In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df = pd.read_csv(r'E:\csv\water.csv')
df
```

Out[2]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.180013
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.868637
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279
...	...	...	...	...	...	...	...
3271	4.668102	193.681735	47580.991603	7.166639	359.948574	526.424171	13.894419
3272	7.808856	193.553212	17329.802160	8.061362	NaN	392.449580	19.903225
3273	9.419510	175.762646	33155.578218	7.350233	NaN	432.044783	11.039070
3274	5.126763	230.603758	11983.869376	6.303357	NaN	402.883113	11.168946
3275	7.874671	195.102299	17404.177061	7.509306	NaN	327.459760	16.140368

3276 rows × 10 columns

In [3]:
```
df.columns
```

Out[3]:
```
Index(['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'Conductivity',
       'Organic_carbon', 'Trihalomethanes', 'Turbidity', 'Potability'],
      dtype='object')
```

In [45]:

```
df = df.dropna(axis=0, how = 'any')
df
```

Out[45]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279
5	5.584087	188.313324	28748.687739	7.544869	326.678363	280.467916	8.399735
6	10.223862	248.071735	28749.716544	7.513408	393.663396	283.651634	13.789695
7	8.635849	203.361523	13672.091764	4.563009	303.309771	474.607645	12.363817
...	...	...	...	...	...	...	...
3267	8.989900	215.047358	15921.412018	6.297312	312.931022	390.410231	9.899115
3268	6.702547	207.321086	17246.920347	7.708117	304.510230	329.266002	16.217303
3269	11.491011	94.812545	37188.826022	9.263166	258.930600	439.893618	16.172755
3270	6.069616	186.659040	26138.780191	7.747547	345.700257	415.886955	12.067620
3271	4.668102	193.681735	47580.991603	7.166639	359.948574	526.424171	13.894419

2011 rows × 10 columns

In [48]:

```
df = df.drop_duplicates()
df
```

Out[48]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279
5	5.584087	188.313324	28748.687739	7.544869	326.678363	280.467916	8.399735
6	10.223862	248.071735	28749.716544	7.513408	393.663396	283.651634	13.789695
7	8.635849	203.361523	13672.091764	4.563009	303.309771	474.607645	12.363817
...	...	...	...	...	...	...	...
3267	8.989900	215.047358	15921.412018	6.297312	312.931022	390.410231	9.899115
3268	6.702547	207.321086	17246.920347	7.708117	304.510230	329.266002	16.217303
3269	11.491011	94.812545	37188.826022	9.263166	258.930600	439.893618	16.172755
3270	6.069616	186.659040	26138.780191	7.747547	345.700257	415.886955	12.067620
3271	4.668102	193.681735	47580.991603	7.166639	359.948574	526.424171	13.894419

2011 rows × 10 columns

In [32]:
```
df.head()
```

Out[32]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279
5	5.584087	188.313324	28748.687739	7.544869	326.678363	280.467916	8.399735
6	10.223862	248.071735	28749.716544	7.513408	393.663396	283.651634	13.789695
7	8.635849	203.361523	13672.091764	4.563009	303.309771	474.607645	12.363817

In [33]:
```
df.describe()
```

Out[33]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_ca
count	2011.000000	2011.000000	2011.000000	2011.000000	2011.000000	2011.000000	2011.0C
mean	7.085990	195.968072	21917.441374	7.134338	333.224672	426.526409	14.35
std	1.573337	32.635085	8642.239815	1.584820	41.205172	80.712572	3.32
min	0.227499	73.492234	320.942611	1.390871	129.000000	201.619737	2.2C
25%	6.089723	176.744938	15615.665390	6.138895	307.632511	366.680307	12.12
50%	7.027297	197.191839	20933.512750	7.143907	332.232177	423.455906	14.32
75%	8.052969	216.441070	27182.587067	8.109726	359.330555	482.373169	16.68
max	14.000000	317.338124	56488.672413	13.127000	481.030642	753.342620	27.0C

In [34]:
```
X=df.drop(['Potability'],axis=1)
X
```

Out[34]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279
5	5.584087	188.313324	28748.687739	7.544869	326.678363	280.467916	8.399735
6	10.223862	248.071735	28749.716544	7.513408	393.663396	283.651634	13.789695
7	8.635849	203.361523	13672.091764	4.563009	303.309771	474.607645	12.363817
...	...	...	...	...	...	...	...
3267	8.989900	215.047358	15921.412018	6.297312	312.931022	390.410231	9.899115
3268	6.702547	207.321086	17246.920347	7.708117	304.510230	329.266002	16.217303
3269	11.491011	94.812545	37188.826022	9.263166	258.930600	439.893618	16.172755
3270	6.069616	186.659040	26138.780191	7.747547	345.700257	415.886955	12.067620
3271	4.668102	193.681735	47580.991603	7.166639	359.948574	526.424171	13.894419

2011 rows × 9 columns

In [35]:
```
Y=df.drop(['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'Conductivity',
       'Organic_carbon', 'Trihalomethanes', 'Turbidity'],axis=1)
Y
```

Out[35]:

	Potability
3	0
4	0
5	0
6	0
7	0
...	...
3267	1
3268	1
3269	1
3270	1
3271	1

2011 rows × 1 columns

In [36]:
```
X = X.values
X
```

Out[36]:
```
array([[8.31676588e+00, 2.14373394e+02, 2.20184174e+04, ...,
         1.84365245e+01, 1.00341674e+02, 4.62877054e+00],
        [9.09222346e+00, 1.81101509e+02, 1.79789863e+04, ...,
         1.15582794e+01, 3.19979927e+01, 4.07507543e+00],
        [5.58408664e+00, 1.88313324e+02, 2.87486877e+04, ...,
         8.39973464e+00, 5.49178618e+01, 2.55970823e+00],
        ...,
        [1.14910109e+01, 9.48125452e+01, 3.71888260e+04, ...,
         1.61727554e+01, 4.15585007e+01, 4.36926431e+00],
        [6.06961576e+00, 1.86659040e+02, 2.61387802e+04, ...,
         1.20676196e+01, 6.04199211e+01, 3.66971170e+00],
        [4.66810169e+00, 1.93681735e+02, 4.75809916e+04, ...,
         1.38944185e+01, 6.66876948e+01, 4.43582091e+00]])
```

In [37]:
```
Y = Y.values
Y
```

Out[37]:
```
array([[0],
        [0],
        [0],
        ...,
        [1],
        [1],
        [1]], dtype=int64)
```

In [38]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=0)
X_train
```

