WATER QUALITY ANALYSIS
PHASE -5

Projects Objective

Access to safe drinking water is essential for good health, The purpose of water quality analysis is to evaluate and ensure the safety, cleanliness, and suitability of water for diverse applications, such as human consumption, agriculture, industrial processes, and ecosystem preservation. In this project, we will be analyzing a given water sample by finding its pH, Hardness, Solids, Chloramine, Sulfate, and conductivity and checking whether the water is fit for drinking

Design Thinking

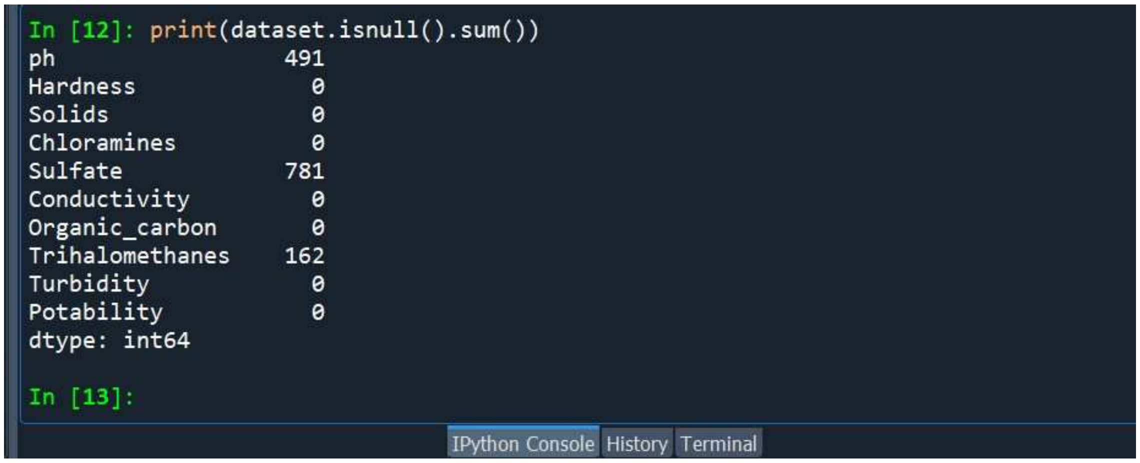
In the phase of design thinking we will understand the motive of the project and make a plan on how to proceed further, in this project we collected data for this project and we understood this project using visualization strategies like correlation analysis, and parameter distribution and we also created a predictive model using a machine learning algorithm.

Data Preprocessing

Data processing transforms data into an effective format by removing null values and outliers before use.

To remove the null Values

```
dataset.columns

dataset.describe

print(dataset.isnull().sum())
```



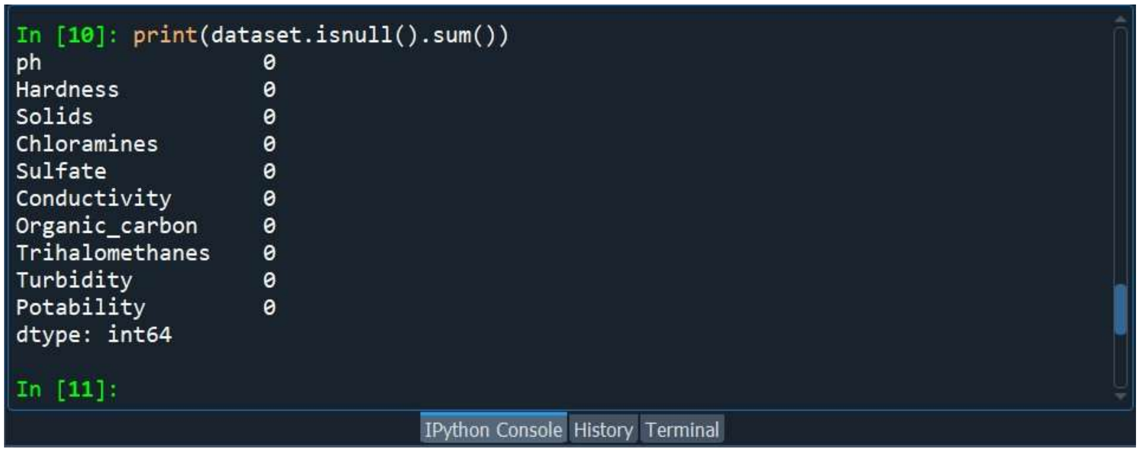
```
In [12]: print(dataset.isnull().sum())
ph                 491
Hardness             0
Solids               0
Chloramines          0
Sulfate            781
Conductivity         0
Organic_carbon       0
Trihalomethanes    162
Turbidity            0
Potability           0
dtype: int64

In [13]:
```

dataset.ph=dataset.ph.fillname("unknown")

dataset.Sulfate=dataset.Sulfate.fillname("unknown")

dataset.Trihalomethanes=dataset.Trihalomethanes.fillname("unknown")

print(dataset.isnull().sum())

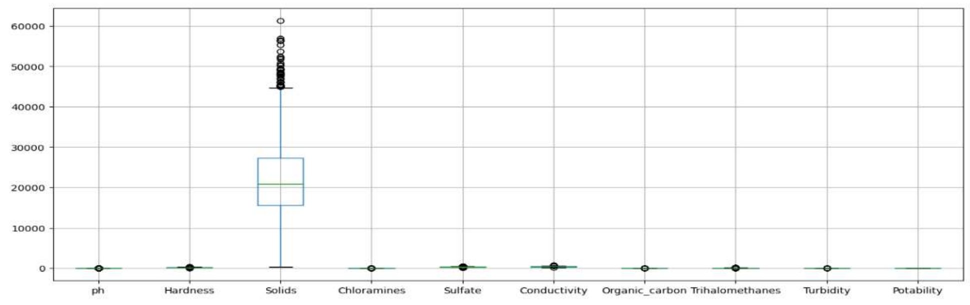


```
In [10]: print(dataset.isnull().sum())
ph                 0
Hardness           0
Solids             0
Chloramines        0
Sulfate            0
Conductivity       0
Organic_carbon     0
Trihalomethanes    0
Turbidity          0
Potability         0
dtype: int64

In [11]:
```

Now let's identify the outliers

data.boxplot(figsize=(15,6))

plt.show()

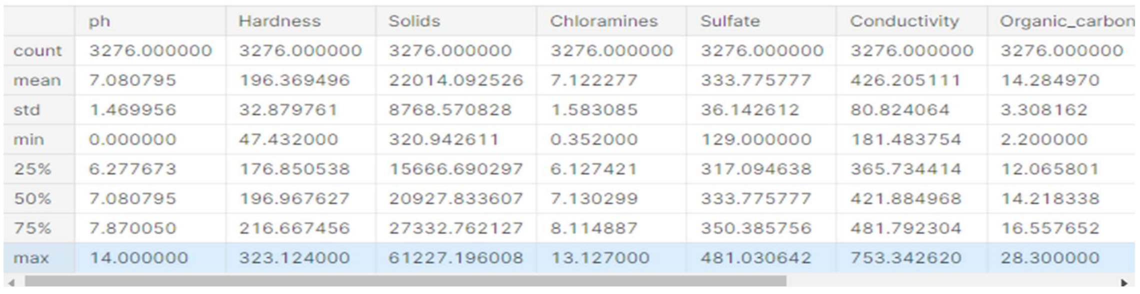


Exploratory Data Analysis(EDA)

Exploratory Data Analysis (EDA) refers to studying and exploring record sets to apprehend their predominant traits, discover patterns, locate outliers, and identify relationships between variables. EDA is normally carried out as a preliminary step before undertaking extra formal statistical analyses or modeling.

```
data.describe()
```

o/p:

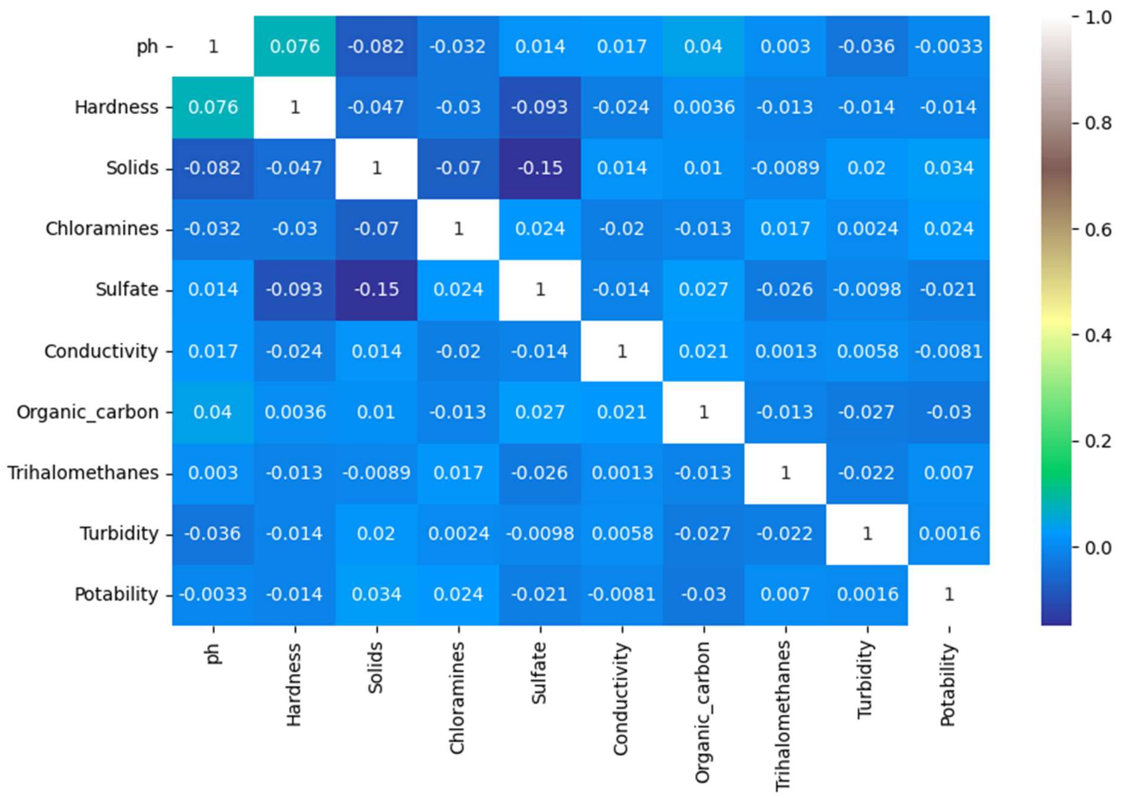


	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon
count	3276.000000	3276.000000	3276.000000	3276.000000	3276.000000	3276.000000	3276.000000
mean	7.080795	196.369496	22014.092526	7.122277	333.775777	426.205111	14.284970
std	1.469956	32.879761	8768.570828	1.583085	36.142612	80.824064	3.308162
min	0.000000	47.432000	320.942611	0.352000	129.000000	181.483754	2.200000
25%	6.277673	176.850538	15666.690297	6.127421	317.094638	365.734414	12.065801
50%	7.080795	196.967627	20927.833607	7.130299	333.775777	421.884968	14.218338
75%	7.870050	216.667456	27332.762127	8.114887	350.385756	481.792304	16.557652
max	14.000000	323.124000	61227.196008	13.127000	481.030642	753.342620	28.300000

Checking if we need to do Dimensionality Reduction

we are trying to reduce the dimensions to which are correlating for that we are looking for the similarity of the features with this chart because fewer features is making easy to predict but we have so small similarities of the features and we can't use the remove feature

```
 sns.heatmap(data.corr(),annot= True,cmap='terrain')

fig= plt.gcf()

fig.set_size_inches(10,6)

plt.show()
```



Data Visualization

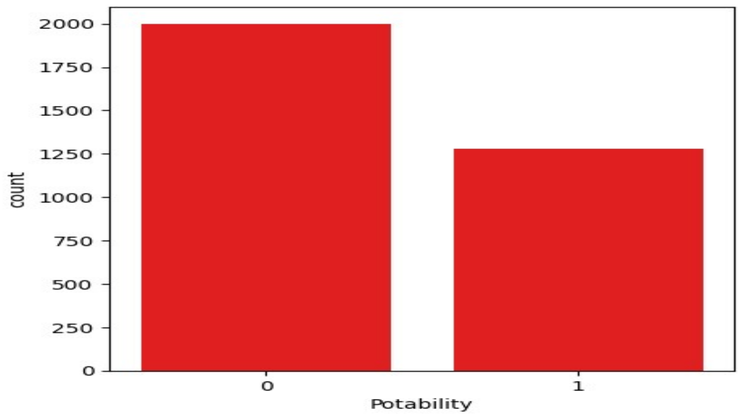
In data visualization, we created histograms and scatter plots to make the given data set more understandable.

```
data['Potability'].value_counts()
```

```
o/p Potability

   0    1998

   1    1278
```

```
plt.figure(figsize=(5,5))

sns.countplot( x=data["Potability"], color="red")

plt.show()
```

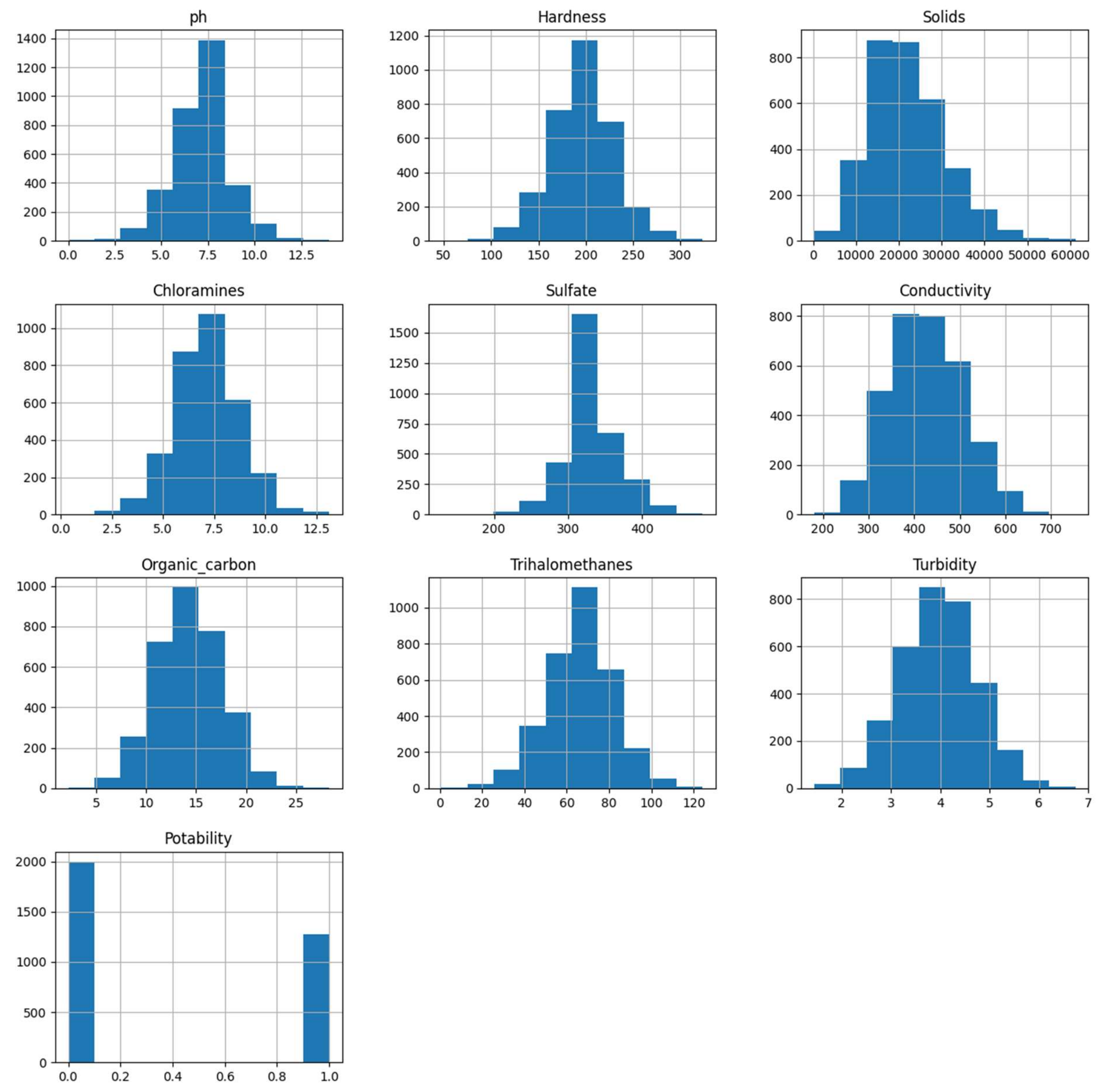
o/p



Histogram

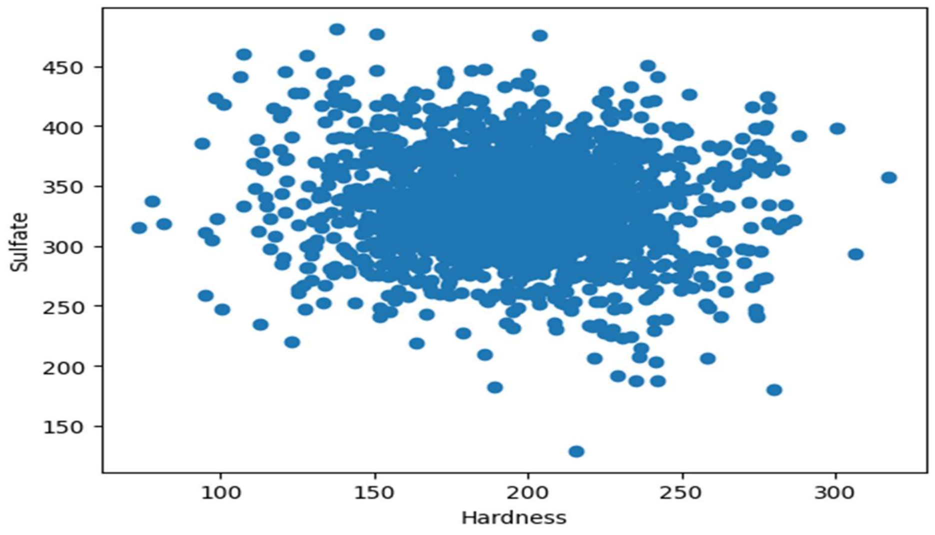
We also plotted hisogram for each column to make them easily understandable

```
 data.hist(figsize=(15,15))

 plt.show()
```



Scatter Plot

```
gp = plt.scatter(ks['Hardness'],ks['Sulfate'])

plt.xlabel('Hardness')

plt.ylabel('Sulfate')

plt.show(gp)
```



sns.scatterplot(x=data['ph'], y=data['Potability'])

plt.show()

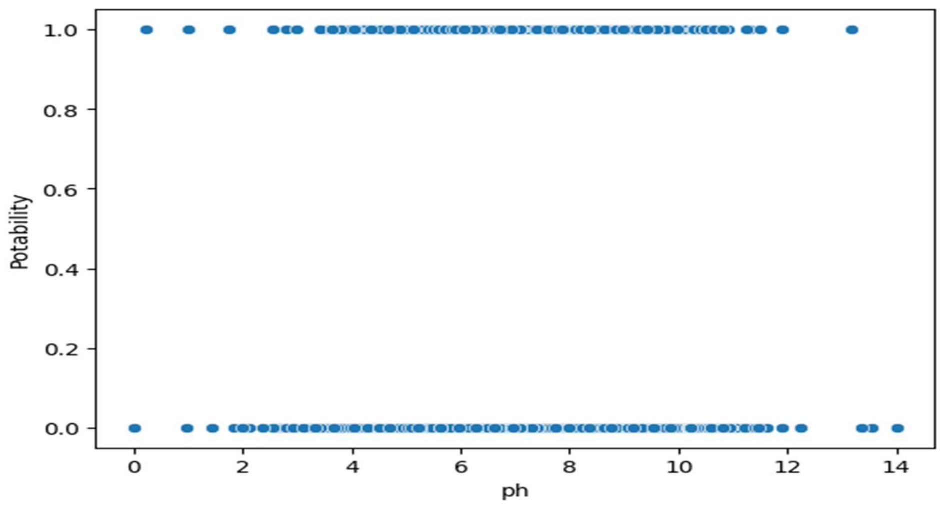
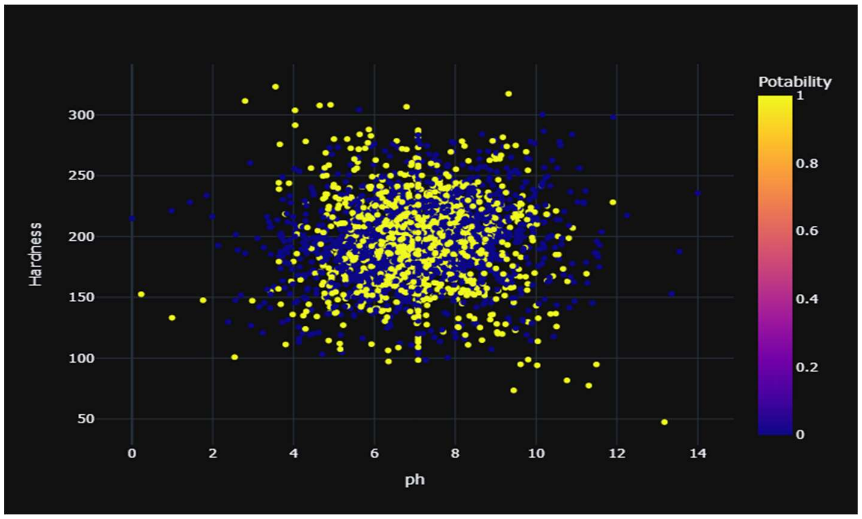
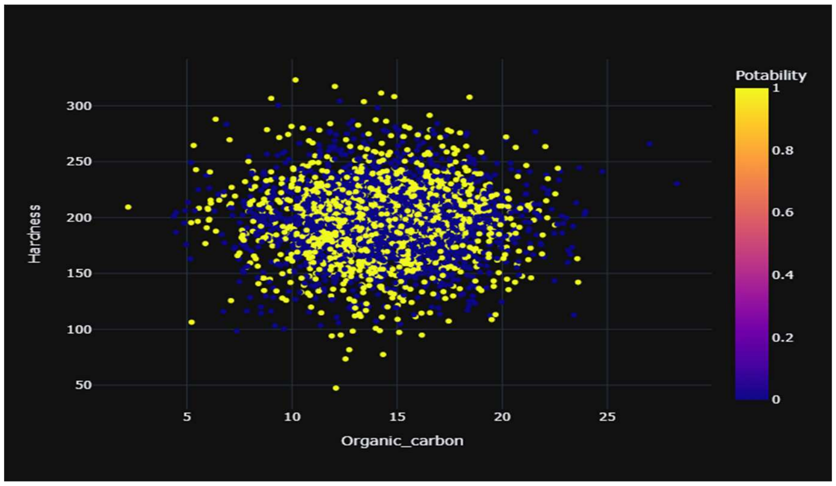


fig = px.scatter(data,x ="ph",y="Hardness",color= "Potability",template="plotly_dark")

fig.show()



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



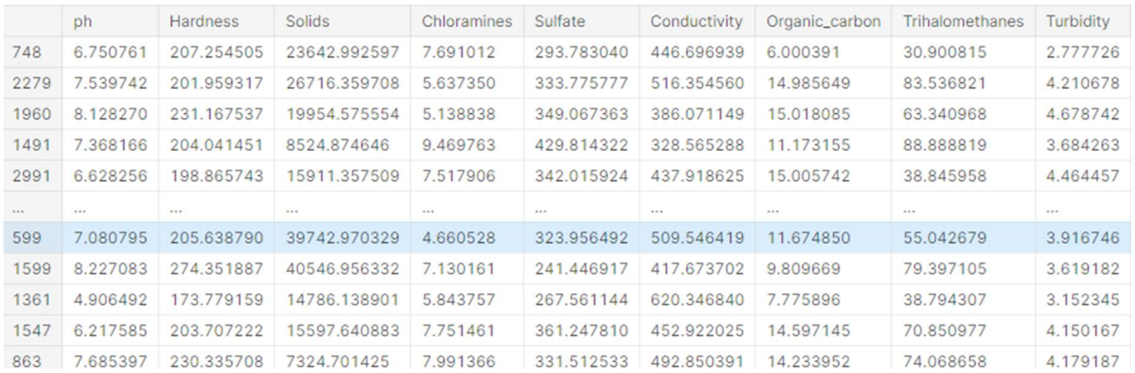
Predictive Model Training

Predictive modeling is the process of using known results to create a statistical model that can be used for predictive analysis

```
Y= data['Potability'] # target variable is potability
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, shuffle=True,random_state=101)
```

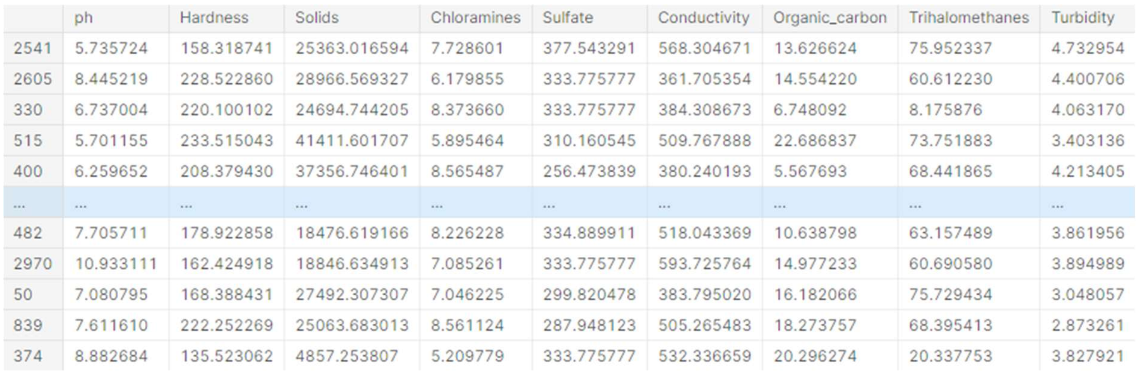
we splited the data to make a prediction on that train data

X_train



	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

X_test



	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

Y_train

o/p

```
748     1

2279    0

1960    1

1491    1

2991    0

      ..

599     0

1599    1

1361    0

1547    1

863     0
```

Y_test

```
o/p

2541    0
2605    0
330     1
515     0
400     1
       ..
482     0
2970    0
50      0
839     0
374     1
```

Random Forest

The random forest algorithm is made up of a collection of decision trees, and each tree in the ensemble is comprised of a data sample drawn from a training set with replacement, called the bootstrap sample

```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(criterion= 'entropy', min_samples_split= 3,)

dt.fit(X_train,Y_train)

Y_test

o/p

2541    0
2605    0
330     1
515     0
400     1
       ..
482     0
2970    0
```

```
50      0
839     0
374     1
```

```
Y_prediction=dt.predict(X_test)

from sklearn.metrics import accuracy_score, confusion_matrix

accuracy_score(Y_prediction,Y_test)*100
```

o/p:59.756097560975604

```
confusion_matrix(Y_prediction,Y_test)
```

```
o/p  array([[262, 124],
            [140, 130]])
```

```
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)

grid_search_cv_dt.fit(X_train,Y_train)
```

o/p:

```
One or more of the test scores are non-finite:
 [       nan        nan 0.57664122        nan 0.5798855         nan
 0.58038168        nan 0.57916031        nan 0.58041985        nan
 0.58118321        nan 0.58053435        nan 0.58049618        nan
        nan        nan 0.58519084        nan 0.58431298        nan
 0.58473282   nan 0.58461832        nan 0.58278626        nan
 0.58534351    nan 0.58305344        nan 0.58793893        nan]
```

```
print(grid_search_cv_dt.best_params_)

prediction_grid=grid_search_cv_dt.predict(X_test)

accuracy_score(Y_test,prediction_grid)*100
```

o/p

```
59.756097560975604
```

How the insights from the analysis can help assess water quality and determine potability?

Water quality analysis provides crucial insights that are fundamental for ensuring safe drinking water. By understanding the composition and contaminants present, we can compare the data with local, national, or international standards (e.g., EPA standards in the U.S.). If the levels of contaminants exceed the permissible limits, the water is considered non-potable.

Conclusion

In this project, we explained the objective of water quality analysis, and what we understood about it and to proceed with the project further we made a design plan. To make the data set more effective, we pre-processed the data by removing all the null values and finding the outliers.
We have also used data visualization and predictive models to enhance our understanding of water portability.
By leveraging visualization libraries such as Matplotlib and Seaborn, we have created informative histograms, scatter plots, and correlation matrices, which have illuminated the relationships between various water quality parameters. These visualizations have provided valuable insights into the data, helping us identify patterns and trends that might not be immediately apparent from raw numbers alone.

Moreover, the development of a predictive model, utilizing techniques like Logistic Regression and Random Forest, has taken our analysis a step further. We have not only assessed the current water quality but also sought to predict water potability based on the collected data. This predictive model has the potential to be a valuable tool for assessing the safety of water sources in real-time and making informed decisions about water treatment and distribution.