COMPARATIVE ANALYSIS OF ALGORITHMS FOR WATER QUALITY PREDICTION

A COURSE PROJECT REPORT

By

THANGALA NITHIN KUMAR REDDY
(RA2111026010117)

KALVA BHAGEERATH
(RA2111026010132)

Under the guidance of

DR. M. FERNI UKRIT

In partial fulfillment for the Course

of

18CSE479T - STATISTICAL

MACHINE LEARNING

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Chengalpattu District

NOVEMBER 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this mini project report "COMPARATIVE ANALYSIS OF ALGORITHMS FOR WATER QUALITY PREDICTION" is the Bonafide work of **THANGALA NITHIN KUMAR REDDY (RA2111026010117), KALVA BHAGEERATH (RA2111026010072)** who carried out the project work under my supervision.

SIGNATURE

Dr. M. Ferni Ukrit

Associate professor

CINTEL

SRM Institute of Science and Technology

Kattankulathur

SIGNATURE

Dr. R Annie Uthra

Professor and Head

CINTEL

SRM Institute of Science and Technology

Kattankulathur

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors. We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We are highly thankful to our Course project Faculty **Dr. M. Ferni Ukrit , Associate Professor , CINTEL,** for his/her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **HoD ,Dr. R Annie Uthra, CINTEL** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

ABSTRACT

The destruction of natural water resources, such as lakes and rivers, is one of the most important and dangerous issues currently plaguing humanity. Polluted water has long-term repercussions on all facets of existence. In order to maximise the quality of your water, it is crucial to manage your water resources. The impacts of water contamination can be efficiently managed if data are analysed and water quality can be forecasted. This study's objective is to develop a model for predicting water quality based on measurements of water quality using machine learning.The use of machine learning can be utilised to create models using algorithms using information acquired. The collected data will be pre-processed, divided into training and testing portions, and exposed to machine learning classification techniques for a better assessment of parametric findings.

TABLE OF CONTENTS

1. INTRODUCTION

Access to clean and safe water is a fundamental human right and a vital resource for sustaining life and ecosystems. Ensuring water quality is of paramount importance to safeguard public health and protect the environment. Monitoring and predicting water quality in various bodies of water, such as rivers, lakes, and reservoirs, is a critical task for governments, environmental agencies, and researchers worldwide. Traditional water quality monitoring methods are often time-consuming, expensive, and limited in spatial coverage. In contrast, machine learning offers a promising solution to address these challenges and enhance our ability to predict and manage water quality effectively.

Water quality prediction using machine learning leverages the power of data-driven modeling to analyze historical water quality data, environmental variables, and other relevant parameters. By developing predictive models, it becomes possible to forecast water quality indicators, such as the concentration of pollutants, the presence of harmful algae blooms, or the levels of dissolved oxygen, with greater accuracy and efficiency.

It explores the significance of water quality prediction using machine learning and its potential benefits across various sectors. It highlights the growing need for accurate, timely, and scalable solutions to address the complex challenges posed by changing environmental conditions, urbanization, and industrial activities. Moreover, it discusses the potential impact of machine learning in improving water resource management, ensuring safe drinking water, and preserving aquatic ecosystems.

2. LITERATURE SURVEY

S.NO	Title	Author name	Journal name	Methodologies	Inference
1.	Water Quality Analysis and prediction using machine learning	1)Dr.D Brindha 2)Viswanath puli 3)Vamsi stephen 4)Bala NVSS 5)Guru Dinesh	IEEE	1) Data Collection 2) Data preprocessing 3)Data Analysis a)Normalization b)Bivariate Analysis c)Correlation Analysis 4)water Quality Index 5) Machine learning Algorithms 6) Data Splitting	By using above methods the work provides high accuracy model, model also accepts user input.as the work is based on ml algorithms, Random forest ,Regression squared error has higher weightage after training and testing,so it predicts the score utilizing squared error.
2.	Water Quality Analysis using Machine Learning	1) Ravi Akshay 2) Gadiraju Tarun 3) uday kiran 4) Durga Devi 5)M Vidhyalakshmi	IEEE	1)Data Description 2)Data pre-processing 3) Histogram 4)Correlation Matrix 5)Data Standardization 6)Training and testing of Data 7)ML Algorithms 8)Performance Metrics	By this paper,they conclude that future cities would benefit on water quality due to ml techniques. Hyperparameter Tuning along with Random forest classifier, given an f1 score of around 70%

3.	Predicting Water Quality Parameters using Machine Learning	1) Nikhil M 2) Ravishankar Holla 3) Manju G	IEEE	1) Dataset 2)preprocessing 3) Levenberg algorithm- mix of gradient decent and Gauss-NewtonAlgorithm 4)Testing	In this paper, their main focus being the use of machine learning algorithm in the field of monitoring, Accuracy- Chlorides- 83.% hardness-87.9% sulphate-81.73% Alkalinity-79.4%
4.	Water Quality Prediction Using Machine Learning Learning Techniques	1)Nithya pagadala 2)Moulika Marri 3)Akshaya Myla 4)Bhargav Abburi 5)k. Shri Ramtej	IEEE	1)Data Collection 2)Exploratory Data Analysis 3)Data Pre-processing 4)check data Imbalance 5)splitting the data 6)model training 7)model evaluation	In this paper, presents a comparative study of ml models used in classifying water as potable or not. By comparing all the algorithm RF model gives promising results. Accuracy: 1)KNN-76.5% 2)SVM-76.5% 3)RF-91.0% 4)DT-92%
5.	Predicting and Analyzing Water Quality using Machine Learning: A comprehensive model	1) Yafra Khan 2) Chai Soo See	IEEE	1) Artificial Neural Network 2) Neural Network for Time series	This paper analyzes and forecasts the values of water parameters. ANN with Nonlinear Autoregressive(NAR) time series has been used with SCG and training algorithms. The proposed model comprising of ANN-NAR, best regression value for specific conductance(0.99)

6.	Predictive Models for river Water Quality using Machine Learning and Big Data Technique- A survey	1) Jitha P Nair 2) Vijaya M S	IEEE	1) water Treatment 2) water Quality Standards 3) Correlation Analysis in Water Quality prediction 4)computational methods- A)Time series Analysis for WQP B)Machine Learning for WQP C)Deep Learning for WQP	This paper, explains the different water treatment methods, water quality standards and source of their emission in detail. Various machine learning,time series analysis and deep learning techniques involved in identifying and evaluating the quality of water.
7.	Water Quality Analysis Using Deep Learning	1) Ankit chahar 2) Ayanesh Chowdhury 3)Bharath Kumar 4) Vishan Reddy 5)Harshal Patel 6)Ninad Patil	IEEE	1) Data preparation 2) Data preprocessing 3) Feature Visualisation 4)Model Architecture	The entire process that the authors had performed, which started from data preparation followed by data preprocessing and feature visualization which was quite impactful in the model performance as the output that the authors got was Exceptional. The authors received an accuracy of 89 percent for deep neural network and 92 percent for artificial neural network.

3. STATISTICAL ANALYIS

Statistical analysis Statistical analysis is the process of collecting, cleaning, summarizing, interpreting, and presenting data in order to discover patterns, relationships, or insights within that data. It involves using various statistical techniques and methods to make sense of data and draw meaningful conclusions. Statistical analysis is commonly used in a wide range of fields, including science, economics, social sciences, business, and many other disciplines, to make informed decisions, test hypotheses, and understand the underlying structures and trends within the data. Some common statistical methods and tools used in statistical analysis include descriptive statistics, inferential statistics, hypothesis testing, regression analysis, and data visualization. It includes the following

- Mean
- Median
- Mode
- Hypothetical Testing:
    1. T-test
    2. Anova (F-Test)
    3. Chi-Squared Test

3.1 MEAN, MEDIAN, MODE

MEAN:

The "**Mean**" refers to the average of a set of numbers. It is one of the most fundamental and commonly used measures of central tendency. To calculate the mean, you add up all the values in a dataset and then divide the sum by the number of values in that dataset. The formula for calculating the mean (often denoted as "μ" for a population mean or "$\bar{x}$" for a sample mean) is as follows:

For a population:

- $\mu = (\Sigma x) / N$

For a sample:

- $\bar{x} = (\Sigma x) / n$

CODE:

```
import pandas as pd

# Load the dataset into a DataFrame
data = pd.read_csv('C:\\Users\\tt263\\Downloads\\sml.csv')

# Calculate and display the mean for a specific column
column_name = 'ph'
mean_value = data[column_name].mean()
print(f"\nThe mean of '{column_name}' is: {mean_value}")

# Calculate and display the mean for a specific column
column_name = 'Hardness'
mean_value = data[column_name].mean()
print(f"\nThe mean of '{column_name}' is: {mean_value}")

# Calculate and display the mean for a specific column
column_name = 'Solids'
mean_value = data[column_name].mean()
print(f"\nThe mean of '{column_name}' is: {mean_value}")

# Calculate and display the mean for a specific column
column_name = 'Sulfate'
mean_value = data[column_name].mean()
print(f"\nThe mean of '{column_name}' is: {mean_value}")
```

Output:

```
The mean of 'ph' is: 7.122250444760001

The mean of 'Hardness' is: 196.21132702027

The mean of 'Solids' is: 21997.396719392003

The mean of 'Sulfate' is: 331.27334278980004
```

Statistical Analysis of Mean:

i) column_name = ph

Total no of observations = 1000

Sum of observations = 8.31 + 9.09 + ... + 6.975
= 7122.25

$$\text{Mean} = \frac{\Sigma x}{N}$$

$$= \frac{7122.25}{1000}$$

$$= 7.12225$$

∴ Mean of ph is 7.12225

ii) column_name = hardness

Sum of observations = 196211.3

$$\text{Mean}(\mu) = \frac{\Sigma x}{N}$$

$$= \frac{196211.3}{1000}$$

$$= 196.2113$$

iii) column_name = sulfate

Sum of observations (Σx) = 331273.3

$$\text{Mean}(\mu) = \frac{\Sigma x}{N}$$

$$= \frac{331273.3}{1000}$$

$$= 331.2733$$

MEDIAN:

- In statistics, the "Median" is a measure of central tendency that represents the middle value in a dataset when the values are arranged in ascending or descending order. To find the median, you first need to order the data from smallest to largest (or largest to smallest), and then identify the middle value.

