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

WATER QUALITY ANALYSIS
BY
DATA ANALYTICS WITH COGNOS

PHASE – 5

NAME : VINOTH S

REG No. : 310121104111

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 Nicholes 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.

Impacts of Not Potable Water :

Using non-potable water (water that is not suitable for drinking) can have various impacts on different aspects of society, the environment, and public health. Here are some key impacts:

Health Risks:

Waterborne Diseases: Non-potable water often contains contaminants and pathogens that can cause waterborne diseases. Using such water for drinking, cooking, or personal hygiene can lead to illnesses such as diarrhea, cholera, and gastroenteritis.

Agricultural and Irrigation Impact:

Crop Contamination: Irrigating crops with non-potable water may introduce contaminants into the soil, affecting the quality of the produce. This can have implications for food safety and agricultural productivity.

Environmental Impact:

Ecosystems: Discharging non-potable water into rivers, lakes, or oceans can harm aquatic ecosystems. Contaminants may disrupt the balance of aquatic life, leading to the decline of fish populations and other aquatic organisms.

Addressing the impacts of non-potable water involves implementing effective water treatment and purification methods, investing in water infrastructure, and promoting water conservation practices. Sustainable water management, including the use of alternative water sources for non-potable purposes (such as irrigation or industrial processes), can contribute to mitigating these impacts and ensuring the responsible use of water resources.

Importance of water quality analysis :

Water quality analysis is crucial for several reasons, as it provides valuable information about the composition and characteristics of water. Here are some key reasons highlighting the importance of water quality analysis:

Public Health Protection:
Safe Drinking Water: Water quality analysis is essential for monitoring and ensuring the safety of drinking water. It helps identify and measure the presence of contaminants, pathogens, and chemicals that could pose health risks to humans if consumed.

Environmental Conservation:
Aquatic Ecosystems: Assessing water quality is vital for the protection of aquatic ecosystems. Monitoring parameters such as nutrient levels, pH, and dissolved oxygen helps understand and mitigate the impact of human activities on rivers, lakes, and oceans.

Agricultural Practices:
Irrigation: Water quality analysis is important for determining the suitability of water for irrigation. Contaminants in water can affect soil quality and crop health, impacting agricultural productivity and food safety.

Research and Education:
Scientific Understanding: Water quality analysis contributes to scientific research and a deeper understanding of ecosystems. It also serves as a basis for educational programs aimed at promoting awareness about water issues and conservation.

Water Quality Analysis plays a pivotal role in safeguarding human health, protecting the environment, and promoting sustainable development. It provides the necessary information for informed decision-making, policy development, and the implementation of measures to maintain or improve water quality.

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

Dataset 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.

Water Quality Parameters :

1. pH value:

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. WHO has recommended maximum permissible limit of pH from 6.5 to 8.5. The current investigation ranges were 6.52–6.83 which are in the range of WHO standards.

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. Hardness was originally defined as the capacity of water to precipitate soap caused by Calcium and Magnesium.

3. Solids (Total dissolved solids - TDS):

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. These minerals produced un-wanted taste and diluted color in appearance of water. This is the important parameter for the use of water. The water with high TDS value indicates that water is highly mineralized. Desirable limit for TDS is 500 mg/l and maximum limit is 1000 mg/l which prescribed for drinking purpose.

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. Sulfate concentration in seawater is about 2,700 milligrams per liter (mg/L). It ranges from 3 to 30 mg/L in most freshwater supplies, although much higher concentrations (1000 mg/L) are found in some geographic locations.

6. Conductivity:

Pure water is not a good conductor of electric current rather's a good insulator. Increase in ions concentration enhances the electrical conductivity of water. Generally, the amount of dissolved solids in water determines the electrical conductivity. Electrical conductivity (EC) actually measures the ionic process of a solution that enables it to transmit current. According to WHO standards, EC value should not exceeded 400 μS/cm.

7. 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. According to US EPA < 2 mg/L as TOC in treated / drinking water, and < 4 mg/Lit in source water which is use for treatment.

8. Trihalomethanes:

THMs are chemicals which may be found in water treated with chlorine. The concentration of THMs in drinking water varies according to the level of organic material in the water, the amount of chlorine required to treat the water, and the temperature of the water that is being treated. THM levels up to 80 ppm is considered safe in drinking water.

9. 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. The mean turbidity value obtained for Wondo Genet Campus (0.98 NTU) is lower than the WHO recommended value of 5.00 NTU.

10. Potability:

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

These insights provide a preliminary understanding of the water quality parameters and their ranges within the dataset. Further analysis and modelling can help extract more meaningful patterns and relationships.

SOLUTION AND TECHNINCAL ARCHITECTURE :

There are basically 10 steps for making our model predict the water quality of the water samples. Those steps are:-

Problem Identification :
    In this step, we identify the problem which is solved by our model. So the problem to be solved by our model is water quality prediction using a dataset.

Data Extraction:-
    In this, we extract the data from the internet to train our data and predict the water quality. So for that, we take the CPCB(Central Pollution Control Board India) dataset which contains 3277 instances of 13 different wellsprings which are collected between 2014 to 2020.

Data Exploration:-
    In this step, we analyze the data visually by comparing some parameters of water with the WHO standards of water. It gives a slight overview of the data.

Data Cleaning :
    In this step, we clean that data like if there are some missing values in it so we replace them with mean and remove noise from the data..

Data Selection :
In this step, we select the data types and source of the data. The essential goal of data selection is deciding fitting data type, source, and instrument that permit agents to respond to explore questions sufficiently

Data Splitting :
In this step, we divide the dataset into smaller subsets for easing the complexity. Normally, with a two-section split, one section is utilized to assess or test the information and the other to prepare the model.

Data Modeling :
In this step, we create a graph of the dataset for visual representation of data for better understanding. A Data Model is this theoretical model that permits the further structure of conceptual models and to set connections between data.

Model Evaluation :
Model Evaluation is a fundamental piece of the model improvement process. In this step, we evaluate our model and check how well our model do in the future.

Loading and Preprocessing the 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.
- 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.

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.

```python
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.

Working with 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 :

```python
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 :

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

For Feature Scaling :

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

To install Python Libraries :

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

Libraray		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 Data

  in 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()
```
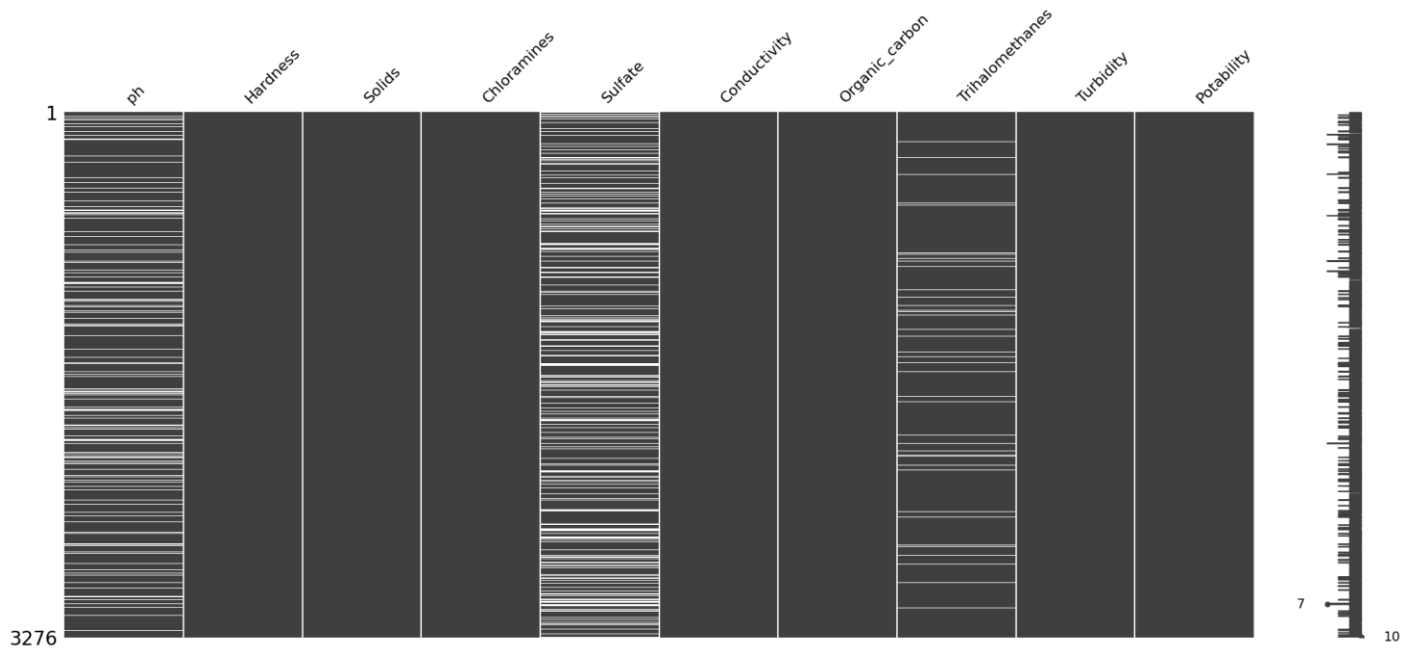

Checking for the Missing Values :

```
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
```

FILLING THE MISSING VALUES USING MEDIAN :

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

AFTER FILLING THE MISSING VALUES :

```python
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 :

```python
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 :

