**Water Quality Analysis**

**Data Analytics with Cognos-Group 1**

**Phase\_3**

**R.Elakya [411521104033]**



Water is a fundamental resource that sustains life and plays a pivotal role in various aspects of our daily existence. It serves as a source of drinking water, supports ecosystems, drives industrial processes, and contributes to agriculture. However, the quality of our water bodies is continually under threat due to various human and natural factors. In light of this, our project, Enhancing Water Quality aims to address this pressing issue and promote the sustainable management of water resources.

Analyzing water quality using data analytics involves the application of advanced computational techniques to extract valuable insights from water quality data. This approach can provide a deeper understanding of water conditions, identify trends, anomalies, and potential issues, and facilitate data-driven decision-making for water resource management. Here's an overview of water quality analysis using data analytics:

The first step of analysis the water quality involves the following steps:

* Data collection
* Data preprocessing
* Data integration
* Exploratory data analysis

**Data collection:**

The dataset used in this project has 3276 rows and 10 columns which has many null values and missing data.

Dataset link: <https://www.kaggle.com/datasets/adityakadiwal/water-potability>

The above dataset has many missing and it can be cleaned by using datacleaninh technique as a first step in the dataset.The data cleaning process was to remove duplicate data in the data set

**Data cleaning:**

The dataset used in the project contains 3276 rows and 10 columns.The first step is to clean the data by removing duplictes and null values in dataset

In[1]:import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

First we need to import the necessary library files to work on the data set

In[2]: data=pd.read\_csv(r'D:\datasets\water\_potability.csv')

Then the dataset path is read using pd.read()function

In[3]:data

|  | **ph** | **Hardness** | **Solids** | **Chloramines** | **Sulfate** | **Conductivity** | **Organic\_carbon** | **Trihalomethanes** | **Turbidity** | **Potability** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | NaN | 204.890455 | 20791.318981 | 7.300212 | 368.516441 | 564.308654 | 10.379783 | 86.990970 | 2.963135 | 0 |
| **1** | 3.716080 | 129.422921 | 18630.057858 | 6.635246 | NaN | 592.885359 | 15.180013 | 56.329076 | 4.500656 | 0 |
| **2** | 8.099124 | 224.236259 | 19909.541732 | 9.275884 | NaN | 418.606213 | 16.868637 | 66.420093 | 3.055934 | 0 |
| **3** | 8.316766 | 214.373394 | 22018.417441 | 8.059332 | 356.886136 | 363.266516 | 18.436524 | 100.341674 | 4.628771 | 0 |
| **4** | 9.092223 | 181.101509 | 17978.986339 | 6.546600 | 310.135738 | 398.410813 | 11.558279 | 31.997993 | 4.075075 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **3271** | 4.668102 | 193.681735 | 47580.991603 | 7.166639 | 359.948574 | 526.424171 | 13.894419 | 66.687695 | 4.435821 | 1 |
| **3272** | 7.808856 | 193.553212 | 17329.802160 | 8.061362 | NaN | 392.449580 | 19.903225 | NaN | 2.798243 | 1 |
| **3273** | 9.419510 | 175.762646 | 33155.578218 | 7.350233 | NaN | 432.044783 | 11.039070 | 69.845400 | 3.298875 | 1 |
| **3274** | 5.126763 | 230.603758 | 11983.869376 | 6.303357 | NaN | 402.883113 | 11.168946 | 77.488213 | 4.708658 | 1 |
| **3275** | 7.874671 | 195.102299 | 17404.177061 | 7.509306 | NaN | 327.459760 | 16.140368 | 78.698446 | 2.309149 | 1 |

3276 rows × 10 columns

It is the dataset before the data cleaning

In[4]:data.head

|  | **ph** | **Hardness** | **Solids** | **Chloramines** | **Sulfate** | **Conductivity** | **Organic\_carbon** | **Trihalomethanes** | **Turbidity** | **Potability** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | NaN | 204.890455 | 20791.318981 | 7.300212 | 368.516441 | 564.308654 | 10.379783 | 86.990970 | 2.963135 | 0 |
| **1** | 3.716080 | 129.422921 | 18630.057858 | 6.635246 | NaN | 592.885359 | 15.180013 | 56.329076 | 4.500656 | 0 |
| **2** | 8.099124 | 224.236259 | 19909.541732 | 9.275884 | NaN | 418.606213 | 16.868637 | 66.420093 | 3.055934 | 0 |
| **3** | 8.316766 | 214.373394 | 22018.417441 | 8.059332 | 356.886136 | 363.266516 | 18.436524 | 100.341674 | 4.628771 | 0 |
| **4** | 9.092223 | 181.101509 | 17978.986339 | 6.546600 | 310.135738 | 398.410813 | 11.558279 | 31.997993 | 4.075075 | 0 |

This data set is after doing datacleaning

**Exploratory Data Analysis:  
Exploratory Data Analysis (EDA)** is a crucial initial step in the data analysis process that involves the summary and visualization of data to gain insights, discover patterns, and identify potential relationships.