Out[38]:
```
array([[8.73352533e+00, 2.03396130e+02, 2.45784212e+04, ...,
         1.67019202e+01, 6.87930618e+01, 3.14920180e+00],
        [6.59244251e+00, 2.42480473e+02, 9.38123993e+03, ...,
         1.40272968e+01, 7.09298795e+01, 3.06082735e+00],
        [6.81760838e+00, 2.19337429e+02, 2.75486142e+04, ...,
         1.94867910e+01, 6.85687912e+01, 3.04829206e+00],
        ...,
        [7.11757866e+00, 1.86199680e+02, 3.15289487e+04, ...,
         1.90739955e+01, 7.59030721e+01, 4.33340174e+00],
        [1.02820680e+01, 1.98546363e+02, 8.10829732e+03, ...,
         1.56616923e+01, 2.87706188e+01, 4.57292300e+00],
        [7.26965225e+00, 1.55157520e+02, 3.11613684e+04, ...,
         1.73855150e+01, 7.31150528e+01, 3.78923676e+00]])
```

In [39]:
```
X_test.shape
```

Out[39]:
```
(403, 9)
```

In [40]:
```
Y_train
```

Out[40]:
```
array([[1],
       [0],
       [1],
       ...,
       [0],
       [0],
       [1]], dtype=int64)
```

In [41]:
```
from sklearn.tree import DecisionTreeClassifier
dc=DecisionTreeClassifier()
dc.fit(X_train,Y_train)
```

Out[41]:
```
DecisionTreeClassifier()
```

In [42]:
```
pred=dc.predict(X_test)
pred
```

Out[42]:
```
array([1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0,
       0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1,
       1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0,
       1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0,
       0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       0, 0, 0, 1, 1, 0, 0], dtype=int64)
```

In [43]:
```
from sklearn.metrics import accuracy_score
accuracy=accuracy_score(Y_test,pred)
accuracy
```

Out[43]:
```
0.6153846153846154
```

In [44]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(Y_test,pred)
```

Out[44]:
```
array([[168,  84],
       [ 71,  80]], dtype=int64)
```

In [49]:

```
df1 = pd.DataFrame({'Actual':Y_test.flatten(),'Predict':pred.flatten()})
df1
```

Out[49]:

	Actual	Predict
0	1	1
1	1	1
2	0	0
3	1	1
4	0	1
...	...	...
398	0	0
399	1	1
400	0	1
401	0	0
402	0	0

403 rows × 2 columns

In [50]:

```
from sklearn import metrics
print("mean absolute error:", metrics.mean_absolute_error(Y_test,pred))
```

```
mean absolute error: 0.38461538461538464
```

In [ ]: