

**ANAND INSTITUTE OF HIGHER TECHNOLOGY
OLD MAHABALIPURAM ROAD,
KALASALINGAM NAGAR,
KAZHIPATTUR,**



**WATER QUALITY ANALYSIS
BY
DATA ANALYTICS WITH COGNOS**

PHASE – 3

NAME : VASUDEVAN K

REG No. : 310121104108

BRANCH : COMPUTER SCIENCE & ENGINEERING

YEAR/SEM : III / V

DATA ANALYTICS WITH COGNOS

PROJECT : WATER QUALITY ANALYSIS

PROJECT ID : PROJ_229794_TEAM_1

TEAM MEMBERS :

- ✧ **SANTHOSH KUMAR P**
- ✧ **YOKESWARAN K**
- ✧ **VINOTH S**
- ✧ **SATHYA MOORTHY A**
- ✧ **VASUDEVAN K**

TEAM LEADER : RICHARD NICHOLAS M

WATER QUALITY ANALYSIS

INTRODUCTION

- Water is a fundamental resource essential for the sustenance of life, and its quality is a critical factor in ensuring the well-being of both humans and the environment.
- Water quality analysis is crucial for ensuring safe drinking water, protecting aquatic ecosystems, managing water resources, and complying with environmental regulations.
- Water quality analysis encompasses a range of parameters and measurements to evaluate the condition of water bodies, including rivers, lakes, oceans, groundwater, and even treated tap water. These parameters may include temperature, pH, turbidity, dissolved oxygen,
- It involves collecting water samples from various sources, conducting laboratory tests, and interpreting the results to make informed decisions about water treatment, pollution control, and resource management.
- Continuous monitoring and analysis help safeguard human health and the environment while ensuring sustainable access to clean water.



PHASE – 3 : { DEVELOPMENT PART 1 }

- Start building the Water Quality Analysis by preprocessing the data and performing exploratory data analysis.
- In this Phase 3, Data preprocessing is a critical step in the analysis of Water Potability data, as it lays the foundation for extracting meaningful insights and patterns from the vast and diverse sources of information related to the water quality.
- This process involves collecting, cleaning, transforming, reduction of null values, visualization, scalability, efficiency and structuring raw data to make it suitable for analysis.
- The goal of Water Quality Analysis in this Phase 3 is to prepare the raw data for analysis, modelling, and decision making.

DATA COLLECTION :

Water Quality Analysis is done by using the Dataset of “**Water_Potability**” provided by the dataset site www.Kaggle.com

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

DATA OBSERVATION :

The water_potability.csv file contains water quality metrics for 3276 different water bodies. It contains Water Quality Parameters such as pH, Hardness, Solids, Chloramines, Sulphate, Conductivity, Organic_Carbon, Trihalomethanes, Turbidity. The Potability value defines the water quality based on the parameters given.

If Potability value is 0 then the water is Potable or the value is 1 then the water is Not Potable.



TITLE : Water Quality Analysis

DATASET ID : adityakadiwal/water-potability

SOURCE : The dataset was created by a Kaggle user named Aditya Kadiwal, collected from various sources about the water parameters.

DESCRIPTION:

1. The dataset provides information about the Water Quality Analysis from various types of water around the world.
2. It includes data on common water parameter and the potability result.

IMPORTANCE OF LOADING AND PREPROCESSING DATASET :

Loading and preprocessing the dataset is an important first step in building any machine learning model. However, it is especially important for water quality prediction models, as water potability datasets are often complex and noisy.

By loading and preprocessing the dataset, we can ensure that the machine learning algorithm is able to learn from the data effectively and accurately.

CHALLENGES INVOLVED IN PREPROCESSING A DATASET :

There are a number of challenges involved in loading and preprocessing a Water_Potability dataset we use for Water Quality Analysis, including:

HANDLING MISSING VALUES :

Water_Potability dataset often contain missing values, which can be due to a variety of factors, such as human error or incomplete data collection. Common methods for handling missing values include dropping the rows with missing values, imputing the missing values with the mean or median of the feature, or using a more sophisticated method such as multiple imputation.

