PROJECT TITLE : WATER QUALITY ANALYSIS

Project Definition: The project involves analyzing water quality data to assess the suitability of water for specific purposes, such as drinking. The objective is to identify potential issues or deviations from regulatory standards and determine water potability based on various parameters. This project includes defining analysis objectives, collecting water quality data, designing relevant visualizations, and building a predictive model.

Design Thinking:

Before diving into the analysis, it's crucial to understand the context and the stakeholders' needs:

1.Identify the stakeholders: Who are the end-users of this analysis (e.g., regulatory bodies, public health agencies, local communities)?

2. Understand their needs and concerns: What are the specific water quality standards and regulations that need to be met? What are the potential health risks associated with poor water quality.

ANALYSIS PHASE:

1.Determine specific goals: Are you primarily focused on assessing water potability, identifying deviations from standards, or both?

2.Define success criteria: What metrics or criteria will be used to measure the success of your analysis?

DATA COLLECTION:

Identify the data sources and collection methods:

1.Identify the relevant parameters: List all the water quality parameters you have access to, such as pH, Hardness, Solids, Chlorine levels, etc.

2.Data collection plan: Determine how you will gather the data, whether it's through field measurements, historical records, or other sources.

3.Data quality assessment: Consider the reliability and completeness of the data. Are there any gaps or inconsistencies that need to be addressed?

VISUALIZATION STRATERGY:

1.Choose visualization tools: Decide which tools or software will be best for creating visualizations (e.g., Python with libraries like Matplotlib and Seaborn).

2.Visualization types: Select appropriate visualization types for parameter distributions, correlations, and potability assessment. For example, histograms, scatter plots, and heatmaps can be useful.

3.Interactive dashboards: Consider creating interactive dashboards for stakeholders to explore the data themselves.

PREDICTIVE MODELING:

1.Feature selection: Determine which water quality parameters are most relevant for predicting potability. This may involve feature engineering to create new variables.

2.Machine learning algorithms: Choose suitable algorithms for classification tasks (since you are predicting potability). Common choices include Decision Trees, Random Forests, Logistic Regression, or even more advanced methods like Neural Networks.

3.Model evaluation: Establish evaluation metrics (e.g., accuracy, precision, recall) to assess the performance of your predictive model.

By following the Design Thinking process, you can ensure that your water quality analysis project is not only technically sound but also addresses the needs and concerns of the stakeholders effectively.

NAAN MUDHALVAN PROJECT PHASE 2: **INNOVATION**

PROJECT TITLE: **WATER QUALITY ANALYSIS**

DATA ANALYTICS OF WATER QUALITY ANALYSIS

<u>VALUABLE INNOVATION STEPS:</u>

STEP1: DATA COLLECTIONS

- Review the initial design concept to ensure it aligns with the identified problem.
- Gather feedback from stakeholders and subject matter experts for improvements.
- Incorporate necessary changes to enhance the design's effectiveness.

STEP2: CLEANING DATA

- Clean and reprocess the data to remove outliers, errors, and inconsistencies.
- Ensure data quality before analysis.
- Transformation data into proper format for further processes

STEP3: EVALUATE AND ANAYLSIS

<u>Machine Learning Models:</u>

- Train machine learning models to predict water quality based on chemical components present in water.
- This can help pinpoint potential sources of water quality.

<u>Cluster Analysis:</u>

- Use clustering algorithms to group similar noise patterns together.
- This can help identify areas with distinct noise characteristics.