```python
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()
```
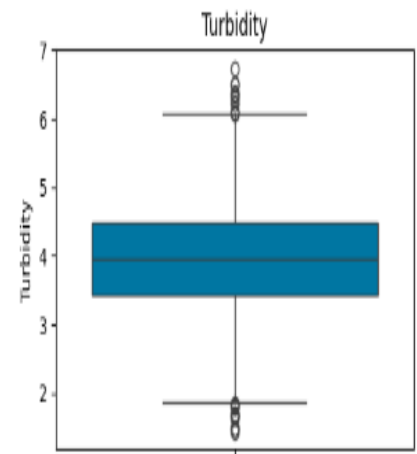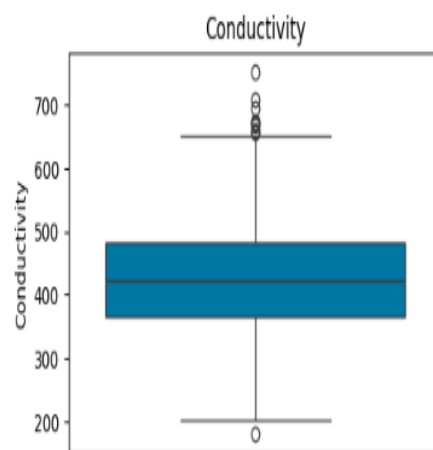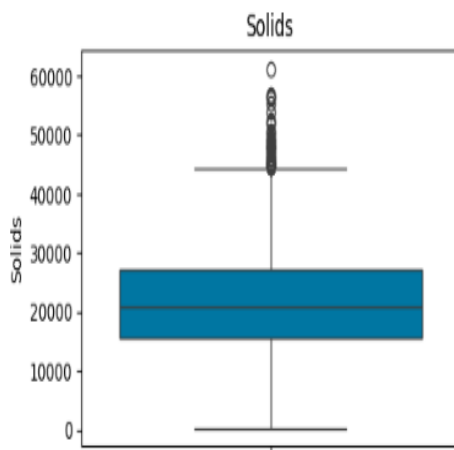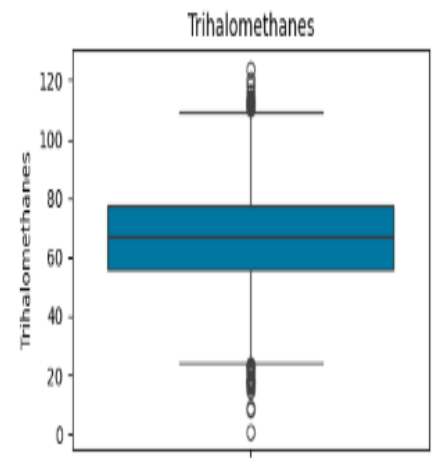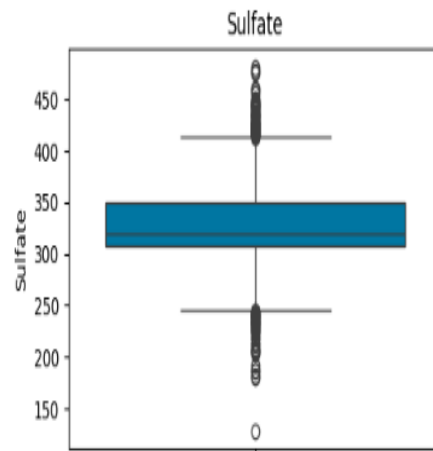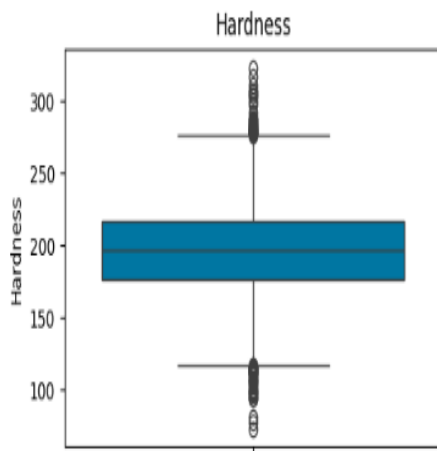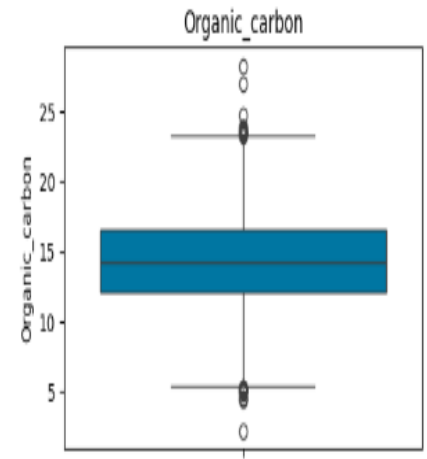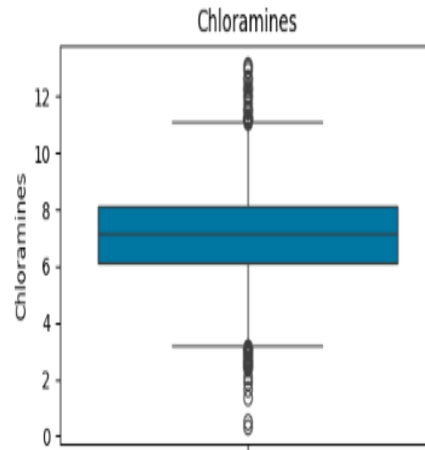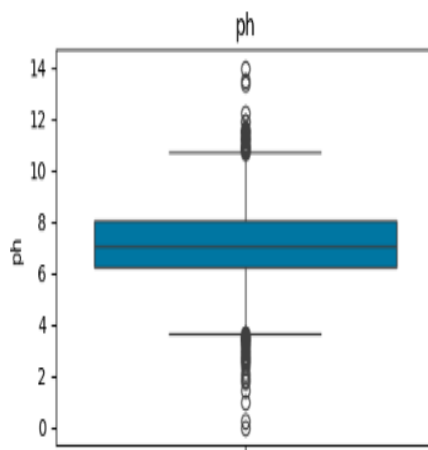
Outlier Detection :

Outliers are found in the following columns.

1. ph
2. Hardness
3. Solids
4. chioramines
5. Sulfate
6. Conductivity
7. Organic_carbon
8. Thrihalomethanes
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 :

```python
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()
```
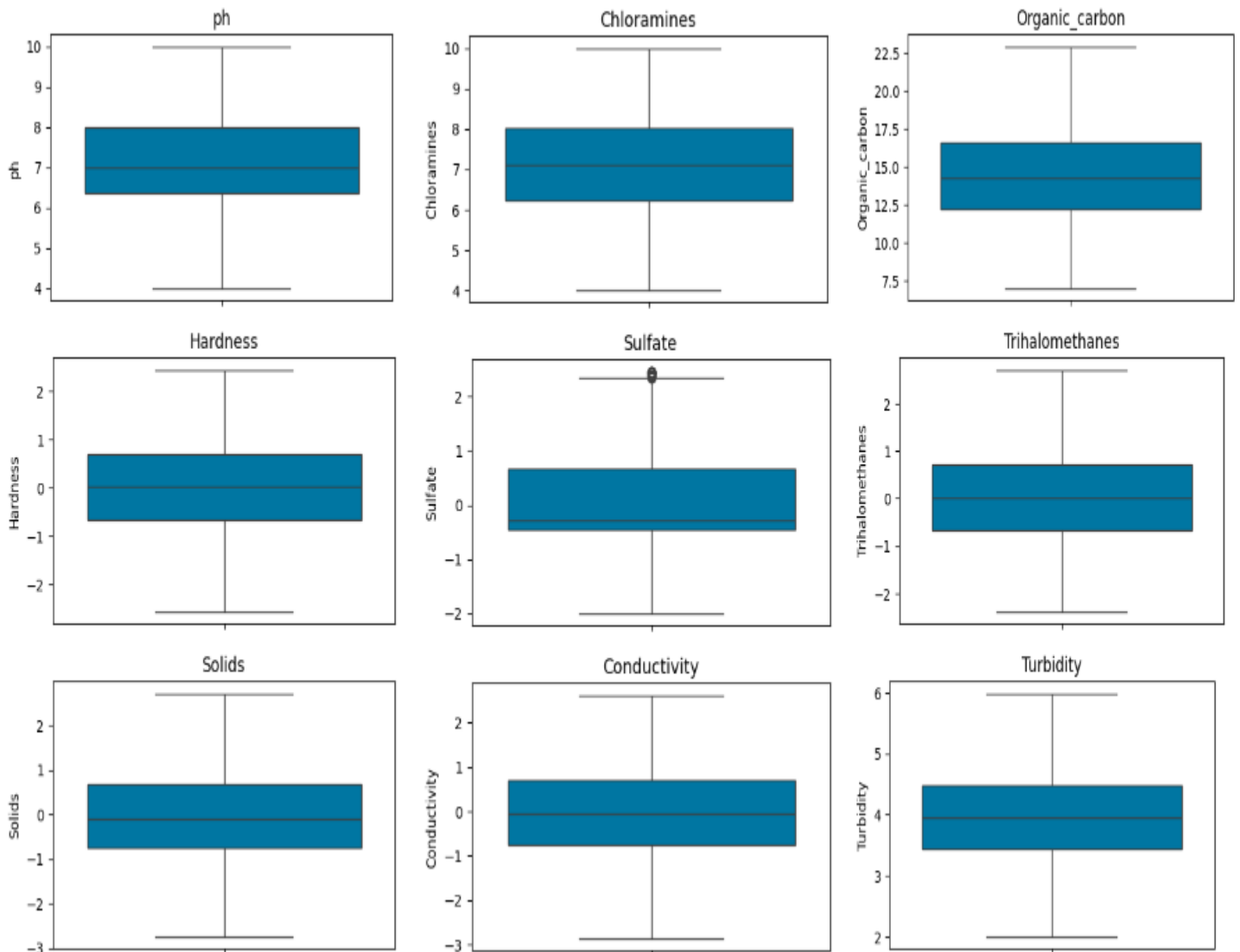

The outliers found in the dataset has been cleaned and the data has been balanced.

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.

EXPLORATORY DATA ANALYSIS :

POTABILITY OF THE DATASET :

```python
plt.pie(df['Potability'].value_counts(),labels = list(df['Potability'].unique()),autopct="%0.1f%%" )
plt.title("Potability")
plt.show()
```
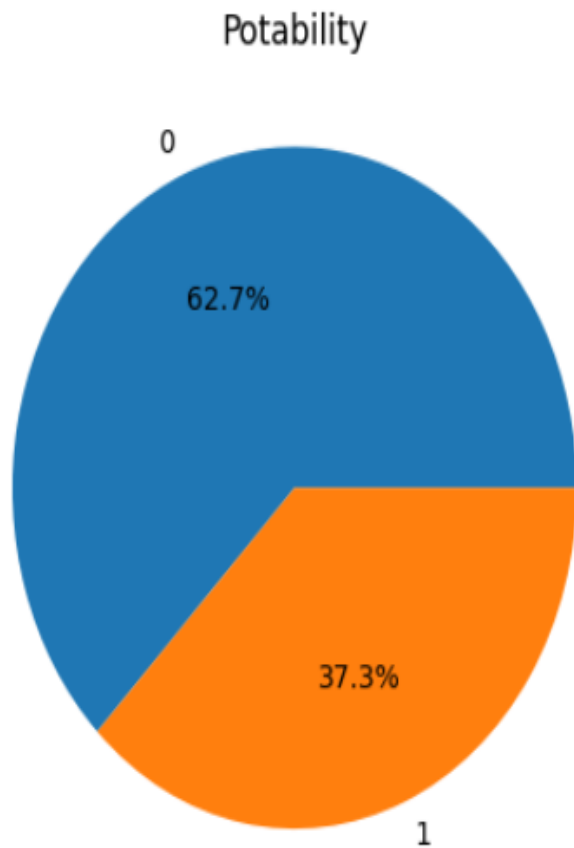
Potability

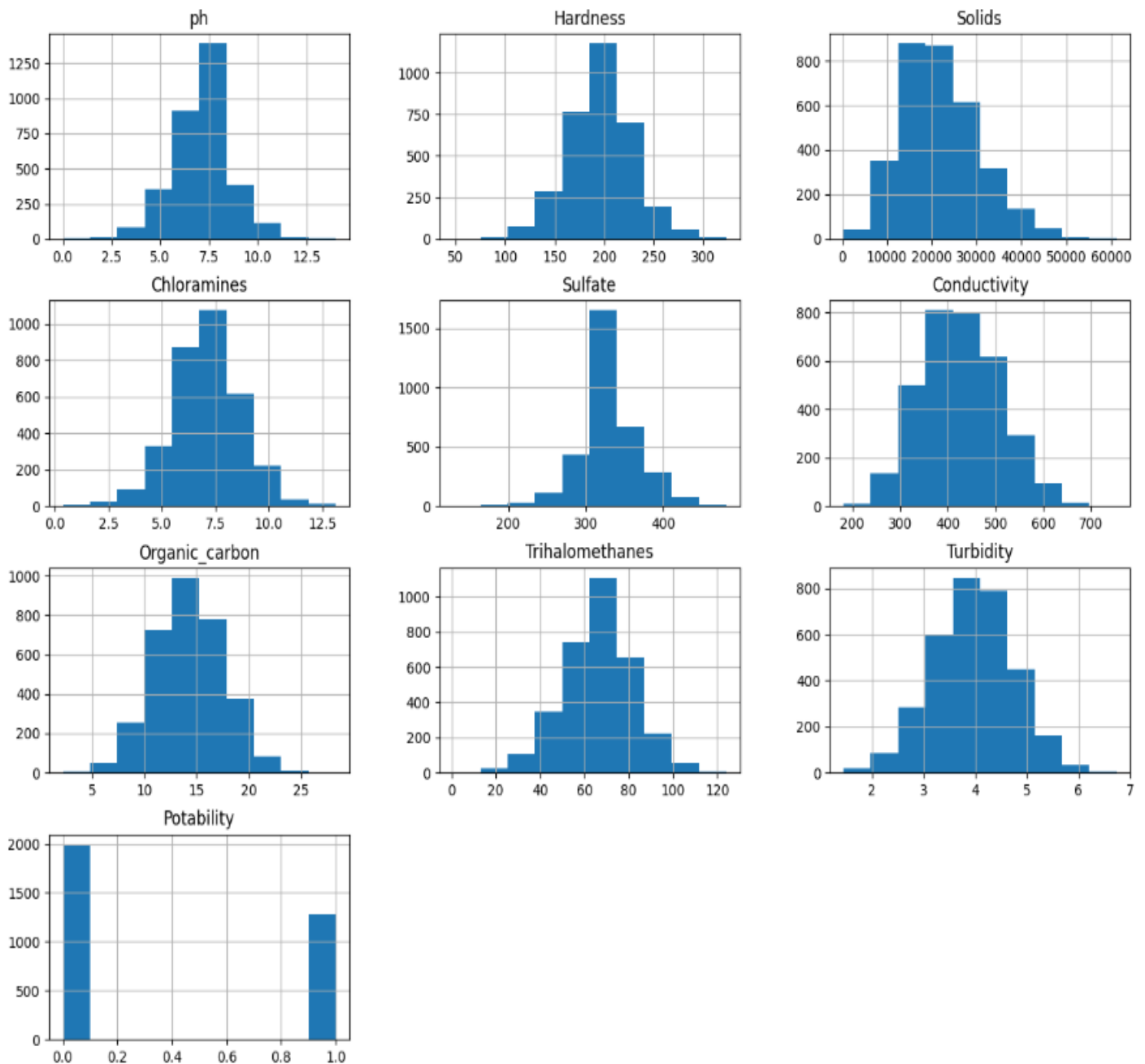# Probability Density by Potability :