SCALING THE FEATURES :

It is often helpful to scale the features before training a machine learning model. This can help to improve the performance of the model and make it more robust to outliers. There are a variety of ways to scale the features, such as min-max scaling and standard scaling.

SPLITTING THE DATASET INTO TRAINING AND TESTING SETS:

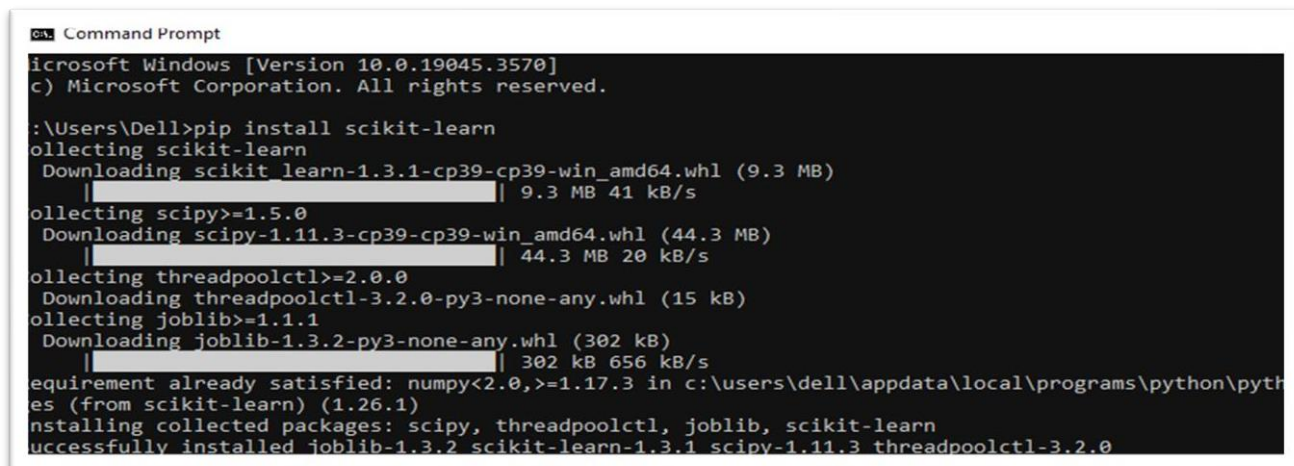
Once the data has been pre-processed, we need to split the dataset into training and testing sets. The training set will be used to train the model, and the testing set will be used to evaluate the performance of the model on unseen data. It is important to split the dataset in a way that is representative of the real world distribution of the data.

HOW TO OVERCOME THE CHALLENGES OF LOADING AND PREPROCESSING WATER_POTABILITY DATASET :

There are a number of things that can be done to overcome the challenges of loading and preprocessing a Water_Potability dataset, including:

USE A DATA PREPROCESSING LIBRARY:

There are a number of libraries available that can help with data preprocessing tasks, such as handling missing values, encoding categorical variables, and scaling the features.



```
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dell>pip install scikit-learn
Collecting scikit-learn
  Downloading scikit_learn-1.3.1-cp39-cp39-win_amd64.whl (9.3 MB)
    | 9.3 MB 41 kB/s
Collecting scipy>=1.5.0
  Downloading scipy-1.11.3-cp39-cp39-win_amd64.whl (44.3 MB)
    | 44.3 MB 20 kB/s
Collecting threadpoolctl>=2.0.0
  Downloading threadpoolctl-3.2.0-py3-none-any.whl (15 kB)
Collecting joblib>=1.1.1
  Downloading joblib-1.3.2-py3-none-any.whl (302 kB)
    | 302 kB 656 kB/s
Requirement already satisfied: numpy<2.0,>=1.17.3 in c:\users\dell\appdata\local\programs\python\python39\lib\site-packages (from scikit-learn) (1.26.1)
Installing collected packages: scipy, threadpoolctl, joblib, scikit-learn
Successfully installed joblib-1.3.2 scikit-learn-1.3.1 scipy-1.11.3 threadpoolctl-3.2.0
```

CAREFULLY CONSIDER THE SPECIFIC NEEDS OF YOUR MODEL :

The best way to preprocess the data will depend on the specific machine learning algorithm that you are using. It is important to carefully consider the requirements of the algorithm and to preprocess the data in a way that is compatible with the algorithm.