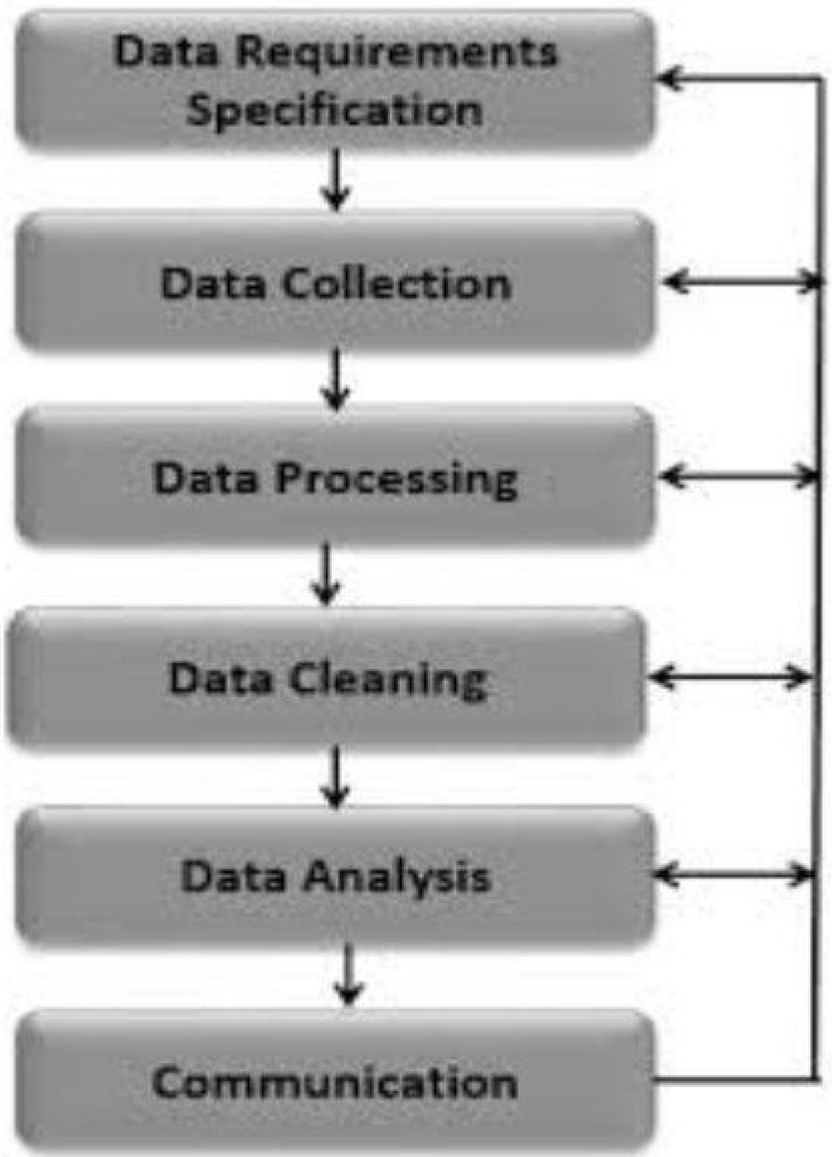
STEP4: DATA VISUALIZATION

- Create interactive maps and visualizations to communicate water quality patterns.
- Finding components present on water such as PH, sodium, carbon, hydrogen etc..

STEP5: DISCRIBE RESULT (COMMUNICATOIN)

- Result is predicted by using appropriate calculation method using statistic evaluation as per our data collections.
- Finally result of water quality is predicted.

WATER QUALITY ANALYSIS DESIGN



da-phase3

November 1, 2023

1 WATER QUALITY ANALYSIS

2 In this part we will begin building your project by loading

[3]:
```
import pandas as pd
file_path = "/content/water_potability.csv"
data = pd.read_csv(file_path)
```

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

```
         ph    Hardness        Solids  Chloramines     Sulfate  Conductivity  \
0       NaN  204.890455  20791.318981     7.300212  368.516441    564.308654
1  3.716080  129.422921  18630.057858     6.635246         NaN    592.885359
2  8.099124  224.236259  19909.541732     9.275884         NaN    418.606213
3  8.316766  214.373394  22018.417441     8.059332  356.886136    363.266516
4  9.092223  181.101509  17978.986339     6.546600  310.135738    398.410813

   Organic_carbon  Trihalomethanes  Turbidity  Potability
0       10.379783        86.990970   2.963135           0
1       15.180013        56.329076   4.500656           0
2       16.868637        66.420093   3.055934           0
3       18.436524       100.341674   4.628771           0
4       11.558279        31.997993   4.075075           0
```

[6]:
```
selected_columns = data[['ph','Hardness','Solids','Chloramines']]
print(selected_columns)
```

```
            ph    Hardness        Solids  Chloramines
0          NaN  204.890455  20791.318981     7.300212
1     3.716080  129.422921  18630.057858     6.635246
2     8.099124  224.236259  19909.541732     9.275884
3     8.316766  214.373394  22018.417441     8.059332
4     9.092223  181.101509  17978.986339     6.546600
…          …           …             …          …
3271  4.668102  193.681735  47580.991603     7.166639
3272  7.808856  193.553212  17329.802160     8.061362
3273  9.419510  175.762646  33155.578218     7.350233
3274  5.126763  230.603758  11983.869376     6.303357
```

```
3275  7.874671  195.102299  17404.177061     7.509306

[3276 rows x 4 columns]
```

