WATER QUALITY ANALYSIS

Problem Definition:

- Deteriorating water quality is damaging the environment, health conditions and the global economy.In addition, here are some of the other consequences. Lack of potable water. Introduce toxins into foods which Are harmful to our health when eaten.

Problem statement:

- IN THE WORLD WATER POLLUTION IS INCREASING DAY BY DAY. PEOPLE ARE NOT AWARE ABOUT HARMFUL EFFECT OF WATER POLLUTION.SO THERE IS NEED OF AWARENESS IN OUR SOCIETY AND IN THE WORLD TO SAVE IT FOR FUTURE.

Design Thinking:

1.Data Collection: We will need a dataset containing labeled examples of Water Quality test Analysis We can use a Kaggle dataset for this purpose.

2.Data Preprocessing: The text data needs to be cleaned and preprocessed. This involves removing special characters, converting text to lowercase, and tokenizing the text into individual words.

3.Feature Extraction: We will convert the tokenized words into numerical features using techniques like TF-IDF (Term Frequency-Inverse Document Frequency).

4.Model Selection: We can experiment with various Data Analytics algorithms such as Cognos technology , and more advanced techniques like data analytics using neural networks

5.Evaluation:We will measure the model's performance using metrics like accuracy, precision, recall, and F1-score.
6.Iterative Improvement: We will fine-tune the model and experiment with hyperparameters to improve its accuracy.

Algorithms Used:

Similarity-based: Filter the sorted water quality analysis parameters using given sort list.
Sample-based: Templates of Water analysis parameters ,testing Parameters allow Data analytics to assess Water parameters .Using these algorithms, we could filter the Water Quality testing .

Dataset used:
https://www.kaggle.com/datasets/adityakadiwal/water-potability

We have used Water_potability.csv file

Data Preprocessing:

Import libraries

Using the dataset perform all the operations.

WATER QUALITY ANALYSIS

Submitted by,
P. Saravana kumar

Team Members :
R. Selva kumar
P. Saravana kumar
M. Sudalai
T. Suba
S. Sathya

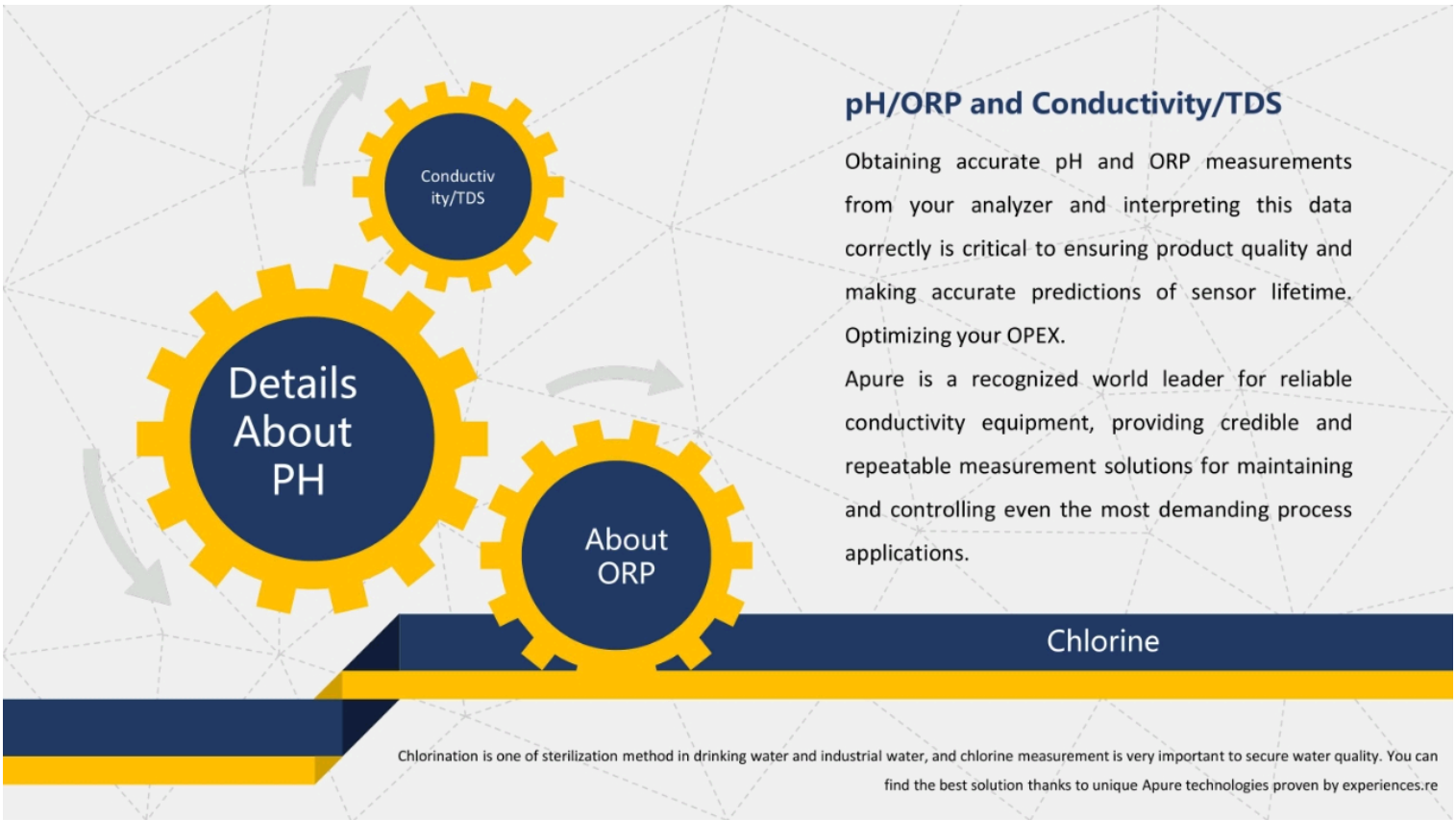
INTRODUCTION

Water quality analyzers are used for monitoring process chemistry including water quality, providing process optimization and control. Water quality parameters are of three types – physical, chemical and biological – and are tested or monitored according to the desired water parameters. Water quality parameters often sampled or monitored include pH, ORP, conductivity, dissolved oxygen, chlorine, salinity, ozone, and corrosion rate. However water monitoring may also include measurement of chlorophyll, blue-green algae, ammonia nitrogen, nitrate, fluoride ions, or laboratory parameters such as BOD, COD and TOC.