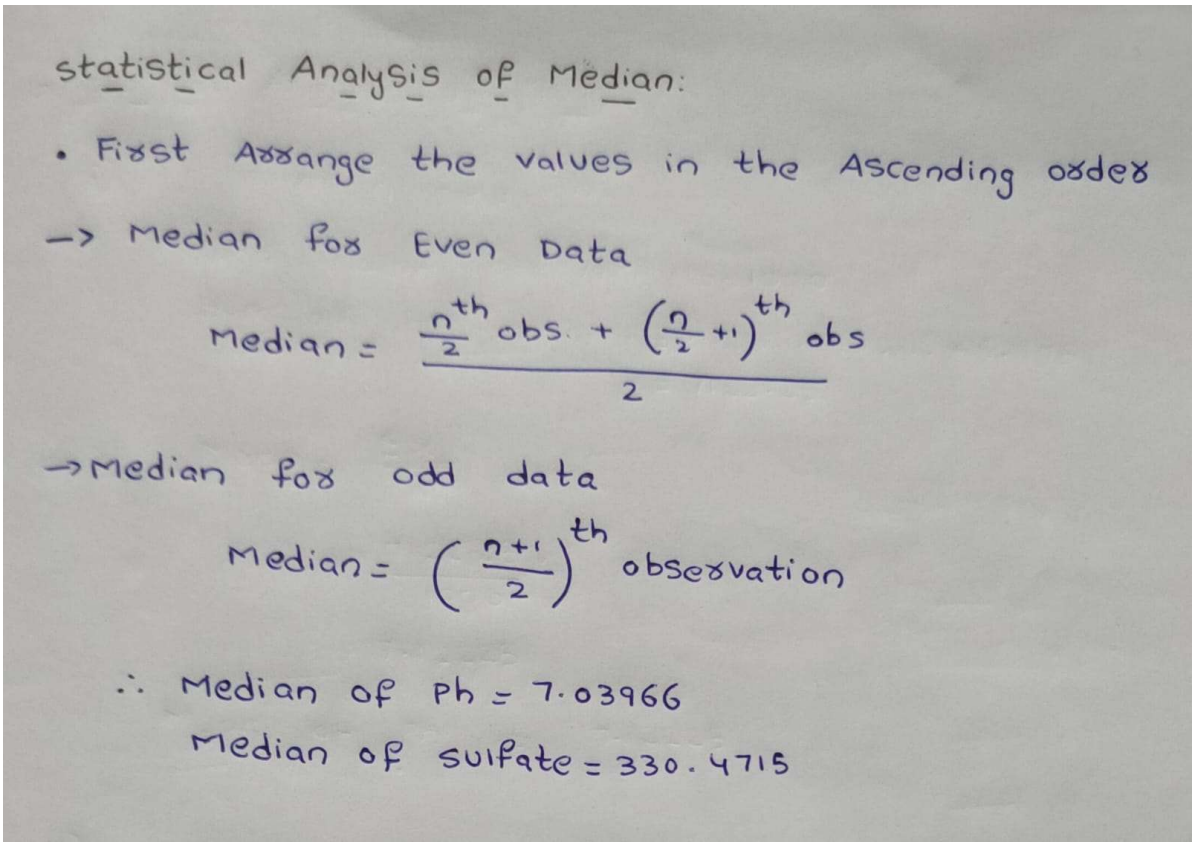
- The median is particularly useful when dealing with datasets that may contain outliers or extreme values because it is not affected by extreme values as much as the mean (average). It divides the data into two equal halves, with half of the values falling below it and half above it.

CODE:

```
import pandas as pd

# Load the dataset into a DataFrame
data = pd.read_csv('C:\\Users\\tt263\\Downloads\\sml.csv')

# Calculate and display the median for a specific column
column_name = 'ph'
median_value = data[column_name].median()
print(f"The median of '{column_name}' is: {median_value}")

column_name = 'Hardness'
median_value = data[column_name].median()
print(f"The median of '{column_name}' is: {median_value}")

column_name = 'Solids'
median_value = data[column_name].median()
print(f"The median of '{column_name}' is: {median_value}")

column_name = 'Sulfate'
median_value = data[column_name].median()
print(f"The median of '{column_name}' is: {median_value}")
```

Output:

```
The median of 'ph' is: 7.0396696495
The median of 'Hardness' is: 197.93481359999998
The median of 'Solids' is: 21065.07204
The median of 'Sulfate' is: 330.4715746
```



MODE:

In statistics, the "**Mode**" is a measure of central tendency that represents the value or values that occur most frequently in a dataset. In other words, it is the value that appears with the highest frequency among all the values in a given dataset.

A dataset can have one mode (unimodal), more than one mode (multimodal), or no mode at all if all values occur with equal frequency. Here are the key characteristics of the mode:

- **Unimodal:** A dataset is said to be unimodal when it has only one mode, which is the most frequently occurring value.
- **Multimodal:** A dataset is considered multimodal when it has more than one mode. In such cases, there can be two or more values that occur with the highest frequency.

No Mode: Some datasets may not have a mode at all if all values occur with the same frequency.

CODE:

```
import pandas as pd
# Load the dataset into a DataFrame
data = pd.read_csv('C:\\Users\\tt263\\Downloads\\sml.csv')

column_of_interest = 'ph'
mode = data[column_of_interest].mode()
print(f"Mode of {column_of_interest}: {mode}")
column_of_interest = 'Hardness'
mode = data[column_of_interest].mode()
print(f"Mode of {column_of_interest}: {mode}")
column_of_interest = 'Solids'
mode = data[column_of_interest].mode()
print(f"Mode of {column_of_interest}: {mode}")
column_of_interest = 'Sulfate'
mode = data[column_of_interest].mode()
print(f"Mode of {column_of_interest}: {mode}")
```

Output:

```
Mode of ph: 0       0.227499
1       0.989912
2       1.757037
3       2.569244
4       2.690831
          ...
995    11.244507
996    11.301794
997    11.534880
998    11.898078
999    12.246928
Name: ph, Length: 1000, dtype: float64
Mode of Hardness: 0       73.492234
1       77.459586
2       81.710895
3       94.091307
4       97.280909
          ...
995    280.089655
996    282.739017
997    283.997284
998    300.292476
999    306.627481
Name: Hardness, Length: 1000, dtype: float64
Mode of Solids: 0       1372.091043
1       2552.962804
2       4111.785432
3       4168.196994
4       4304.492483
          ...
995    48621.563950
996    49074.730410
997    55334.702800
998    56351.396300
999    56488.672410
Name: Solids, Length: 1000, dtype: float64
Mode of Sulfate: 0      129.000000
1      180.206746
2      182.397370
3      187.170714
4      187.424131
          ...
995    444.375731
996    444.970552
997    445.938391
998    475.737460
999    476.539717
Name: Sulfate, Length: 1000, dtype: float64
```

Statistical Analysis of Mode:

- It is a measure of value (or) values that occurs most frequently in the data set.

-> here in our data set all the coloumns are in the float value.

-> due to the presence of float value (upto 5 decimals) our data set doesn't contains mode value

∴ Mode of ph coloumn = 0

Mode of sulfate coloumn = 0

3.2 F-Test (ANNOVA)

The **F-test**, specifically in the context of analysis of variance (**ANOVA**), is a statistical test used to determine whether there are significant differences among the means of two or more groups or populations. **ANOVA**, which stands for "analysis of variance," is a statistical technique that partitions the total variance observed in a dataset into different sources of variation to assess whether there are statistically significant differences between the group means.

In summary, the **F-test** in **ANOVA** helps answer questions about whether there are statistically significant differences among group means, making it a valuable tool for exploring the impact of categorical independent variables on a continuous dependent variable.