```python
for feature in df.columns:
    if feature == "Potability":
        pass
    else:
        plt.figure(figsize=(5,3))
        bar = sns.histplot(df[feature] , kde_kws = {'bw' : 1} ,)
        bar.legend(["Skewness: {:0.2f}".format(df[feature].skew())])
        plt.xlabel(feature)
        plt.ylabel("Probability density")
        plt.title(feature)
        plt.show()
```
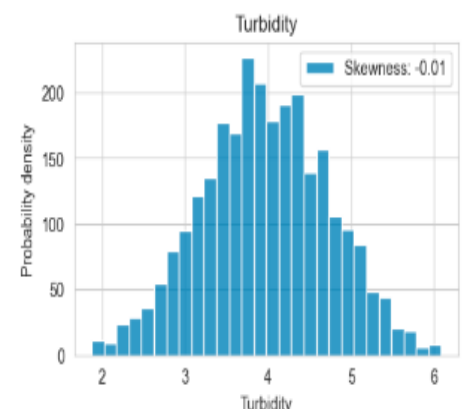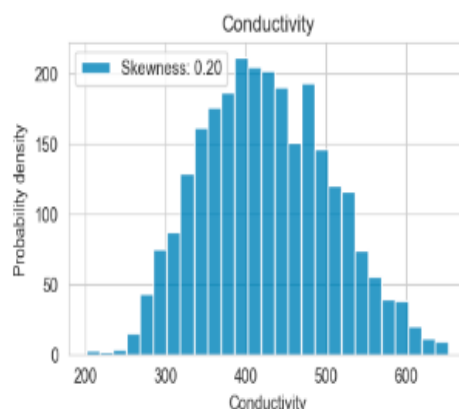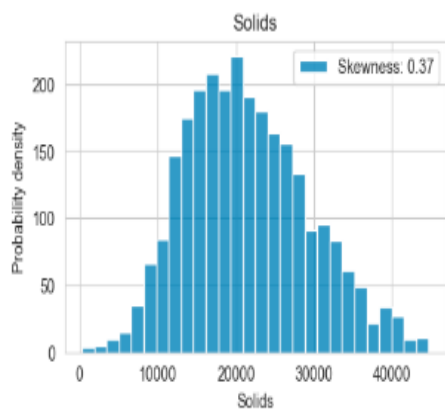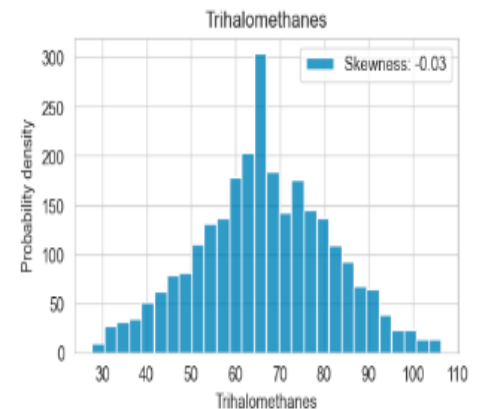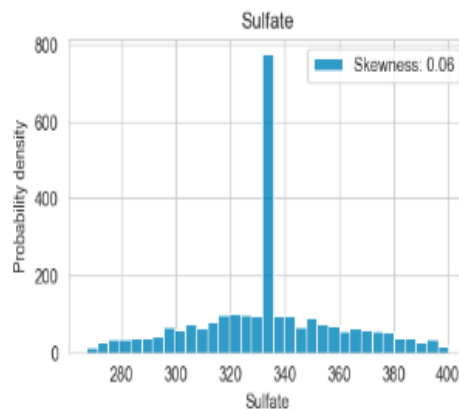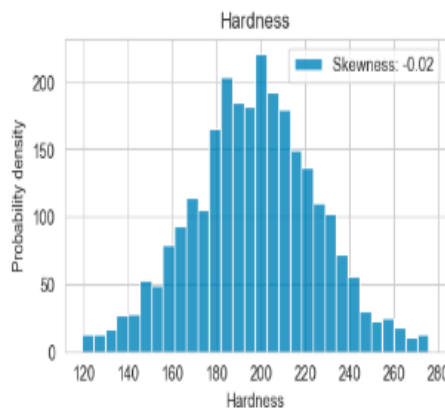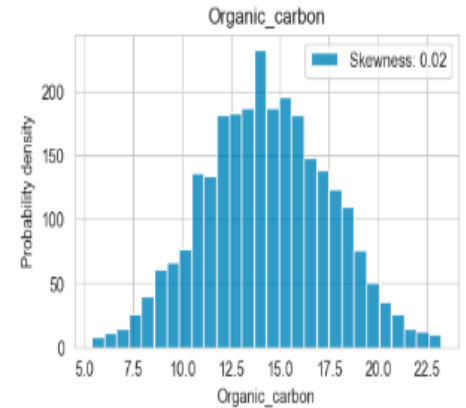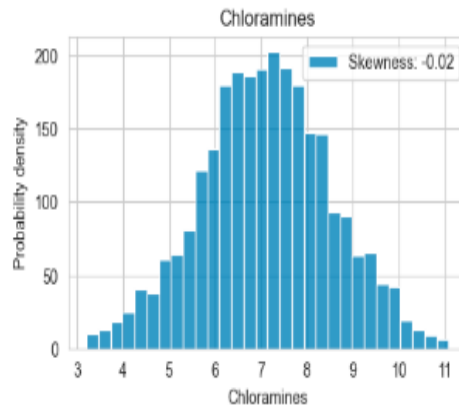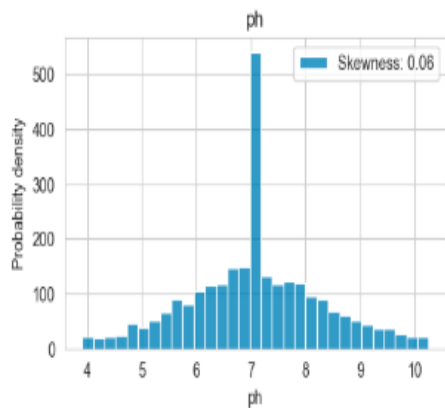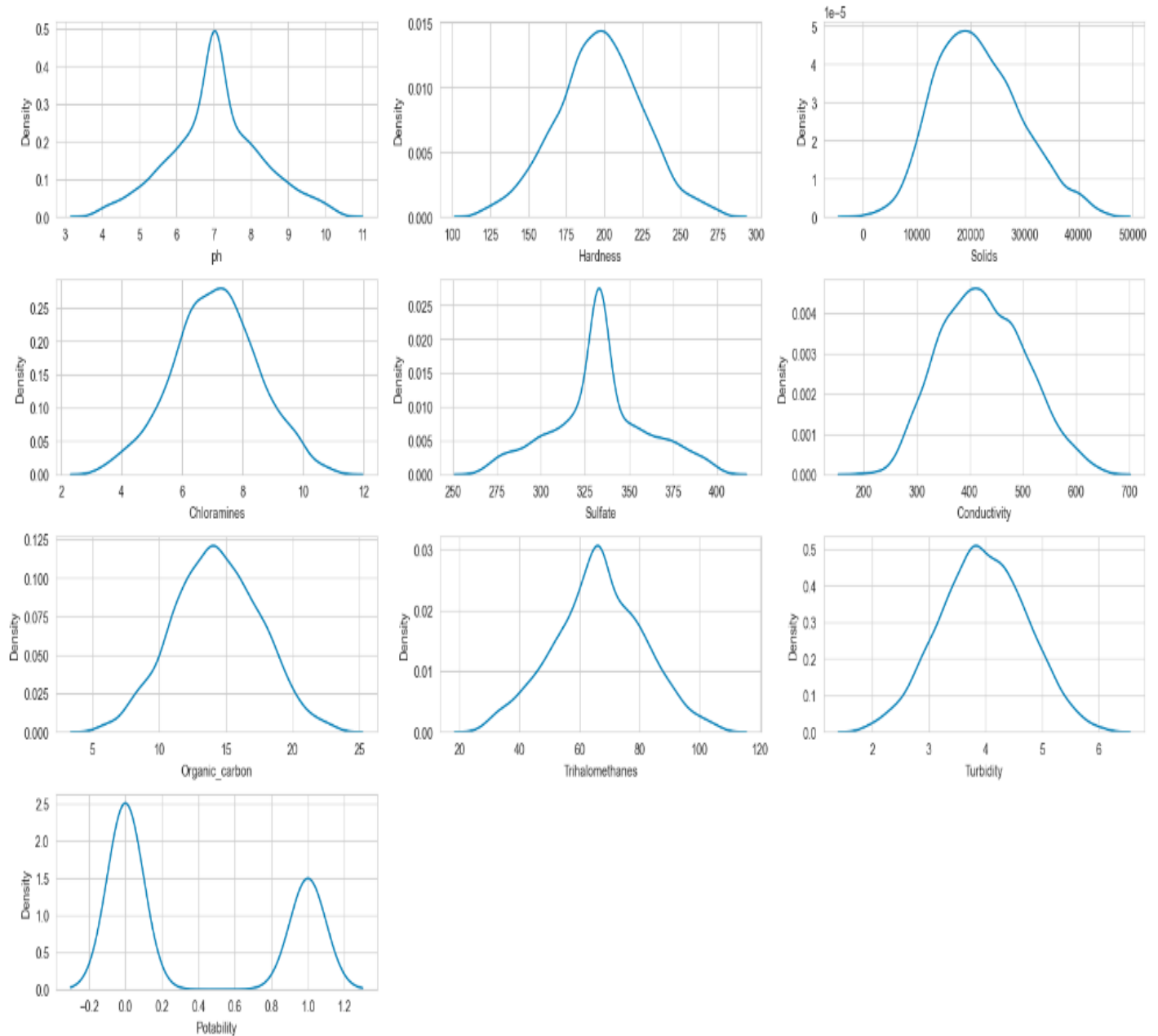
Parameters range by Histogram :

```
df.hist(figsize=(15,12))
```

```
array([[<Axes: title={'center': 'ph'}>,
        <Axes: title={'center': 'Hardness'}>,
        <Axes: title={'center': 'Solids'}>],
       [<Axes: title={'center': 'Chloramines'}>,
        <Axes: title={'center': 'Sulfate'}>,
        <Axes: title={'center': 'Conductivity'}>],
       [<Axes: title={'center': 'Organic_carbon'}>,
        <Axes: title={'center': 'Trihalomethanes'}>,
        <Axes: title={'center': 'Turbidity'}>],
       [<Axes: title={'center': 'Potability'}>, <Axes: >, <Axes: >]],
      dtype=object)
```
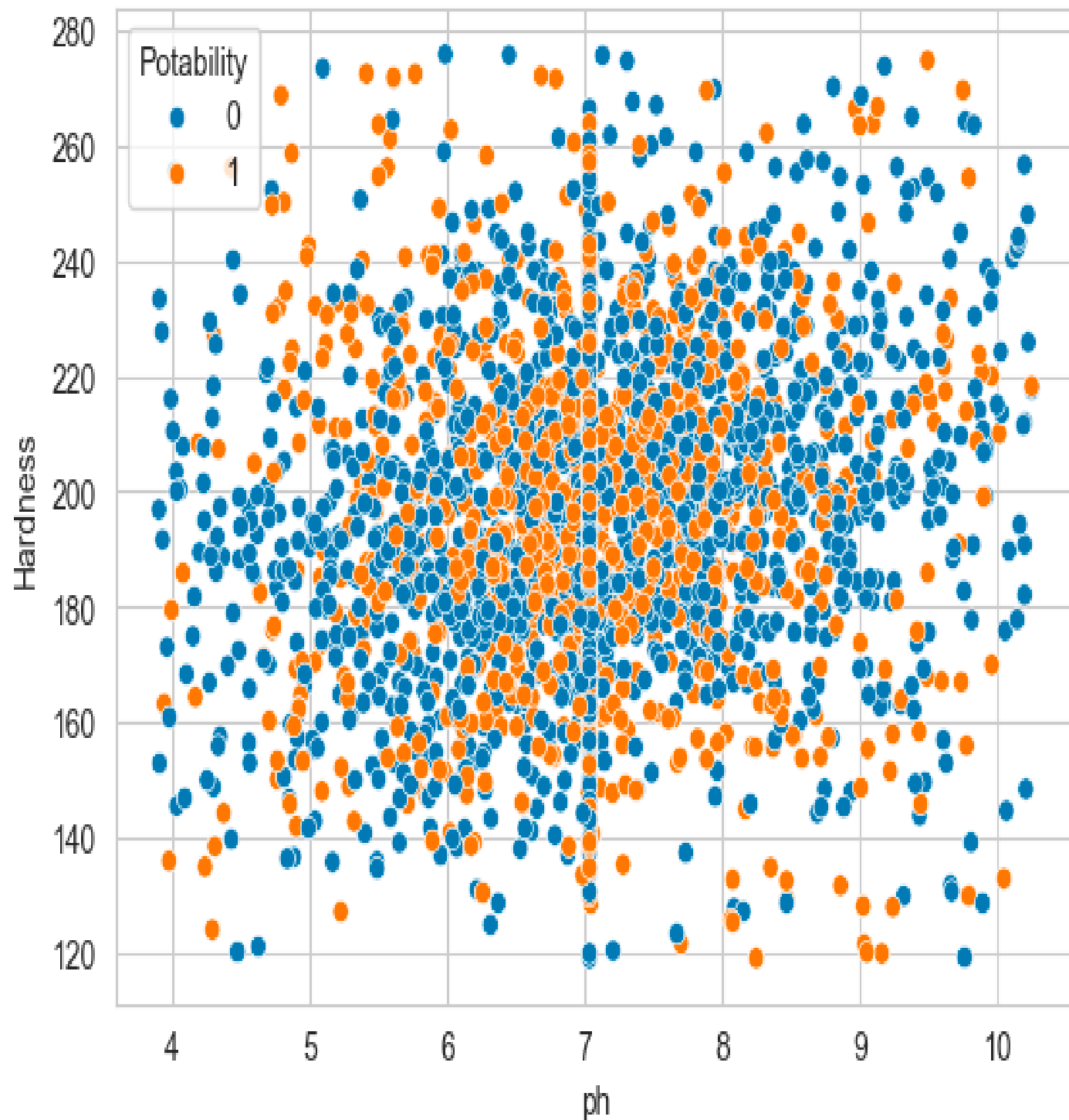
Parameters range by kdeplot :

```python
plt.figure(figsize=(16,10))
for i,col in enumerate(df.columns):
    plt.subplot(4,3,i+1)
    sns.kdeplot(data=df[col])
    plt.tight_layout()
```
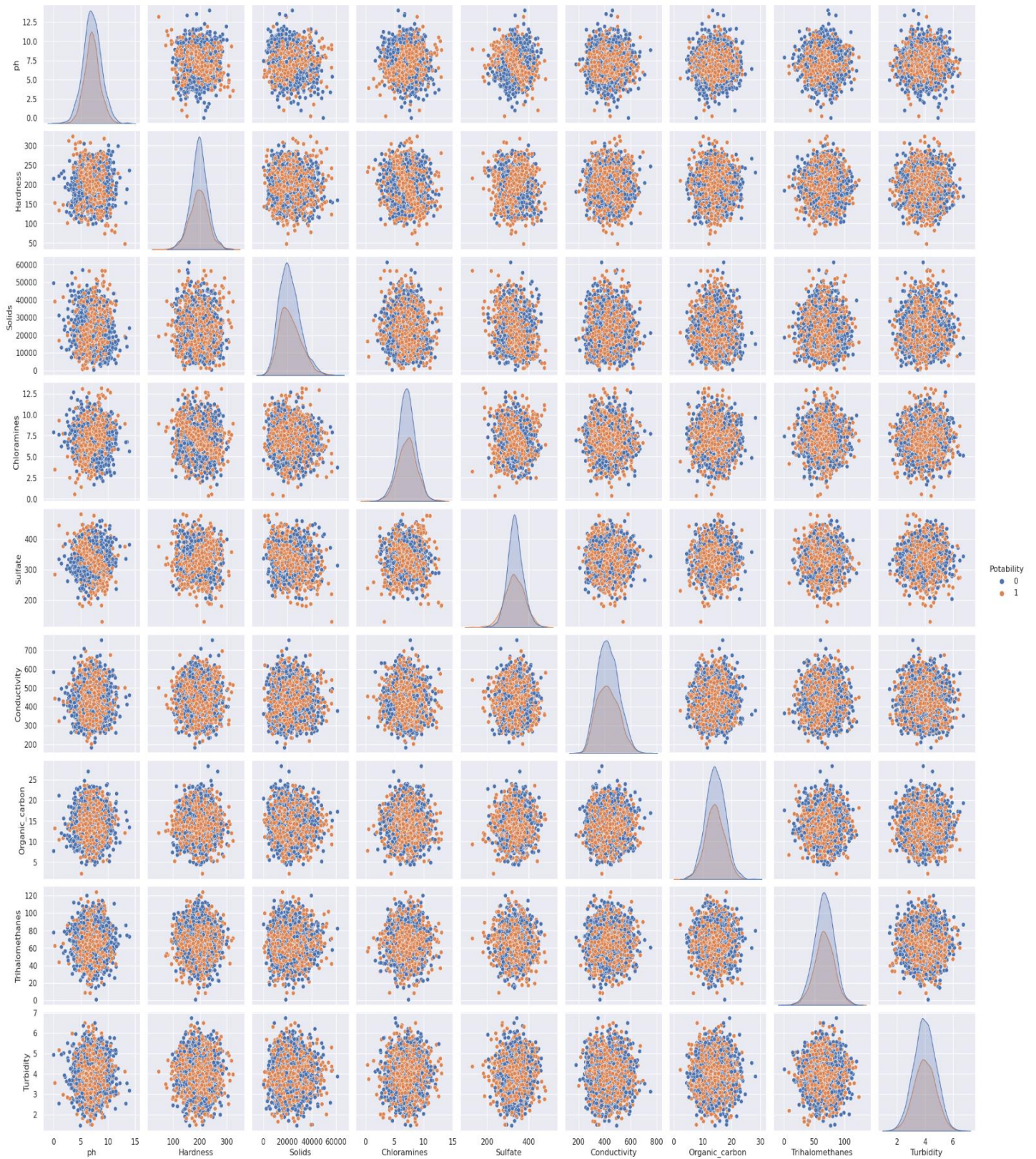
Distribution of pH and Hardness by Scatterplot :

```python
sns.scatterplot(df,x="ph",y="Hardness",hue="Potability")
plt.show()
```

Parameter-wise Realtionship by Pairplot :

```
sns.pairplot(df,hue='Potability')
```

TRAIN TEST SPLITTING :

```python
### splitting data into x and y
X = df.iloc[: , : -1]
y = df.iloc[ : , -1]
```

```python
# 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.

```python
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 :

```python
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.

Importance of Visualizing the Dataset :

The importance of visualizing datasets cannot be overstated in the realm of data analysis. Visualization serves as a powerful bridge between raw data and human comprehension. It enhances our ability to understand, interpret, and extract meaningful insights from complex datasets.

By transforming numbers and statistics into charts, graphs, and interactive displays, visualization offers several key advantages. Firstly, it enables us to detect patterns, trends, and outliers that might remain hidden in tabular data, facilitating more accurate and timely decision-making. Moreover, it supports data exploration by allowing users to interact with the data, making it easier to uncover specific details and refine analysis.

This feature is particularly valuable in the age of big data, where sifting through vast datasets can be a formidable challenge. It is a time-saving tool that provides a rapid overview of data, streamlining the analysis process.

Visualizing the dataset :

- Visualizing a dataset in Python is the process of using data visualization libraries like Matplotlib, Seaborn, or Plotly to create graphical representations of data. The goal is to gain insights, identify patterns, and present data in a visual format that is easy to understand.

- Visualizing a dataset in Cognos is a fundamental aspect of data analysis and interpretation. It involves creating graphical representations of data to uncover patterns, relationships, and insights, making it an essential tool in data-driven decision-making and storytelling.

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.
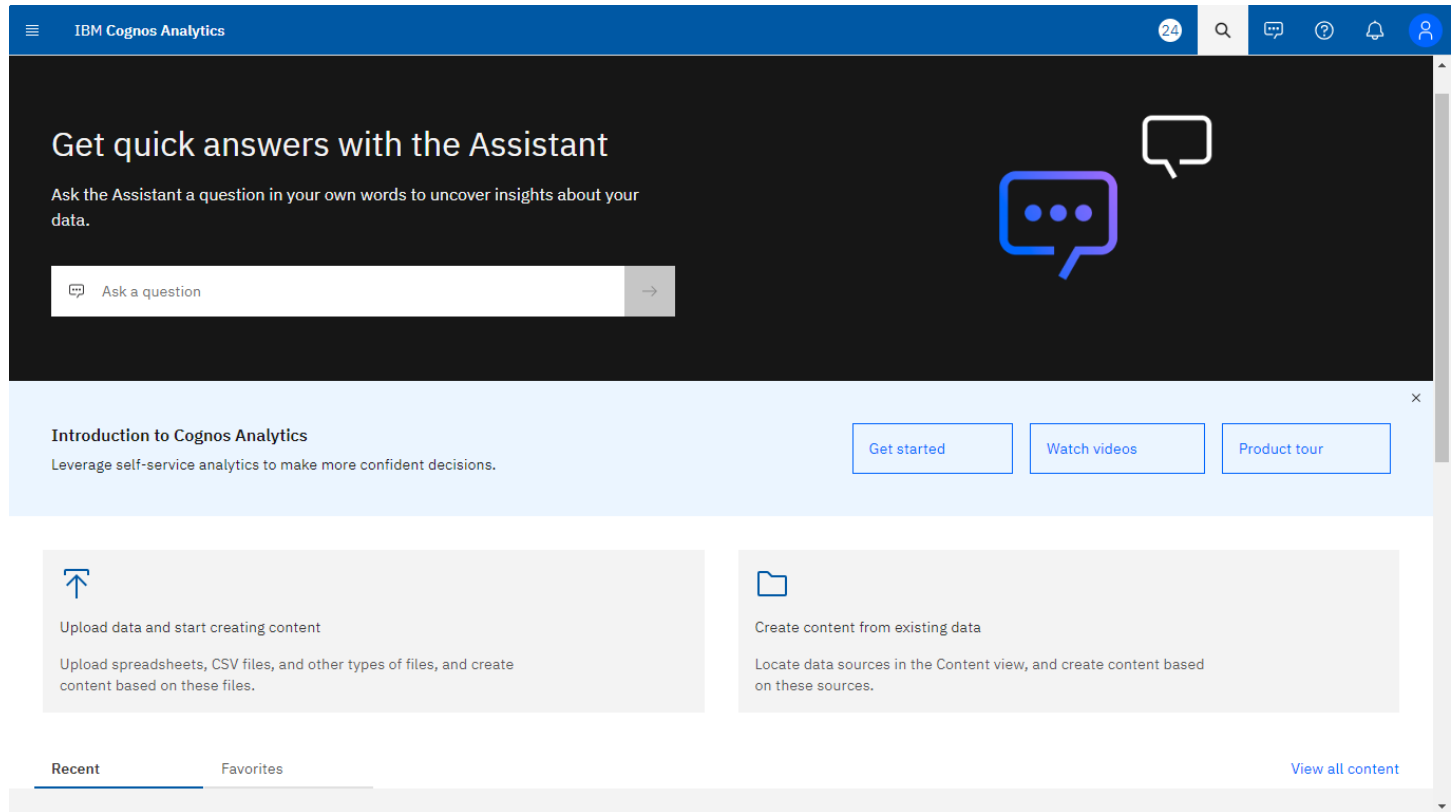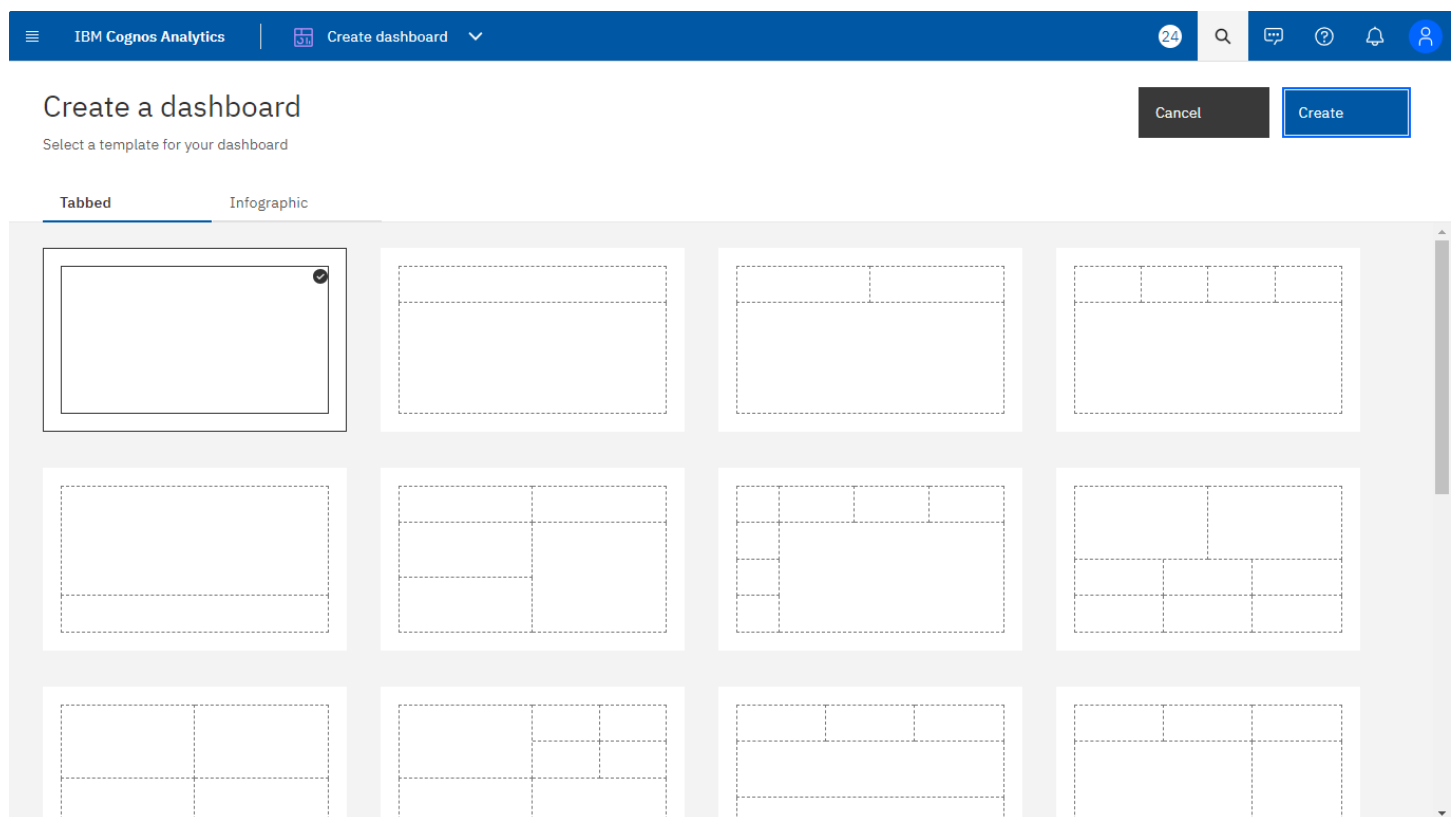
Load The Dataset:

Once you have identified the dataset, you need to load it into the IBM Cognos Analytics. This may involve using a built-in function in the machine learning library, or it may involve writing our own code.

Preprocess The Dataset:

Once the dataset is loaded into the IBM Cognos Analytics Environment, you may need to think a logic before you can start visualizing the dataset. It involves creating graphical representations of data to uncover patterns, relationships, and insights, making it an essential tool in data-driven decision-making and storytelling.

Introduction to IBM Cognos Analytics

1.Upload the Dataset in Cognos Analytics

2.Select the Dataset and Create a Dashboard to Visualize

3.Select the Visualization tool and Visualize respective data

Edit

Point

Analytics | Filters | Fields | Properties

Selected sources / **trained.csv**

🔍 Search

All tabs — Drag and drop data here to filter all tabs.

This tab — Drag and drop data here to filter this tab.

▢ Navigation paths
▾ ⊞ trained.csv
  count
  ph
  Hardness
  Solids
  Chloramines
  Sulfate
  Conductivity
  Organic_carbon
  Trihalomethanes
  Turbidity
  Potability

ph, Chloramines, Organic_carbon and Turbidity by count

Measures
● ph   ● Chloramines   ● Organic_carbon   ● Turbidity

Values

20

15

10

5

0

0   82   162   239   334   429   500   573   641   714   800   872   943   1,011   1,090   1,172   1...
  39   121   200   290   391   466   537   606   676   752   839   906   977   1,047   1,130

count

Visualization of the dataset Using IBM Cognos Analytics

Visualization using Pie Chart :
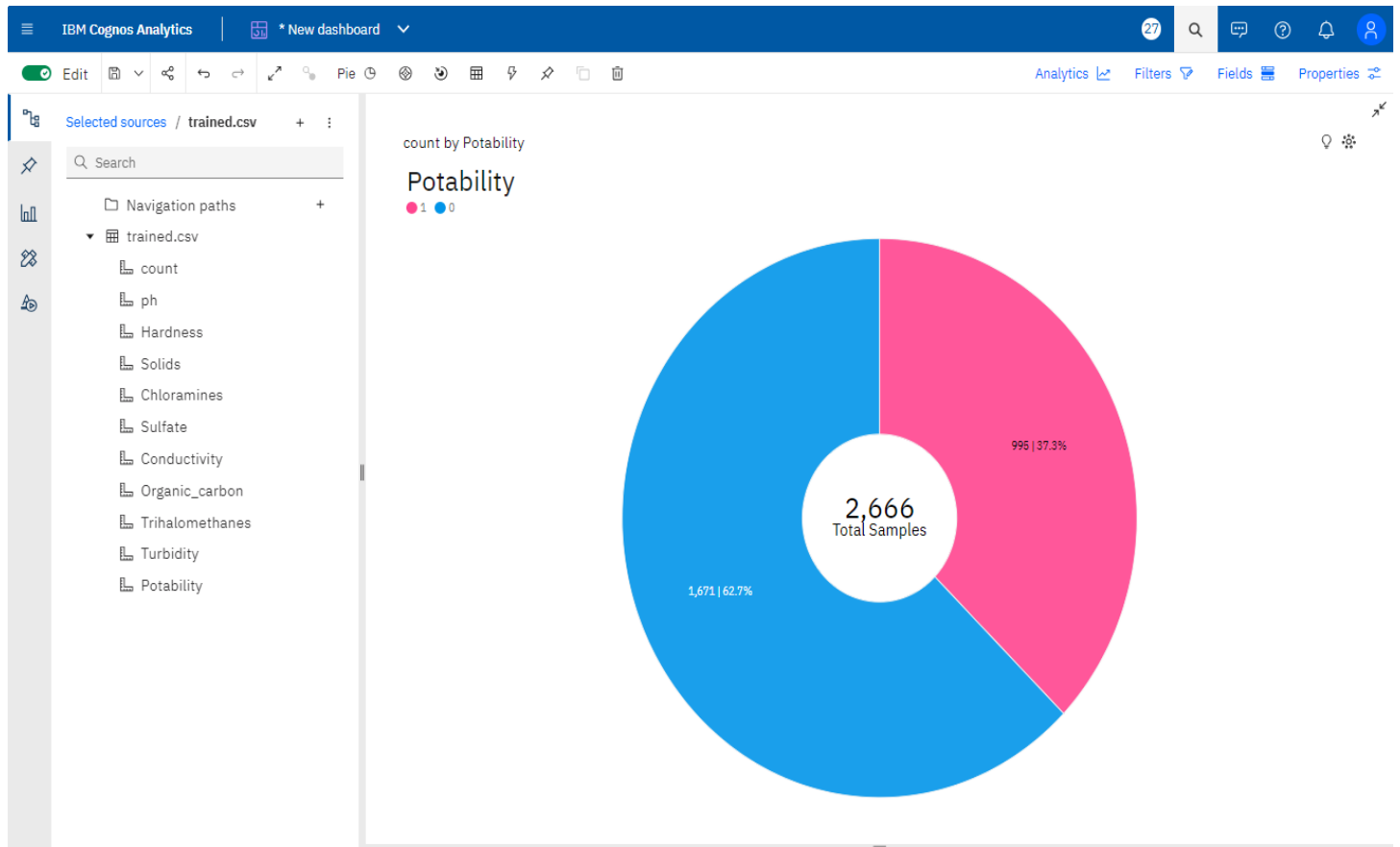
     Visualizing the Potability data using Pie Chart provided in the Cognos.

     Here the insights of the Pie chart is the dataset contains 62.7% Not Potable and 37.3% Potable datas among the 2666 Water Samples.