Water quality analysis is essential for protecting capital assets including boilers and cooling towers, by preventing corrosion, minimizing maintenance, and maximizing uptime.

1

PHYSICAL ASPECTS


Conductiv
ity/TDS
Details
About
PH
About
ORP
pH/ORP and Conductivity/TDS
Obtaining accurate pH and ORP measurements from your analyzer and interpreting this data correctly is critical to ensuring product quality and making accurate predictions of sensor lifetime. Optimizing your OPEX.
Apure is a recognized world leader for reliable conductivity equipment, providing credible and repeatable measurement solutions for maintaining and controlling even the most demanding process applications.
Chlorine
Chlorination is one of sterilization method in drinking water and industrial water, and chlorine measurement is very important to secure water quality. You can find the best solution thanks to unique Apure technologies proven by experiences.re


TURBIDITY
TASTE
ODOUR
TASTE , TURBIDITY AND ODOUR
Obtaining accurate pH and ORP measurements from your analyzer and interpreting this data correctly is critical to ensuring product quality and making accurate predictions of sensor lifetime. Optimizing your OPEX.
Apure is a recognized world leader for reliable conductivity equipment, providing credible and repeatable measurement solutions for maintaining and controlling even the most demanding process applications.Obtaining
Colour
Colour in water may be caused by the presence of minerals such as iron and manganese or by substances of vegetable origin such as algae and weeds. Colour tests indicate the efficacy of the water treatment system.orination

2

CHEMICAL ASPECTS

Alkalinity

Alkalinity indicates the water's acid-neutralizing capacity. Likely the most common reason to measure the alkalinity of a sample of water is to identify how much soda and lime must be added to the water for water softening purposes. The water softening process is particularly beneficial for mitigating corrosion in boilers.

PH

When measuring the quality of water, pH is one of the first measurements that you should take. The pH of water is measured with a simple pH sensor or test kit, which will tell you how acidic or basic the water is. Acidic water will invariably be comprised of more hydrogen ions. On the other hand, basic water contains more hydroxyl ions.

Acidity

This refers to the measure of how much acids are in a specific solution. The water's acidity is the quantitative capacity that it has to neutralize a base at a certain pH level. Acidity is commonly caused by the presence of mineral acids, hydrolyzed salts, and carbon dioxide. When acids are introduced to water, they can influence many different processes, which include everything from biological activities and chemical reactions to corrosion. The acidity of water is measured with a pH sensor.

Acidity

This refers to the measure of how much acids are in a specific solution. The water's acidity is the quantitative capacity that it has to neutralize a base at a certain pH level. Acidity is commonly caused by the presence of mineral acids, hydrolyzed salts, and carbon dioxide. When acids are introduced to water, they can influence many different processes, which include everything from biological activities and chemical reactions to corrosion. The acidity of water is measured with a pH sensor.

CHEMICAL ASPECTS

Hardness

Hardness occurs when water contains high mineral levels. If left untended, the dissolved minerals in your water could create scale deposits on hot water pipes. If you take a shower with water that has high mineral content, you may find it difficult to produce a lather with the soap you're using. Hardness in water is mainly caused by the presence of magnesium and calcium ions, which can enter water from rock and soil. In most cases, groundwater has more hardness to it than surface water. You can measure water hardness with a colorimeter or test strip.

Chemical Parameters of Water Quality

PH

When measuring the quality of water, pH is one of the first measurements that you should take. The pH of water is measured with a simple pH sensor or test kit, which will tell you how acidic or basic the water is. Acidic water will invariably be comprised of more hydrogen ions. On the other hand, basic water contains more hydroxyl ions.

It's possible for pH levels to range from 0-14. If you receive a reading of 7.0, this means that the water is neutral. Any readings below 7.0 are acidic, while any readings above 7.0 are alkaline. Pure water has a neutral pH. However, rainfall is somewhat more acidic and typically has a 5.6 pH. Water is considered to be safe to drink if it has a pH of 6.5-8.5. The many effects that changing pH levels can have on plants and animals include:

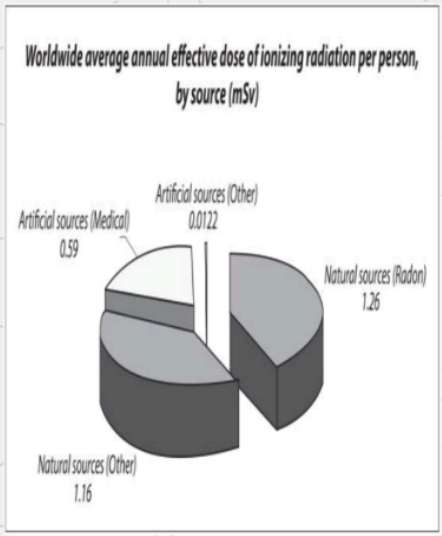
3

RADIOLOGY ASPECTS

/core/mathjax/2.7.9/MathJax.js?config=/core/html/pmc/js/mathjax-config-classic.3.4.js

Table 9.1 Average radiation dose from naturally occurring

Source	Worldwide average annual ef dose (mSv)
External exposure	
Cosmic rays	0.39
Terrestrial radiation (outdoors and indoors)	0.48
Internal exposure	
Inhalation (mainly radon)	1.26



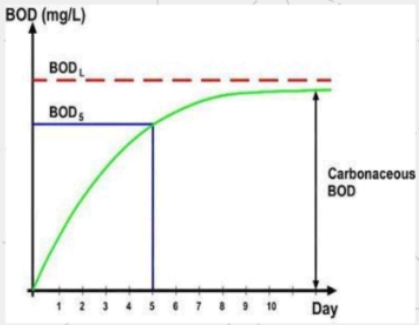
Drinking-water may contain radioactive substances (“radionuclides”) that could present a risk to human health. These risks are normally small compared with the risks from microorganisms and chemicals that may be present in drinking-water. Except in extreme circumstances, the radiation dose resulting from the ingestion of radionuclides in drinking-water is much lower than that received from other sources of radiation. The objective of this chapter is to provide criteria with which to assess the safety of drinking-water with respect to its radionuclide content and to provide guidance on reducing health risks by taking measures to decrease radionuclide concentrations, and therefore radiation doses, in situations where this is considered necessary.