VALIDATE THE PREPROCESSED DATA:

It is important to validate the preprocessed data to ensure that it is in a format that can be used by the machine learning algorithm and that it is of high quality. This can be done by inspecting the data visually or by using statistical methods.

LOADING THE DATASET :

- Loading the dataset using machine learning is the process of bringing the data into the machine learning environment so that it can be used to train and evaluate a model.
- The specific steps involved in loading the dataset will vary depending on the machine learning library or framework that is being used. However, there are some general steps that are common to most machine learning frameworks:

IDENTIFY THE DATASET:

The first step is to identify the dataset that you want to load. This dataset may be stored in a local file, in a database, or in a cloud storage service.

▲ Water_potability.csv

LOAD THE DATASET:

Once you have identified the dataset, you need to load it into the machine learning environment. To load the dataset into Machine Learning Environment, Python's Pandas library is used.

```
import pandas as pd
df = pd.read_csv("water_potability.csv")
print(df)
```

PREPROCESS THE DATASET:

Once the dataset is loaded into the machine learning environment, you may need to preprocess it before you can start training and evaluating your model. This may involve cleaning the data, transforming the data into a suitable format, and splitting the data into training and test sets.

Let's see, How the Water_Potability Dataset Loaded and Accessed using Python Jupyter Notebook.

IMPORT NECESSARY LIBRARIES :

To perform the data preprocessing, splitting, scaling, and other tasks as described, several libraries in Python are needed to be imported. Here are the required libraries for the code:

FOR LOADING AND PREPROCESSING THE DATASET :

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

FOR SPLITTING THE DATASET INTO TRAINING AND TEST SETS :

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

FOR FEATURE SCALING :

```
from sklearn.preprocessing import StandardScaler
```

TO INSTALL PYTHON LIBRARIES :

To install the necessary libraries, run the given command in the Command Prompt

LIBRARY		COMMANDS
Pandas	—	pip install pandas
Numpy	—	pip install numpy
Matplotlib.pyplot	—	pip install matplotlib
Seaborn	—	pip install seaborn
Sklearn.model_selection	—	pip install scikit-learn
Sklearn.preprocessing	—	pip install scikit-learn

IMPORT AND LOAD THE DATASET :

Use Pandas library to read the dataset file you downloaded and convert into DataFrame :

```
df = pd.read_csv("water_potability.csv")  
print(df)
```

	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

PREPROCESSING THE DATASET :

- Data preprocessing is the process of
 1. Cleaning
 2. Transforming
 3. Integrating Datain order to make it ready for analysis.
- This may involve removing errors and inconsistencies, handling missing values, transforming the data into a consistent format, and scaling the data to a suitable range.

Some common data preprocessing tasks include:

Data Cleaning: This involves identifying and correcting errors and inconsistencies in the data. For example, this may involve removing duplicate records, correcting typos, and filling in missing values.

Data Transformation: This involves converting the data into a format that is suitable for the analysis task. For example, this may involve converting categorical data to numerical data, or scaling the data to a suitable range.

Feature Engineering: This involves creating new features from the existing data. For example, this may involve creating features that represent interactions between variables, or features that represent summary statistics of the data