**Handling missing data:**

The handling missing data involves Identify the missing values and fill them will nan so that the data consistency is maintain throughout the process

In[5]:data.shape()

Out[5]:3276,10

In[6]: data.isnull().sum()

Out[6]:

ph 491

Hardness 0

Solids 0

Chloramines 0

Sulfate 781

Conductivity 0

Organic\_carbon 0

Trihalomethanes 162

Turbidity 0

Potability 0

dtype: int64

In[7]:data.info()

Out[7]: <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]: data.fillna(data.mean(),inplace=True)

data.isnull().sum()

Out[8]: ph 0

Hardness 0

Solids 0

Chloramines 0

Sulfate 0

Conductivity 0

Organic\_carbon 0

Trihalomethanes 0

Turbidity 0

Potability 0

dtype: int64

Now The dataset has no missing value after datacleaning

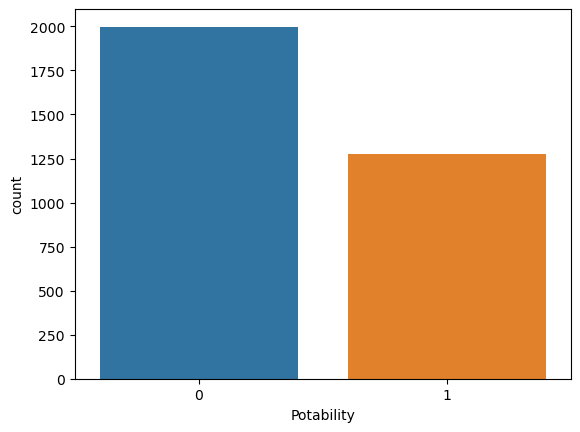
**Parameter visualizing using EDA:**

In[9]: sns.countplot(data['Potability'])

plt.show()

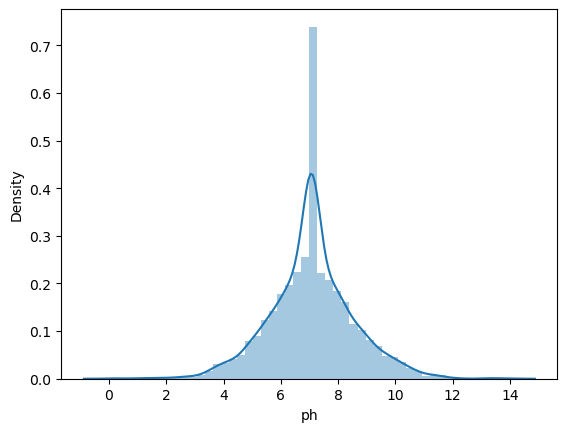
Output for countplot

The Potability of the water is represented graphically using countplot function in python



In[10]: sns.distplot(data['ph'])