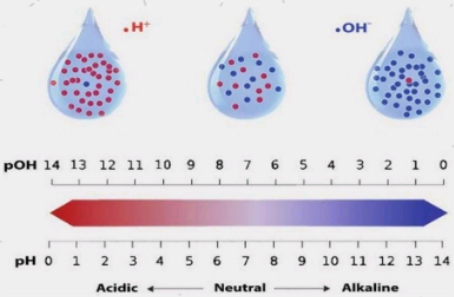
3

Water Quality Parameters

Physical parameters of water qualityChemical parameters of water quality

Since the industrial revolution in the late eighteenth century, the world has discovered new sources of pollution nearly every day. So, air and water can potentially become polluted everywhere. Little is known about changes in pollution rates. The increase in water-related diseases provides a real assessment of the degree of pollution in the environment. This chapter summarizes water quality parameters from an ecological perspective not only for humans but also for other living things. According to its quality, water can be classified into four types. Those four water quality types are discussed through an extensive review of their important common attributes including physical, chemical, and biological parameters. These water quality parameters are reviewed in terms of definition, sources, impacts, effects, and measuring methods.





Toxic inorganic substances

Toxic inorganic substances
A wide variety of inorganic toxic substances may be found in water in very small or trace amounts. Even in trace amounts, they can be a danger to public health [11]. Some toxic substances occur from natural sources but many others occur due to industrial activities and/or improper management of hazardous waste [22]. They can be divided into two groups:

Metallic compounds

This group includes some heavy metals that are toxic, namely, cadmium (Cd), chromium (Cr), lead (Pb), mercury (Hg), silver (Ag), arsenic (As), barium (Ba), thallium (Tl), and selenium (Se) [22, 28]. They have a wide range of dangerous effects that differ from one metal to another. They may be acute fatal poisons such as (As) and (Cr6+) or may produce chronic diseases such as (Cd, Hg, Pb, and Tl) [21, 29, 30, 31, 32]. The heavy metals concentration can be determined by atomic absorption photometers, spectrophotometer, or inductively coupled plasma (ICP) for very low concentration [10].

Nonmetallic compounds

This group includes nitrates (NO_3^-) and cyanides (CN^-), nitrate has been discussed with the nitrogen in the previous section. Regarding cyanide, as Mackenzie stated [11] it causes oxygen deprivation by binding the hemoglobin sites and prevents the red blood cell from carrying the oxygen [11]. This causes a blue skin color syndrome, which is called cyanosis [33]. It also causes chronic effects on the central nervous system and thyroid [33]. Cyanide is normally measured by colorimetric, titrimetric, or electrometric methods [10].

In [1]:
```
import pandas as pd # data processing, CSV file I/O
import numpy as np # linear algebra
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.preprocessing import StandardScaler
```

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

In [3]:
```
data.shape #rows,columns
```

Out[3]:
```
(3276, 10)
```

In [4]:
```
data.head()
```

Out[4]:

	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

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3276 entries, 0 to 3275
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype
```

In [5]:
```
data.info()
```

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

In [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.fillna(data.mean(),inplace=True)
```

In [8]:
```
data
```

Out[8]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	7.080795	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	333.775777	592.885359	15.180013	56.329076	4.500656	0
2	8.099124	224.236259	19909.541732	9.275884	333.775777	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	333.775777	392.449580	19.903225	66.396293	2.798243	1
3273	9.419510	175.762646	33155.578218	7.350233	333.775777	432.044783	11.039070	69.845400	3.298875	1
3274	5.126763	230.603758	11983.869376	6.303357	333.775777	402.883113	11.168946	77.488213	4.708658	1
3275	7.874671	195.102299	17404.177061	7.509306	333.775777	327.459760	16.140368	78.698446	2.309149	1

3276 rows × 10 columns

In [11]:
```
data.describe()
```

Out[11]:

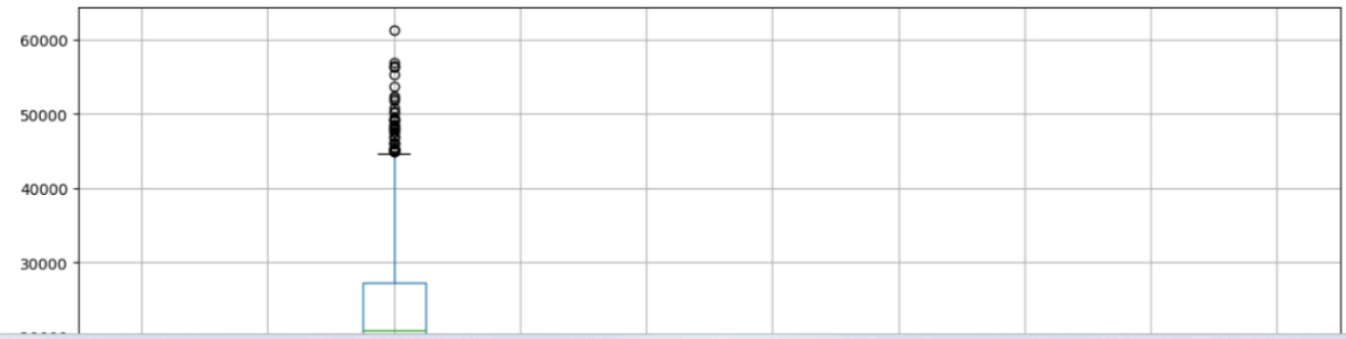
	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
count	3276.000000	3276.000000	3276.000000	3276.000000	3276.000000	3276.000000	3276.000000	3276.000000	3276.000000	3276.000000
mean	7.080795	196.369496	22014.092526	7.122277	333.775777	426.205111	14.284970	66.396293	3.966786	0.390110
std	1.469956	32.879761	8768.570828	1.583085	36.142612	80.824064	3.308162	15.769881	0.780382	0.487849