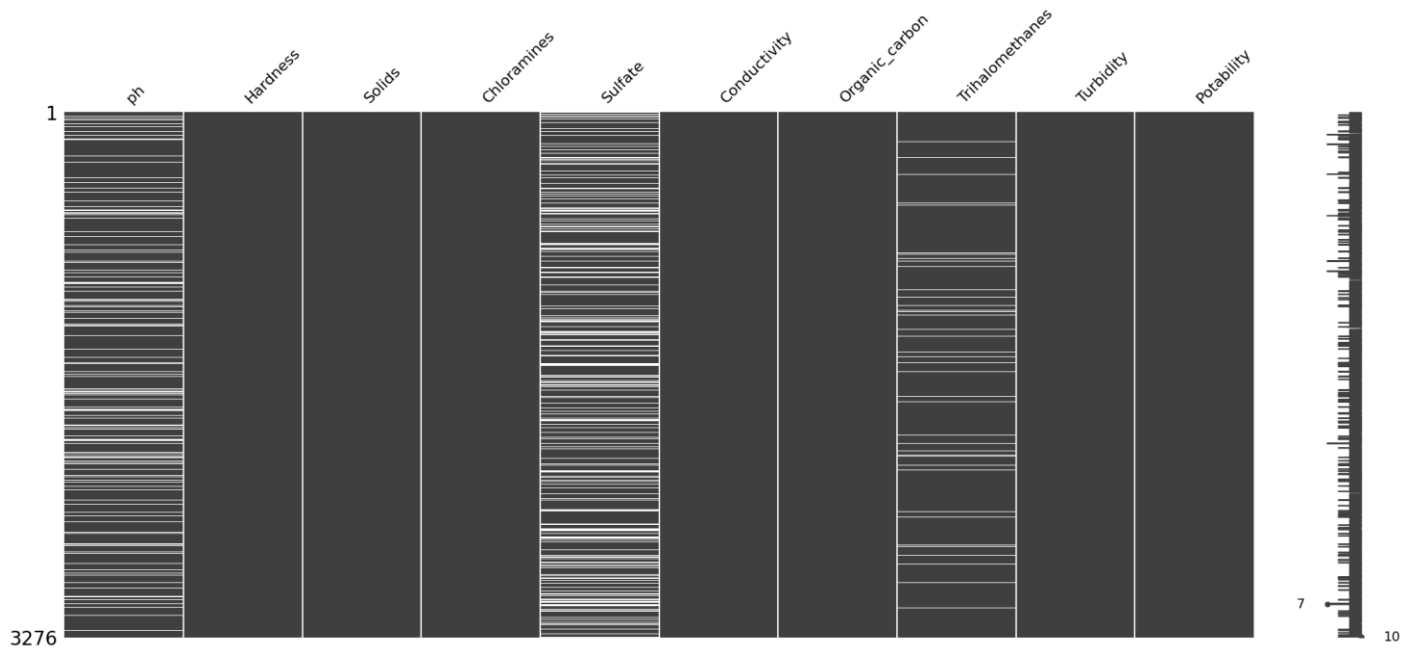
Data Integration: This involves combining data from multiple sources into a single dataset. This may involve resolving inconsistencies in the data, such as different data formats or different variable names.

Data preprocessing is an essential step in many data science projects. By carefully preprocessing the data, data scientists can improve the accuracy and reliability of their results.

HANDLING MISSING VALUES :

CHECKING FOR THE MISSING VALUES USING MISSINGNO LIBRARY :

```
import missingno as msno
msno.matrix(df)
plt.show()
```



CHECKING FOR THE MISSING VALUES :

```
df.isnull().sum()
```

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

FILL MISSING VALUES USING MEDIAN :

```
for feature in df.columns:  
    df[feature].fillna(df[feature].median() , inplace = True)
```

The Missing values of the dataset

```
df.isnull().sum()
```

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

FINDING DUPLICATE VALUES :

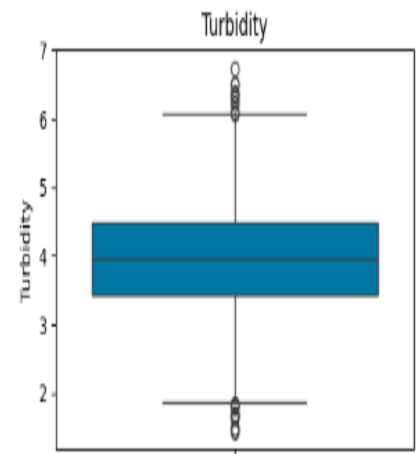
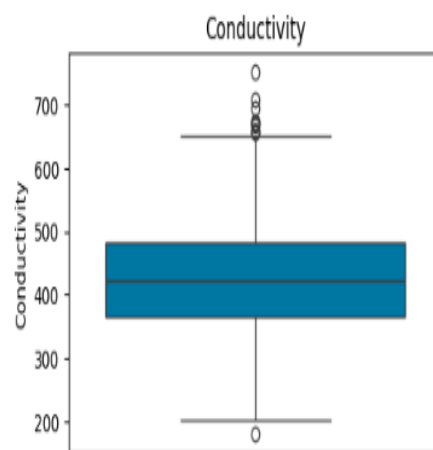
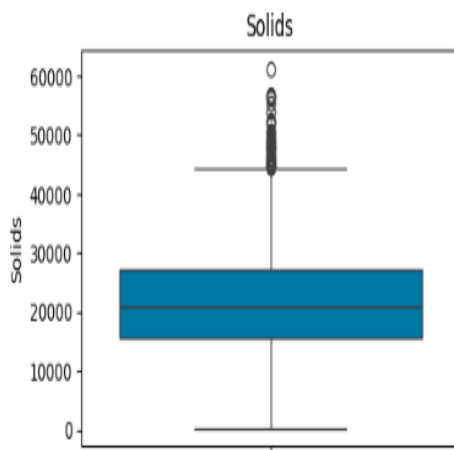
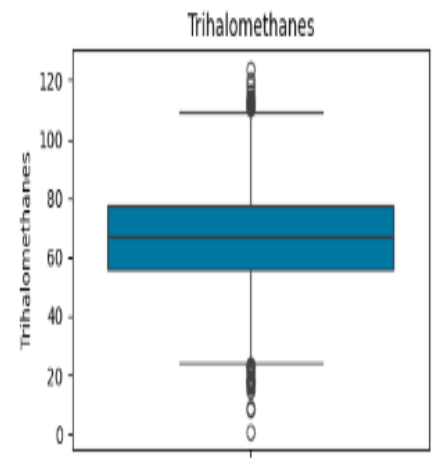
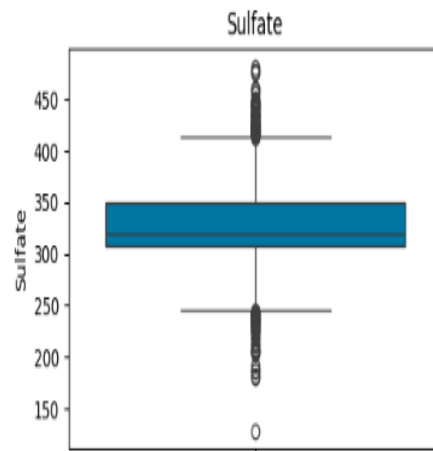
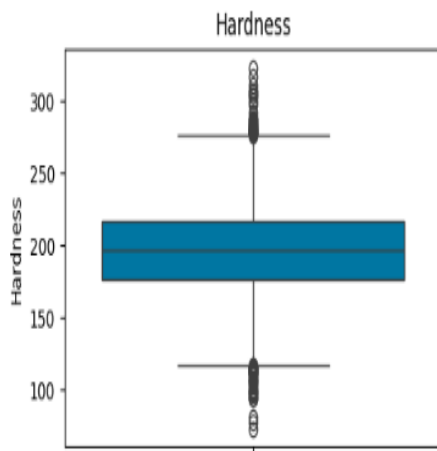
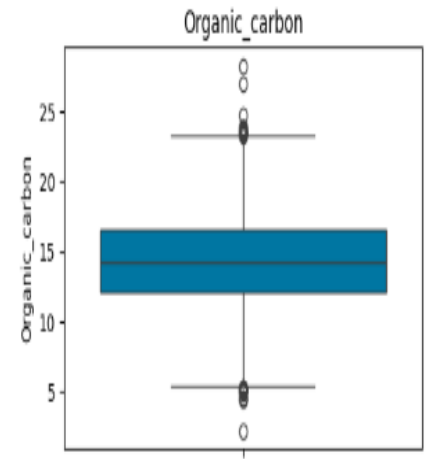
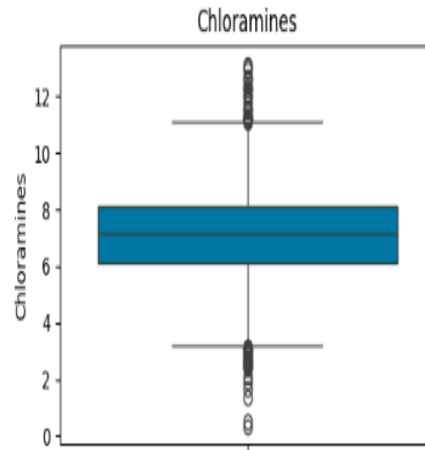
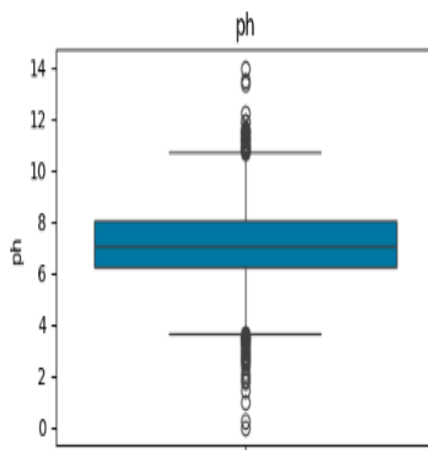
```
duplicate = df[df.duplicated()]  
duplicate
```

```
ph  Hardness  Solids  Chloramines  Sulfate  Conductivity  Organic_carbon  Trihalomethanes  Turbidity  Potability
```