Visualizing using the Points :
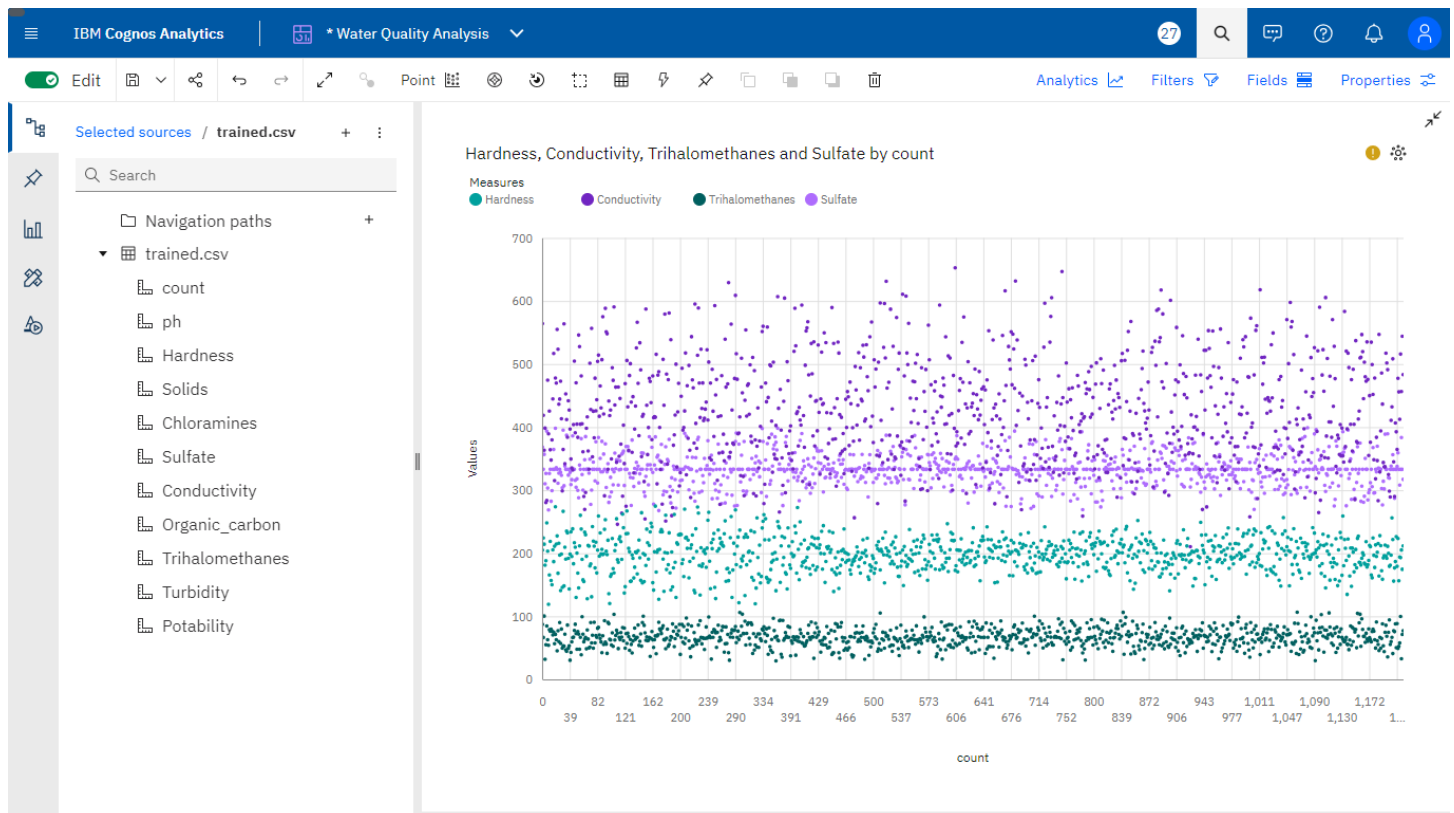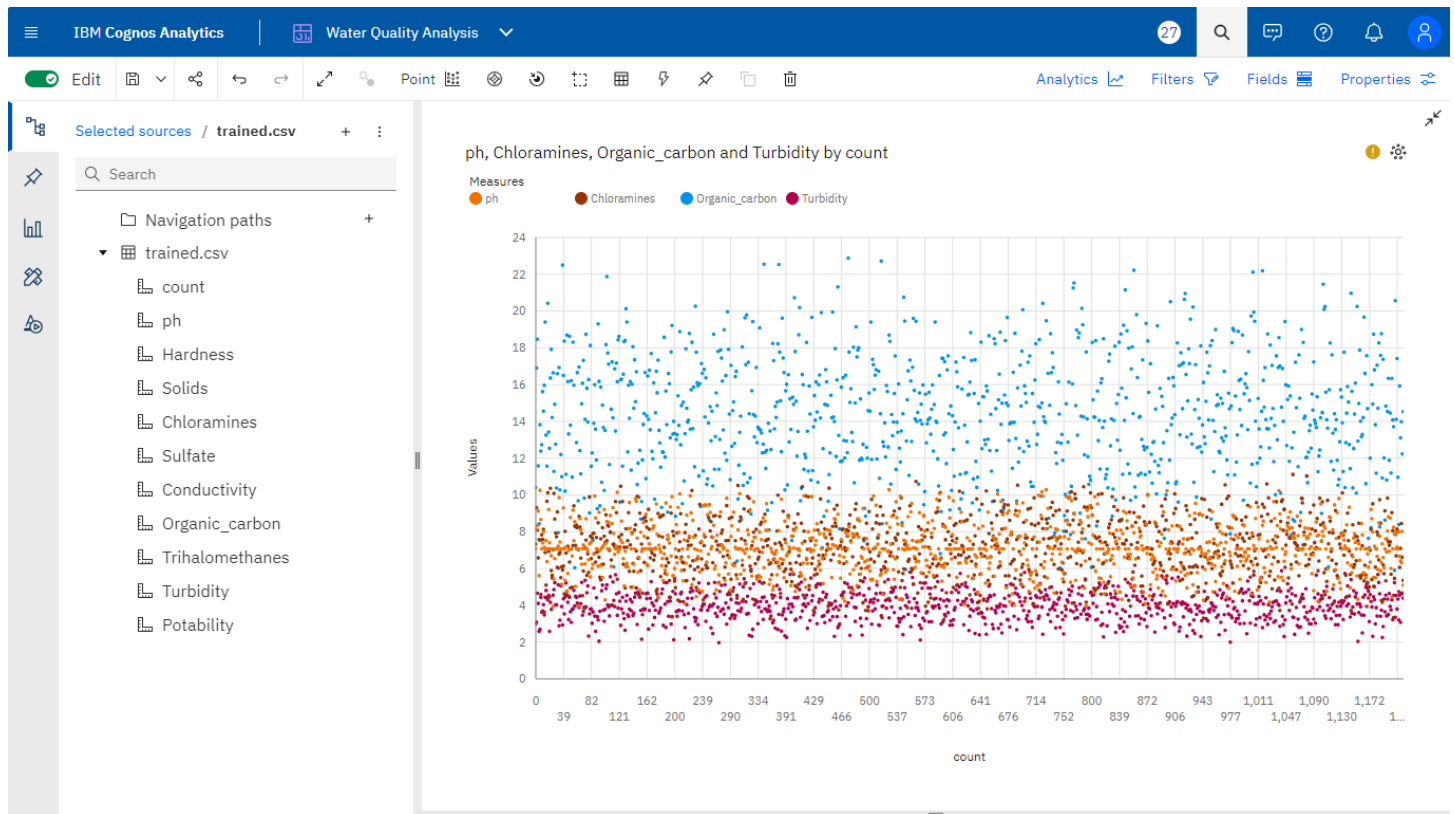
Visualizing the Distribution of each Water Quality Parameters provided in the Dataset using Point Chart provided in the Cognos.

Visualization using Line Chart :
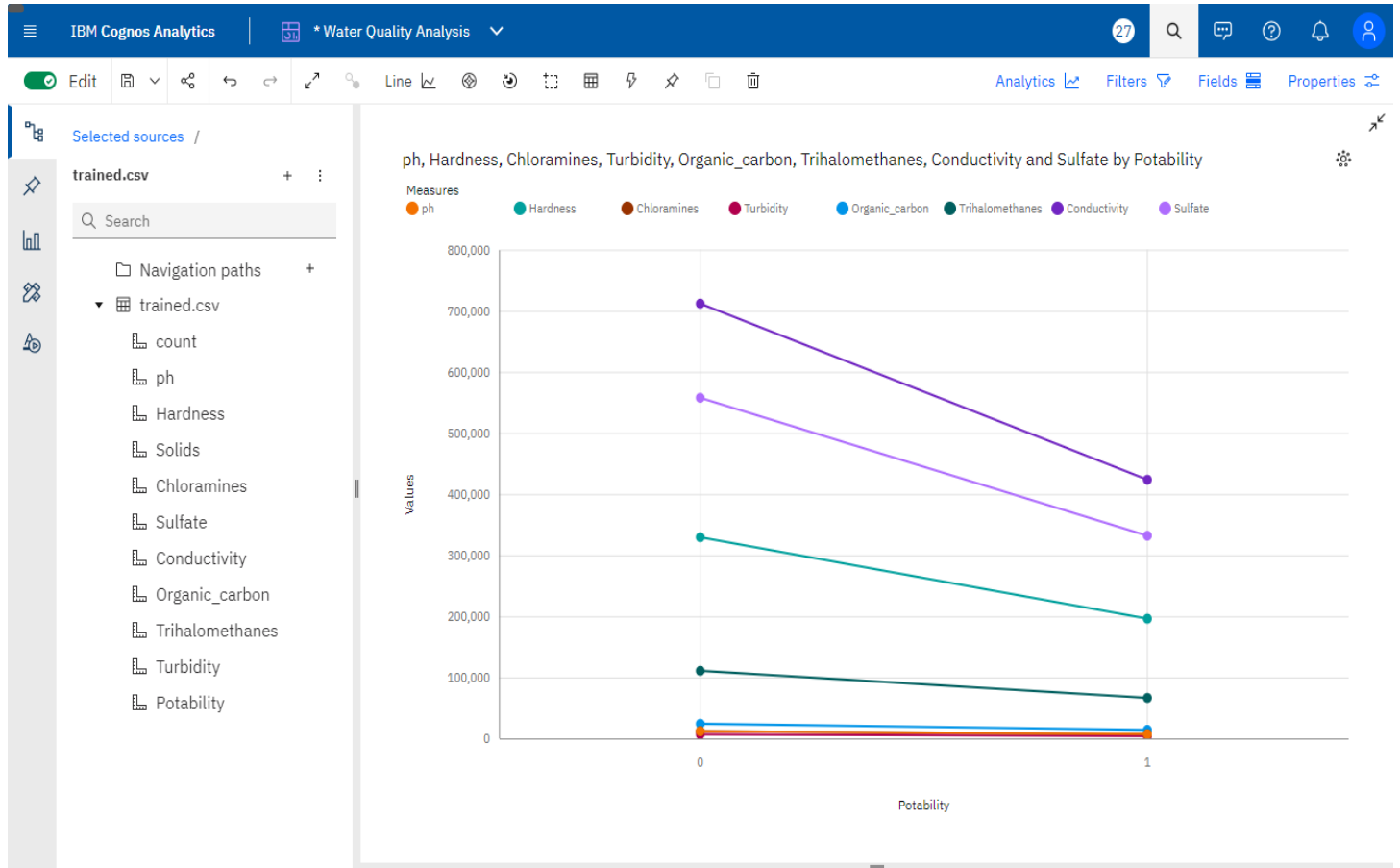
    Visualizing the Potability of each water quality parameters using Line Chart provided in the Cognos.

Visualization using the Pie Chart :
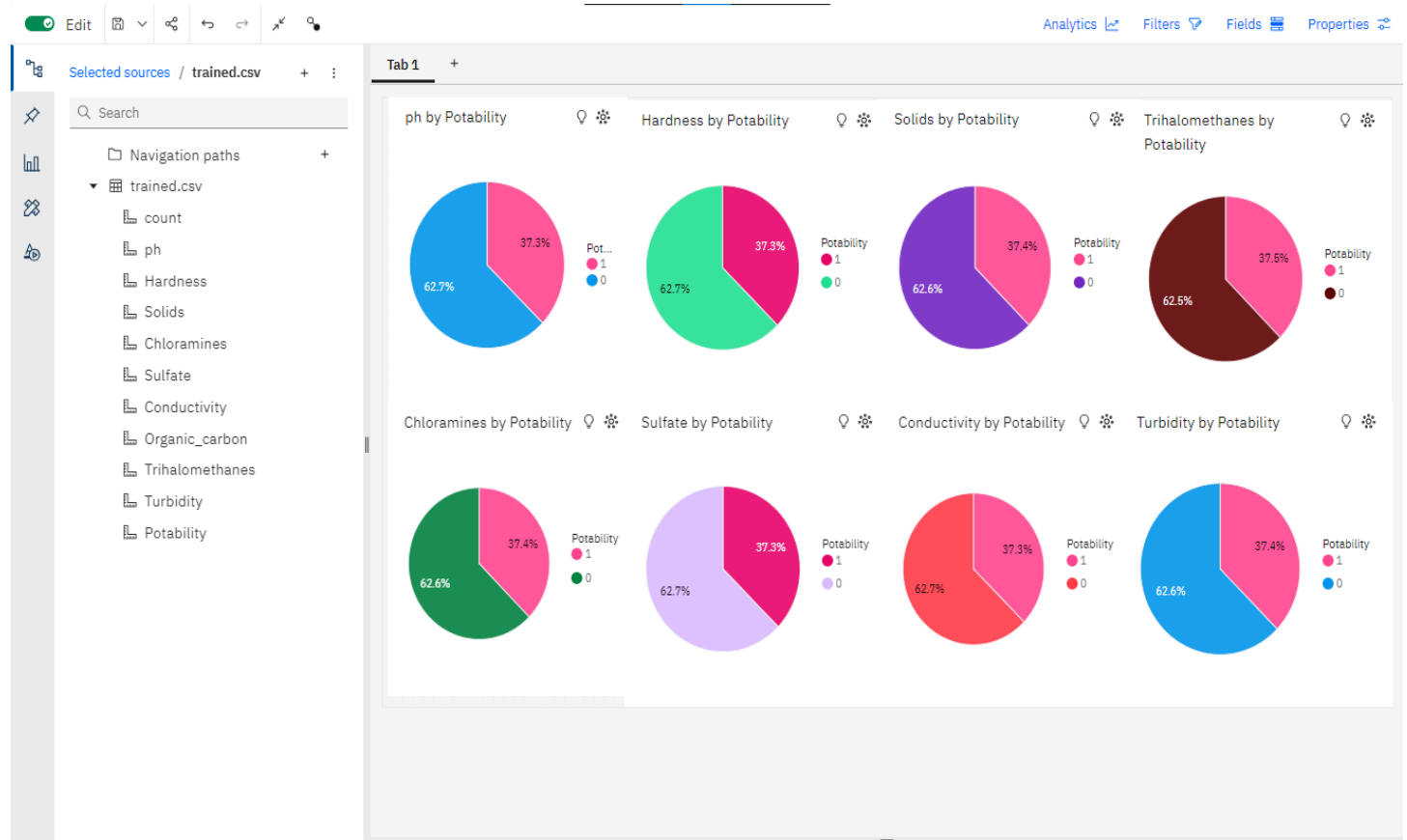
      Visualizing the Potability of each water quality parameters using Pie Chart provided in the Cognos.