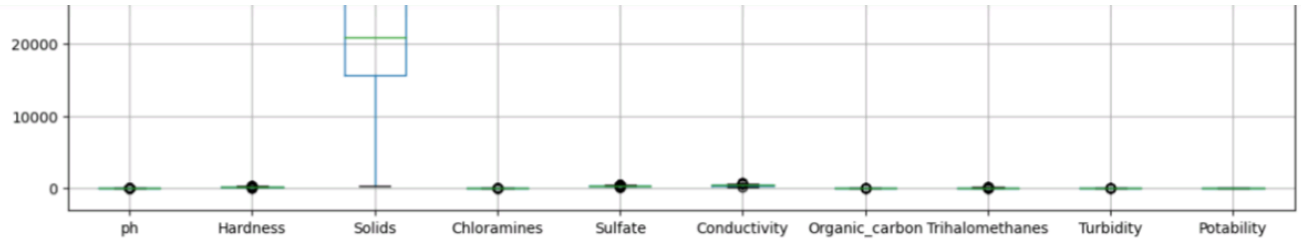
In [11]:
```
data.describe()
```

Out[11]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
count	3276.000000	3276.000000	3276.000000	3276.000000	3276.000000	3276.000000	3276.000000	3276.000000	3276.000000	3276.000000
mean	7.080795	196.369496	22014.092526	7.122277	333.775777	426.205111	14.284970	66.396293	3.966786	0.390110
std	1.469956	32.879761	8768.570828	1.583085	36.142612	80.824064	3.308162	15.769881	0.780382	0.487849
min	0.000000	47.432000	320.942611	0.352000	129.000000	181.483754	2.200000	0.738000	1.450000	0.000000
25%	6.277673	176.850538	15666.690297	6.127421	317.094638	365.734414	12.065801	56.647656	3.439711	0.000000
50%	7.080795	196.967627	20927.833607	7.130299	333.775777	421.884968	14.218338	66.396293	3.955028	0.000000
75%	7.870050	216.667456	27332.762127	8.114887	350.385756	481.792304	16.557652	76.666609	4.500320	1.000000
max	14.000000	323.124000	61227.196008	13.127000	481.030642	753.342620	28.300000	124.000000	6.739000	1.000000

In [10]:
```
data.boxplot(figsize=(15,6))
plt.show()
```





In [12]:
```
data['Solids'].describe()
```

Out[12]:
```
count     3276.000000
mean     22014.092526
std       8768.570828
min        320.942611
25%      15666.690297
50%      20927.833607
75%      27332.762127
max      61227.196008
Name: Solids, dtype: float64
```

In [13]:
```
data
```

Out[13]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	7.080795	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	333.775777	592.885359	15.180013	56.329076	4.500656	0
2	8.099124	224.236259	19909.541732	9.275884	333.775777	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	333.775777	392.449580	19.903225	66.396293	2.798243	1
3273	9.419510	175.762646	33155.578218	7.350233	333.775777	432.044783	11.039070	69.845400	3.298875	1
3274	5.126763	230.603758	11983.869376	6.303357	333.775777	402.883113	11.168946	77.488213	4.708658	1
3275	7.874671	195.102299	17404.177061	7.509306	333.775777	327.459760	16.140368	78.698446	2.309149	1

3276 rows × 10 columns

In [14]:
```
data.shape
```

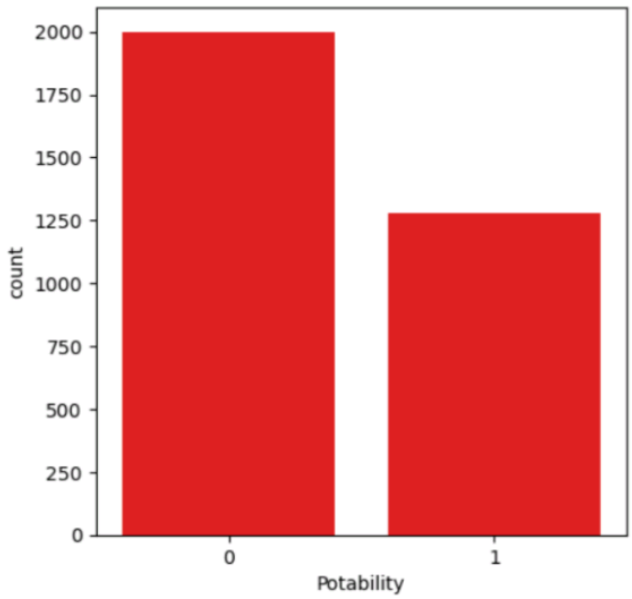
Out[14]:
```
(3276, 10)
```

In [15]:
```
data['Potability'].value_counts()
```

Out[15]:
```
Potability
0    1998
1    1278
Name: count, dtype: int64
```

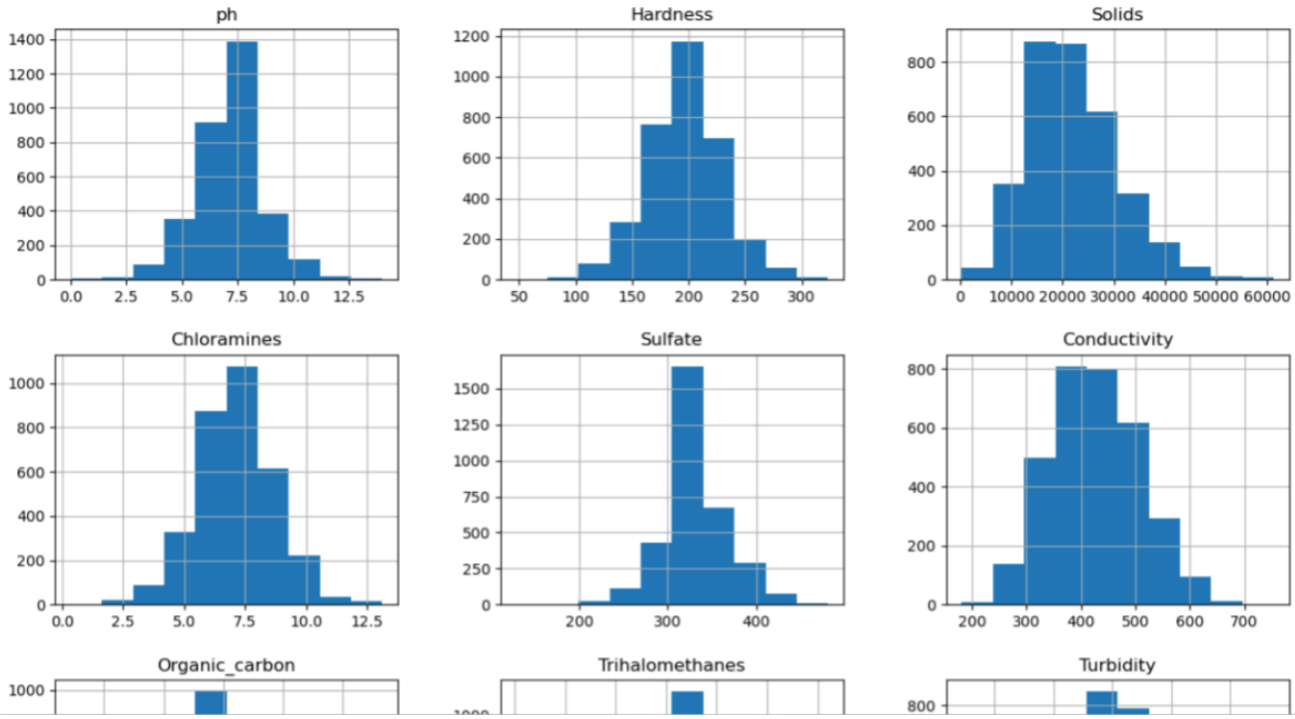
In [16]:
```
plt.figure(figsize=(5,5))
sns.countplot( x=data["Potability"], color="red")
plt.show()
```