No Duplicates present in the dataset.

CHECKING FOR THE OUTLIERS :

```
cont = ["ph","Hardness","Solids","Chloramines","Sulfate","Conductivity",  
        "Organic_carbon","Trihalomethanes","Turbidity"]  
for i in cont:  
    plt.figure(figsize=(5,3))  
    sns.boxplot(data=df,y=i)  
    plt.title(i)  
    plt.show()
```



OUTLIER DETECTION :

Outliers are found in the following columns.

1. ph
 2. Hardness
 3. Solids
 4. chioramines
 5. Sulfate
 6. Conductivity
 7. Organic_carbon
 8. Trihalomethanes
 9. Turbidity
- When we examine the boxplots, we can see that there are some outlier values. We will clean these in the next step.
 - As a result, I decided to delete the outlier values from the dataset.

REMOVING OUTLIERS :

```
# removing outliers
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
ph                1.592377
Hardness          39.816918
Solids            11666.071830
Chloramines       1.987466
Sulfate           33.291119
Conductivity      116.057890
Organic_carbon    4.491850
Trihalomethanes   20.018954
Turbidity         1.060609
Potability        1.000000
dtype: float64
```

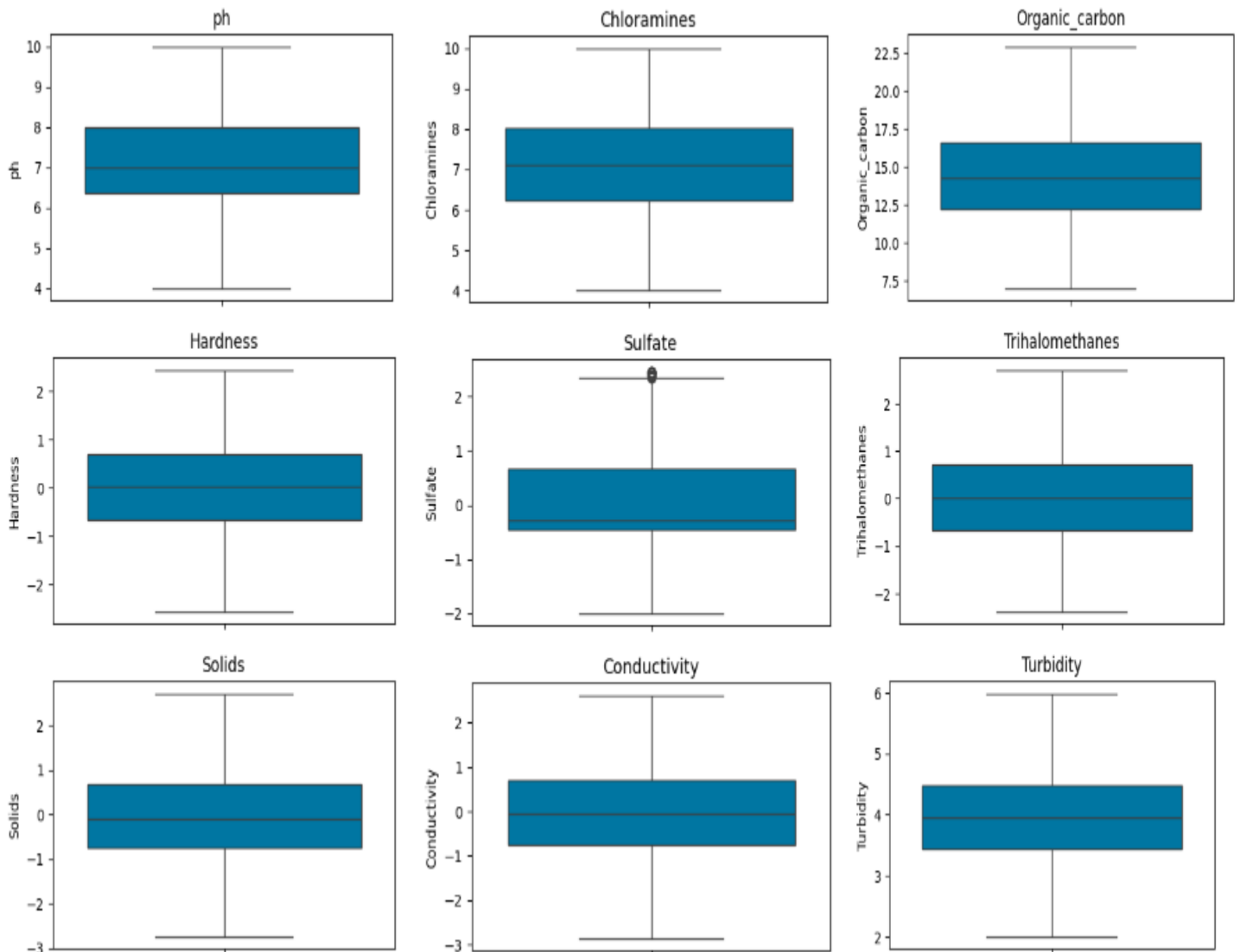
```
df = df[~((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))).any(axis=1)]
df.shape
```

```
(2666, 10)
```

The method you've used to remove outliers from a dataset, based on the Interquartile Range (IQR) and the 1.5 times IQR rule, is a commonly used and reasonable approach for outlier removal.