VISUALIZATION USING THE HISTPLOT :
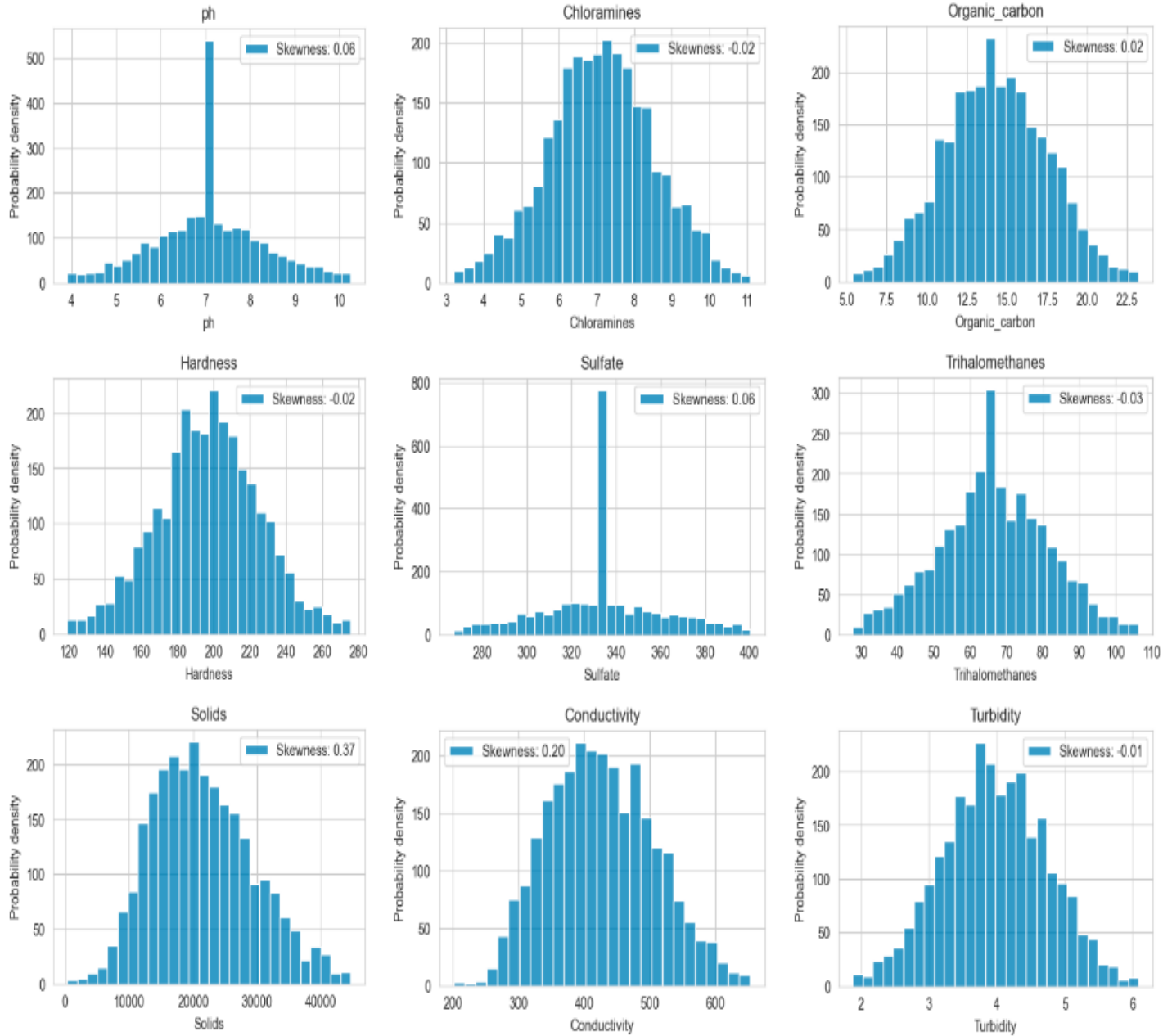
Visualizing the Potability Density of each water quality parameters using Histplot provided in the Cognos.

Visualizing Correaltion :

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
ph	1	0.13	-0.087	-0.063	0.016	0.0022	0.026	0.0093	-0.043	-0.0055
Hardness	0.13	1	-0.046	-0.036	-0.028	-0.03	0.012	-0.0071	-0.022	-0.00071
Solids	-0.087	-0.046	1	-0.04	-0.11	0.0071	0.027	-0.024	0.023	0.0056
Chloramines	-0.063	-0.036	-0.04	1	0.025	-0.015	-0.00068	0.017	-0.0054	0.013
Sulfate	0.016	-0.028	-0.11	0.025	1	-0.0059	-0.0073	-0.022	-0.017	-0.0039
Conductivity	0.0022	-0.03	0.0071	-0.015	-0.0059	1	0.0065	-0.0012	0.0077	-0.0019
Organic_carbon	0.026	0.012	0.027	-0.00068	-0.0073	0.0065	1	-5e-05	-0.017	-0.027
Trihalomethanes	0.0093	-0.0071	-0.024	0.017	-0.022	-0.0012	-5e-05	1	-0.024	0.014
Turbidity	-0.043	-0.022	0.023	-0.0054	-0.017	0.0077	-0.017	-0.024	1	0.0048
Potability	-0.0055	-0.00071	0.0056	0.013	-0.0039	-0.0019	-0.027	0.014	0.0048	1

Splitting the Dataset into Training and Test sets :

```python
### splitting data into x and y
X = df.iloc[: , : -1]
y = df.iloc[ : , -1]
```

```python
# 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)
```

In this code, we first separate the features (X) and the target variable (Y) from the dataset. Then, we use train_test_split to split the data into training and test sets. The test_size parameter determines the proportion of the data that will be allocated to the test set, and random_state is set to a specific value (e.g., 42) to ensure reproducibility.

Feature Scaling :

Feature scaling is an important preprocessing step in many machine learning algorithms. You can use the StandardScaler from scikit-learn to scale your features so that they have a mean of 0 and a standard deviation of 1.

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

In this code, we first create a StandardScaler instance. We then fit the scaler on the training data using the fit_transform method, and apply the same transformation to both the training and test data using the transform method. This ensures that the scaling is consistent between the two sets.

Machine Learning Algorithms :

Machine learning algorithms play a crucial role in data analysis by enabling automated data modeling, pattern recognition, and predictive analytics. Machine learning algorithms enhance data analysis by automating complex tasks, uncovering hidden patterns, and providing data-driven insights.

Library :

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

Logistic Regression :

Logistic Regression is a commonly used algorithm for binary and multiclass classification problems. It is a fundamental classification algorithm that models the probability of class membership using the sigmoid function. It estimates parameters to create a decision boundary that separates data points into different classes. Accuracy is used to evaluate the model's performance by comparing its predictions to actual class labels.

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression()
log_reg.fit(X_train_final, y_train)
y_pred = log_reg.predict(X_test_final)
log_acc=accuracy_score(y_test, y_pred)
print(classification_report(y_test, y_pred))
print("Test Set Accuracy : ",log_acc)
```

	precision	recall	f1-score	support
0	0.62	1.00	0.77	497
1	0.00	0.00	0.00	303
accuracy			0.62	800
macro avg	0.31	0.50	0.38	800
weighted avg	0.39	0.62	0.48	800

```
Test Set Accuracy :  0.62125
```

The Test Accuracy of Logistic Regression is 62%

K-Nearest Neighbor Classifier :

     KNN is a simple yet effective supervised machine learning algorithm used for both classification and regression tasks. It operates on the principle of similarity and is based on the idea that data points with similar features are more likely to belong to the same class or have similar target values. KNN is a straightforward algorithm that relies on the concept of similarity to classify or predict data points. It is non-parametric and lazy (as it doesn't build a model during training), making it suitable for various tasks.

```python
from sklearn.neighbors import KNeighborsClassifier


knn = KNeighborsClassifier(n_neighbors=9)
knn.fit(X_train_final, y_train)
y_pred = knn.predict(X_test_final)
knn_acc = accuracy_score(y_test,y_pred)


print(classification_report(y_test, y_pred))
print("Test Set Accuracy : ", knn_acc)
```

	precision	recall	f1-score	support
0	0.65	0.86	0.74	497
1	0.51	0.24	0.33	303
accuracy			0.62	800
macro avg	0.58	0.55	0.53	800
weighted avg	0.60	0.62	0.58	800

Test Set Accuracy : 0.62375

The Test Accuracy of K-Nearest Neighbor is 62%

SUPPORT VECTOR CLASSIFIER :

    SVM is a powerful algorithm for classification and regression tasks that aims to find an optimal hyperplane to separate different classes while maximizing the margin between them. It can handle both linear and nonlinear data, and its performance is evaluated using accuracy metrics on training and test datasets.

```python
from sklearn.svm import SVC

svc_classifier = SVC(class_weight = "balanced"  , C=100, gamma=0.01)
svc_classifier.fit(X_train_final, y_train)
y_pred_scv = svc_classifier.predict(X_test_final)
svm_acc = accuracy_score(y_test, y_pred_scv)

print(classification_report(y_test, y_pred))
print("The Test Accuracy is : ",svm_acc)
```

```
              precision    recall  f1-score   support

           0       0.66      0.87      0.75       497
           1       0.55      0.26      0.35       303

    accuracy                           0.64       800
   macro avg       0.60      0.56      0.55       800
weighted avg       0.62      0.64      0.60       800


The Test Accuracy is :  0.6325
```

The Test Accuracy of Support Vector Classifier is 63%

DECISION TREE CLASSIFIER :

   A Decision Tree is a machine learning algorithm used for both classification and regression tasks. It builds a tree-like structure of decisions based on feature values to classify data. It makes use of impurity measures like entropy to determine the best feature splits. By controlling the tree's depth and evaluating its accuracy, one can create a model that balances between fitting the training data well and generalizing to new data.

```python
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier(criterion='entropy',max_depth=5)
dtc.fit(X_train_final, y_train)
y_pred = dtc.predict(X_test_final)
dtc_acc= accuracy_score(y_test,dtc.predict(X_test_final))

print(classification_report(y_test, y_pred))
print("Test Set Accuracy : ", dtc_acc)
```

	precision	recall	f1-score	support
0	0.63	0.92	0.75	497
1	0.46	0.11	0.17	303
accuracy			0.61	800
macro avg	0.54	0.51	0.46	800
weighted avg	0.56	0.61	0.53	800

Test Set Accuracy : 0.61375

The Test Accuracy of Decision Tree Classifier is 61%

Algorithm Analysis :

This code facilitates the comparison of different machine learning models by recording and presenting their training and test accuracy in a structured table, sorted by test accuracy for easy assessment.

```python
models = pd.DataFrame({
    'Model': ['Logistic','KNN', 'SVM',  'Decision Tree Classifier'],
    'Test': [ log_acc,knn_acc, svm_acc, dtc_acc]
})

models.sort_values(by = 'Test', ascending = False)
```

	Model	Test
2	SVM	0.63250
1	KNN	0.62375
0	Logistic	0.62125
3	Decision Tree Classifier	0.61375

This code organizes the performance metrics of different machine learning models in a structured tabular format, sorts the models based on their test accuracy, and provides a clear comparison of how well each model performed on the test dataset. This is a common practice to help data analysts and machine learning practitioners choose the best model for a given task.

ALGORITHM VISUALIZATION :

This code generates a bar plot using Seaborn to visually compare and present the test accuracy of different machine learning algorithms. The choice of colors, style, and figure size enhances the readability and presentation of the plot.