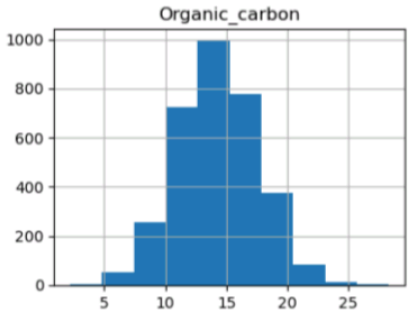
In [16]:
```
plt.figure(figsize=(5,5))
sns.countplot( x=data["Potability"], color="red")
plt.show()
```

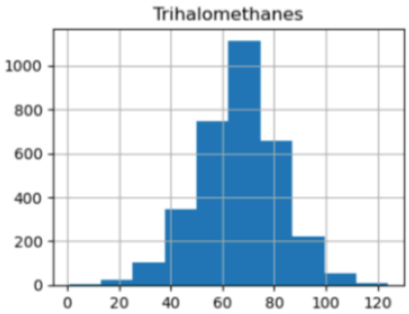


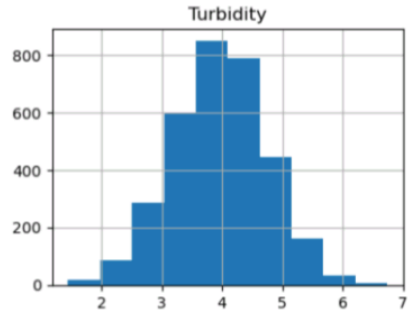
In [17]:
```
data.hist(figsize=(15,15))
plt.show()
```

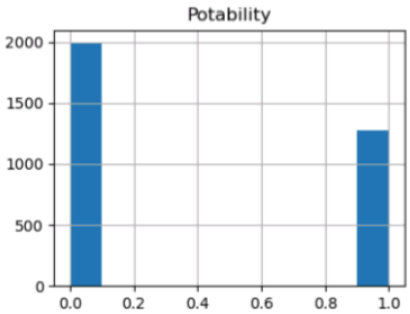
In [17]:
```
data.hist(figsize=(15,15))
plt.show()
```