AFTER DELETING OUTLIERS :

```
cont = ["ph","Hardness","Solids","Chloramines","Sulfate","Conductivity",  
        "Organic_carbon","Trihalomethanes","Turbidity"]  
for i in cont:  
    plt.figure(figsize=(5,3))  
    sns.boxplot(data=df,y=i)  
    plt.title(i)  
    plt.show()
```



The outliers found in the dataset has been cleaned and the data has been balanced.

AFTER BALANCING :

Now the data has been cleaned by filling the Missing Values and Removing the Outliers of the data.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
Index: 2666 entries, 0 to 3275  
Data columns (total 10 columns):  
#   Column                Non-Null Count  Dtype  
---  -  
0   ph                    2666 non-null   float64  
1   Hardness              2666 non-null   float64  
2   Solids                2666 non-null   float64  
3   Chloramines           2666 non-null   float64  
4   Sulfate               2666 non-null   float64  
5   Conductivity          2666 non-null   float64  
6   Organic_carbon        2666 non-null   float64  
7   Trihalomethanes       2666 non-null   float64  
8   Turbidity             2666 non-null   float64  
9   Potability            2666 non-null   int64  
dtypes: float64(9), int64(1)  
memory usage: 229.1 KB
```

After balancing the dataset data as follows :

- In the dataset there are total 2666 entries.
- All column are numeric values.
- Column Dtype are float64(9), int64(1).

DESCRIPTIVE ANALYSIS :