CODE:

```
import statsmodels.api as sm
from statsmodels.formula.api import ols
import pandas as pd
# Load the dataset into a DataFrame
data = pd.read_csv('C:\\Users\\tt263\\Downloads\\sml.csv')

# Assuming 'target' is your dependent variable
model = ols('Potability ~ ph + Hardness + Solids +Chloramines+ Sulfate+ Organic_carbon+Turbidity ', data=data).fit()

# Perform the ANOVA
anova_table = sm.stats.anova_lm(model, typ=2)

# Extract the F-statistic and associated p-value
f_statistic = anova_table['F'][0]
p_value = anova_table['PR(>F)'][0]

print("F-Statistic:", f_statistic)
print("P-Value:", p_value)
```

Output:

```
F-Statistic: 2.799127220907765
P-Value: 0.09463092551889118
```

Statistical Analysis of F-Test:

Step 1: fix the NULL hypothesis (H_0) and the Alternative hypothesis (H_1) and $\alpha = 0.05$

step 2: find

$$\text{Df between} = a - 1$$
$$\text{Df within} = N - a$$
$$\text{Df Total} = N - 1$$

and F critical $= F(df \text{ between}, df \text{ within})$

Step 3:

- calculate mean of all groups
- And also calculate Grand mean
- SS Total $= \Sigma (x - \bar{x})^2$

  SS within $= \Sigma (x_i - \bar{x}_i)^2$

  SS between = SS Total − SS within

Step 4:

$$MS_{between} = \frac{SS_{between}}{dF_{between}}$$

$$MS_{within} = \frac{SS_{within}}{dF_{within}}$$

step 5:

$$F_{statistics} = \frac{MS_{between}}{MS_{within}}$$

→ After following the above stastics, we got the F-stastics value as 2.79912

3.3 T-TEST

A t-test, or Student's t-test, is a statistical hypothesis test used to determine whether there is a significant difference between the means of two groups or populations. It is particularly useful when you want to compare the means of two samples to assess whether any observed difference is statistically significant or if it could have occurred by chance.

The t-test is based on the t-distribution, which is similar to the normal distribution but is used when dealing with small sample sizes or when the population standard deviation is unknown. The t-distribution has heavier tails, which makes it more appropriate for small samples.

CODE-1:

```
import scipy.stats as stats
group_a_column = data['ph']
group_b_column = data['Hardness']

# Perform a two-sample t-test assuming unequal variances (Welch's t-test)
t_statistic, p_value = stats.ttest_ind(group_a_column, group_b_column, equal_var=False)

# Display the results
print("T-Statistic:", t_statistic)
print("P-Value:", p_value)

# Determine whether the difference is statistically significant at a significance level (e.g., alpha = 0.05)
alpha = 0.05
if p_value < alpha:
    print("The difference is statistically significant.")
else:
    print("The difference is not statistically significant.")
```

```
T-Statistic: -197.44600605736852
P-Value: 0.0
The difference is statistically significant.
```

CODE-2(for ph and organic_carbon):

```
import scipy.stats as stats
group_a_column = data['ph']
group_b_column = data['Organic_carbon']

# Perform a two-sample t-test assuming unequal variances (Welch's t-test)
t_statistic, p_value = stats.ttest_ind(group_a_column, group_b_column, equal_var=False)

# Display the results
print("T-Statistic:", t_statistic)
print("P-Value:", p_value)

# Determine whether the difference is statistically significant at a significance level (e.g., alpha = 0.05)
alpha = 0.05
if p_value < alpha:
    print("The difference is statistically significant.")
else:
    print("The difference is not statistically significant.")
```

```
T-Statistic: -60.94236185551237
P-Value: 0.0
The difference is statistically significant.
```

CODE-3(for Turbidity & solids**):**

```
import scipy.stats as stats
group_a_column = data['Turbidity']
group_b_column = data['Solids']

# Perform a two-sample t-test assuming unequal variances (Welch's t-test)
t_statistic, p_value = stats.ttest_ind(group_a_column, group_b_column, equal_var=False)

# Display the results
print("T-Statistic:", t_statistic)
print("P-Value:", p_value)

# Determine whether the difference is statistically significant at a significance level (e.g., alpha = 0.05)
alpha = 0.05
if p_value < alpha:
    print("The difference is statistically significant.")
else:
    print("The difference is not statistically significant.")
```

```
T-Statistic: -78.41265882485668
P-Value: 0.0
The difference is statistically significant.
```

3.4 CHI-TEST

A chi-squared test (χ^2 test) is a statistical hypothesis test used to determine whether there is a significant association or relationship between two categorical variables in a contingency table. It is a non-parametric test, meaning it does not assume a specific distribution for the data, and it is commonly used in statistics to analyse data where variables are categorical rather than continuous.

CODE:

```
import scipy.stats as stats

# Perform the chi-square test of independence
chi2_stat, p_value, dof, expected = stats.chi2_contingency(contingency_table)

# Display the results
print("Chi-Square Statistic:", chi2_stat)
print("P-Value:", p_value)
print("Degrees of Freedom:", dof)
alpha = 0.05
if p_value < alpha:
    print("There is a significant association between the variables.")
else:
    print("There is no significant association between the variables.")
```

OUTPUT:

```
Chi-Square Statistic: 999000.0000000017
P-Value: 0.23967673147112914
Degrees of Freedom: 998001
There is no significant association between the variables.
```

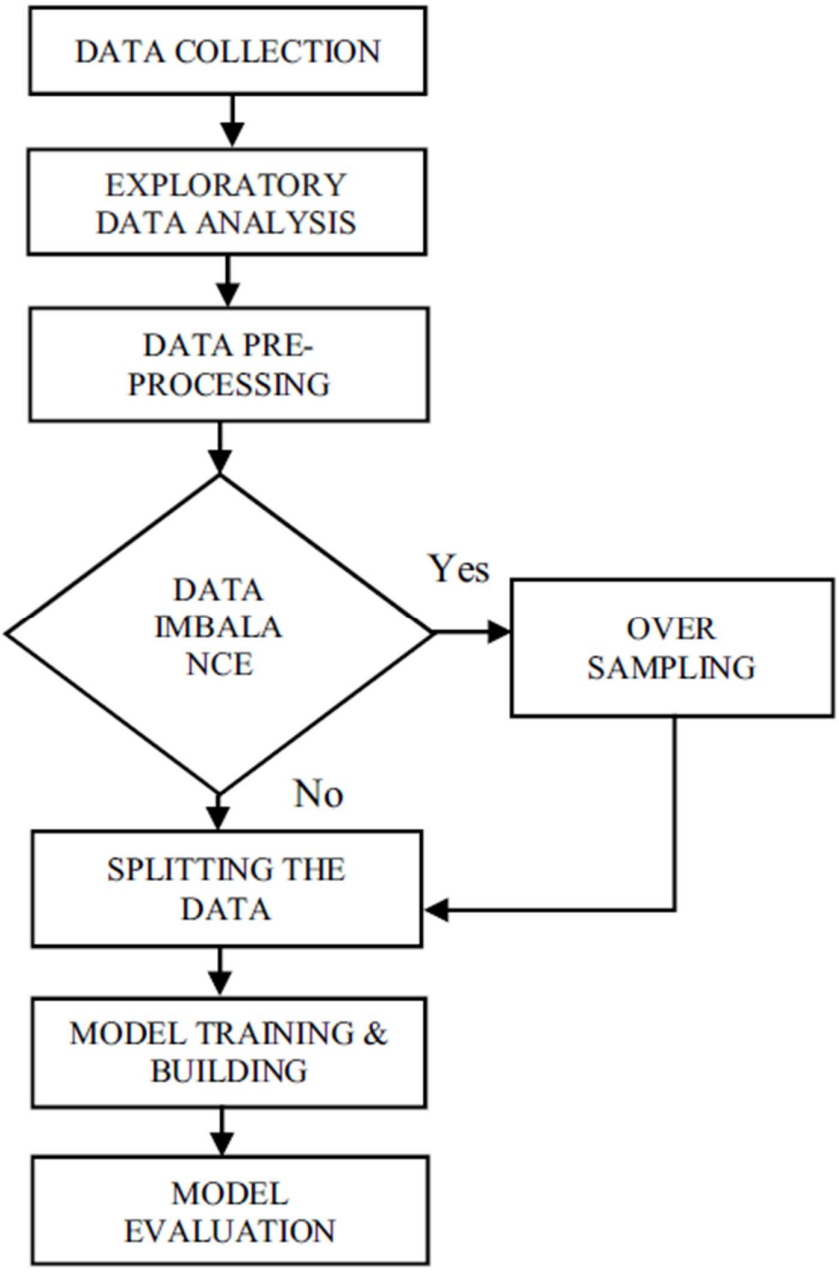
4. SUPERVISED LEARNING

Supervised learning is a type of machine learning paradigm where an algorithm learns to make predictions or decisions based on input data that is labelled or paired with the corresponding correct output. In other words, the algorithm is provided with a dataset in which the input data and their associated target or output values are provided. The primary goal of supervised learning is to learn a mapping or relationship between the input and output so that the algorithm can make accurate predictions or classifications on new, unseen data.