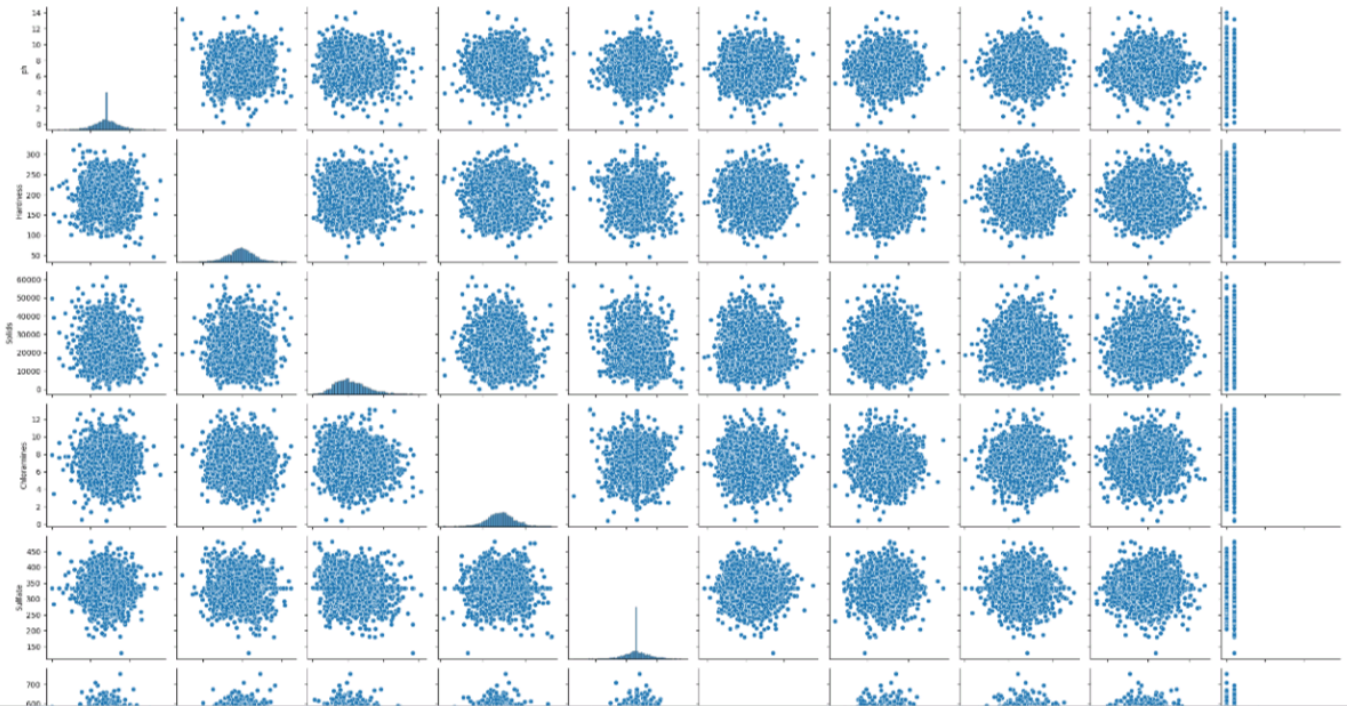


In [22]:
```
sns.pairplot(data)
```

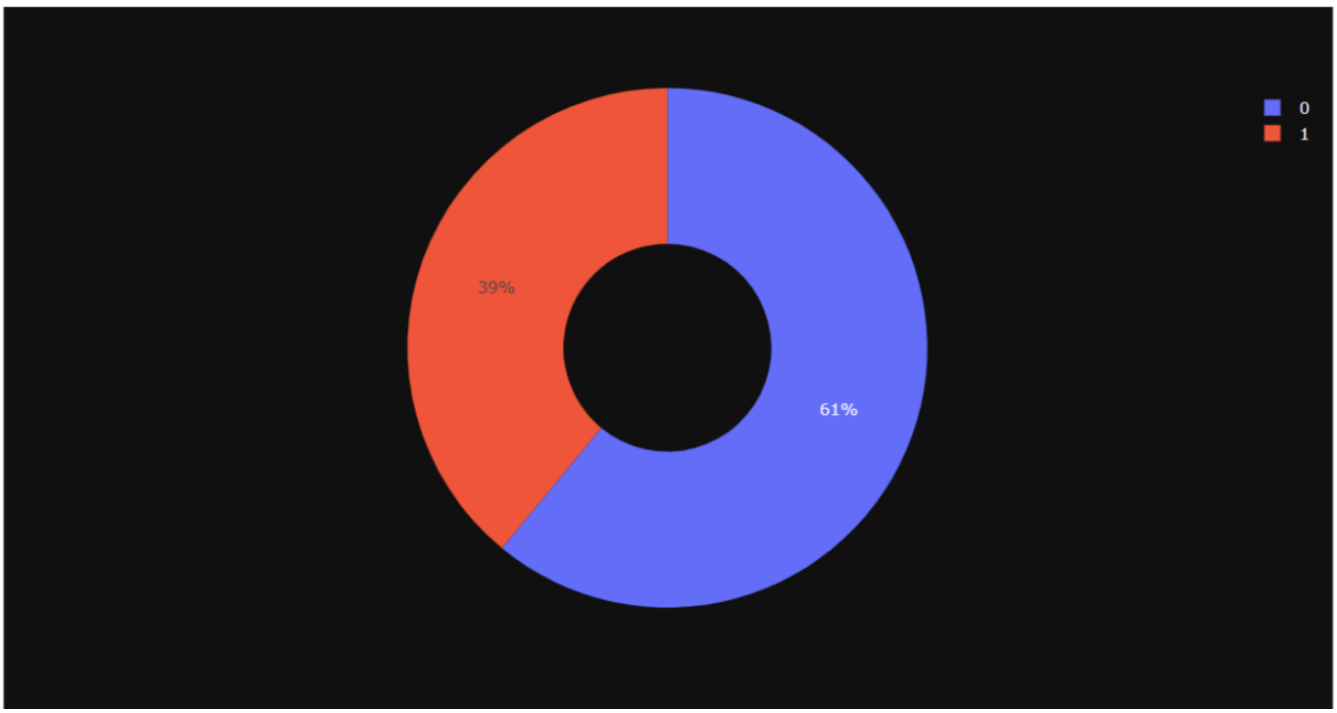
```
D:\New folder\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning:

The figure layout has changed to tight
```

Out[22]: <seaborn.axisgrid.PairGrid at 0x225102df950>

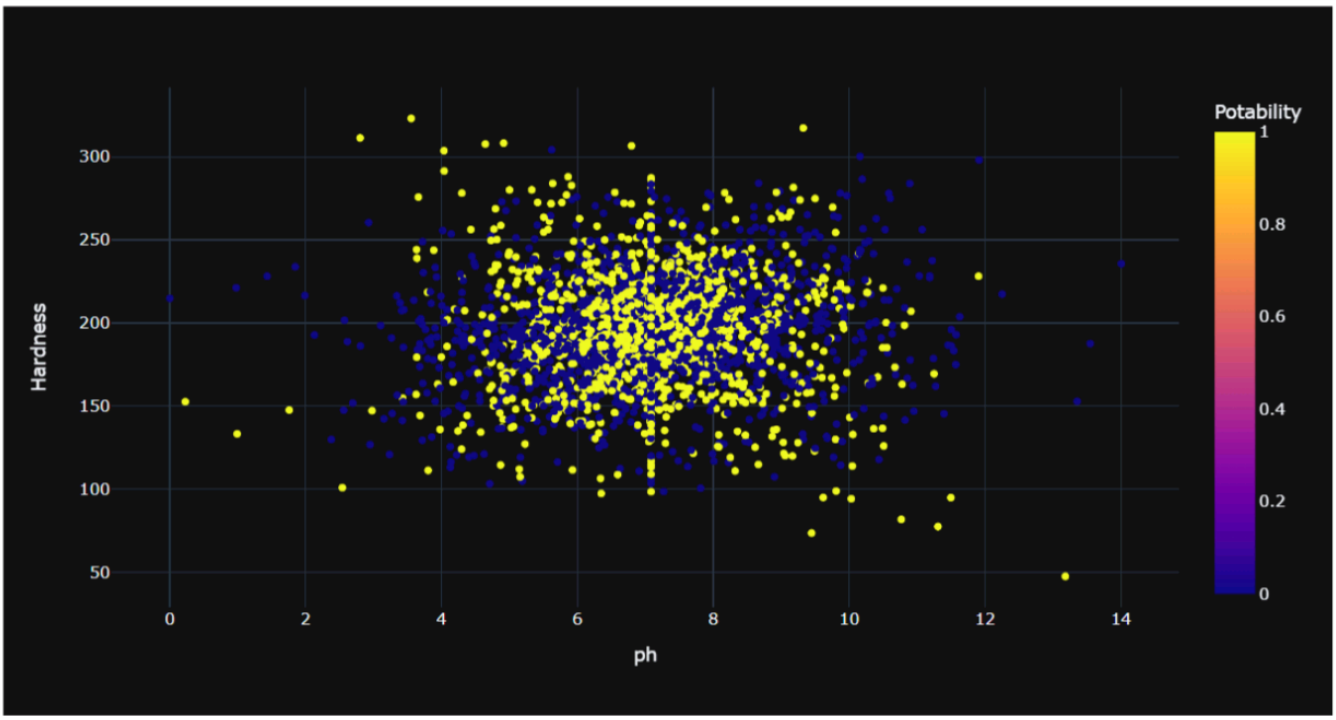
In [19]:
```
sns.scatterplot(x=data['ph'], y=data['Potability'])
plt.show()
```



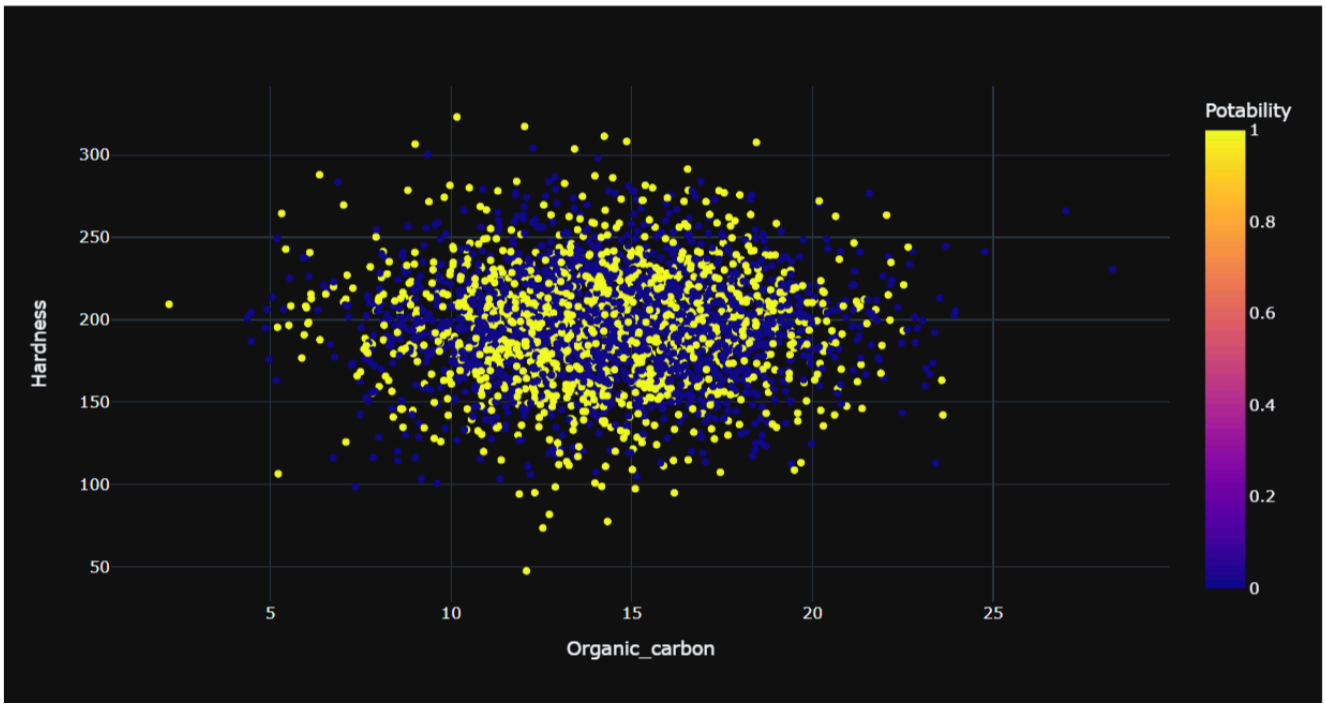
In [20]:
```
fig = px.pie(data,names ="Potability",hole = 0.4,template ="plotly_dark")
fig.show()
```



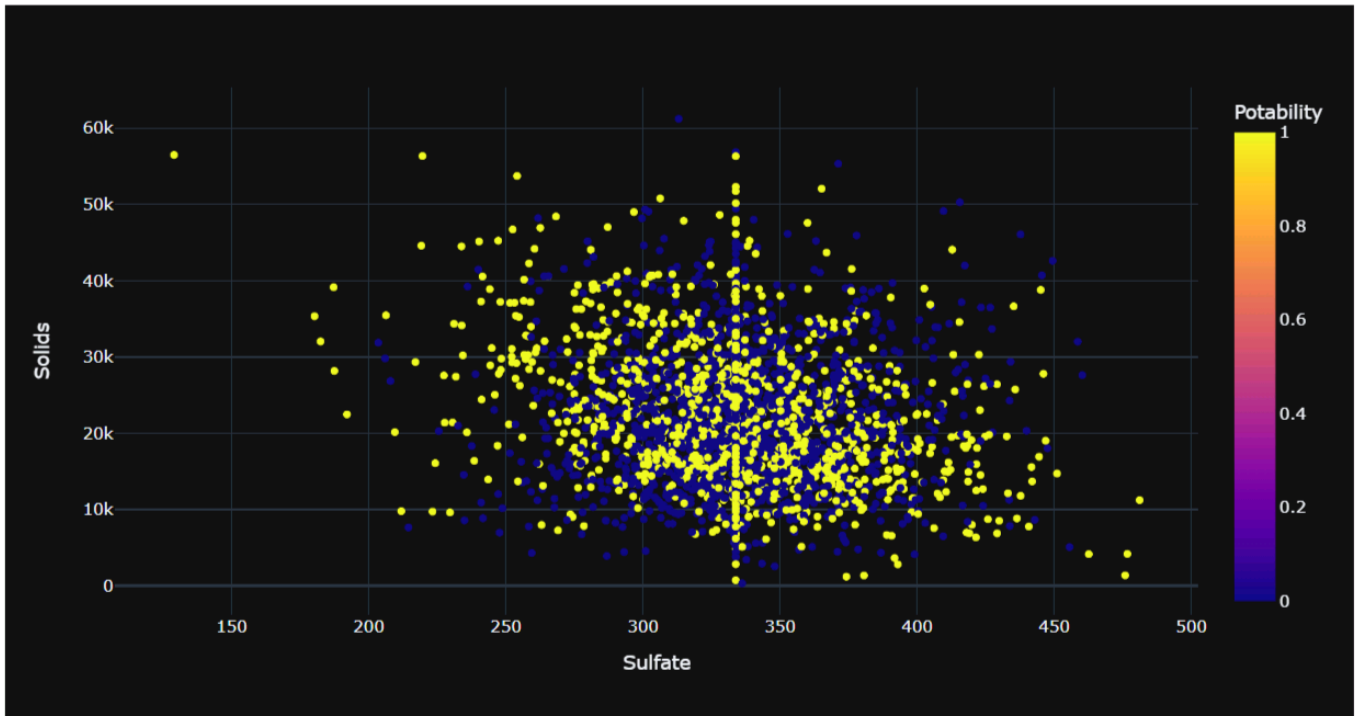
In [21]:
```
fig = px.scatter(data,x ="ph",y="Hardness",color= "Potability",template="plotly_dark")
fig.show()
```



In [23]:
```
fig = px.scatter(data,x ="Organic_carbon",y="Hardness",color= "Potability",template="plotly_dark")
fig.show()
```



In [24]:
```
fig = px.scatter(data,x ="Sulfate",y="Solids",color= "Potability",template="plotly_dark")
fig.show()
```



In [25]:
```
X= data.drop('Potability',axis=1)
```

In [26]:
```
Y= data['Potability']
```

In [27]:
```
from sklearn.model_selection import train_test_split
```

In [28]:
```
X_train , X_test , Y_train , Y_test = train_test_split(X,Y,test_size=0.2, shuffle=True,random_state=101)
```

In [29]:
```
X_train
```

Out[29]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity
748	6.750761	207.254505	23642.992597	7.691012	293.783040	446.696939	6.000391	30.900815	2.777726
2279	7.539742	201.959317	26716.359708	5.637350	333.775777	516.354560	14.985649	83.536821	4.210678
1960	8.128270	231.167537	19954.575554	5.138838	349.067363	386.071149	15.018085	63.340968	4.678742
1491	7.368166	204.041451	8524.874646	9.469763	429.814322	328.565288	11.173155	88.888819	3.684263
2991	6.628256	198.865743	15911.357509	7.517906	342.015924	437.918625	15.005742	38.845958	4.464457
...	...	...	...	...	...	...	...	...	...
599	7.080795	205.638790	39742.970329	4.660528	323.956492	509.546419	11.674850	55.042679	3.916746
1599	8.227083	274.351887	40546.956332	7.130161	241.446917	417.673702	9.809669	79.397105	3.619182
1361	4.906492	173.779159	14786.138901	5.843757	267.561144	620.346840	7.775896	38.794307	3.152345
1547	6.217585	203.707222	15597.640883	7.751461	361.247810	452.922025	14.597145	70.850977	4.150167
863	7.685397	230.335708	7324.701425	7.991366	331.512533	492.850391	14.233952	74.068658	4.179187

2620 rows × 9 columns

In [30]:
```
X_test
```

Out[30]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity
2541	5.735724	158.318741	25363.016594	7.728601	377.543291	568.304671	13.626624	75.952337	4.732954
2605	8.445219	228.522860	28966.569327	6.179855	333.775777	361.705354	14.554220	60.612230	4.400706
330	6.737004	220.100102	24694.744205	8.373660	333.775777	384.308673	6.748092	8.175876	4.063170
515	5.701155	233.515043	41411.601707	5.895464	310.160545	509.767888	22.686837	73.751883	3.403136
400	6.259652	208.379430	37356.746401	8.565487	256.473839	380.240193	5.567693	68.441865	4.213405
...	...	...	...	...	...	...	...	...	...
482	7.705711	178.922858	18476.619166	8.226228	334.889911	518.043369	10.638798	63.157489	3.861956
2970	10.933111	162.424918	18846.634913	7.085261	333.775777	593.725764	14.977233	60.690580	3.894989
50	7.080795	168.388431	27492.307307	7.046225	299.820478	383.795020	16.182066	75.729434	3.048057
839	7.611610	222.252269	25063.683013	8.561124	287.948123	505.265483	18.273757	68.395413	2.873261
374	8.882684	135.523062	4857.253807	5.209779	333.775777	532.336659	20.296274	20.337753	3.827921

656 rows × 9 columns

In [31]:
```
Y_train
```

Out[31]:
```
748     1
2279    0
1960    1
1491    1
2991    0
       ..
599     0
1599    1
1361    0
```

```
2970    0
50      0
839     0
374     1
Name: Potability, Length: 656, dtype: int64
```

In [33]:
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion= 'entropy', min_samples_split= 3,)
```

In [34]:
```
dt.fit(X_train,Y_train)
```

Out[34]:
```
DecisionTreeClassifier(criterion='entropy', min_samples_split=3)
```

In [35]:
```
Y_test
```

Out[35]:
```
2541    0
2605    0
330     1
515     0
400     1
       ..
482     0
2970    0
50      0
839     0
374     1
Name: Potability, Length: 656, dtype: int64
```

In [36]:
```
Y_prediction=dt.predict(X_test)
```

In [37]:
```
from sklearn.metrics import accuracy_score, confusion_matrix
```

In [37]:
```
from sklearn.metrics import accuracy_score, confusion_matrix
```

In [38]:
```
accuracy_score(Y_prediction,Y_test)*100
```

Out[38]:
```
58.84146341463414
```

In [39]:
```
confusion_matrix(Y_prediction,Y_test)
```

Out[39]:
```
array([[262, 130],
       [140, 124]], dtype=int64)
```

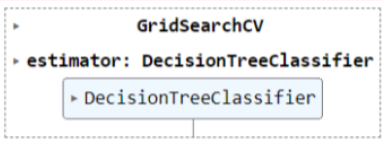
In [40]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RepeatedStratifiedKFold

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 [41]:
```
grid_search_cv_dt.fit(X_train,Y_train)
```

```
D:\New folder\Lib\site-packages\sklearn\model_selection\_validation.py:425: FitFailedWarning:


1000 fits failed out of a total of 1800.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
```

```
  File "D:\New folder\Lib\site-packages\sklearn\base.py", line 637, in _validate_params
    validate_parameter_constraints(
  File "D:\New folder\Lib\site-packages\sklearn\utils\_param_validation.py", line 95, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'splitter' parameter of DecisionTreeClassifier must be a str among
{'best', 'random'}. Got 'random ' instead.


D:\New folder\Lib\site-packages\sklearn\model_selection\_search.py:976: UserWarning:

One or more of the test scores are non-finite: [       nan        nan 0.57900763        nan 0.57801527        nan
 0.57969466        nan 0.58099237        nan 0.58049618        nan
 0.58091603        nan 0.5809542         nan 0.58183206        nan
        nan        nan 0.58408397        nan 0.58435115        nan
 0.5840458         nan 0.58709924        nan 0.58530534        nan
 0.58614504        nan 0.58423664        nan 0.58744275        nan]
```

Out[41]:



In [42]:
```
print(grid_search_cv_dt.best_params_)
```

```
{'criterion': 'entropy', 'min_samples_split': 9, 'splitter': 'best'}
```

In [43]:
```
prediction_grid=grid_search_cv_dt.predict(X_test)
```

In [44]:
```
accuracy_score(Y_test,prediction_grid)*100
```

Out[44]:
```
58.993902439024396
```

THANK YOU

Thanks for your patience to see it