3 In this part we begin with the preprocessing od datasae

[7]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
```

[8]:
```
data = pd.read_csv('/content/water_potability.csv')
```

[9]:
```
data['ph'].fillna(data['ph'].mean(), inplace=True)
```

[10]:
```
label_encoder = LabelEncoder()
data['Hardness'] = label_encoder.fit_transform(data['Hardness'])
```

[11]:
```
X = data.drop('Potability', axis=1)
y = data['Potability']
```

[12]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[13]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

4 In this part we beigns with the exploratory data analysis

[22]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[17]:
```
data = pd.read_csv('/content/water_potability.csv')
```

[18]:
```
print("First 5 rows of the dataset:")
print(data.head())
```

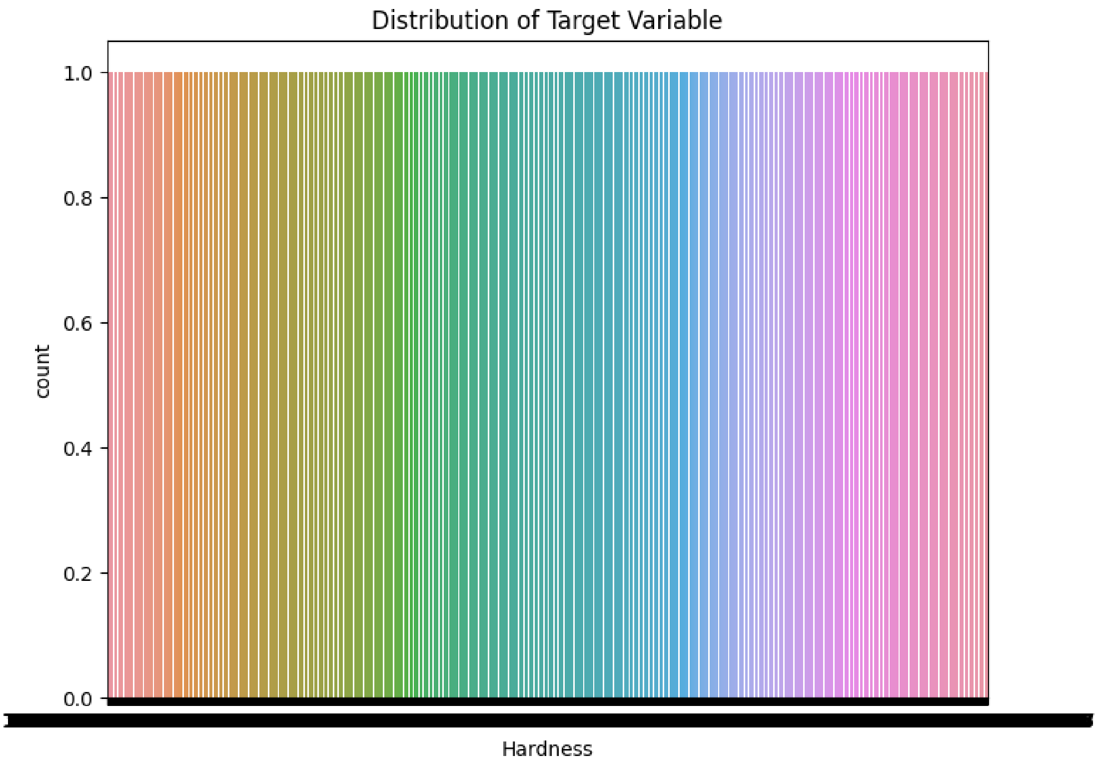
```
First 5 rows of the dataset:
         ph    Hardness        Solids  Chloramines     Sulfate  Conductivity  \
0       NaN  204.890455  20791.318981     7.300212  368.516441    564.308654
1  3.716080  129.422921  18630.057858     6.635246         NaN    592.885359
2  8.099124  224.236259  19909.541732     9.275884         NaN    418.606213
3  8.316766  214.373394  22018.417441     8.059332  356.886136    363.266516
```

```
4  9.092223  181.101509  17978.986339     6.546600  310.135738    398.410813

   Organic_carbon  Trihalomethanes  Turbidity  Potability
0       10.379783        86.990970   2.963135           0
1       15.180013        56.329076   4.500656           0
2       16.868637        66.420093   3.055934           0
3       18.436524       100.341674   4.628771           0
4       11.558279        31.997993   4.075075           0
```

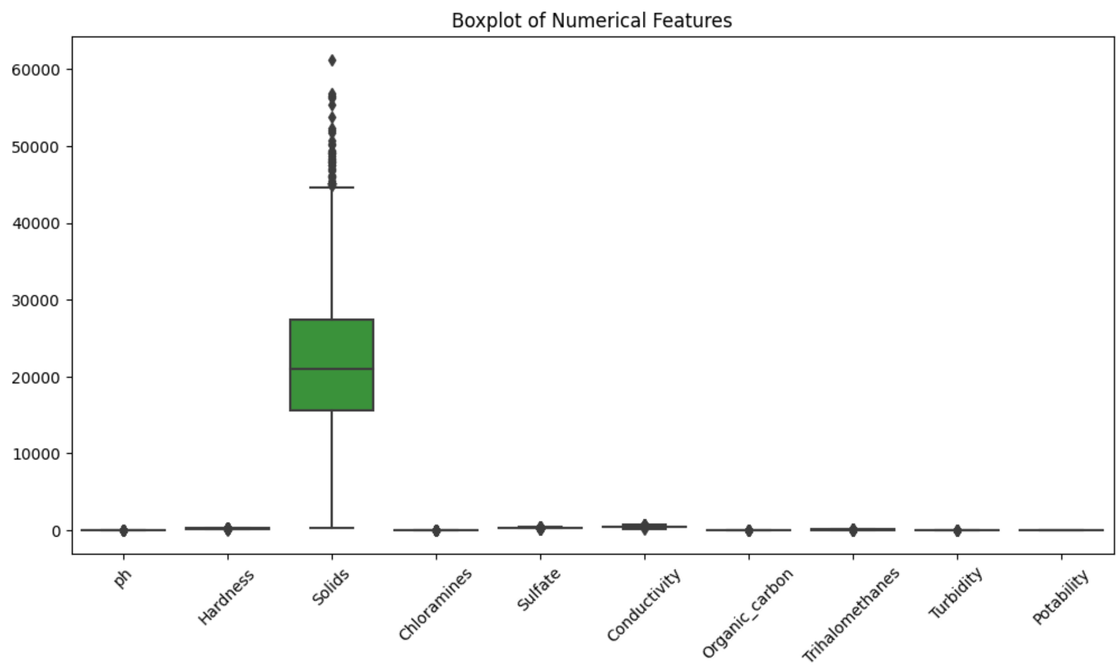
[5]:
```python
print(data.describe())
```