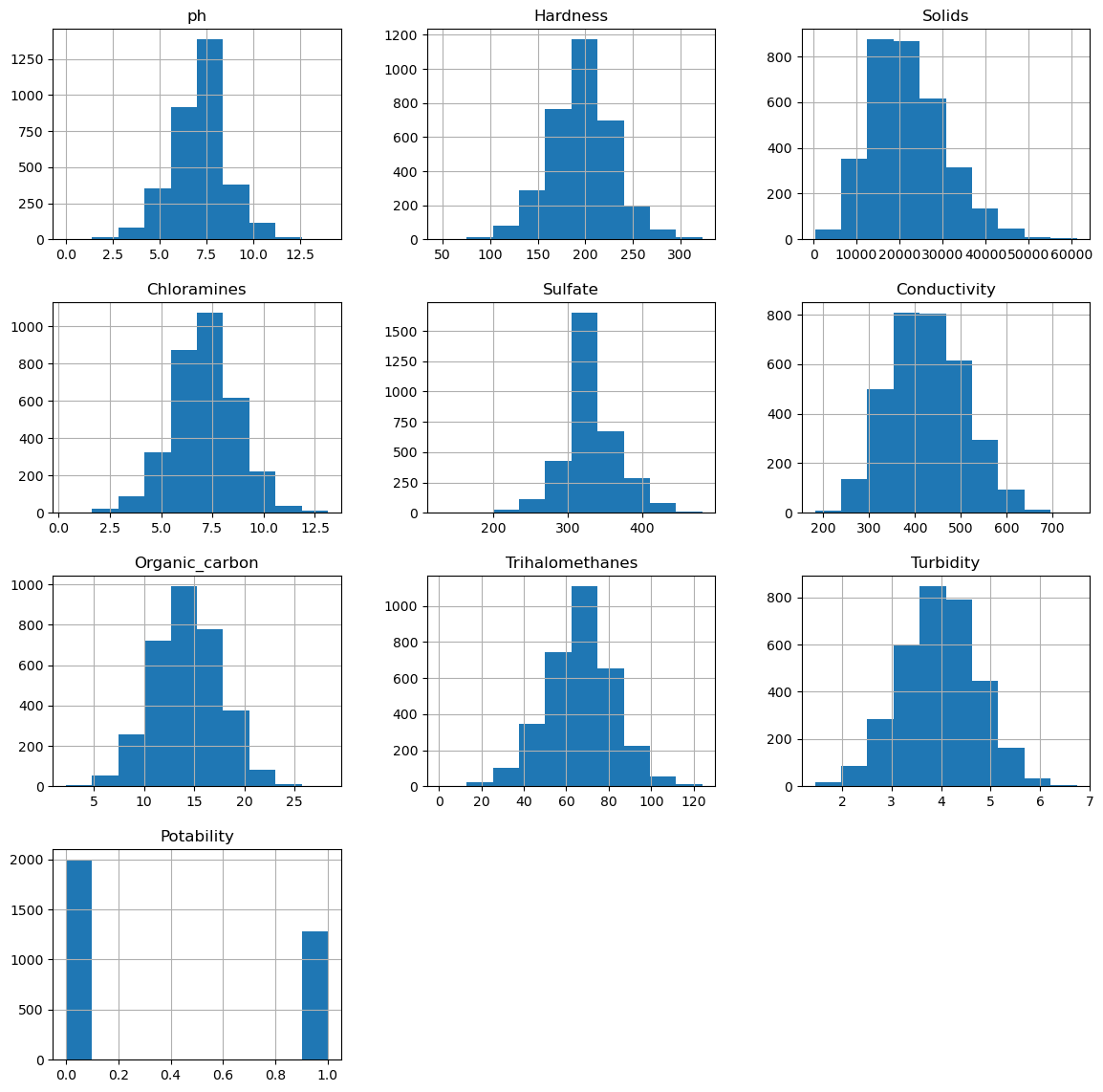
plt.show()

Out[10]: 

Now the parameters are visualized using hist() in Pyhton for visualization

In[11]: data.hist(figsize=(14,14))

plt.show()

Out[11]: 

