Drinking Water Potability

Access to safe drinking water is essential to health, a basic human right, and a component of effective policy for health protection. This is important as a health and development issue at a national, regional, and local level. In some regions, it has been shown that investments in water supply and sanitation can yield a net economic benefit, since the reductions in adverse health effects and health care costs outweigh the costs of undertaking the interventions.

Attributes

- pH value: PH is an important parameter in evaluating the acid-base balance of water
- Hardness: Hardness is mainly caused by calcium and magnesium salts. These salts are dissolved from geologic deposits through which water travels.
- Solids : 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 an unwanted taste and diluted color in the appearance of water. This is the important parameter for the use of water. The water with a high TDS value indicates that water is highly mineralized.
- Chloramines : Chlorine and chloramine are the major disinfectants used in public water systems.
- Sulfate : Sulfates are naturally occurring substances that are found in minerals, soil, and rocks.
- Conductivity : Pure water is not a good conductor of electric current rather’s a good insulator.
- Organic_carbon : Total Organic Carbon (TOC) in source waters comes from decaying natural organic matter (NOM) as well as synthetic sources.
- Trihalmomethanes : THMs are chemicals that may be found in water treated with chlorine.THM levels up to 80 ppm are considered safe in drinking water.
- Turbidity : The turbidity of water depends on the quantity of solid matter present in the suspended state.
- Potability : Indicates if water is safe for human consumption where 1 means potable and 0 means not potable

Let's get Started

Import all necessary libraries

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Read 'water_potability.csv' and store it in a DataFrame

In [4]:
```
df = pd.read_csv('water_potability.csv')
```

View the top 5 rows

In [5]:
```
df.head()
```

Out[5]:

	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

View info of the data

In [6]:
```
df.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
```

View basic statistical information about the dataset

In [7]:
```
df.describe()
```

Out[7]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic
count	2785.000000	3276.000000	3276.000000	3276.000000	2495.000000	3276.000000	327
mean	7.080795	196.369496	22014.092526	7.122277	333.775777	426.205111	1
std	1.594320	32.879761	8768.570828	1.583085	41.416840	80.824064	
min	0.000000	47.432000	320.942611	0.352000	129.000000	181.483754	
25%	6.093092	176.850538	15666.690297	6.127421	307.699498	365.734414	1
50%	7.036752	196.967627	20927.833607	7.130299	333.073546	421.884968	1
75%	8.062066	216.667456	27332.762127	8.114887	359.950170	481.792304	1
max	14.000000	323.124000	61227.196008	13.127000	481.030642	753.342620	2

Check if there are any null values

In [8]:
```
df.isna().sum()
```

Out[8]:
```
ph                 491
Hardness             0
Solids               0
Chloramines          0
Sulfate            781
Conductivity         0
Organic_carbon       0
Trihalomethanes    162
Turbidity            0
Potability           0
dtype: int64
```

Fill all missing values in 'ph', 'Sulfate' and 'Trihalomethanes' with mean value

In [16]:
```
for i in df.columns.tolist():
    df[i].fillna(df[i].mean(), inplace = True)
```