```python
colors = ["green","yellow", "black", "cyan"]
sns.set_style("whitegrid")
plt.figure(figsize=(8,3))
sns.barplot(x=models['Model'],y=models['Test'], palette = colors )
plt.ylabel("Accuracy")
plt.xlabel("Algorithms")
plt.show()
```
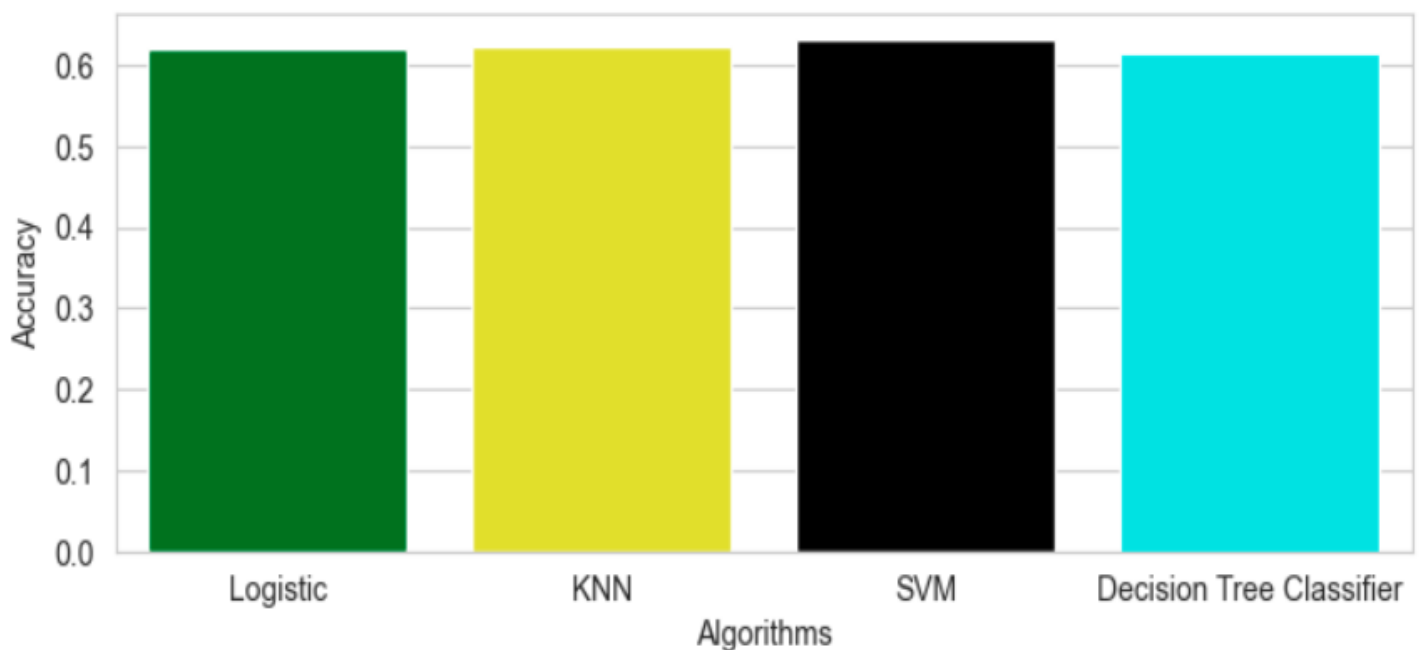

The accuracy score for support vector machine is 63%. As compare with other models the accuracy score is much higher in support vector machine.

Water Quality Predictive Model

In this project, we developed a water quality predictive model using the Support Vector Machine (SVM) algorithm within a Tkinter-based graphical user interface in Python. The model leverages historical water quality data to predict water quality levels based on various parameters, such as pH, Hardness, Solids, Chloramines, Sulphate, Conductivity, Organic_Carbon, Trihalomethanes and Turbidity. The Tkinter GUI allows users to input these parameters and receive real-time predictions, making it a valuable tool for environmental monitoring and ensuring the safety of water resources.

Code :

```python
import tkinter as tk
from tkinter import Entry, Button, Label, Frame

import pickle
import pandas as pd
import numpy as np
import joblib

scaler = joblib.load("final_scaler.save")
model = pickle.load(open('final_model.pkl', 'rb'))

# Function to make a prediction
def predict():
    input_features = [float(entry.get()) for entry in entry_widgets]
    features_value = [np.array(input_features)]
    feature_names = ["ph", "Hardness", "Solids", "Chloramines", "Sulfate", "Conductivity",
"Organic_carbon","Trihalomethanes", "Turbidity"]
    df = pd.DataFrame(features_value, columns=feature_names)
    df = scaler.transform(df)
    output = model.predict(df)

    if output[0] == 1:
        prediction = "safe"
        result_label.config(text="Water is Safe for Human Consumption", fg="#68FF00")
    else:
        prediction = "not safe"
        result_label.config(text="Water is Not Safe for Human Consumption", fg="red")

# Create a Tkinter application window
app = tk.Tk()
app.title("Water Quality Prediction")

# Set the window geometry and background color
app.geometry("470x480")
```

```python
app.configure(bg='#ABFFF1')

# Create a frame for the input fields and labels
input_frame = Frame(app, bg='#ABFFF1')
input_frame.pack(pady=10)

# Create input entry fields with labels
entry_labels = ["pH:", "Hardness:", "Solids:", "Chloramines:", "Sulfate:",
                "Conductivity:", "Organic Carbon:", "Trihalomethanes:", "Turbidity"]
entry_widgets = []

for label_text in entry_labels:
    label = Label(input_frame, text=label_text, bg='#ABFFF1', font=("copperplate gothic
bold", 14,"bold"), fg='black')
    label.grid(row=entry_labels.index(label_text), column=0, sticky='w', padx=10, pady=5)
    entry = Entry(input_frame, font=("Arial", 12))
    entry.grid(row=entry_labels.index(label_text), column=1, padx=10, pady=5)
    entry_widgets.append(entry)

# Create a prediction button with custom style
predict_button = Button(app, text="Predict", command=predict, font=("copperplate gothic
bold", 16,"bold"), bg='#63F07B', fg='black',
                        width = 12)
predict_button.pack(pady=10)

# Create a label to display the prediction with increased font size and custom style
result_label = Label(app, text="", font=("copperplate gothic bold", 14), bg="#ABFFF1")
result_label.pack()

# Start the Tkinter main loop
app.mainloop()
```
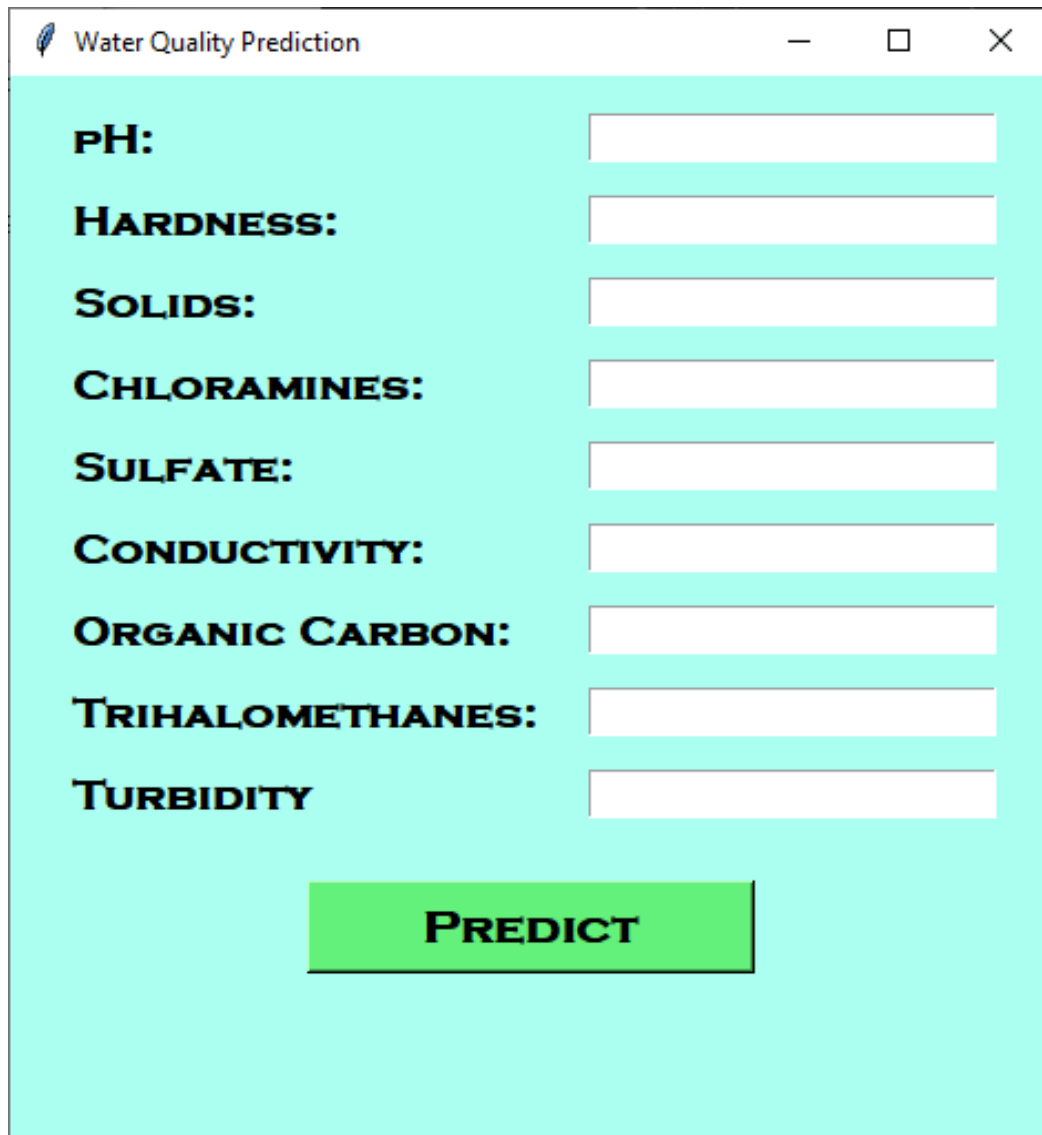
OUTPUT :

HOME INTERFACE :

   Let's test the predictive model using the water parameters provided in the dataset.

TESTING WITH POTABLE VALUES :

Here, we tested the Potable values provided in the dataset. So the Predictive model shows the result i.e "Water is Safe for Human Consumption".

Testing with Not Potable Values :

Here, we tested the Not Potable values provided in the dataset. So the Predictive model shows the result i.e "Water is Not Safe for Human Consumption".

Advantages :

- **Protection of Public Health:** Ensuring that drinking water is of high quality is essential for preventing waterborne diseases. Water quality analysis helps identify and mitigate potential health risks associated with contaminants and pathogens in water sources.
- **Environmental Conservation:** Monitoring water quality is vital for protecting aquatic ecosystems. It helps in identifying pollution sources, preventing habitat degradation, and preserving biodiversity.
- **Safe Recreation:** Water quality analysis is critical for recreational water bodies like swimming pools, lakes, and beaches. It helps in preventing recreational waterborne diseases and ensuring safe swimming and other activities.
- **Agricultural and Irrigation Use**: For agriculture, the quality of water used for irrigation is crucial. Poor water quality can negatively impact crop growth. Water analysis helps in making informed decisions about water use in agriculture.
- **Early Detection of Pollution:** Regular monitoring and analysis of water quality can help detect pollution incidents or spills early, allowing for rapid response and containment, reducing the environmental impact.
- **Research and Data:** Water quality data is essential for scientific research and understanding long-term trends. Researchers can use this data to study the effects of climate change, pollution, and other environmental factors.
- **Resource Management:** In regions with water scarcity, water quality analysis helps in efficient resource management. It can guide decisions on water treatment and distribution strategies.
- **Public Awareness:** Water quality analysis results can be used to inform and educate the public about the importance of water conservation, pollution prevention, and the significance of clean water for various purposes.

Overall, water quality analysis is a fundamental tool for maintaining a clean and safe water supply, protecting the environment, and supporting various sectors of society. It allows for informed decision-making and the protection of public health and natural resources.

Disadvantages :

   While water quality analysis is essential for various reasons, it is not without its disadvantages and challenges. Here are some of the potential disadvantages of water quality analysis:

- **Cost:** Water quality analysis can be expensive, especially when it involves the purchase of equipment, chemicals, and specialized instruments. The cost may be a barrier for some organizations or individuals.
- **Complexity:** Conducting water quality analysis can be technically challenging and requires skilled personnel. Training and expertise are needed to operate instruments, interpret data, and ensure accurate results.
- **Time-Consuming:** Water quality analysis can be time-consuming, particularly when multiple parameters need to be tested, and samples must be collected and processed. This can lead to delays in obtaining results.
- **Sample Collection and Preservation:** Proper sample collection and preservation are critical for accurate analysis. Failing to collect samples correctly or preserve them adequately can lead to misleading results.
- **Data Interpretation:** Interpreting water quality data requires an understanding of the specific context and local regulations. Misinterpretation of data can lead to incorrect decisions or regulatory violations.
- **Inaccuracies and Errors:** Analytical instruments and methods may have limitations and sources of error. Calibration issues, contamination, or equipment malfunctions can lead to inaccurate results.
- **Limited Coverage:** Water quality analysis typically focuses on a specific set of parameters, and not all potential contaminants or pollutants may be tested. Some emerging contaminants may not yet be included in standard tests.
- **Data Management:** Handling and managing large datasets generated from water quality analysis can be challenging and may require dedicated data management systems.

   Despite these disadvantages, the benefits of water quality analysis generally outweigh the drawbacks. Efforts to address these challenges, such as improved technology, standardized methods, and better training, are ongoing to ensure that water quality analysis remains a valuable tool for protecting human health and the environment.

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.

- Using multiple Machine Learning Algorithms, we concluded highest accuracy algorithm to make the water quality predictive model.

- The Water Quality Predictive Model is trained using SVM algorithm with the provided water samples data.

- Then the Water Quality Predictive Model is tested with several Water Quality Parameters and it provides the result whether its Potable or Not.