediction using *Machine Learning with Python* under the guidance of **Prof. Arnab Chakraborty**.

Prof. Arnab Chakraborty

Certificate

This is to certify that Mr. Bikram Roy, of Asansol Engineering College, registration number: 201080571010002, has successfully completed a project on Water Quality Prediction using *Machine Learning with Python* under the guidance of **Prof. Arnab Chakraborty**.

Prof. Arnab Chakraborty

Certificate

This is to certify that Mr. Deep Mondal, of Asansol Engineering College, registration number: 201080571010045, has successfully completed a project on Water Quality Prediction using *Machine Learning with Python* under the guidance of **Prof. Arnab Chakraborty**.

Prof. Arnab Chakraborty

Certificate

This is to certify that Mr. Sanjib Mondal of Asansol Engineering College, registration number: 201080571010012, has successfully completed a project on Water Quality Prediction using *Machine Learning with Python* under the guidance of **Prof. Arnab Chakraborty**.

Prof. Arnab Chakraborty

Certificate

This is to certify that Mr. Sudip Chatterjee of Asansol Engineering College, registration number: 201080571010016, Water Quality Prediction using *Machine Learning with Python* under the guidance of **Prof. Arnab Chakraborty**.

Prof. Arnab Chakraborty