```
df.describe()
```

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
count	2666.000000	2666.000000	2666.000000	2666.000000	2666.000000	2666.000000	2666.000000	2666.000000	2666.000000	2666.000000
mean	7.066770	197.051982	21486.829049	7.111191	333.581956	425.811207	14.305441	66.449377	3.961669	0.373218
std	1.215605	28.366547	7924.092724	1.413403	26.507237	79.988085	3.217070	14.898614	0.760980	0.483750
min	3.902476	118.988579	320.942611	3.194969	267.202392	201.619737	5.362371	27.559355	1.872573	0.000000
25%	6.348026	179.010144	15596.765222	6.188575	319.481628	365.641745	12.082883	56.915951	3.439135	0.000000
50%	7.036752	197.561474	20583.142637	7.114169	333.073546	421.320293	14.219418	66.622485	3.945844	0.000000
75%	7.792306	215.744047	26742.195037	8.053054	347.921235	481.446065	16.575501	76.628761	4.494523	1.000000
max	10.252816	275.886513	44652.363872	11.086526	400.274579	652.537592	23.234326	106.371720	6.083772	1.000000

This method provides summary statistics for the columns in a dataset, typically a DataFrame in Python. It's a useful way to quickly understand the distribution and characteristics of the data within each column.

1. **Count:** The number of non-missing (non-null) values in each column. This tells you how many data points are available for each feature.
2. **Mean (Average):** The average value of the data in each column. It gives you an idea of the central tendency of the data.
3. **Standard Deviation (std):** A measure of the dispersion or spread of the data. It indicates how much individual data points typically deviate from the mean.
4. **Minimum:** The smallest (minimum) value in the column.
5. **25th Percentile (Q1):** The value below which 25% of the data falls. It's the first quartile.
6. **Median (50th Percentile or Q2):** The middle value when the data is sorted. It's the second quartile and also the median of the data.
7. **75th Percentile (Q3):** The value below which 75% of the data falls. It's the third quartile.
8. **Maximum:** The largest (maximum) value in the column.

TRAIN TEST SPLITTING :

```
### splitting data into x and y
```

```
X = df.iloc[:, :-1]
```

```
y = df.iloc[:, -1]
```

```
# split dataset into train and test
```

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

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state= 5)
```

The use of this code is to prepare the data for machine learning by creating distinct datasets for training and evaluation. The training set is used to train the machine learning model, while the test set is used to evaluate the model's performance and assess its ability to make accurate predictions on unseen data.

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((1894, 9), (812, 9), (1894,), (812,))
```

The data is split as follows:

Training data : 1894 Rows

Testing data : 812 Rows

FEATURE SCALING :

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train_final = sc.fit_transform(X_train)
X_test_final = sc.transform(X_test)
```

The main use of this code is to ensure that your features have similar scales, which can be important for many machine learning algorithms. Scaling can help improve the convergence and performance of algorithms that are sensitive to the scale of the input features. Standardization, as performed by the StandardScaler, transforms your data so that it has a mean of 0 and a standard deviation of 1.

Now the data is ready to work effectively with Machine Learning algorithms to evaluate the performance of the model and to build a Water Quality Predictive Model.

CONCLUSION :

- In the quest to build a Water Quality Prediction Model, we have embarked on a critical journey that begins with loading and preprocessing the Water_Potability dataset. We have traversed through essential steps, starting with importing the necessary libraries to facilitate data manipulation and analysis.
- Understanding the data's structure, characteristics, and any potential issues through exploratory data analysis (EDA) is essential for informed decision-making.
- Data preprocessing emerged as a pivotal aspect of this process. It involves cleaning, transforming, and refining the dataset to ensure that it aligns with the requirements of machine learning algorithms.
- With these foundational steps completed, our dataset is now primed for the subsequent stages of building and training a Water Quality Prediction Model.
- Water quality analysis, as exemplified by this dataset, is a vital component of responsible water resource management in the face of evolving environmental challenges and needs for clean and sustainable water supplies.