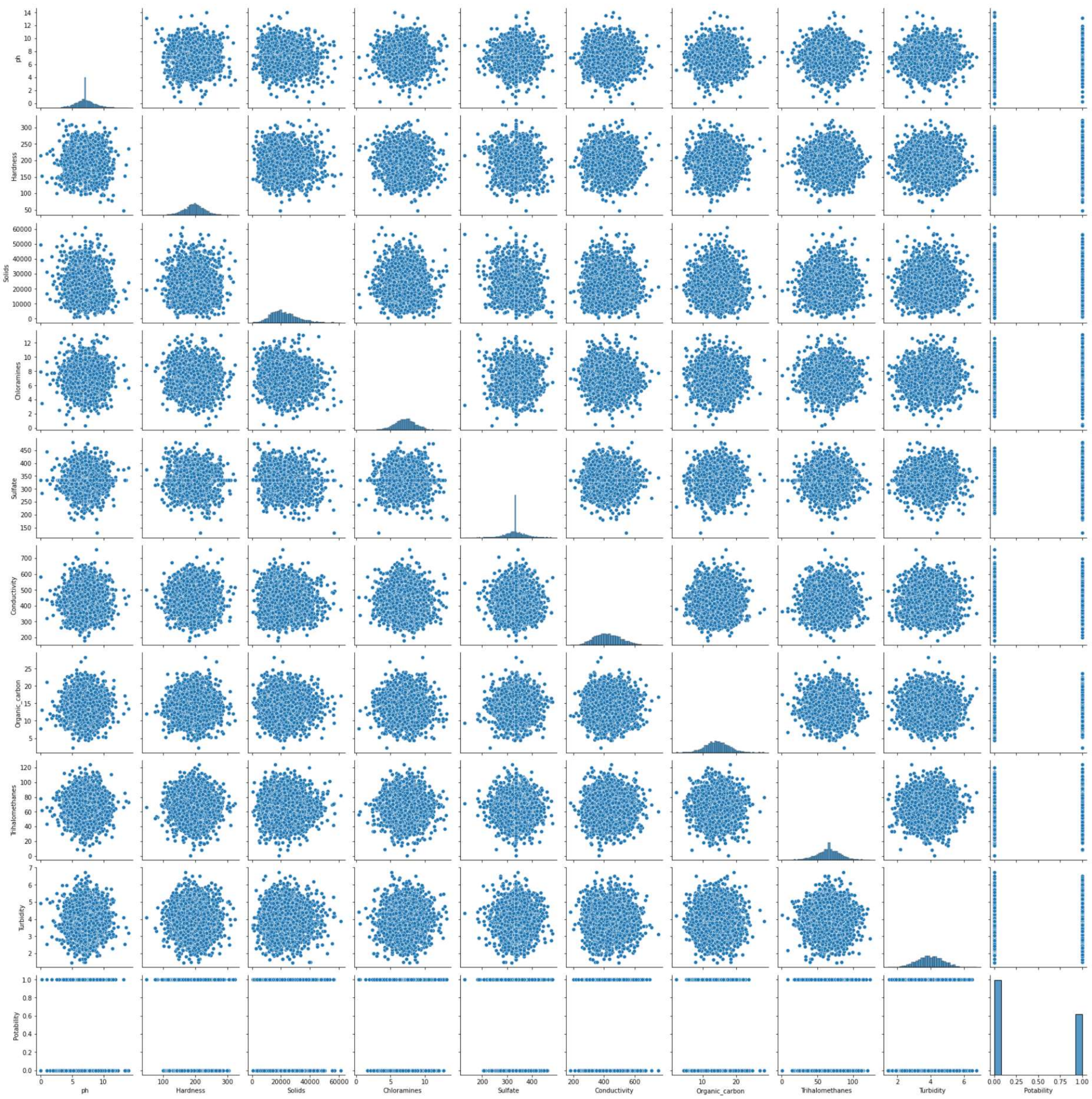
In [18]:
```
df.isna().sum()
```

Out[18]:
```
ph                 0
Hardness           0
Solids             0
Chloramines        0
Sulfate            0
Conductivity       0
Organic_carbon     0
Trihalomethanes    0
Turbidity          0
Potability         0
dtype: int64
```