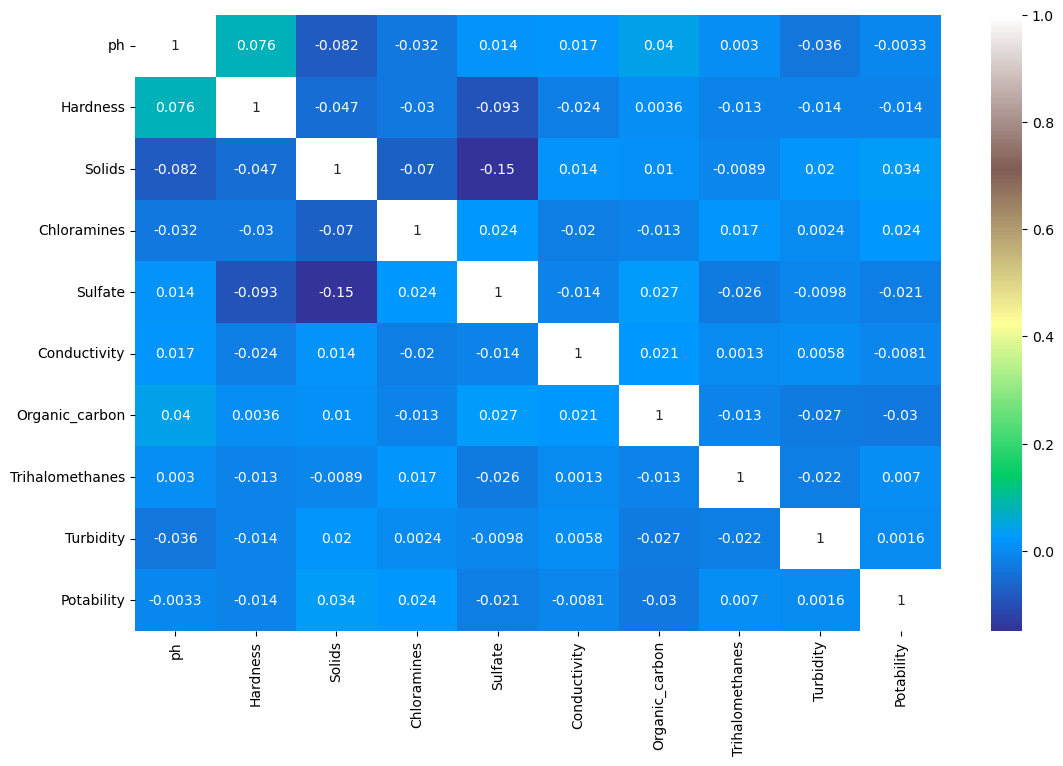
**Correlation and distribution:**

Correlation refers to a statistical measure that quantifies the strength and direction of the linear relationship between two or more variables. It is a valuable tool for understanding how changes in one variable relate to changes in another. Python provides several libraries and methods for calculating correlation, with one of the most commonly used libraries being **numpy** and **scipy**.

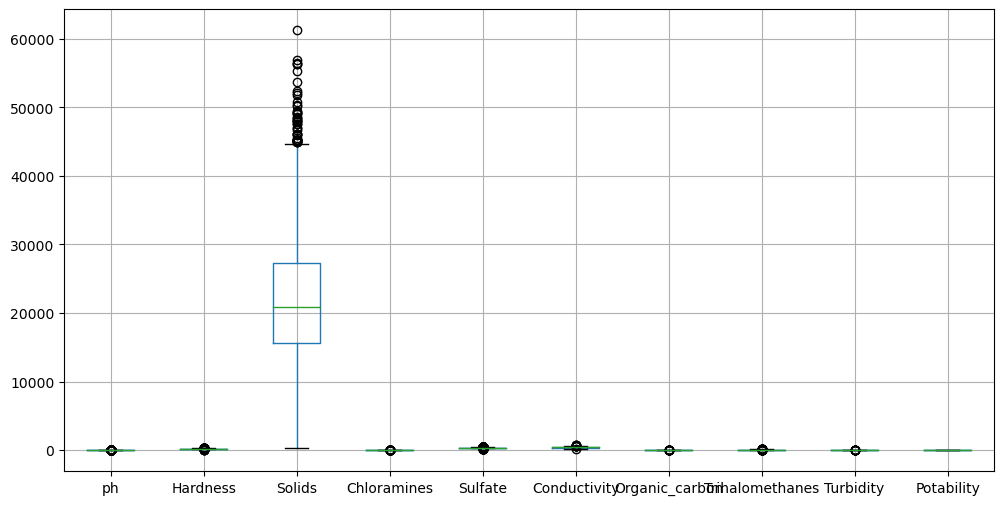
In[12]: plt.figure(figsize=(13,8))

sns.heatmap(data.corr(),annot=True,cmap='terrain')

plt.show()

out[12]: 

In[13]: data.boxplot(figsize=(12,6))

Out[13]: 

**Model Training:**

In[14]: x=data.drop('Potability',axis=1)

X

Out[14]:

|  | **ph** | **Hardness** | **Solids** | **Chloramines** | **Sulfate** | **Conductivity** | **Organic\_carbon** | **Trihalomethanes** | **Turbidity** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 7.080795 | 204.890455 | 20791.318981 | 7.300212 | 368.516441 | 564.308654 | 10.379783 | 86.990970 | 2.963135 |
| **1** | 3.716080 | 129.422921 | 18630.057858 | 6.635246 | 333.775777 | 592.885359 | 15.180013 | 56.329076 | 4.500656 |
| **2** | 8.099124 | 224.236259 | 19909.541732 | 9.275884 | 333.775777 | 418.606213 | 16.868637 | 66.420093 | 3.055934 |
| **3** | 8.316766 | 214.373394 | 22018.417441 | 8.059332 | 356.886136 | 363.266516 | 18.436524 | 100.341674 | 4.628771 |
| **4** | 9.092223 | 181.101509 | 17978.986339 | 6.546600 | 310.135738 | 398.410813 | 11.558279 | 31.997993 | 4.075075 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **3271** | 4.668102 | 193.681735 | 47580.991603 | 7.166639 | 359.948574 | 526.424171 | 13.894419 | 66.687695 | 4.435821 |
| **3272** | 7.808856 | 193.553212 | 17329.802160 | 8.061362 | 333.775777 | 392.449580 | 19.903225 | 66.396293 | 2.798243 |
| **3273** | 9.419510 | 175.762646 | 33155.578218 | 7.350233 | 333.775777 | 432.044783 | 11.039070 | 69.845400 | 3.298875 |
| **3274** | 5.126763 | 230.603758 | 11983.869376 | 6.303357 | 333.775777 | 402.883113 | 11.168946 | 77.488213 | 4.708658 |
| **3275** | 7.874671 | 195.102299 | 17404.177061 | 7.509306 | 333.775777 | 327.459760 | 16.140368 | 78.698446 | 2.309149 |

In[15]: y=data['Potability']y

Out[15]: 0 0

1 0

2 0

3 0

4 0

..

3271 1

3272 1

3273 1

3274 1

3275 1

Name: Potability, Length: 3276, dtype: int64

In[16]: from sklearn.model\_selection import train\_test\_split

In[17]: x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=101,shuffle=True)

**Decision tree classification:**

In[18]: from sklearn.tree import DecisionTreeClassifier

dt=DecisionTreeClassifier()

In[19]:dt.fit(x\_train,y\_train)

Out[19]: DecisionTreeClassifier()

In[20]: dt.predict(x\_test)

Out[20]: array([1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1,

0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,

0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1,

0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1,

0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,

0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1,

0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,

0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0,

1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1,

1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0,

0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0,

0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1,

0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1,

1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0,

1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,

0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,

1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0,

0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0,

0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1,

1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0,

1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,

0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0,

0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0,

0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0,

0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,

0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0,

1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1,

1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0,

0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1], dtype=int64)

In[21]: y\_prediction=dt.predict(x\_test)

In[22]: from sklearn.metrics import accuracy\_score,confusion\_matrix

accuracy\_score(y\_prediction,y\_test)

Out[22]: 0.5823170731707317

In[23]: confusion\_matrix(y\_prediction,y\_test)

Out[23]: array([[268, 140],

[134, 114]], dtype=int64)

In[24]:y\_test.shape

Out[24]:(656,)

**Hyperparameter tuning:**

In[25]: from sklearn.model\_selection import GridSearchCV

from sklearn.model\_selection import RepeatedStratifiedKFold

dt=DecisionTreeClassifier()

criterion=["gini","entropy"]

splitter=["best","random"]

min\_samples\_split=range(1,10)

parameters=dict(criterion=criterion,splitter=splitter,min\_samples\_split=min\_samples\_split)

cv= RepeatedStratifiedKFold(n\_splits=5,random\_state=101)

grid\_search\_cv\_dt=GridSearchCV(estimator=dt,param\_grid=parameters,scoring='accuracy',cv=cv)

In[26]: grid\_search\_cv\_dt.fit(x\_train,y\_train)

Out[26]: GridSearchCV(cv=RepeatedStratifiedKFold(n\_repeats=10, n\_splits=5, random\_state=101),

estimator=DecisionTreeClassifier(),

param\_grid={'criterion': ['gini', 'entropy'],

'min\_samples\_split': range(1, 10),

'splitter': ['best', 'random']},

scoring='accuracy')

In[27]: print(grid\_search\_cv\_dt.best\_params\_)

In[28]: from sklearn.metrics import accuracy\_score,confusion\_matrix

accuracy\_score(y\_prediction,y\_test)\*100

Out[28]: 58.231707317073166

**Conclusion:**

Enhancing Water Quality in project is not just an analysis but a comprehensive effort to ensure the sustainable use of this invaluable resource. With the support of our dedicated team, engaged stakeholders, and the local community, we are committed to improving water quality, protecting the environment, and fostering a healthier and more sustainable future.

In this Phase the dataset is cleaned and visualized in different methods and Trains the Dataset by Different algorithms