```
                ph     Hardness        Solids  Chloramines      Sulfate  \
count  2785.000000  3276.000000   3276.000000  3276.000000  2495.000000
mean      7.080795   196.369496  22014.092526     7.122277   333.775777
std       1.594320    32.879761   8768.570828     1.583085    41.416840
min       0.000000    47.432000    320.942611     0.352000   129.000000
25%       6.093092   176.850538  15666.690297     6.127421   307.699498
50%       7.036752   196.967627  20927.833607     7.130299   333.073546
75%       8.062066   216.667456  27332.762127     8.114887   359.950170
max      14.000000   323.124000  61227.196008    13.127000   481.030642

       Conductivity  Organic_carbon  Trihalomethanes    Turbidity   Potability
count   3276.000000     3276.000000      3114.000000  3276.000000  3276.000000
mean     426.205111       14.284970        66.396293     3.966786     0.390110
std       80.824064        3.308162        16.175008     0.780382     0.487849
min      181.483754        2.200000         0.738000     1.450000     0.000000
25%      365.734414       12.065801        55.844536     3.439711     0.000000
50%      421.884968       14.218338        66.622485     3.955028     0.000000
75%      481.792304       16.557652        77.337473     4.500320     1.000000
max      753.342620       28.300000       124.000000     6.739000     1.000000
```

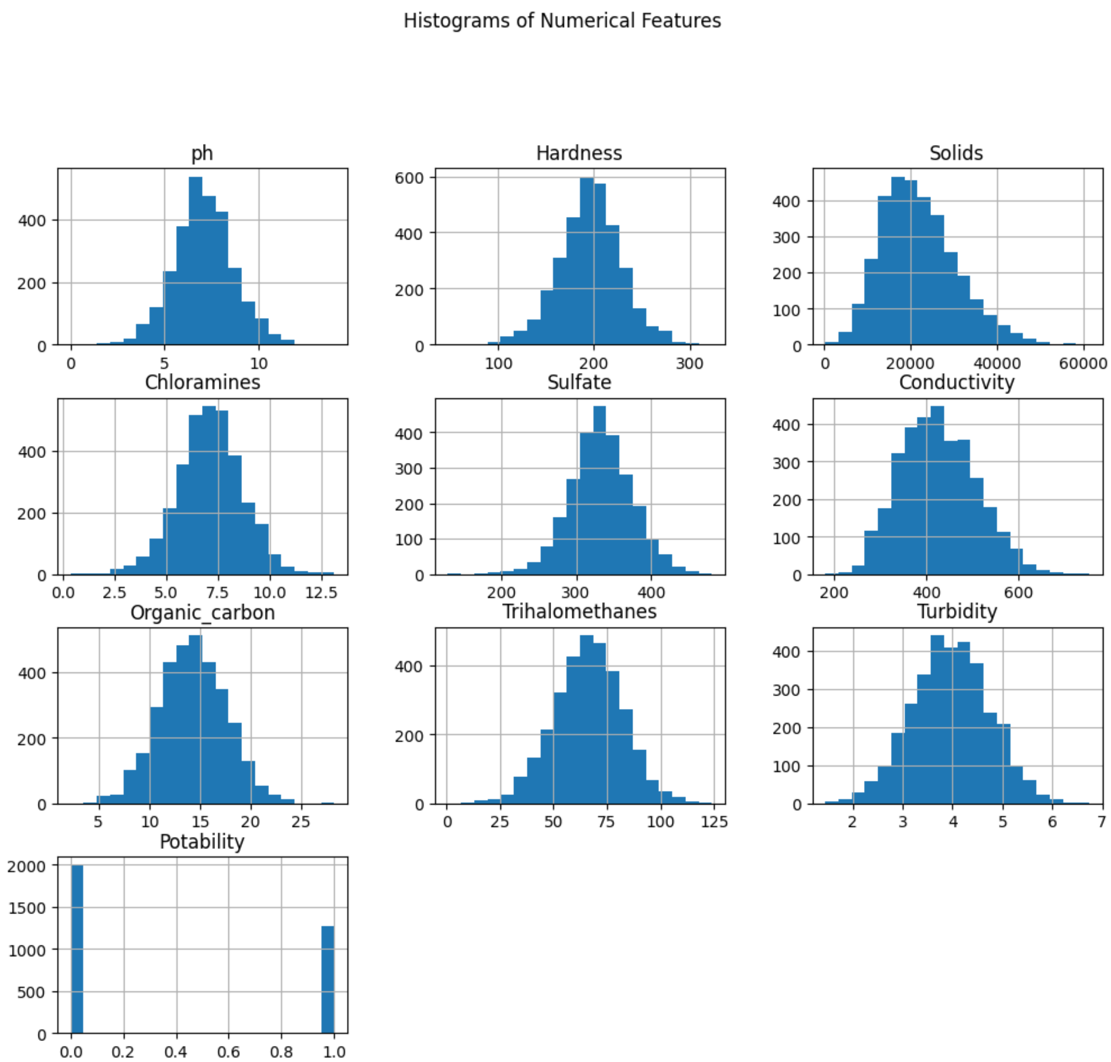
[19]:
```python
plt.figure(figsize=(8, 6))
sns.countplot(x='Hardness', data=data)
plt.title('Distribution of Target Variable')
plt.show()
```



[20]:
```
plt.figure(figsize=(12, 6))
sns.boxplot(data=data.select_dtypes(include=['float64', 'int64']))
plt.title('Boxplot of Numerical Features')
plt.xticks(rotation=45)
plt.show()
```



[21]:
```
data.select_dtypes(include=['float64', 'int64']).hist(bins=20, figsize=(12, 10))
plt.suptitle("Histograms of Numerical Features", y=1.02)
plt.show()
```



[ ]:
```python
data.fillna(data.mean(), inplace=True)
```

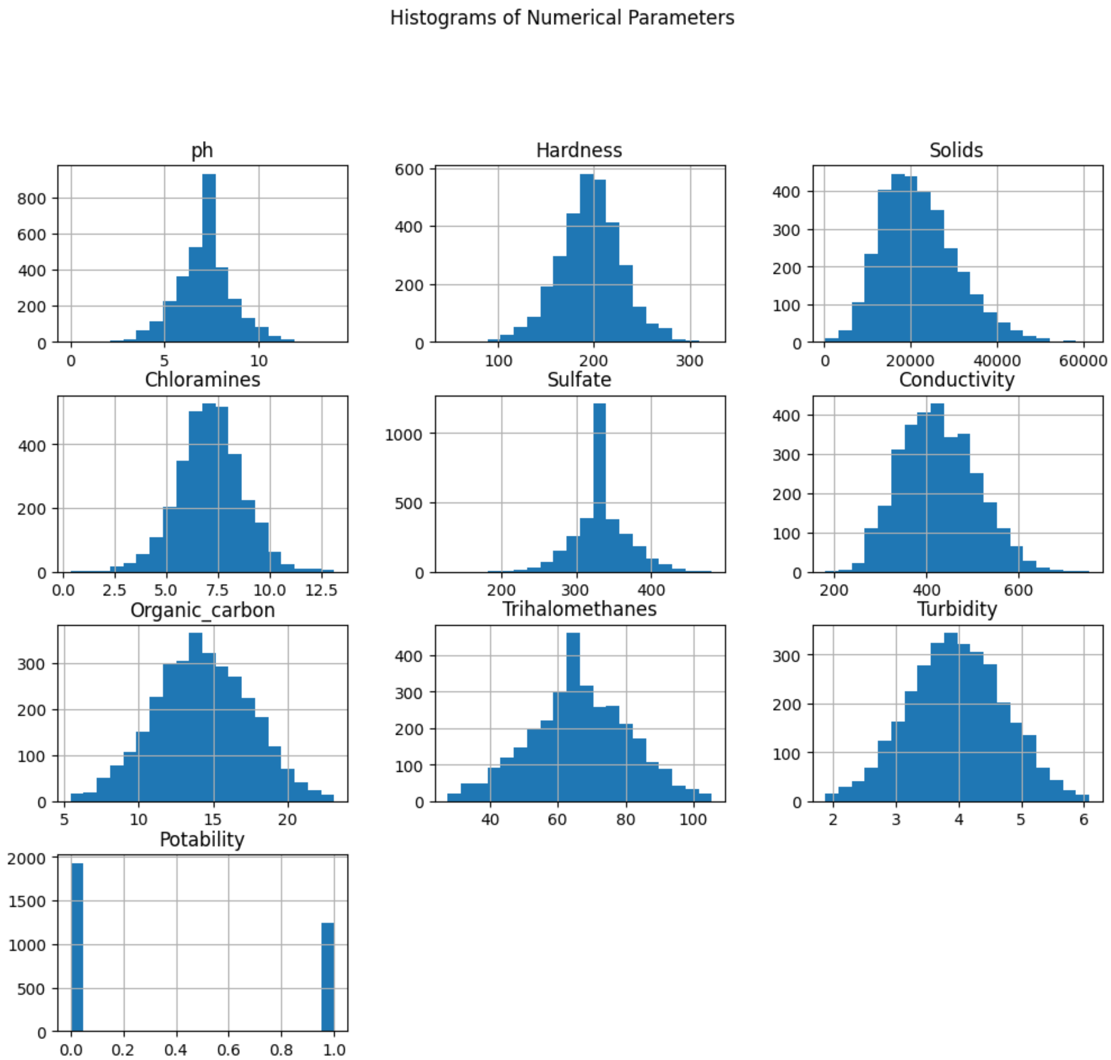
[25]:
```python
def handle_outliers_iqr(data, column):
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    data = data[(data[column] >= lower_bound) & (data[column] <= upper_bound)]
    return data
```

[26]:
```python
numeric_columns = ['Organic_carbon','Trihalomethanes','Turbidity']
```

```
[29]: for column in numeric_columns:
          data = handle_outliers_iqr(data, column)
```

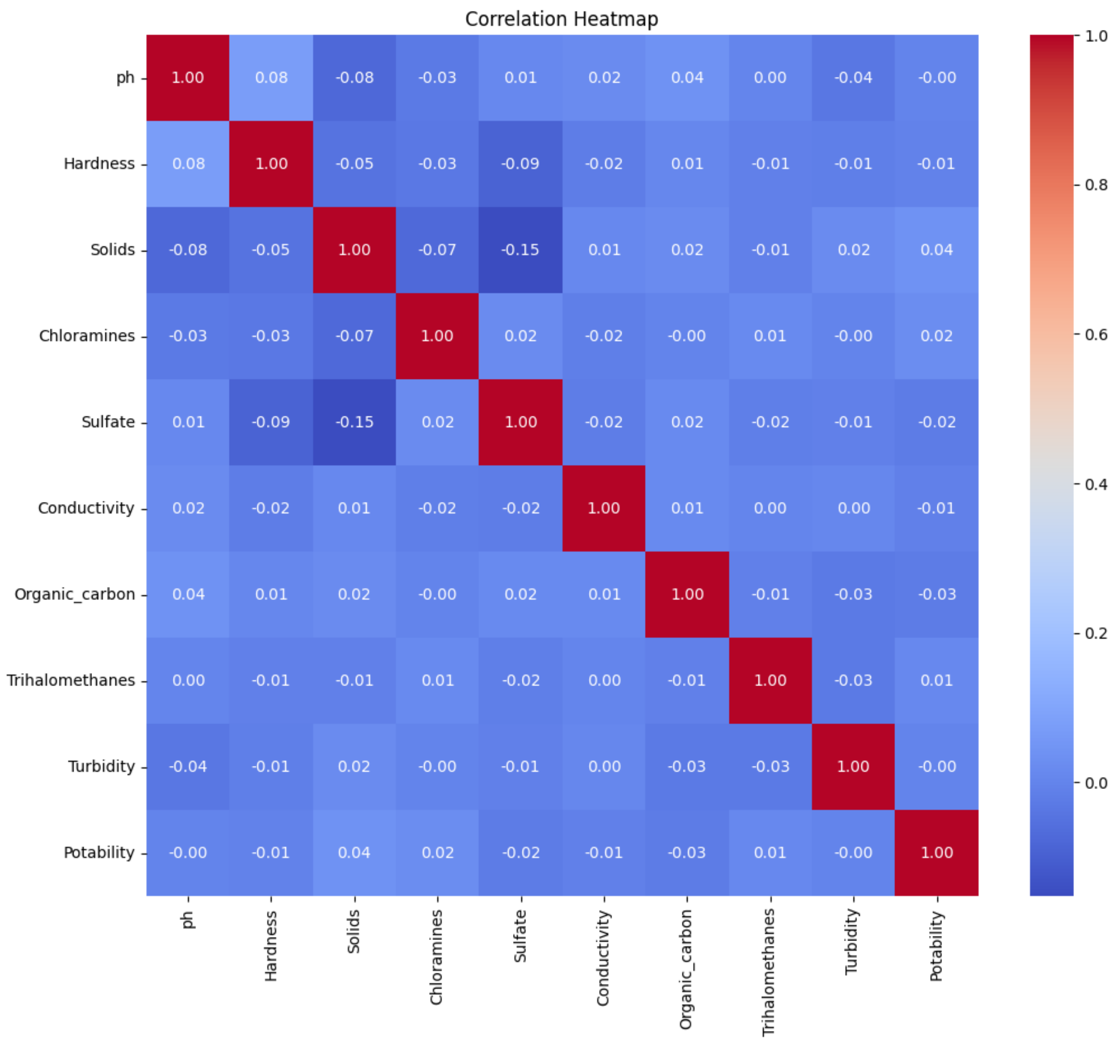
5 Visualize Parameter Distributions

```
[30]: data.select_dtypes(include=['float64', 'int64']).hist(bins=20, figsize=(12, 10))
      plt.suptitle("Histograms of Numerical Parameters", y=1.02)
      plt.show()
```



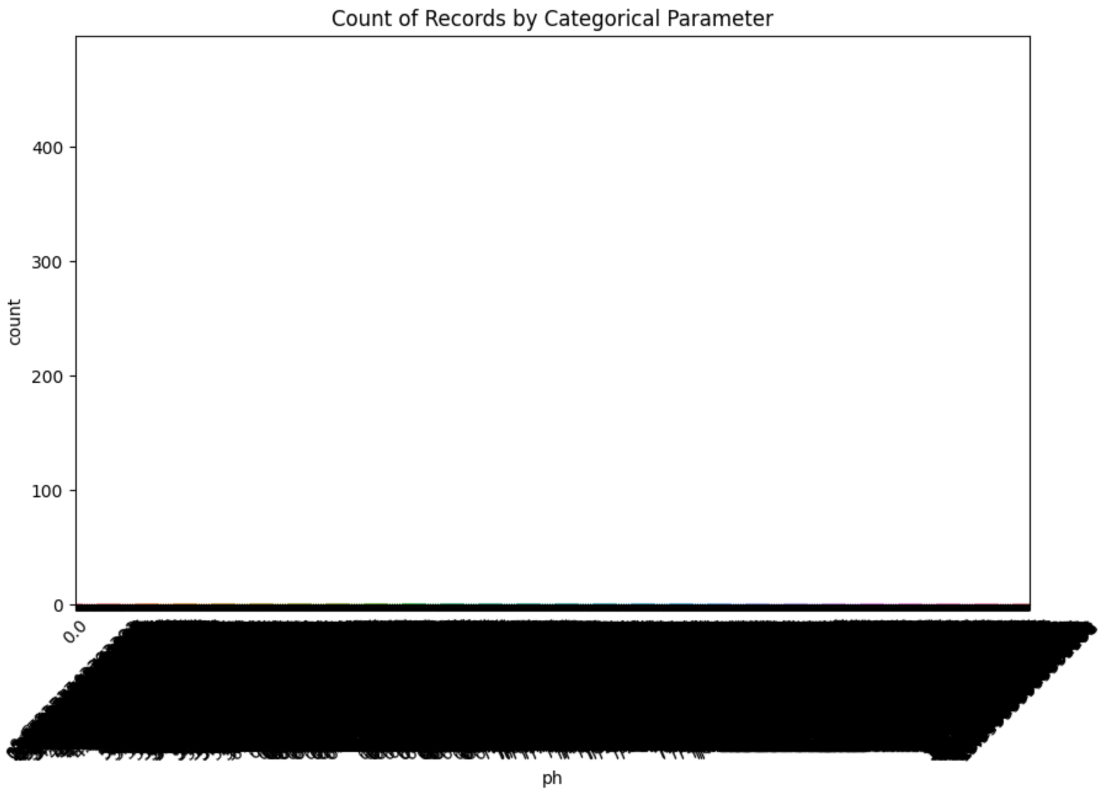
6 Visualize Correlations

[31]:
```
correlation_matrix = data.corr()
# Plot correlation heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```



7 Identify Potential Deviations from Standards

[34]:
```
plt.figure(figsize=(10, 6))
sns.countplot(x='ph', data=data)
plt.title('Count of Records by Categorical Parameter')
plt.xticks(rotation=45)
plt.show()
```



PROJECT TITLE: WATER QUALITY ANALYSIS

PHASE 4: CREATING VISUALIZATIONS AND BUILDING A PREDICTIVE MODEL

In this model we are going to create visualizations of the previous loaded dataset and to building a predictive model.

STEPS FOLLOWED:

STEP 1: LOAD THE DATASET

Loading the given dataset from the source shared, by using the library functions.

STEP 2: HANDLING THE MISSING VALUES

After loading the dataset we have to identify the missing values by using the functions like isnull()

Drop the missing values based upon the nature of the dataset.

STEP 3: CREATING VISUALIZATIONS

After preprocessing the data, we have to create the visualizations of the given dataset.

To create the visualizations, use the library functions such as the matplotlib, seaborn. These libraries can be used to create histograms, scatter plots

STEP 4: BUILDING A PREDICTIVE MODEL

The machine learning models such as the logistic regression and the random forest to determine the water portability based upon the water quality parameters.

VISUALIZATION OF DATA

Importing libraries:

Importing necessary libraries for loading the dataset.

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
```

Load the dataset:

```
In [2]: data=pd.read_csv("/kaggle/input/water-potability/water_potability.csv")
```

DATA PREPROCESSING:

Perform data cleaning and preprocessing. This may include handling missing values, converting data types, and ensuring data quality.

Here already the dataset is cleaned and loaded, so no preprocessing is needed.

In [3]:
```
data.head()
```

Out[3]:

	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

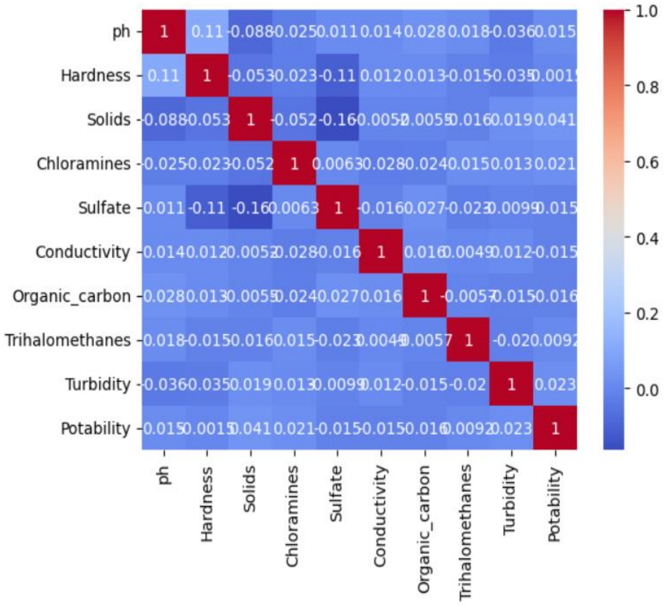
In [4]:
```
data.info()
```

```
<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 [7]:
```
data.shape
```

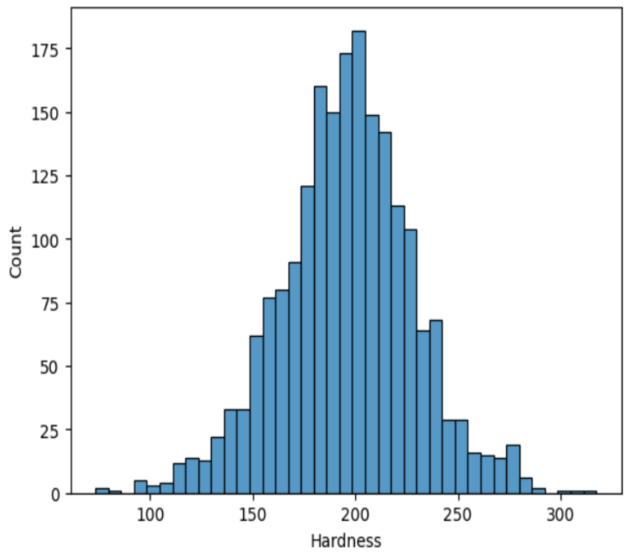
Out[7]: (3276, 10)

In [15]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(cor,annot=True,cmap='coolwarm')
plt.show()
```



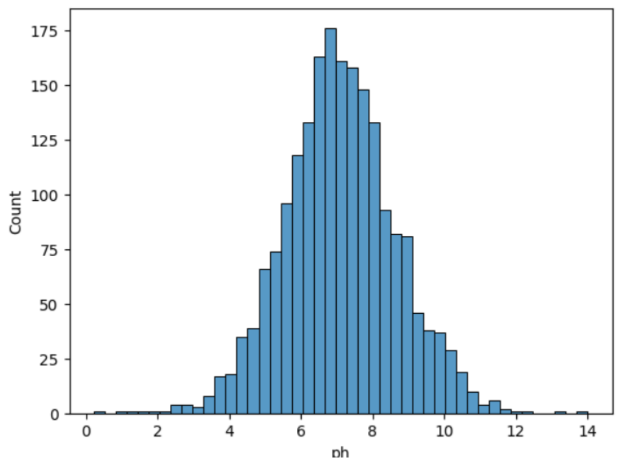
In [19]:
```
sns.histplot(data['Hardness'])
```

Out[19]: <Axes: xlabel='Hardness', ylabel='Count'>

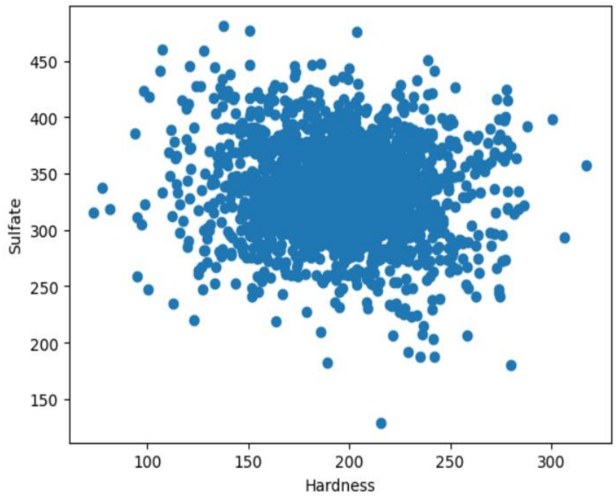


In [20]:
```
sns.histplot(data['ph'])
```

Out[20]: <Axes: xlabel='ph', ylabel='Count'>



In [21]:
```
gp = plt.scatter(data['Hardness'],data['Sulfate'])
plt.xlabel('Hardness')
plt.ylabel('Sulfate')
plt.show(gp)
```



DATA NORMALIZATION AND STANDARDIZATION

In [17]:
```
from sklearn.preprocessing import MinMaxScaler,StandardScaler

normalizer=MinMaxScaler()
standardizer=StandardScaler()
X= normalizer.fit_transform(X)
X=standardizer.fit_transform(X)
```

In [18]:
```
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=62)
```

MODEL BUILDING

In [19]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import ExtraTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score

models = {
    'Logistic Regression': LogisticRegression(),
    'Naive Bayes': GaussianNB(),
    'Support Vector Machine': SVC(),
    'K-Nearest Neighbors': KNeighborsClassifier(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
    'Bagging': BaggingClassifier(),
    'AdaBoost': AdaBoostClassifier(),
    'Gradient Boosting': GradientBoostingClassifier(),
    'Extra Trees': ExtraTreeClassifier(),
}
```

```python
for name, md in models.items():
    md.fit(X_train,Y_train)
    ypred = md.predict(X_test)

    print(f"{name}  with accuracy : {accuracy_score(Y_test,ypred)}")
```

```
Logistic Regression  with accuracy : 0.6178660049627791
Naive Bayes  with accuracy : 0.6327543424317618
Support Vector Machine  with accuracy : 0.7245657568238213
K-Nearest Neighbors  with accuracy : 0.6550868486352357
Decision Tree  with accuracy : 0.6104218362282878
Random Forest  with accuracy : 0.7096774193548387
Bagging  with accuracy : 0.6650124069478908
AdaBoost  with accuracy : 0.6004962779156328
Gradient Boosting  with accuracy : 0.6898263027295285
Extra Trees  with accuracy : 0.5806451612903226
```