WATER QUALITY ANALYSIS
PHASE 4: DEVELOPMENT PART 2

INTRODUCTION

Water quality analysis is a crucial process that plays a fundamental role in ensuring the safety and sustainability of our planet's most precious resource - water. It involves the systematic examination of various physical, chemical, and biological characteristics of water to determine its suitability for various purposes, including drinking, recreation, agriculture, and industrial use. As the world's population continues to grow and environmental pressures increase, understanding, and monitoring water quality has become more important than ever.

Water quality analysis enables us to assess the presence of contaminants, pollutants, and pathogens in water bodies, which can have a significant impact on both human health and the ecosystem. It helps us identify potential threats to water sources and informs decisions regarding water treatment and conservation strategies. By providing valuable data, water quality analysis empowers policymakers, scientists, and environmentalists to make informed choices and develop effective solutions to address water-related challenges.

This introduction sets the stage for a deeper exploration of the methods, tools, and significance of water quality analysis in safeguarding our environment, health, and the sustainability of one of our most vital resources.

Dataset Used:

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

1	ph	Hardness	Solids	Chloramir	Sulfate	Conductiv	Organic_c	Trihalome	Turbidity	Potability
2		204.8905	20791.32	7.300212	368.5164	564.3087	10.37978	86.99097	2.963135	0
3	3.71608	129.4229	18630.06	6.635246		592.8854	15.18001	56.32908	4.500656	0
4	8.099124	224.2363	19909.54	9.275884		418.6062	16.86864	66.42009	3.055934	0
5	8.316766	214.3734	22018.42	8.059332	356.8861	363.2665	18.43652	100.3417	4.628771	0
6	9.092223	181.1015	17978.99	6.5466	310.1357	398.4108	11.55828	31.99799	4.075075	0
7	5.584087	188.3133	28748.69	7.544869	326.6784	280.4679	8.399735	54.91786	2.559708	0
8	10.22386	248.0717	28749.72	7.513408	393.6634	283.6516	13.7897	84.60356	2.672989	0
9	8.635849	203.3615	13672.09	4.563009	303.3098	474.6076	12.36382	62.79831	4.401425	0
10		118.9886	14285.58	7.804174	268.6469	389.3756	12.70605	53.92885	3.595017	0
11	11.18028	227.2315	25484.51	9.0772	404.0416	563.8855	17.92781	71.9766	4.370562	0
12	7.36064	165.5208	32452.61	7.550701	326.6244	425.3834	15.58681	78.74002	3.662292	0
13	7.974522	218.6933	18767.66	8.110385		364.0982	14.52575	76.48591	4.011718	0
14	7.119824	156.705	18730.81	3.606036	282.3441	347.715	15.92954	79.50078	3.445756	0
15		150.1749	27331.36	6.838223	299.4158	379.7618	19.37081	76.51	4.413974	0
16	7.496232	205.345	28388	5.072558		444.6454	13.22831	70.30021	4.777382	0
17	6.347272	186.7329	41065.23	9.629596	364.4877	516.7433	11.53978	75.07162	4.376348	0
18	7.051786	211.0494	30980.6	10.0948		315.1413	20.39702	56.6516	4.268429	0
19	9.18156	273.8138	24041.33	6.90499	398.3505	477.9746	13.38734	71.45736	4.503661	0
20	8.975464	279.3572	19460.4	6.204321		431.444	12.88876	63.82124	2.436086	0
21	7.37105	214.4966	25630.32	4.432669	335.7544	469.9146	12.50916	62.79728	2.560299	0
22		227.435	22305.57	10.33392		554.8201	16.33169	45.38282	4.133423	0
23	6.660212	168.2837	30944.36	5.858769	310.9309	523.6713	17.88424	77.04232	3.749701	0
24		215.9779	17107.22	5.60706	326.944	436.2562	14.18906	59.85548	5.459251	0
25	3.902476	196.9032	21167.5	6.996312		444.4789	16.60903	90.18168	4.528523	0
26	5.400302	140.7391	17266.59	10.05685	328.3582	472.8741	11.25638	56.93191	4.824786	0
27	6.514415	198.7674	21218.7	8.670937	323.5963	413.2905	14.9	79.84784	5.200885	0
28	3.445062	207.9263	33424.77	8.782147	384.007	441.7859	13.8059	30.2846	4.184397	0
29		145.7682	13224.94	7.906445	304.002	298.9907	12.72952	49.53685	4.004871	0
30		266.421	26362.97	7.700063	395.3895	364.4801	10.34895	53.00838	3.991564	0
31		148.1531	15193.41	9.046833	307.0118	563.8047	16.56866	52.67619	6.038185	0
32	7.181449	209.6256	15196.23	5.994679	338.3364	342.1113	7.922598	71.53795	5.08886	0
33	9.82549	190.7566	19677.89	6.757541		452.8362	16.89904	47.08197	2.857472	0
34	10.43329	117.7912	22326.89	8.161505	307.7075	412.9868	12.89071	65.73348	5.057311	0
35	7.414148	235.0445	32555.85	6.845952	387.1753	411.9834	10.24482	44.4893	3.160624	0
36		232.2805	14787.21	5.474915		383.9817	12.16694	86.08073	5.029167	0
37	5.115817	191.9527	19620.55	6.060713	323.8364	441.7484	10.96649	49.23823	3.902089	0

In this phase we have to building the analysis by creating visualizations and building a predictive model.

- Use visualization libraries (e.g., Matplotlib, Seaborn) to create histograms, scatter plots, and correlation matrices.
- Build a predictive model (e.g., Logistic Regression, Random Forest) to determine water potability based on water quality parameters.

Histogram:

A **Histogram** is a graphical representation of the distribution of data. The histogram is represented by a set of rectangles, adjacent to each other, where each bar represent a kind of data.

Scatter Plot:

Scatter plots are the graphs that present the relationship between two variables in a data-set. It represents data points on a two-dimensional plane or on a Cartesian system

Correlation Matrices :

A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. A correlation matrix is used to summarize data, as an input into a more advanced analysis, and as a diagnostic for advanced analyses.

Logistic Regression:

Logistic regression is a supervised machine learning algorithm mainly used for classification tasks where the goal is to predict the probability that an instance of belonging to a given class

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

Import Libraries

```
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
data=pd.read_csv('/kaggle/input/water-quality-and-potability/water_potability.csv')
data.head()
```

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidit
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783	86.990970	2.9631
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.180013	56.329076	4.5006
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.868637	66.420093	3.0559
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524	100.341674	4.6287
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279	31.997993	4.0750

Data Cleaning:

we have null values so for the right solution we have to clean this values

```
data.info()
```

o/p

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

We have 3276 rows and there is a 781 row missing value surface data this is highest we already have small data and there is almost 1000 rows is null value . we remove the null values permanently

```
data.isnull().sum()
```

o/p

```
ph               491
Hardness           0
Solids             0
Chloramines        0
Sulfate          781
Conductivity       0
Organic_carbon     0
Trihalomethanes  162
Turbidity          0
Potability         0
```

dtype: int64

so where ever ph=null it will get replaced with 7.080795, sulfate=null get replaced with 333.775777, Trihalomethanes=null get replaced with 66.396293

```
data.fillna(data.mean(),inplace=True)
Data # to print data without null
```

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomet
	7.080795	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783	86.99097
	3.716080	129.422921	18630.057858	6.635246	333.775777	592.885359	15.180013	56.32907
	8.099124	224.236259	19909.541732	9.275884	333.775777	418.606213	16.868637	66.42009
	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524	100.3416
	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279	31.99799
	...	...	...	...	...	...	...	...
271	4.668102	193.681735	47580.991603	7.166639	359.948574	526.424171	13.894419	66.68769
272	7.808856	193.553212	17329.802160	8.061362	333.775777	392.449580	19.903225	66.39629
273	9.419510	175.762646	33155.578218	7.350233	333.775777	432.044783	11.039070	69.84540
274	5.126763	230.603758	11983.869376	6.303357	333.775777	402.883113	11.168946	77.48821
275	7.874671	195.102299	17404.177061	7.509306	333.775777	327.459760	16.140368	78.69844

Exploratoy Data Analysis

Exploratory Data Analysis (EDA) refers to the method of 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()
```

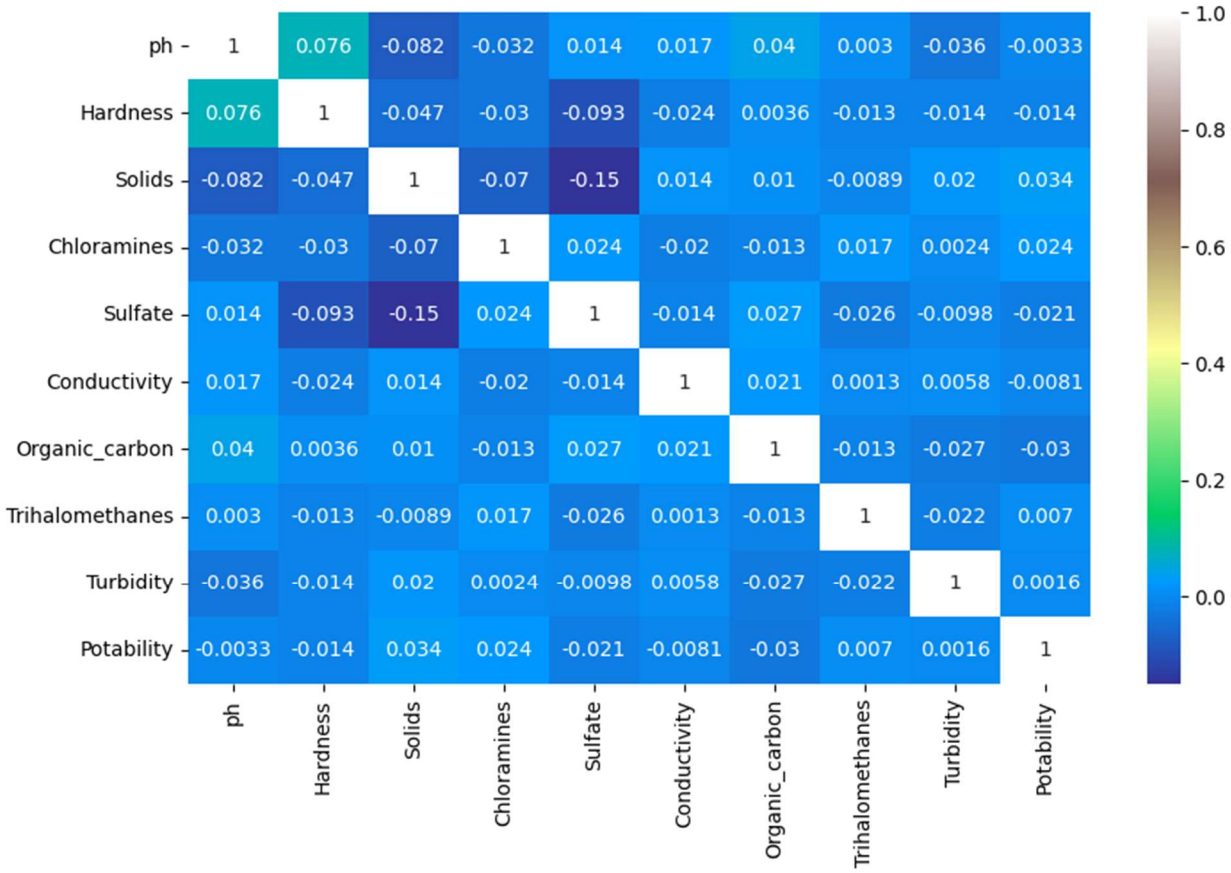
o/p:

	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 Dimensonility Reduction

we are trying the reduce the dimension to which are correlating for that we are looking for the similarity of the features with this chart because less feature is making easy to the predict but we have so small similarity of the features and we cant use the remove feature

```
sns.heatmap(data.corr(),annot= True,cmap='terrain')
fig= plt.gcf()
fig.set_size_inches(10,6)
plt.show()
```

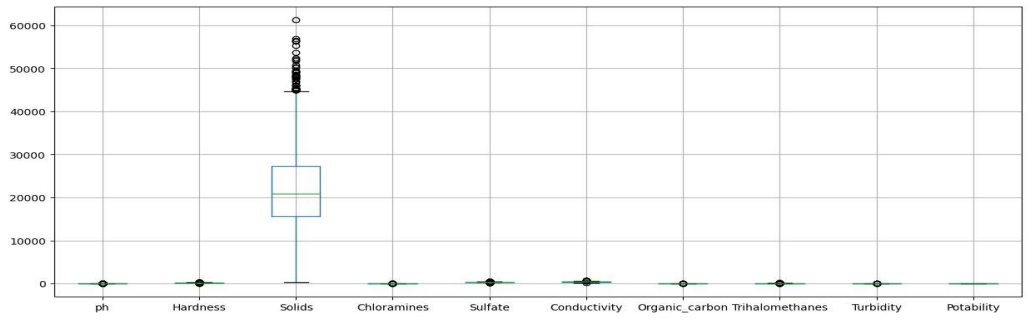


Next Step Is To Check The Outlier Using Box Plot:

You can see outliers but if we remove this outliers we cant have good predict the result will be closer to good water

```
data.boxplot(figsize=(15,6))
plt.show()
```

o/p




```
data['Solids'].describe()
```

o/p:

```
count     3276.000000
mean     22014.092526
std       8768.570828
min        320.942611
25%      15666.690297
50%      20927.833607
75%      27332.762127
max      61227.196008
```

visualization (Histogram , scatter plot)

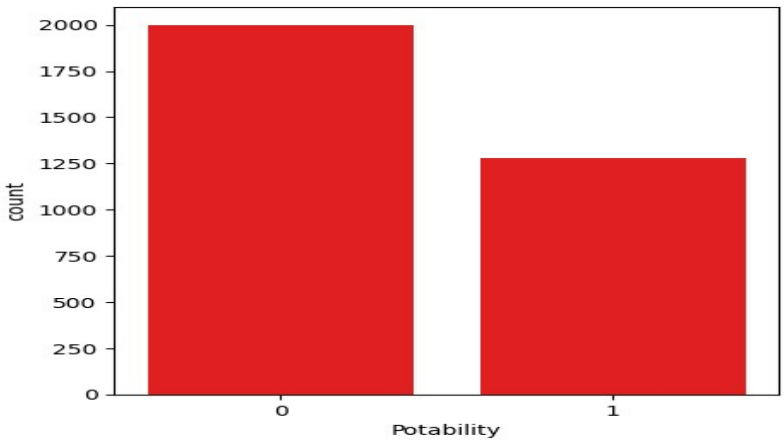
```
 data['Potability'].value_counts()
```

o/p Potability

```
   0    1998
   1    1278
```

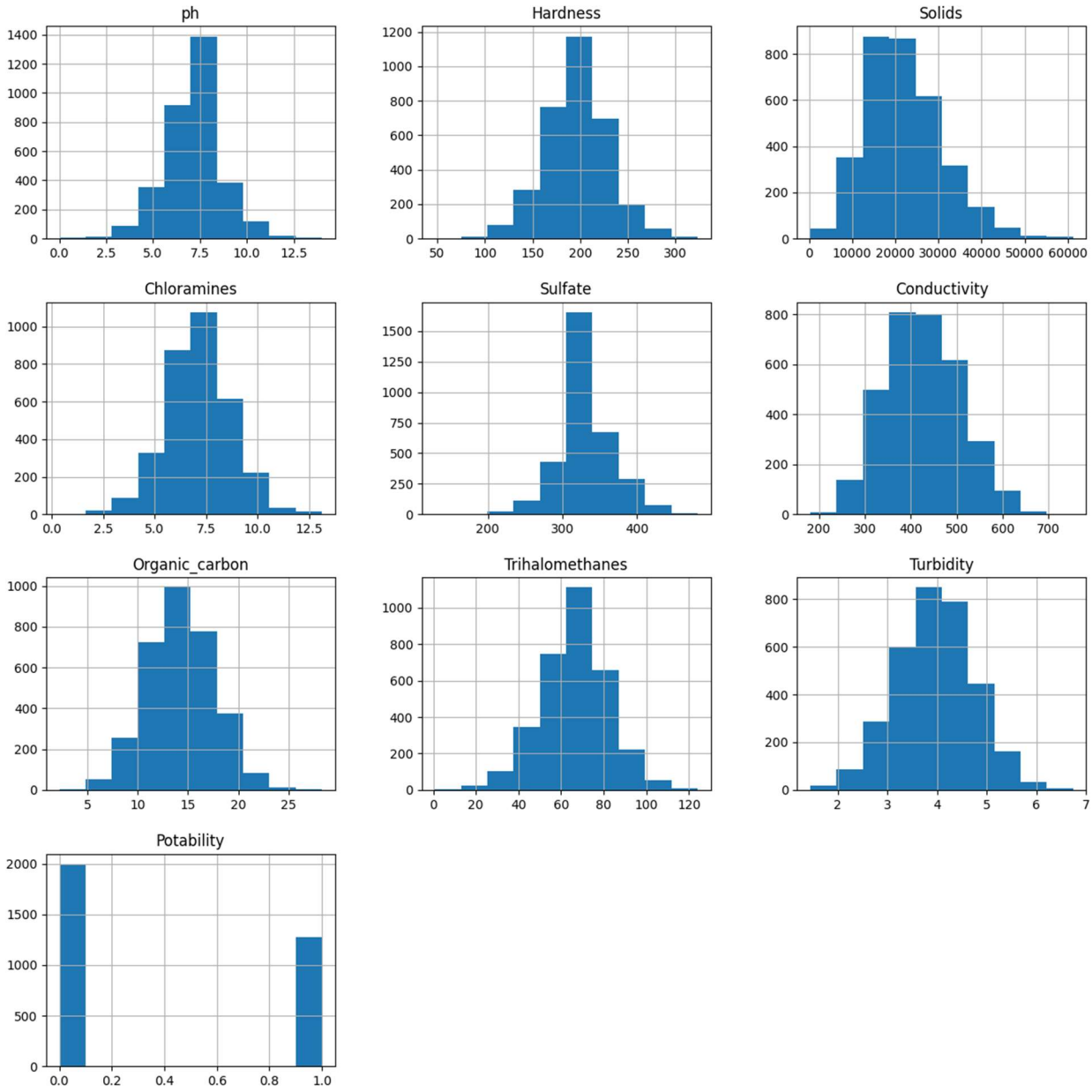
```
plt.figure(figsize=(5,5))
sns.countplot( x=data["Potability"], color="red")
plt.show()
```

o/p



Histogram

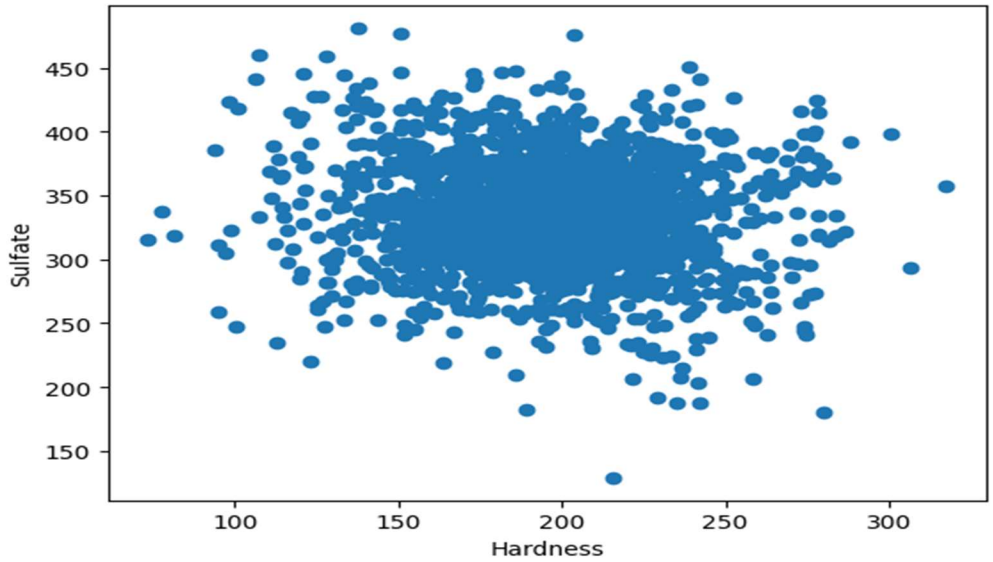
```
data.hist(figsize=(15,15))
plt.show()
```



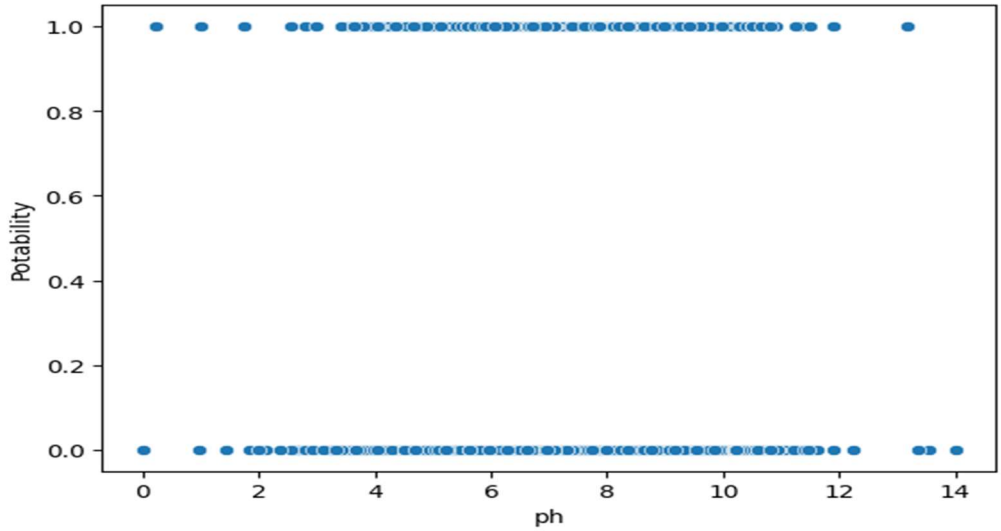
Scatter Plot

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

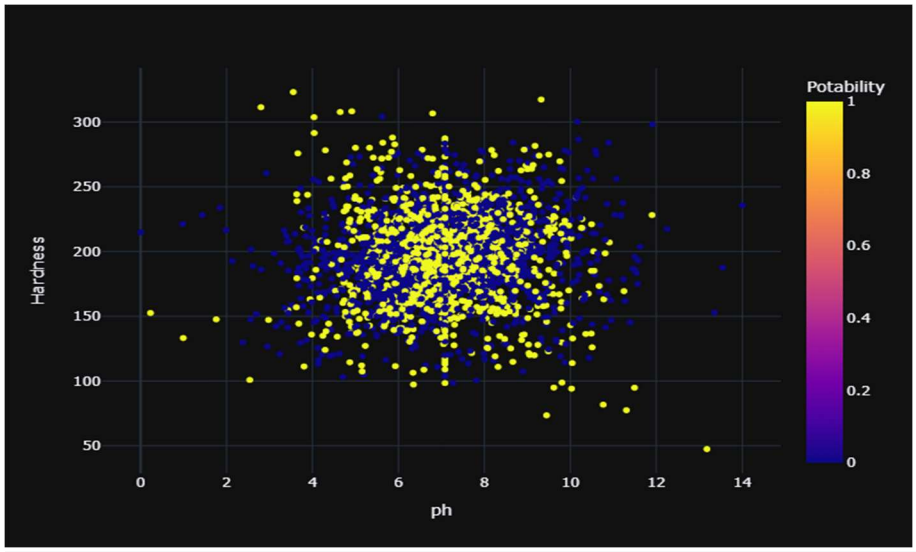
```
plt.show(gp)
```



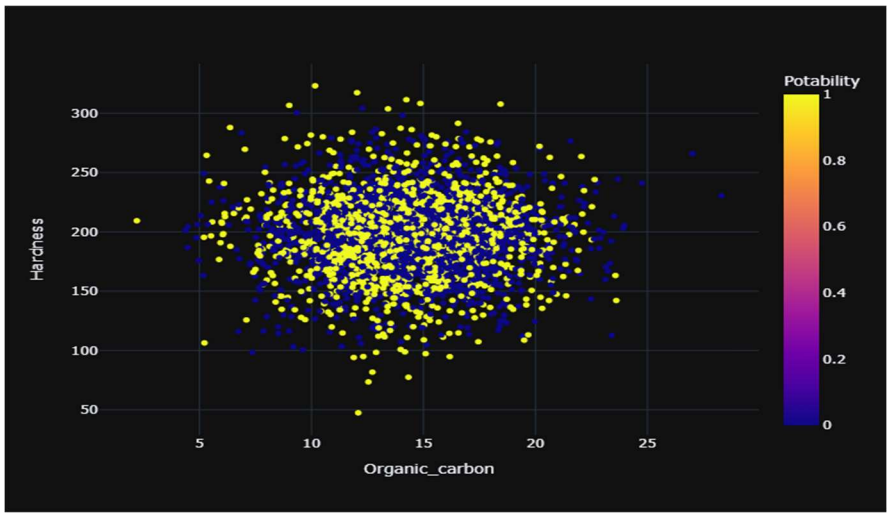

```
sns.scatterplot(x=data['ph'], y=data['Potability'])
plt.show()
```




```
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

Conclusion:

This project has been a comprehensive exploration of water quality analysis, combining the power of data visualization and predictive modeling 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.