Visualization

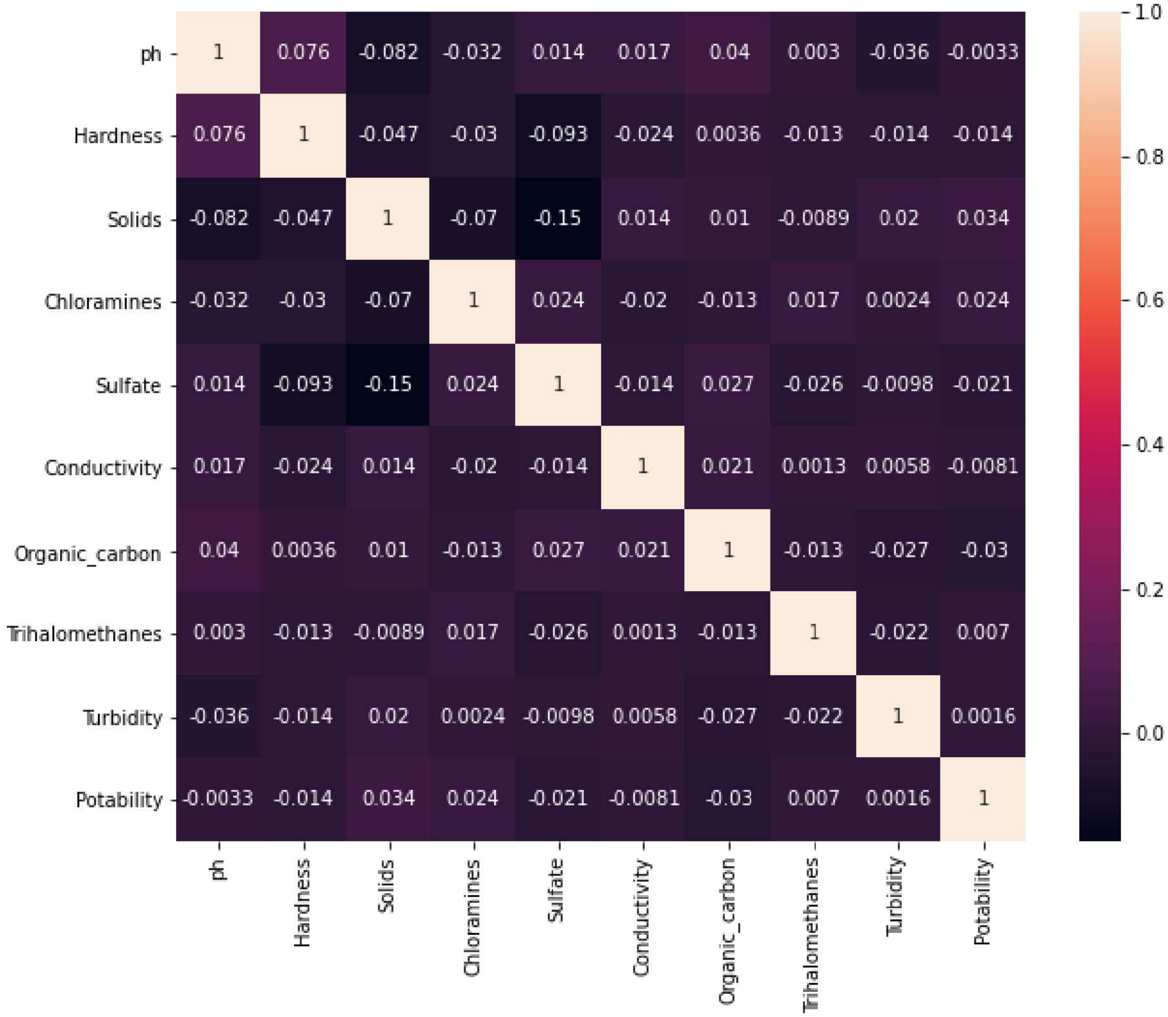
Plot a pairplot of the dataset

In [21]:
```
sns.pairplot(df)
plt.show()
```



Plot a heatmap to visualize the correlation between the features

In [47]:
```
plt.figure(figsize = (10,8))
sns.heatmap(df.corr(), annot = True)
plt.show()
```



Split the data into Input and Target Variables

In [19]:
```
X = df.drop(columns = ['Potability'])
y = df['Potability']
```

Standardise the data with StandardScaler

In [22]:
```
from sklearn.preprocessing import StandardScaler
```

In [23]:
```
scaler = StandardScaler()
```

In [24]:
```
xcolumns = X.columns
```

In [25]:
```
X = scaler.fit_transform(X)
X = pd.DataFrame(X, columns = xcolumns)
```

In [26]:
```
X.head()
```

Out[26]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Tı
0	-1.027333e-14	0.259195	-0.139471	0.112415	0.961357	1.708954	-1.180651	
1	-2.289339e+00	-2.036414	-0.385987	-0.307694	0.000000	2.062575	0.270597	
2	6.928678e-01	0.847665	-0.240047	1.360594	0.000000	-0.094032	0.781117	
3	8.409504e-01	0.547651	0.000493	0.592008	0.639519	-0.778830	1.255134	
4	1.368569e+00	-0.464429	-0.460249	-0.363698	-0.654177	-0.343939	-0.824357	

Split the dataset into Training and Testing set

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 = 0.3, rar
```

Check the shape of X_train and X_test

In [29]:
```
X_train.shape
```

Out[29]: (2293, 9)

In [30]:
```
X_test.shape
```

Out[30]: (983, 9)

Create a SVM model and train it

In [31]:
```
from sklearn.svm import SVC
```

In [32]:
```
model = SVC()
```

In [33]:
```
#Train the model
model.fit(X_train, y_train)
```

Out[33]:
```
SVC()
```

Check the score of our model

In [34]:
```
model.score(X_train,y_train)
```

Out[34]:
```
0.7313563017880506
```

Make predictions using X_test

In [35]:
```
y_pred = model.predict(X_test)
```

Check the accuracy of the prediction

In [36]:
```
from sklearn import metrics
```

In [37]:
```
metrics.accuracy_score(y_test, y_pred)
```

Out[37]:
```
0.6734486266531028
```

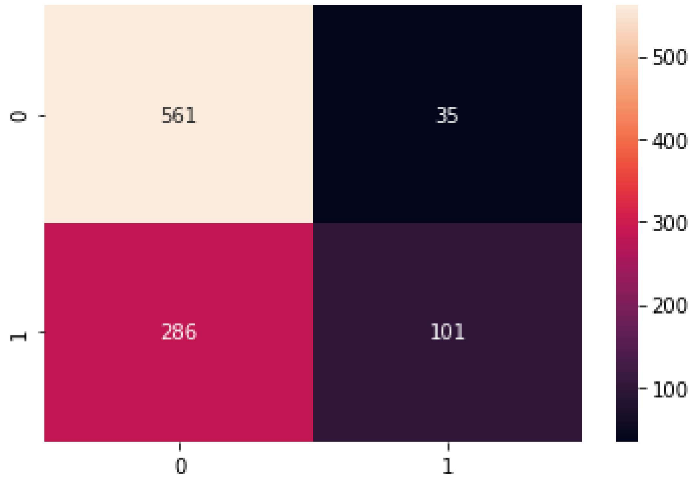
Create confusion matrix

In [39]:
```
metrics.confusion_matrix(y_test,y_pred)
```

Out[39]:
```
array([[561,  35],
       [286, 101]], dtype=int64)
```

Plot the confusion matrix

In [42]:
```
sns.heatmap(metrics.confusion_matrix(y_test,y_pred), annot = True, fmt = 'd')
plt.show()
```



Create a classification report

In [43]:
```
print(metrics.classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.66      0.94      0.78       596
           1       0.74      0.26      0.39       387

    accuracy                           0.67       983
   macro avg       0.70      0.60      0.58       983
weighted avg       0.69      0.67      0.62       983
```