SUPERVISED LEARNING ALGORITHMS:

- Linear Regression
- Logistic Regression
- Decision Tree
- Random Forest
- K-Nearest Neighbor(KNN)
- Support Vector Machine
- Artificial Neural Network

BLOCK DIAGRAM:



4.1 LINEAR REGRESSION

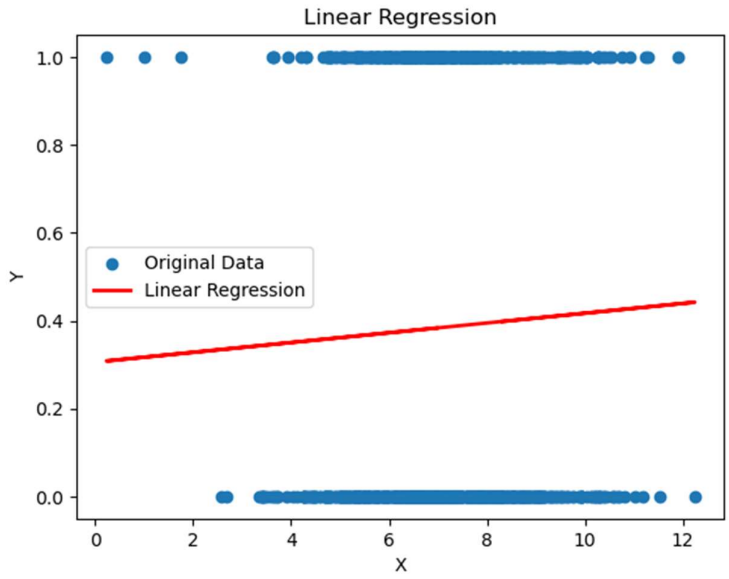
Linear regression is a statistical and machine learning model used for analyzing and modeling the relationship between a dependent variable and one or more independent variables. The goal of linear regression is to find the linear relationship that best describes how changes in the independent variables are associated with changes in the dependent variable. In essence, it seeks to fit a straight line to the data points in a way that minimizes the difference between the observed and predicted values of the dependent variable.

CODE(ph vs Potability)**:**

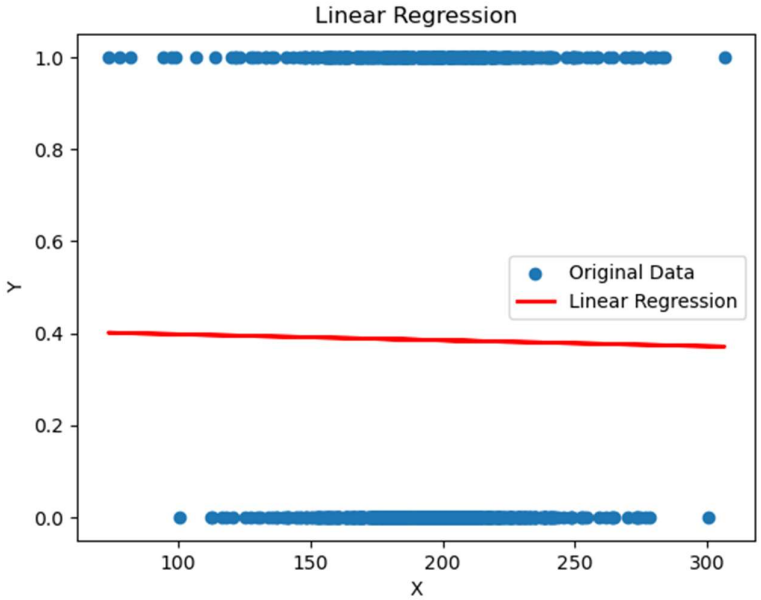
```
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
X = data[['ph']]  # Independent variable
Y = data['Potability']     # Dependent variable
# Create a linear regression model
model = LinearRegression()
# Fit the model to the data
model.fit(X, Y)
# Get the coefficients (slope and intercept)
slope = model.coef_[0]
intercept = model.intercept_

# Predict Y values based on the model
predicted_Y = model.predict(X)

# Print the coefficients
print("Slope (Coefficient):", slope)
print("Intercept:", intercept)

# Plot the original data points and the regression line
plt.scatter(X, Y, label='Original Data')
plt.plot(X, predicted_Y, color='red', linewidth=2, label='Linear Regression')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.title('Linear Regression')
plt.show()
```

output:

```
Slope (Coefficient): 0.011121849892867805
Intercept: 0.3057873996539683
```



Similarly for Hardness and potability:




```
Slope (Coefficient): -0.00012932022813911035
Intercept: 0.4103740935737389
```

Statistical Analysis of Linear Regression:

- for multiple linear Regression -

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots\ldots \beta_n x_n$$

here,

Y → dependent variable

$x_1, x_2, x_3 \ldots x_n$ → Independent variable

$\beta_0, \beta_1, \ldots\ldots \beta_n$ → Regression co-efficients

→ If there are two groups (x & Y)

step 1: calculate mean of x and Y

step 2: calculate slope (b_1) and intercept (b_0)

$$\text{Slope, } b_1 = \frac{\Sigma(x_i - \bar{x})(Y_i - \bar{Y})}{\Sigma(x_i - \bar{x})^2}$$

$$\text{Intercept } b_0 = \bar{Y} - b_1 \cdot \bar{x}$$

step 3: find the Regression equation,

$$\boxed{Y = b_0 + b_1 x}$$

→ By using the above stastics for the Ph and Potability we got -

Slope : 0.01112

Intercept : 0.30578

4.2 LOGISTIC REGRESSION

Logistic regression is a statistical and machine learning model used for binary classification tasks, which involve predicting one of two possible outcomes or classes (typically represented as 0 and 1). Despite its name, logistic regression is a classification algorithm rather than a regression algorithm.

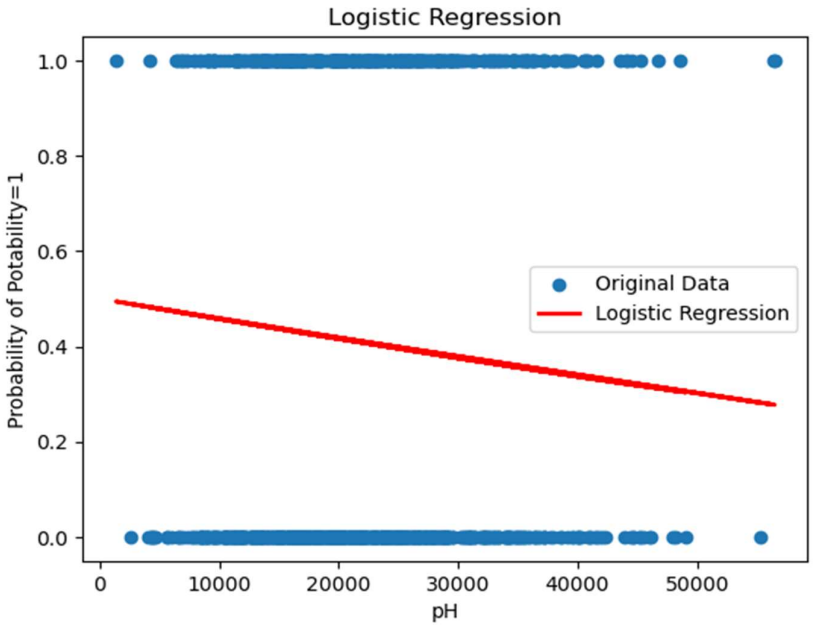
The fundamental idea behind logistic regression is to model the probability of an observation belonging to one of the two classes as a function of one or more predictor variables. It uses the logistic function (also called the sigmoid function) to transform a linear combination of the predictor variables into a value between 0 and 1, representing the probability of the positive class. The logistic function has an S-shaped curve, which is useful for modeling probabilities.

CODE:

```
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt

# Assuming 'pH' is your independent variable and 'Potability' is your dependent variable
X = data[['Solids']]  # Independent variable (features)
Y = data['Potability']  # Binary dependent variable (0 or 1)

# Create a logistic regression model
model = LogisticRegression()
# Fit the model to the data
model.fit(X, Y)

# Get the coefficients (slope and intercept)
slope = model.coef_[0][0]
intercept = model.intercept_[0]

# Predict probabilities of class 1 (Potability=1) based on the model
predicted_probs = model.predict_proba(X)[:, 1]

# Print the coefficients
print("Coefficient (Slope):", slope)
print("Intercept:", intercept)

# Plot the original data points and the logistic regression curve
plt.scatter(X, Y, label='Original Data', marker='o')
plt.plot(X, predicted_probs, color='red', linewidth=2, label='Logistic Regression')
plt.xlabel('pH')
plt.ylabel('Probability of Potability=1')
plt.legend()
plt.title('Logistic Regression')
plt.show()
```

output:

```
Coefficient (Slope): -1.6953267335935837e-05
Intercept: -1.3681556177533695e-09
```



ACCURACY FOR LOGISTIC REGRESSION:

CODE:

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Load your dataset (replace 'your_dataset.csv' with your data file)
data = pd.read_csv('C:\\Users\\tt263\\Downloads\\sml.csv')

# Assuming 'X' contains your independent features, and 'y' contains your binary labels (0 or 1)
X = data[['ph', 'Hardness', 'Solids','Chloramines','Sulfate','Organic_carbon','Turbidity']]  # Independent variables
y = data['Potability']  # Binary labels

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a logistic regression model
model = LogisticRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Predict the labels for the testing data
y_pred = model.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)
```

Output:

```
Logistic Regression model accuracy (in %): 67.0
```

Statistical Analysis of Logistic Regression:

- Logistic function:

$$P = \frac{1}{1+e^{-L}}$$

$$L = b_0 + b_1 * x_1 + b_2 * x_2$$

b_0 - intercept
b_1 - first regression co-efficient
b_2 - second regression co-efficient
x_1 - first predictor (input)
x_2 - second predictor (input

→ The new co-efficients values can be calculated using -

$$b_1 = \frac{\Sigma (x_1 - \bar{x}_1)(x_2 - \bar{x}_2)}{\Sigma (x_1 - \bar{x}_1)^2}$$

$$b_2 = \frac{\Sigma (x_1 - \bar{x}_1)(x_2 - \bar{x}_2)}{\Sigma (x_2 - \bar{x}_2)^2}$$

$$b_0 = \bar{x}_2 - b_1 * \bar{x}_1$$

Factors influencing the accuracy of a logistic regression model for water quality prediction include:

- **Data Quality & Size**: The quality and size of the training data significant ly impact the model's accuracy. A high-quality, adequately sized training dataset is crucial for the model to learn the underlying patterns effectively .

- **Feature Selection**: Selecting the most relevant features from the available attributes can improve the model's accuracy by reducing noise and focusin g on the most informative predictors.

- **Model Regularization**: Regularization techniques, such as L1 or L2 regularization, can help prevent overfitting and improve the model's generalizability.

- **Hyperparameter Optimization**: Tuning hyperparameters, such as the regularization strength or the optimization algorithm, can further enhance the model's performance.

4.3 DECISION TREE

A decision tree is a supervised machine learning algorithm used for both classification and regression tasks. It is a tree-like structure that recursively splits the dataset into subsets based on the most significant attribute or feature at each node. Each node in the decision tree represents a decision or test on one or more input features, and each branch from a node represents a possible outcome of that decision. The leaves of the tree represent the final decisions or predictions.

Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,classification_report, confusion_matrix

# Load the dataset
data = pd.read_csv('C:\\Users\\tt263\\Downloads\\sml.csv')

# Split the dataset into features and target variable
X = data[['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'Organic_carbon', 'Turbidity']]
y = data['Potability']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create the decision tree classifier
clf = DecisionTreeClassifier()

# Train the classifier using the training data
clf.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = clf.predict(X_test)

# Calculate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Confusion Matrix:\n", confusion_mat)
print("Classification Report:\n", classification_rep)
```

Output:

```
Accuracy: 0.715
Confusion Matrix:
 [[161  70]
 [115  57]]
Classification Report:
              precision    recall  f1-score   support

           0       0.58      0.70      0.64       231
           1       0.45      0.33      0.38       172

    accuracy                           0.54       403
   macro avg       0.52      0.51      0.51       403
weighted avg       0.53      0.54      0.53       403
```

Factors that affect the accuracy of a decision tree model for water quality prediction:

- **The quality and size of the training data:** The decision tree algorithm can only learn from the data that it is given. Therefore, it is important to have a high-quality training dataset that is large enough to represent the full range of water quality conditions that the model will encounter in the real world.
- **The choice of decision tree algorithm**: Different decision tree algorithms have different strengths and weaknesses. Therefore, it is important to choose an algorithm that is appropriate for your dataset.
- **The hyperparameters of the decision tree algorithm:** Hyperparameters are parameters that control the learning behavior of the decision tree algorithm. There are many different hyperparameters that can be tuned, and the optimal values for these hyperparameters will vary depending on the dataset.

4.4 RANDOM FOREST

A Random Forest Algorithm is a supervised machine learning algorithm that is extremely popular and is used for Classification and Regression problems in Machine Learning. We know that a forest comprises numerous trees, and the more trees more it will be robust. Similarly, the greater the number of trees in a Random Forest Algorithm, the higher its accuracy and problem-solving ability.

Random Forest is a classifier that contains several decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. It is based on the concept of ensemble learning which is a process of combining multiple classifiers to solve a complex problem and improve the performance of the model.

CODE:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Load the dataset
data = pd.read_csv('C:\\Users\\tt263\\Downloads\\sml.csv')

# Split the dataset into features and target variable
X = data[['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'Organic_carbon', 'Turbidity']]
y = data['Potability']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create the random forest classifier
clf = RandomForestClassifier()

# Train the classifier using the training data
clf.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = clf.predict(X_test)

# Calculate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Output:

```
Accuracy: 0.8
```

Factors that affect the accuracy of a random forest model for water quality prediction:

- The quality and size of the training data. The random forest algorithm can only learn from the data that it is given. Therefore, it is important to have a high-quality training dataset that is large enough to represent the full range of water quality conditions that the model will encounter in the real world.
- The number of trees in the forest. The number of trees in the forest is a hyperparameter that controls the complexity of the model. More trees can lead to a more accurate model, but it can also make the model slower to train and predict.
- The maximum depth of each tree. The maximum depth of each tree is another hyperparameter that controls the complexity of the model. Deeper trees can lead to a more accurate model, but they can also make the model more prone to overfitting.

4.5 K-NEAREST NEIGHBOR

K-nearest neighbors (KNN) is a supervised machine learning algorithm used for classification and regression tasks. It is a simple and intuitive algorithm that can be used for both types of problems.

In K-NN, the fundamental idea is that data points with similar characteristics are typically close to each other in the feature space. The algorithm makes predictions based on the majority class (for classification) or the average of nearby data points (for regression) among the K-nearest neighbors to a given data point.

CODE:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Load the dataset
data = pd.read_csv('C:\\Users\\tt263\\Downloads\\sml.csv')

# Split the dataset into features and target variable
X = data[['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'Organic_carbon', 'Turbidity']]
y = data['Potability']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create the KNN classifier
clf = KNeighborsClassifier()

# Train the classifier using the training data
clf.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = clf.predict(X_test)

# Calculate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Output:

```
Algorithm: K-Nearest Neighbors
Accuracy: 0.705
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.79      0.78       130
           1       0.58      0.54      0.56        70

    accuracy                           0.70       200
   macro avg       0.67      0.67      0.67       200
weighted avg       0.70      0.70      0.70       200
```

Factors that affect the accuracy of a KNN model for water quality prediction:

- **The quality and size of the training data:** The KNN algorithm can only learn from the data that it is given. Therefore, it is important to have a high-quality training dataset that is large enough to represent the full range of water quality conditions that the model will encounter in the real world.

- **The choice of distance metric:** The distance metric is a measure of the similarity between two water samples. The choice of distance metric can have a significant impact on the accuracy of the KNN model.

- **The value of K:** The value of K is a hyperparameter that controls the complexity of the model. Higher values of K can lead to a more accurate model, but they can also make the model more prone to overfitting.

4.6 SUPPORT VECTOR MACHINE

Support Vector Machine or SVM is one of the Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points or vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

CODE:

```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
X = data.drop(columns=['Potability'])
y = data['Potability']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = SVC(kernel='linear', random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
confusion_mat = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
print("Accuracy:", accuracy)
print("Confusion Matrix:\n", confusion_mat)
print("Classification Report:\n", classification_rep)
```

Output:

```
Algorithm: Support Vector Machine
Accuracy: 0.815
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.93      0.87       130
           1       0.82      0.60      0.69        70

    accuracy                           0.81       200
   macro avg       0.82      0.77      0.78       200
weighted avg       0.82      0.81      0.81       200
```

Factors that affect the accuracy of an SVM model for water quality prediction:

- **The quality and size of the training data:** SVMs are data-driven models ,so the quality and size of the training data have a significant impact on the accuracy of the model. SVMs require large datasets to train effectively .
- **The choice of kernel function:** The kernel function that you choose will also affect the accuracy of the model. The kernel function should be able to capture the relationships between the input attributes in a way that is relevant to the target variable.
- **The hyperparameters of the SVM model:** SVM models have a number of hyperparameters that can be tuned to improve the accuracy of the mode l. These hyperparameters include the regularization parameter and the C parameter.

4.7 ARTIFICAL NEURAL NETWORK

- Artificial neural networks are a type of machine learning algorithm that are modelled after the structure and function of the human brain.

- They consist of interconnected nodes, or "neurons," that process and transmit information through a network of layers.

- Each neuron receives input from other neurons, processes that input, and then sends its output to other neurons in the network.

- Neural networks can be trained to recognize patterns and make predictions based on input data, and they are used in a wide range of applications such as image and speech recognition, natural language processing, and predictive analytics.

CODE:

```
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report

# Load your water quality dataset using pandas
data = pd.read_csv('C:\\Users\\tt263\\Downloads\\sml.csv')

# Assuming 'target_column_name' is the name of the target variable column.
# You can replace it with the actual column name from your dataset.
X = data.drop(columns=['Potability']) # Features
y = data['Potability']  # Target variable

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the features (mean=0, std=1)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Create an ANN model
model = tf.keras.models.Sequential([
    tf.keras.layers.Input(shape=(X_train.shape[1],)),  # Input layer
    tf.keras.layers.Dense(128, activation='relu'),  # Hidden layer with 128 neurons and ReLU activation
    tf.keras.layers.Dense(64, activation='relu'),   # Hidden layer with 64 neurons and ReLU activation
    tf.keras.layers.Dense(1, activation='sigmoid')  # Output layer with 1 neuron and Sigmoid activation
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=20, batch_size=32, validation_data=(X_test, y_test))

# Evaluate the model
y_pred = model.predict(X_test)
y_pred_binary = (y_pred > 0.5).astype(int)
accuracy = accuracy_score(y_test, y_pred_binary)
report = classification_report(y_test, y_pred_binary)

print(f"Accuracy: {accuracy}")
print("Classification Report:\n", report)
```

Output:

```
Accuracy: 0.78
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.84      0.83       130
           1       0.69      0.67      0.68        70

    accuracy                           0.78       200
   macro avg       0.76      0.75      0.76       200
weighted avg       0.78      0.78      0.78       200
```

Factors that affect the accuracy of an ANN model for water quality prediction:

- **The quality and size of the training data:** ANNs are data-driven models, so the quality and size of the training data have a significant impact on the accuracy of the model. ANNs require large datasets to train effectively.
- **The choice of ANN architecture:** The ANN architecture that you choose will also affect the accuracy of the model. The architecture should be complex enough to learn the relationships between the input and output data, but not so complex that it overfits the training data.
- **The choice of ANN algorithm:** The ANN algorithm that you choose will also affect the accuracy of the model. The algorithm should be able to train the ANN model effectively without overfitting the training data.
- **The hyperparameters of the ANN model**: ANN models have a number of hyperparameters that can be tuned to improve the accuracy of the model. These hyperparameters include the learning rate, the number of epochs, and the regularization parameters.

5. UNSUPERVISED LEARNING

Unsupervised learning is a machine learning technique in which algorithms are used to identify patterns and structures in data without the need for labelled output or human intervention. It is often contrasted with supervised learning, where models are trained on labelled data.

Unsupervised learning is valuable for data exploration, anomaly detection, and preparing data for further analysis. It plays a crucial role in various domains, including data mining, natural language processing, and recommendation systems.

Unsupervised learning Algorithms:

1) K-means
2) Principal Component Analysis

5.1 K-MEANS

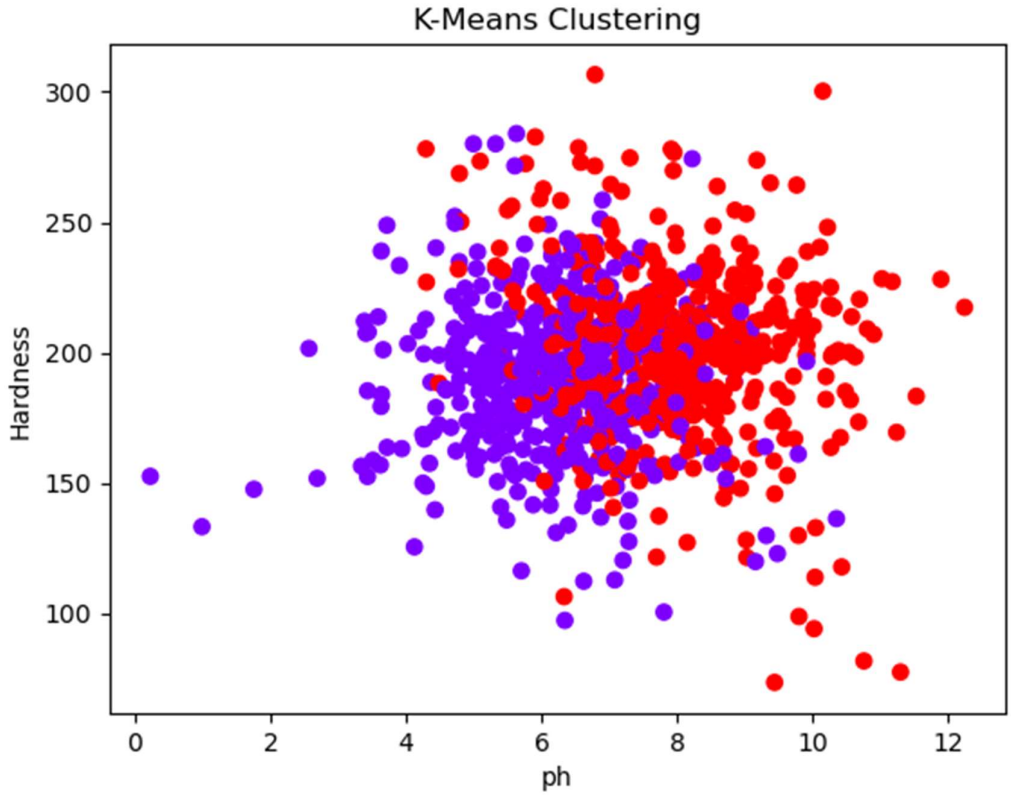
K-means clustering is a machine learning technique used for grouping data into distinct clusters based on their similarity. It involves selecting an initial number of clusters (K), assigning data points to the nearest cluster center, and iteratively updating these centers to minimize the variance within each cluster. K-means is widely employed for data segmentation and pattern recognition, although it assumes spherical, equally-sized clusters and can be sensitive to the initial centroid selection.

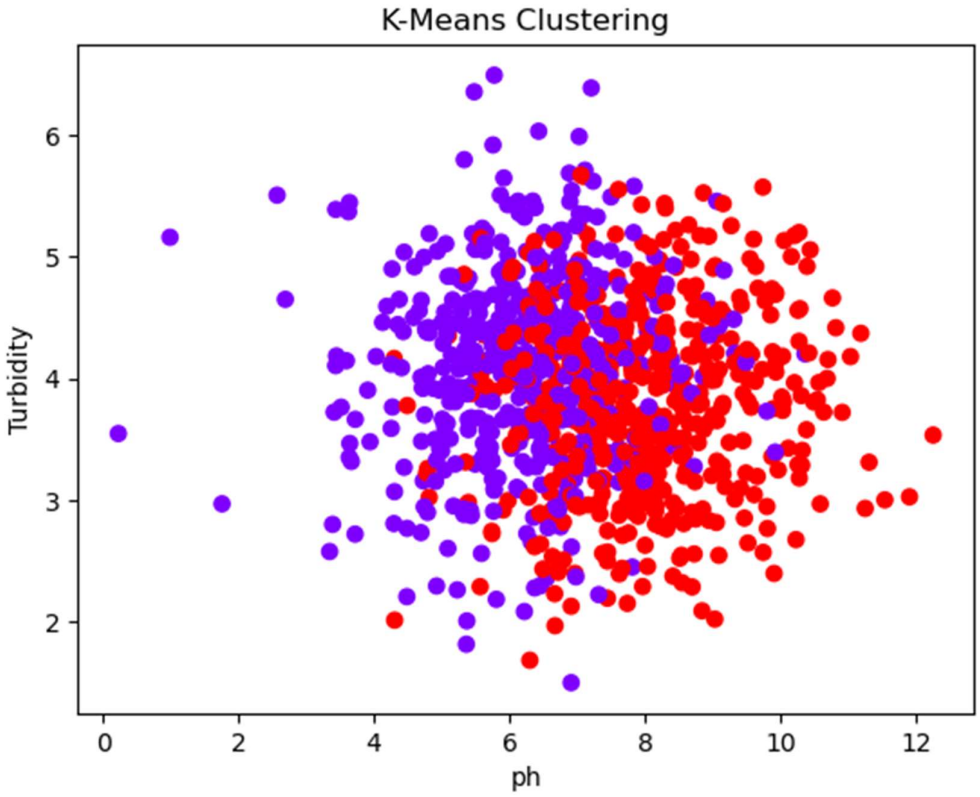
To use K-means effectively, it's important to carefully choose the value of K and consider its limitations, such as sensitivity to initialization and assumptions about cluster shapes and sizes. Despite these drawbacks, K-means remains a popular and efficient unsupervised learning method for various applications, including customer segmentation, image compression, and data analysis.

CODE:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

# Load your water quality dataset using pandas
data = pd.read_csv('C:\\Users\\tt263\\Downloads\\sml.csv')

# Assuming 'target_column_name' is the name of the target variable column.
# You can replace it with the actual column name from your dataset.
X = data.drop(columns=['Potability'])  # Features

# Standardize the features (mean=0, std=1)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Create a K-Means model with the desired number of clusters
k = 2  # You can change the number of clusters as needed
kmeans = KMeans(n_clusters=k, random_state=0)

# Fit K-Means to the standardized data
kmeans.fit(X_scaled)

# Get cluster labels for each data point
cluster_labels = kmeans.labels_

# Add the cluster labels to your dataset
data['Cluster'] = cluster_labels

# Visualize the clusters (for 2D data, you can adjust this code for higher-dimensional data)
plt.scatter(X['ph'], X['Hardness'], c=cluster_labels, cmap='rainbow')
plt.xlabel('ph')
plt.ylabel('Hardness')
plt.title('K-Means Clustering')
plt.show()
```

Output:


K-Means Clustering
Hardness
ph
300
250
200
150
100
0
2
4
6
8
10
12


K-Means Clustering
Turbidity
ph
6
5
4
3
2
0
2
4
6
8
10
12

Factors that affect the accuracy of a K-means model for water quality prediction:

- **The quality and size of the training data:** The K-means algorithm can only learn from the data that it is given. Therefore, it is important to have a high-quality training dataset that is large enough to represent the full range of water quality conditions that the model will encounter in the real world.
- **The choice of value for K:** The value of K is a hyperparameter that controls the number of clusters in the model. Choosing an incorrect value for K can lead to a decrease in the accuracy of the model.
- **The initialization of the cluster centroids:** The way that the cluster centroids are initialized can also affect the accuracy of the model. Initializing the cluster centroids randomly can lead to suboptimal results.

5.2 PRINCIPAL COMPONENT ANALYSIS

Principal Component Analysis (PCA) is a dimensionality reduction technique widely used in data analysis and machine learning. Its primary objective is to transform a dataset with a potentially large number of correlated variables into a new set of uncorrelated variables called principal components. These components are ordered in terms of the variance they explain, with the first principal component capturing the most variance in the data. By selecting a subset of these components, you can effectively reduce the dimensionality of your data while preserving as much relevant information as possible. PCA is often employed for visualization, noise reduction, and simplifying complex data for subsequent analysis.

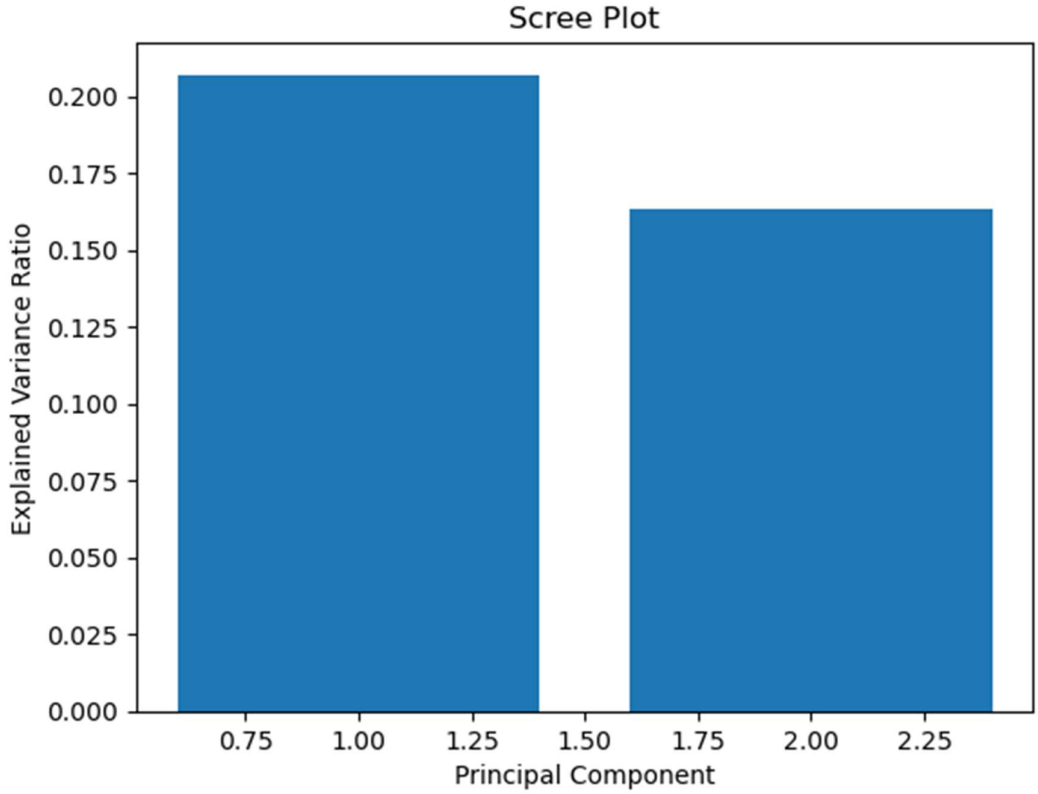
In summary, PCA is a valuable tool for simplifying high-dimensional data into a more manageable and interpretable form, making it easier to uncover patterns, reduce redundancy, and enhance the efficiency of various machine learning and statistical techniques.

CODE:

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Load your water quality dataset using pandas
data = pd.read_csv('C:\\Users\\tt263\\Downloads\\sml.csv')

# Assuming 'target_column_name' is the name of the target variable column.
# You can replace it with the actual column name from your dataset.
X = data.drop(columns=['Potability'])  # Features

# Standardize the features (mean=0, std=1)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Create a PCA model with the number of components you want to retain
# You can adjust the number of components based on your requirements.
n_components = 2  # Example: retaining 2 principal components
pca = PCA(n_components=n_components)

# Fit the PCA model to the standardized data and transform the data
X_pca = pca.fit_transform(X_scaled)

# Create a DataFrame with the principal components
pca_df = pd.DataFrame(data=X_pca, columns=[f'PC{i+1}' for i in range(n_components)])

# Visualize the explained variance ratio
explained_variance = pca.explained_variance_ratio_
print(f'Explained Variance Ratios: {explained_variance}')

# Plot the explained variance in a scree plot
plt.bar(range(1, n_components + 1), explained_variance)
plt.xlabel('Principal Component')
plt.ylabel('Explained Variance Ratio')
plt.title('Scree Plot')
plt.show()

# You can now use the reduced PCA data for further analysis or visualization
```

Output:

```
Explained Variance Ratios: [0.2069236  0.16302229]
```



Factors that affect the accuracy of a PCA model for water quality prediction:

- **The quality and size of the training data**: PCA is a data-driven model, so the quality and size of the training data have a significant impact on the accuracy of the model. PCA models require large datasets to train effectively.
- **The number of principal components to retain**: If too few principal components are retained, the model may not be able to capture the important relationships between the input attributes. If too many principal components are retained, the model may overfit the training data.
- **The choice of machine learning algorithm**: The performance of the PCA model will also depend on the choice of machine learning algorithm. It is important to choose a machine learning algorithm that is appropriate for the specific problem that you are trying to solve.

6. PERFORMANCE ANALYSIS

Performance analysis in machine learning is crucial for assessing the effectiveness of trained models. It involves the evaluation of a model's predictive capabilities, often using metrics like accuracy, precision, recall, F1 score, and ROC-AUC, depending on the specific task. Performance analysis helps data scientists and researchers fine-tune models, identify overfitting or underfitting, and select the best algorithms and hyperparameters to achieve optimal results. Additionally, it aids in comparing different models and making informed decisions about their deployment in real-world applications.

Furthermore, performance analysis also includes the exploration of training and inference times, memory usage, and scalability of models to ensure they meet computational requirements in production environments. The assessment of performance is an iterative process, essential for building reliable and effective machine learning solutions that fulfill their intended purpose with high accuracy and efficiency.

ACCURACY:

In machine learning, accuracy is a common metric used to evaluate the performance of a classification model. It measures the proportion of correctly predicted instances among all the instances in a dataset. It is typically expressed as a percentage and can be calculated using the following formula:

Accuracy = (Number of Correct Predictions) / (Total Number of Predictions)

F1 SCORE:

The F1 score is a commonly used metric in machine learning for evaluating the performance of a binary classification model. It combines two other important metrics: precision and recall, into a single value. The F1 score provides a balance between these two metrics, and it's especially useful when dealing with imbalanced datasets, where one class significantly outweighs the other.

The F1 score is calculated using the following formula:

F1 Score = 2 * (Precision * Recall) / (Precision + Recall)

RECALL:

In machine learning, recall is a performance metric used to evaluate the ability of a classification model to correctly identify all relevant instances of a particular class within a dataset. It is also known as true positive rate or sensitivity. Recall measures the proportion of actual positive instances that the model correctly predicted as positive. The formula for calculating recall is as follows:

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

PRECISION:

In machine learning, precision is a commonly used evaluation metric for classification models, particularly in binary classification problems. Precision measures the accuracy of positive predictions made by the model, specifically the ratio of correctly predicted positive instances to all instances predicted as positive. It is often used in conjunction with other metrics like recall and F1 score to assess the model's performance comprehensively.

The precision formula is defined as follows:

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

Algorithm	Accuracy	F1 score (0)	F1 score (1)	Recall (0)	Recall (1)	Precision (0)	Precision (1)
Logistic Regression	0.67	0.77	0.14	0.93	0.14	0.65	0.40
Decision Tree	0.715	0.64	0.38	0.70	0.33	0.58	0.45
Random Forest	0.80	0.83	0.65	0.85	0.61	0.80	0.69
K-Nearest Neighbour	0.705	0.78	0.56	0.79	0.54	0.76	0.58
Support Vector Machine	0.815	0.87	0.69	0.93	0.60	0.81	0.82
Artificial Neural Network	0.78	0.83	0.68	0.84	0.67	0.83	0.69
K-Means	0.37	0.77	0.14	0.93	0.09	0.65	0.40

6.1 COMPARISON ANALYSIS OF MACHINE LEARNING ALGORITHMS

ABOUT DATASET:

1.PH:
PH is an important parameter in evaluating the acid–base balance of water. It is also the indicator of acidic or alkaline condition of water status.

2. HARDNESS:
Hardness is mainly caused by calcium and magnesium salts. These salts are dissolved from geologic deposits through which water travels. The length of time water is in contact with hardness producing material helps determine how much hardness there is in raw water.

3.SOLIDS:
Water has the ability to dissolve a wide range of inorganic and some organic minerals or salts such as potassium, calcium, sodium, bicarbonates, chlorides, magnesium, sulfates etc.

4.CHLORAMINES:

Chlorine and chloramine are the major disinfectants used in public water systems. Chloramines are most commonly formed when ammonia is added to chlorine to treat drinking water. Chlorine levels up to 4 milligrams per liter (mg/L or 4 parts per million (ppm)) are considered safe in drinking water.

5.SULFATE:
Sulfates are naturally occurring substances that are found in minerals, soil, and rocks. They are present in ambient air, groundwater, plants, and food. The principal commercial use of sulfate is in the chemical industry.

6.ORGANIC CARBON:
Total Organic Carbon (TOC) in source waters comes from decaying natural organic matter (NOM) as well as synthetic sources. TOC is a measure of the total amount of carbon in organic compounds in pure water.

7.TURBIDITY:
The turbidity of water depends on the quantity of solid matter present in the suspended state. It is a measure of light emitting properties of water and the test is used to indicate the quality of waste discharge with respect to colloidal matter.

8.POTABILITY:
Indicates if water is safe for human consumption where 1 means Potable and 0 means Not potable.

- By applying the above data set to the different machine learning algorithms, the SVM got the highest accuracy (81.5%) when compared to other machine learning algorithm, as we can see the accuracy of all the machine learning algorithm models in the fig-1

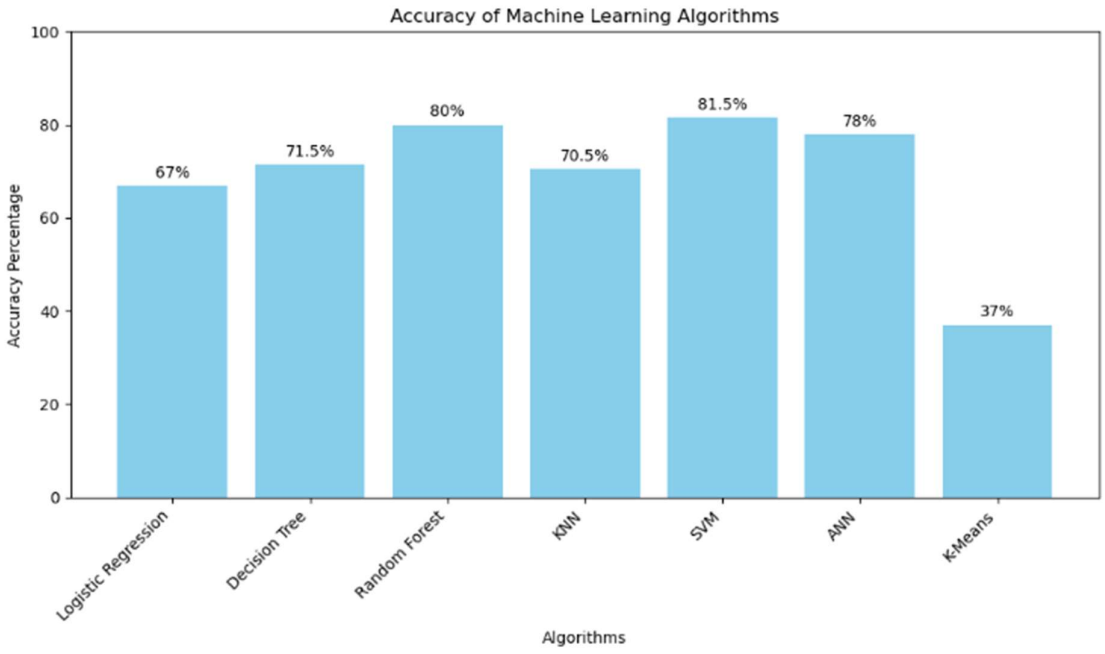


FIG-1(ACCURACY OF ML ALGORITHMS)

6.2 RESULT AND DISCUSSION

MACHINE LEARNING ALGORITHM ACCURACY:

1.Logistic Regression (67%):

Logistic regression provides a basic baseline for classification problems. An accuracy of 67% indicates that it captures some of the relationships in the data but may not be the best model for this problem. It's a simple linear model and may not handle the complexity of water quality prediction well.

Decision Tree (69%):

Decision trees are capable of capturing non-linear relationships in the data. The accuracy of 69% is slightly better than logistic regression but still moderate. Decision trees can sometimes overfit the data, so it's important to consider hyperparameter tuning and pruning.

Random Forest (80%):

Random forests are an ensemble of decision trees, which tend to improve generalization. An accuracy of 80% is a significant improvement over the previous models. It's a promising result, and random forests are known for their robustness and ability to handle complex data.

K-Nearest Neighbors (KNN) (70.5%):

KNN is a simple instance-based algorithm that classifies data points based on the majority class of their k-nearest neighbors. The 70.5% accuracy suggests moderate performance. The choice of the right k value may influence results.

Support Vector Machine (SVM) (81.5%):

SVMs are powerful for separating data into distinct classes. An accuracy of 81.5% is a strong result, indicating that the SVM model is effective at capturing the underlying patterns in the data. SVMs are known for their ability to work well with both linear and non-linear data.

Artificial Neural Network (ANN) (78%):

ANNs are a class of deep learning models that can capture complex relationships in data. The 78% accuracy shows that the neural network performs well. Fine-tuning the network architecture and hyperparameters might further improve the results.

K-Means (37%):

K-Means is a clustering algorithm, not typically used for classification tasks. An accuracy of 37% is not suitable for classification. It's important to use clustering models for their intended purposes, like finding natural groupings in data, rather than classification.

In summary, the results indicate that some machine learning algorithms like SVM, Random Forest, and ANN are performing well with accuracy scores above 75%. These models are capable of capturing complex patterns in the data. It's important to continue fine-tuning hyperparameters, assessing model performance using different evaluation metrics, and conducting feature engineering to potentially improve results further.

The poor performance of K-Means (37%) is expected because it's not designed for classification tasks. You may want to re-evaluate its usage in this context.

CONCLUSION & FUTURE ENHANCEMENTS

The examination of water quality is a complex issue because of the many variables that affect it. This idea, in particular, is inextricably linked to the many purposes for which water is used. Different needs call for various standards. There is a lot of study being done on predicting water quality. Normally, a collection of chemical factors and physical factors that are closely connected to the water's intended use must be used to determine water quality. Then, acceptable values and unacceptable values for each variable should be established. Water is deemed acceptable for an application when its variables comply with the requirements set out in advance. The water must be treated before usage if it doesn't match these requirements. Numerous physical and chemical traits may be utilized to assess the level of water pollution or the quality of the water. As a result, it is not practical to describe quality of water precisely, either on a geographical or temporal basis, by analyzing the behavior of each individual variable independently. The challenging option is to combine the values of a number of physical factors and chemical factors to create a single value, such as an overall index. The water quality index measures the effectiveness of any water, that may be created by only identifying the requirements needed for that usage. This indicator has a number of chemical and physical properties. Each variable in the index has a quality value function that indicates how well the parameter and its quality level are equivalent. Direct measurements of a substance's or an entity's concentration are used to define a physical variable's value as determined by water sample tests. The primary goal of this research to analyze machine learning methods for predicting the quality of water using eight different factors, including ph, Hardness, solids, chloramines, sulphate, conductivity, organic carbon, trihalomethanes, Turbidity, potability has been achieved.

In future, a lot of improvements can be done in terms of upgrading the model’s accuracy and prediction. The data which is entered by user could be stored in database, so that it can be used for future research analysis. This model requires eight parameters to predict the water quality, and in future, the parameters can be reduced and can make it available for all users. More functionalities and stylings can be added to the web interface, so that the web application can be more interactive. For now, the model predicts the water quality with few suggestions for the purification and treatment of water; however, in the future work, the suggestions can be elaborated with what minerals can be added to the water.

REFERENCES

IEEE JOURNAL PAPERS:

1.Water Quality Analysis and prediction using machine learning

Journal: IEEE

Authors: Dr.D Brindha, Viswanath puli, Vamsi Stephen, Bala NVSS, Guru Dinesh
https://ieeexplore.ieee.org/document/10083776

2. Water Quality Analysis using Machine Learning

Journal: IEEE

Authors: Ravi Akshay, Gadiraju Tarun, uday kiran, Durga Devi, M Vidhyalakshmi
https://ieeexplore.ieee.org/abstract/document/10047533

3.Predicting Water Quality Parameters using Machine Learning

Journal: IEEE

Authors: Nikhil M, Ravishankar Holla, Manju G
https://ieeexplore.ieee.org/document/9016825

4.Water Quality Prediction Using Machine Learning Learning Techniques

Journal: IEEE

Authors: Nithya pagadala, Moulika Marri, Akshaya Myla, Bhargav Abburi, k. Shri Ramtej
https://ieeexplore.ieee.org/document/10117415

5.Predicting and Analyzing Water Quality using Machine Learning: A comprehensive model

Journal: IEEE

Authors: Yafra Khan, Chai Soo See
https://ieeexplore.ieee.org/document/7494106

6.Predictive Models for river Water Quality using Machine Learning and Big Data Technique- A survey

Journal: IEEE

Authors: Jitha P Nair, Vijaya M
https://ieeexplore.ieee.org/document/9395832

7.Water Quality Analysis Using Deep Learning

Journal: IEEE

Authors: Ankit chahar, Ayanesh Chowdhury, Bharath Kumar, Vishan Reddy,Harshal Patel,Ninad Patil

https://ieeexplore.ieee.org/document/9785189