PROJECT DOCUMENT – PHASE 5

WATER QUALITY ANALYSIS

Madras Institute of Technology, Anna University

Harishma Sabu Ranjana S Ranjini S Roshni N Vithya S

Project Objective

The project aims to analyze water quality data to determine the suitability of water for specific purposes, particularly for drinking. It involves identifying potential issues or deviations from regulatory standards and predicting water potability based on various water quality parameters. The project will encompass the following key components:

1. Analysis Objectives:

   The primary objectives are to assess water potability, identify deviations from regulatory standards, and understand relationships among water quality parameters.

2. Data Collection:

   Gather the provided water quality data, which includes parameters such as pH, Hardness, Solids, etc.

3. Visualization Strategy:

   Plan how to visualize parameter distributions, correlations, and potability status using appropriate visualization tools.

4. Predictive Modeling:

   Determine the machine learning algorithms and features to use for predicting water potability.

Design Thinking Process

1. Analysis Objectives

Objective 1: Assess Water Potability

- Description: This objective involves classifying water samples as either potable or non-potable (binary classification).
- Steps:
  - Establish criteria for water potability based on regulatory standards (e.g., WHO guidelines).
  - Develop a classification model to predict potability using supervised learning techniques.
  - Evaluate the model's performance using metrics such as accuracy, precision, recall, and F1-score.

Objective 2: Identify Deviations from Standards

- Description: Identify water quality parameters that deviate significantly from established regulatory standards.
- Steps:
  - Calculate summary statistics for each parameter.
  - Define thresholds or ranges for acceptable values based on regulations.
  - Generate reports or alerts for samples with out-of-spec parameters.

Objective 3: Understand Parameter Relationships

- Description: Explore relationships between different water quality parameters to gain insights.
- Steps:
  - Perform correlation analysis to assess the degree of association between parameters.
  - Visualize these relationships using scatter plots, heatmaps, or network graphs.
  - Interpret the visualizations to identify potential patterns and dependencies.

2. Data Collection

Data Source

- Dataset: The dataset for this project can be accessed from the following link: Water Potability Dataset.

- Data Integrity: Verify the dataset's integrity and completeness, checking for missing values, duplicates, and data types.

Data Preprocessing

- Missing Values: Handle missing values through techniques such as imputation or removal.
- Outliers: Identify and address outliers in the data.
- Data Types: Ensure consistent data types for all variables.

Data Split

- Divide the dataset into training and testing subsets for model training and evaluation, typically using an 80-20 or 70-30 split ratio.

3. Visualization Strategy

Parameter Distributions

- Utilize data visualization tools (e.g., Matplotlib, Seaborn) to create histograms, box plots, or density plots to visualize the distributions of water quality parameters (e.g., pH, Hardness).
- Explore central tendencies, spread, and skewness of the distributions to understand their characteristics.

Correlation Analysis

- Calculate and visualize correlation coefficients (e.g., Pearson, Spearman) between pairs of parameters.
- Create correlation matrices and heatmaps to identify strong positive or negative correlations.
- Use scatter plots to visually represent relationships between selected pairs of parameters.

Potability Visualization

- Create visualizations to represent the potability status of water samples.
- Bar charts, pie charts, or stacked bar plots can be used to display the proportion of potable and non-potable samples.

4. Predictive Modeling

Algorithm Selection

- Choose suitable machine learning algorithms for binary classification tasks, such as Logistic Regression, Random Forest, Support Vector Machine (SVM), or Gradient Boosting.

Feature Engineering

- Engineer relevant features that may enhance the model's predictive performance, e.g., creating interaction terms or polynomial features.

Model Development

- Split the data into training and testing sets to facilitate model development and evaluation.
- Train the chosen classification algorithm(s) on the training data.
- Implement hyperparameter tuning to optimize model performance.

Model Evaluation

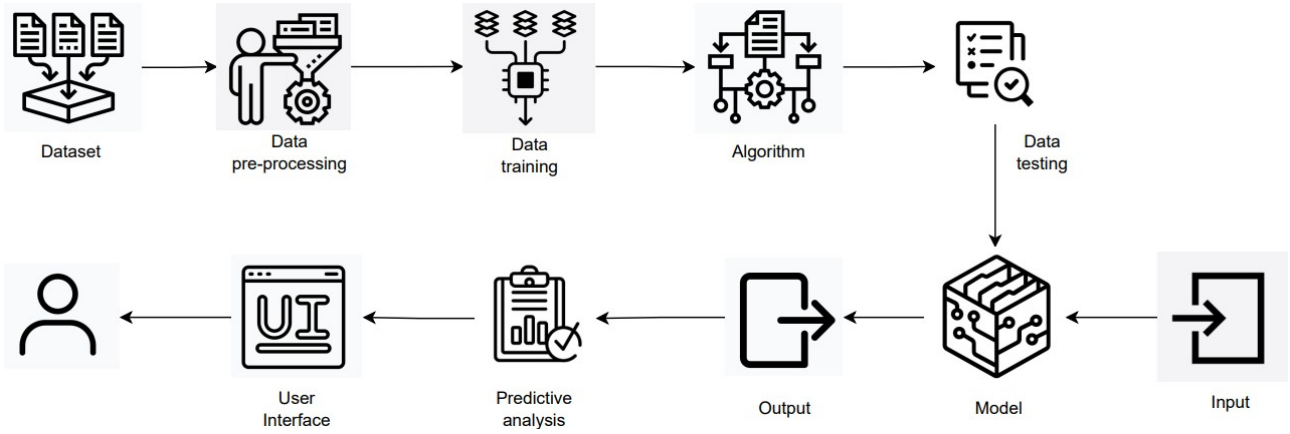
- Assess the model's performance using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.
- Use cross-validation techniques to ensure model robustness and avoid overfitting.

Recommendations

- Provide recommendations and insights based on the model's predictions.
- Highlight key features or parameters that strongly influence water potability.

Development Phases

The ever-expanding process of urbanization has brought forth a pressing issue - ensuring access to safe drinking water. This challenge is amplified by the contamination of water sources due toa multitude of factors, underscoring the importance of water quality analysis and real-time monitoring.



Analytic Flowchart

Solution Description

We have introduced a solution that employs a spectrum of machine learning algorithms to predict the Water Quality Index (WQI). Our predictive model assesses key water quality parameters, including alkalinity, pH levels, temperature, turbidity, dissolved oxygen, and mineral and nutrient content (such as nitrogen and phosphorous). These calculated WQI values are instrumental in determining the suitability of water for various specific applications.

Novelty

This approach stands out by extending beyond the conventional water sample analysis. In addition to this analysis, we harness advanced machine learning techniques to gauge the water's usability and its applicability for specific use cases.

Social Impact

Prioritizing customer satisfaction is paramount, particularly within the framework of total quality management. Given the recent deterioration of water quality attributed to various pollutants, predicting water quality becomes indispensable for managing water pollution and safeguarding the quality of water for consumers. Our evaluation model plays a pivotal role in quantifying and enhancing customer satisfaction.

Business Model (Revenue Model)

Our business model focuses on the enhancement of technology and production processes, translating to increased profitability and streamlined logistics. Furthermore, it offers a revenue stream by aiding users in identifying potential risks associated with water bodies and categorizing nearby water sources according to their suitability for various purposes.

Scalability of the Solution

Our solution boasts impressive scalability, capable of handling vast volumes of data collected from water sources, ranging from smaller water bodies to expansive aquatic ecosystems. This scalable system can adeptly process and analyze the data to meet the real-time needs of a substantial user base.

Data Collection:

- **Web Browsers**: Download data from the dataset.
- **Data Collection Tools**: We need web scraping tools or APIs to collect data if it's not available in a downloadable format.

Data Preprocessing:

- **Python**: For data manipulation and preprocessing.
- **Jupyter Notebook**: For interactive data analysis.
- **Pandas**: A Python library for data cleaning and manipulation.
- **NumPy**: For numerical operations.

Data Visualization:

- **Matplotlib** and **Seaborn**: For creating static visualizations.
- **Plotly** or **Bokeh**: For interactive visualizations.
- **Tableau** or **Power BI**: For advanced data visualization and dashboard creation.

Machine Learning and Predictive Modeling:

- **Python**: For implementing machine learning models.
- **Scikit-Learn**: A Python library for machine learning.
- **XGBoost**, **LightGBM**, or **CatBoost**: For gradient boosting algorithms.

About Dataset

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.

Content

The water_potability.csv file contains water quality metrics for 3276 different water bodies.

1. pH value:

PH is an important parameter in evaluating the acid–base balance of water. It is also the indicator of acidic or alkaline condition of water status. WHO has recommended maximum permissible limit of pH from 6.5 to 8.5. The current investigation ranges were 6.52–6.83 which are in the range of WHO standards.

2. Hardness:

Hardness is mainly caused by calcium and magnesium salts. These salts are dissolved from geologic deposits through which water travels. The length of time water is in contact with hardness producing material helps determine how much hardness there is in raw water. Hardness was originally defined as the capacity of water to precipitate soap caused by Calciumand Magnesium.

3. Solids (Total dissolved solids - TDS):

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 un-wanted taste and diluted color in appearance of water. This is the important parameter for the use of water. The water with high TDS value indicates that water is highly mineralized. Desirable limit for TDS is 500 mg/l and maximum limit is 1000 mg/l which prescribed for drinking purpose.

4. Chloramines:

Chlorine and chloramine are the major disinfectants used in public water systems. Chloramines are most commonly formed when ammonia is added to chlorine to treat drinking water. Chlorine levels up to 4 milligrams per liter (mg/L or 4 parts per million (ppm)) are considered safe in drinking water.

5. Sulfate:

Sulfates are naturally occurring substances that are found in minerals, soil, and rocks. They are present in ambient air, groundwater, plants, and food. The principal commercial use of sulfate is in the chemical industry. Sulfate concentration in seawater is about 2,700 milligrams per liter (mg/L). It ranges from 3 to 30 mg/L in most freshwater supplies, although much higher concentrations (1000 mg/L) are found in some geographic locations.

6. Conductivity:

Pure water is not a good conductor of electric current rather's a good insulator. Increase in ions concentration enhances the electrical conductivity of water. Generally, the amount of dissolved solids in water determines the electrical conductivity. Electrical conductivity (EC) actually measures the ionic process of a solution that enables it to transmit current. According to WHO standards, EC value should not exceeded 400 μS/cm.

7. Organic carbon:

Total Organic Carbon (TOC) in source waters comes from decaying natural organic matter (NOM) as well as synthetic sources. TOC is a measure of the total amount of carbon in organic compounds in pure water. According to US EPA < 2 mg/L as TOC in treated / drinking water, and < 4 mg/Lit in source water which is use for treatment.

8. Trihalomethanes:

THMs are chemicals which may be found in water treated with chlorine. The concentration of THMs in drinking water varies according to the level of organic material in the water, the amount of chlorine required to treat the water, and the temperature of the water that is being treated. THM levels up to 80 ppm is considered safe in drinking water.

9. Turbidity:

The turbidity of water depends on the quantity of solid matter present in the suspended state.It is a measure of light emitting properties of water and the test is used to indicate the quality of waste discharge with respect to colloidal matter. The mean turbidity value obtained for Wondo Genet Campus (0.98 NTU) is lower than the WHO recommended value of 5.00 NTU.

10. Potability:

Indicates if water is safe for human consumption where 1 means Potable and 0 means notpotable.

Data Gathering

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

In [2]:
```
df = pd.read_csv('C:\\Users\\Roshni\\Downloads\\archive\\water_potability.csv')
df.head()
```

Out[2]:

	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

Exploratory Data Analysis

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

Out[3]:
```
(3276, 10)
```

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
ph                 491
Hardness             0
Solids               0
Chloramines          0
Sulfate            781
Conductivity         0
Organic_carbon       0
Trihalomethanes    162
Turbidity            0
Potability           0
dtype: int64
```

In [5]:
```
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
```

In [6]:
```
df.describe()
```

Out[6]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
count	2785.000000	3276.000000	3276.000000	3276.000000	2495.000000	3276.000000	3276.000000	3114.000000	3276.000000	3276.000000
mean	7.080795	196.369496	22014.092526	7.122277	333.775777	426.205111	14.284970	66.396293	3.966786	0.390110
std	1.594320	32.879761	8768.570828	1.583085	41.416840	80.824064	3.308162	16.175008	0.780382	0.487849
min	0.000000	47.432000	320.942611	0.352000	129.000000	181.483754	2.200000	0.738000	1.450000	0.000000
25%	6.093092	176.850538	15666.690297	6.127421	307.699498	365.734414	12.065801	55.844536	3.439711	0.000000
50%	7.036752	196.967627	20927.833607	7.130299	333.073546	421.884968	14.218338	66.622485	3.955028	0.000000
75%	8.062066	216.667456	27332.762127	8.114887	359.950170	481.792304	16.557652	77.337473	4.500320	1.000000
max	14.000000	323.124000	61227.196008	13.127000	481.030642	753.342620	28.300000	124.000000	6.739000	1.000000

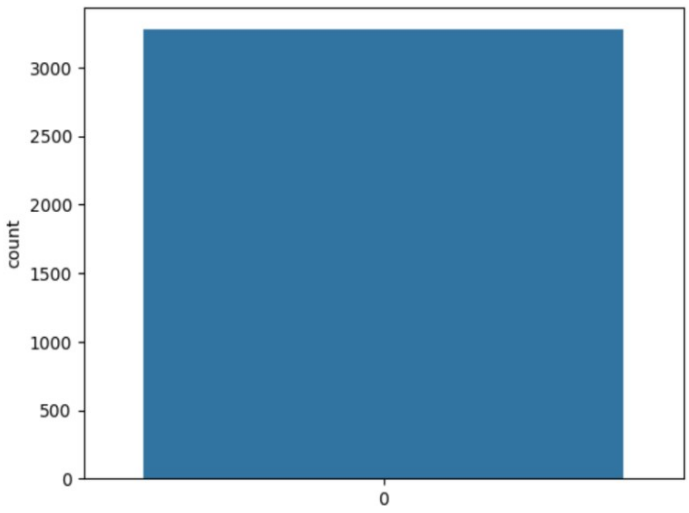
In [7]:
```
df.fillna(df.mean(), inplace=True)
df.isnull().sum()
```

Out[7]:
```
ph                0
Hardness          0
Solids            0
Chloramines       0
Sulfate           0
Conductivity      0
Organic_carbon    0
Trihalomethanes   0
Turbidity         0
Potability        0
dtype: int64
```

In [8]:
```
df.Potability.value_counts()
```

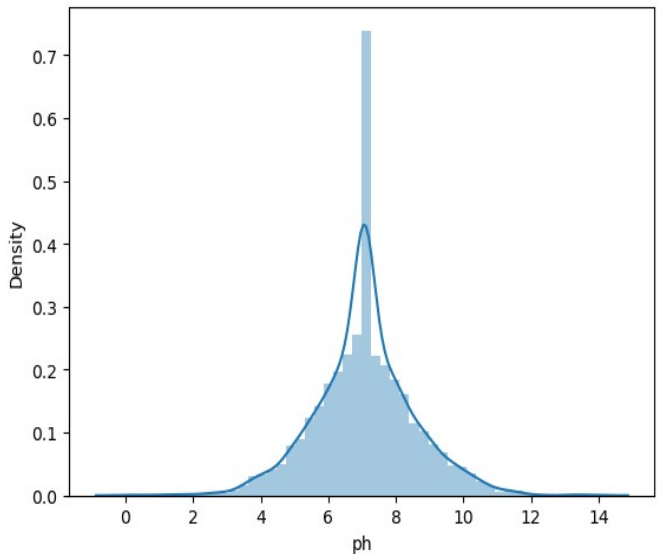
Out[8]:
```
0    1998
1    1278
Name: Potability, dtype: int64
```

In [9]:
```
sns.countplot(df['Potability'])
plt.show()
```

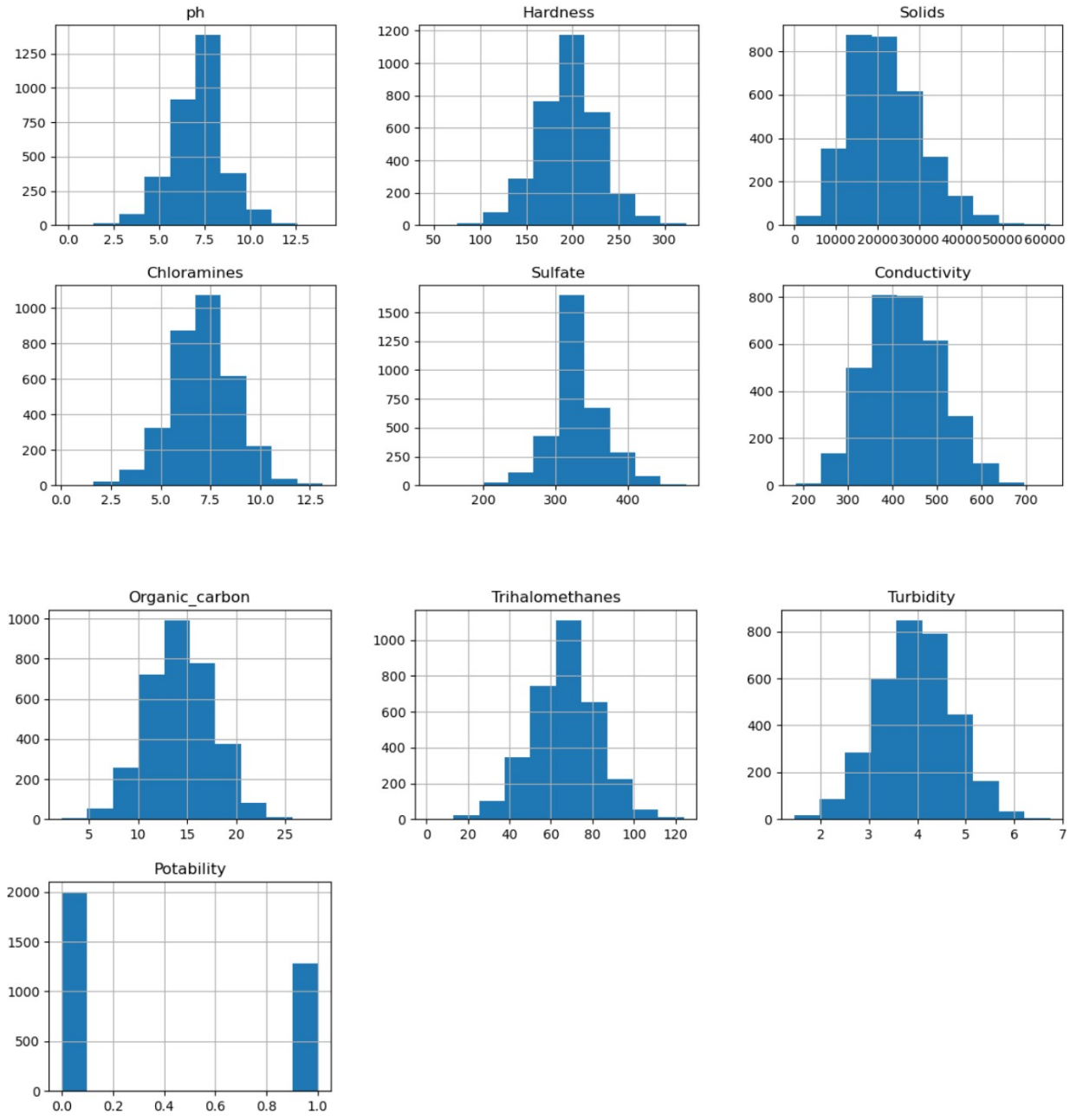


In [10]:
```
sns.distplot(df['ph'])
plt.show()
```

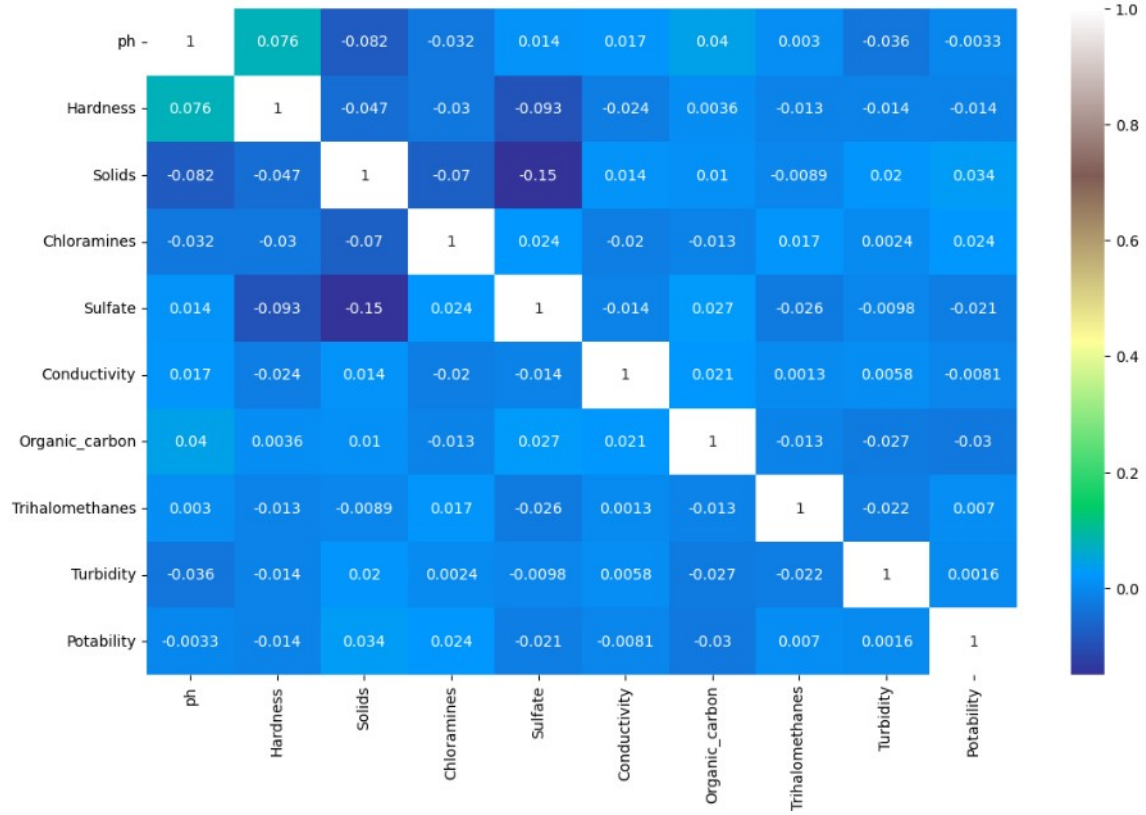
```
C:\Users\Roshni\AppData\Local\Temp\ipykernel_5356\3057384885.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['ph'])
```



In [11]:
```
df.hist(figsize=(14,14))
plt.show()
```

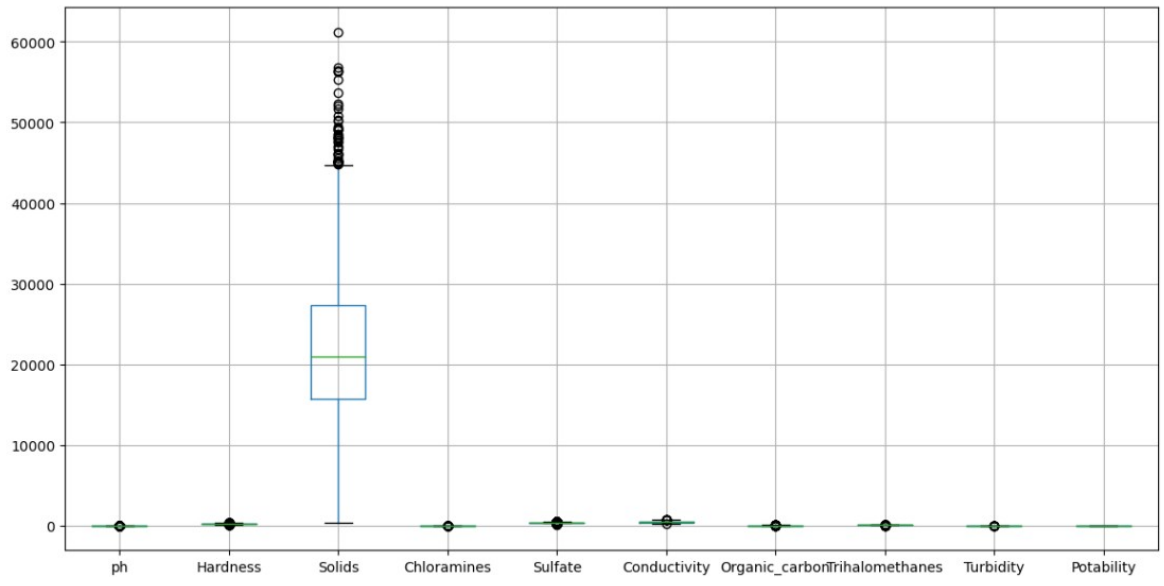


In [12]:
```
plt.figure(figsize=(13,8))
sns.heatmap(df.corr(),annot=True,cmap='terrain')
plt.show()
```



In [13]:
```
df.boxplot(figsize=(14,7))
```

Out[13]: <Axes: >



Train Decision Tree Classifier and checking accuracy

In [14]:
```
X = df.drop('Potability',axis=1)
Y= df['Potability']
```

In [15]:
```
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=101,shuffle=True)
```

In [16]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
dt=DecisionTreeClassifier(criterion= 'gini', min_samples_split= 10, splitter= 'best')
dt.fit(X_train,Y_train)
```

Out[16]:
```
DecisionTreeClassifier(min_samples_split=10)
```

In [17]:
```
prediction=dt.predict(X_test)
print(f"Accuracy Score = {accuracy_score(Y_test,prediction)*100}")
print(f"Confusion Matrix =\n {confusion_matrix(Y_test,prediction)}")
print(f"Classification Report =\n {classification_report(Y_test,prediction)}")
```

```
Accuracy Score = 59.45121951219512
Confusion Matrix =
 [[276 126]
 [140 114]]
Classification Report =
              precision    recall  f1-score   support

           0       0.66      0.69      0.67       402
           1       0.47      0.45      0.46       254

    accuracy                           0.59       656
   macro avg       0.57      0.57      0.57       656
weighted avg       0.59      0.59      0.59       656
```

In [18]:
```
res = dt.predict([[5.735724, 158.318741,25363.016594,7.728601,377.543291,568.304671,13.626624,75.952337,4.732954]])[0]
res
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\base.py:420: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

Out[18]: 1

Apply hyper parameter tuning

In [20]:
```
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import GridSearchCV

# define models and parameters
model = DecisionTreeClassifier()
criterion = ["gini", "entropy"]
splitter = ["best", "random"]
min_samples_split = [2,4,6,8,10,12,14]

# define grid search
grid = dict(splitter=splitter, criterion=criterion, min_samples_split=min_samples_split)
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
grid_search_dt = GridSearchCV(estimator=model, param_grid=grid, n_jobs=-1, cv=cv,
                              scoring='accuracy',error_score=0)
grid_search_dt.fit(X_train, Y_train)
```

Out[20]:
```
GridSearchCV
estimator: DecisionTreeClassifier
DecisionTreeClassifier
```

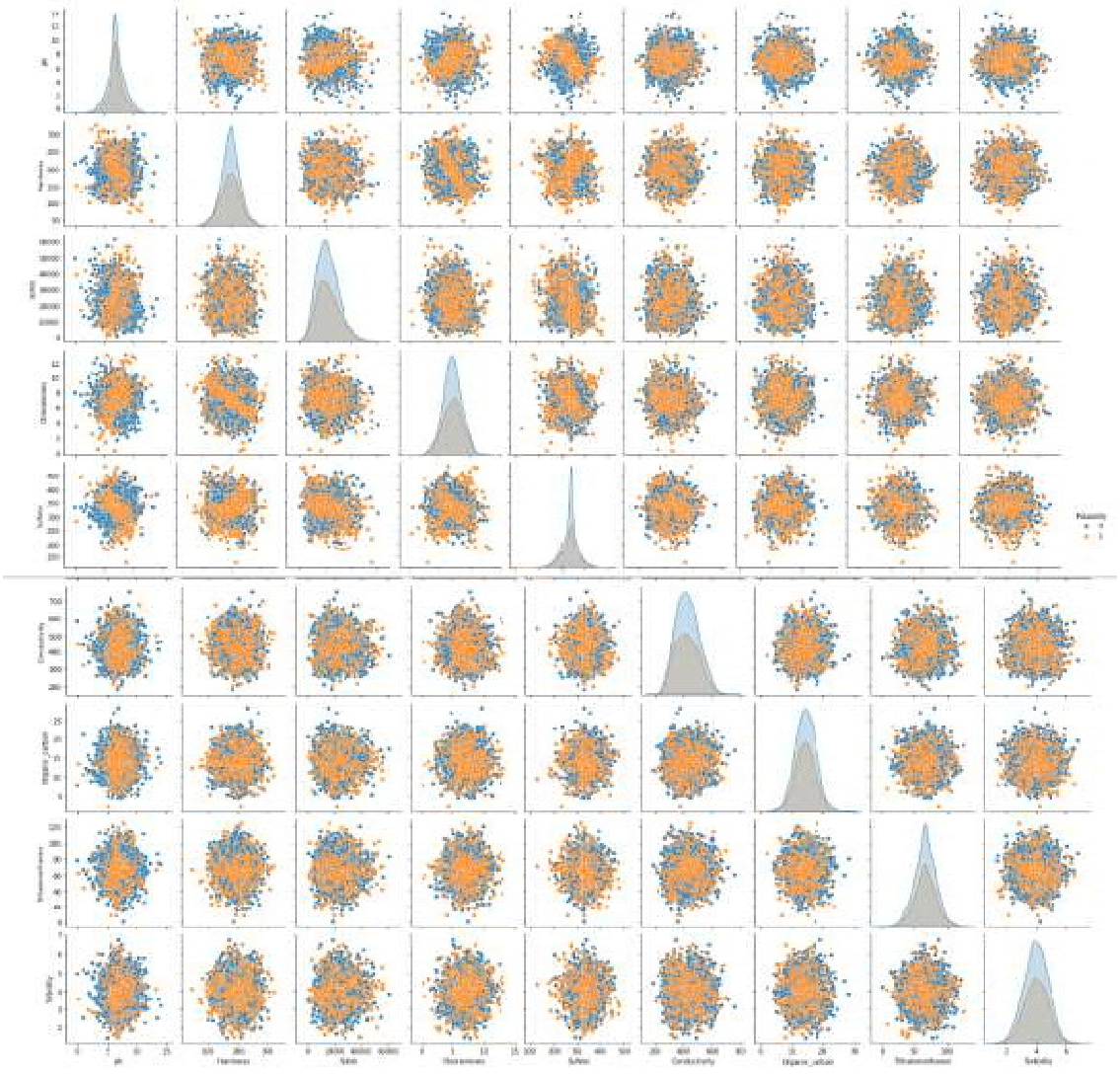
In [21]:

```
print(f"Best: {grid_search_dt.best_score_:.3f} using {grid_search_dt.best_params_}")
means = grid_search_dt.cv_results_['mean_test_score']
stds = grid_search_dt.cv_results_['std_test_score']
params = grid_search_dt.cv_results_['params']

for mean, stdev, param in zip(means, stds, params):
    print(f"{mean:.3f} ({stdev:.3f}) with: {param}")

print("Training Score:",grid_search_dt.score(X_train, Y_train)*100)
print("Testing Score:", grid_search_dt.score(X_test, Y_test)*100)
```

```
Best: 0.593 using {'criterion': 'gini', 'min_samples_split': 6, 'splitter': 'random'}
0.583 (0.030) with: {'criterion': 'gini', 'min_samples_split': 2, 'splitter': 'best'}
0.567 (0.026) with: {'criterion': 'gini', 'min_samples_split': 2, 'splitter': 'random'}
0.586 (0.031) with: {'criterion': 'gini', 'min_samples_split': 4, 'splitter': 'best'}
0.581 (0.023) with: {'criterion': 'gini', 'min_samples_split': 4, 'splitter': 'random'}
0.582 (0.033) with: {'criterion': 'gini', 'min_samples_split': 6, 'splitter': 'best'}
0.593 (0.031) with: {'criterion': 'gini', 'min_samples_split': 6, 'splitter': 'random'}
0.588 (0.034) with: {'criterion': 'gini', 'min_samples_split': 8, 'splitter': 'best'}
0.583 (0.031) with: {'criterion': 'gini', 'min_samples_split': 8, 'splitter': 'random'}
0.589 (0.029) with: {'criterion': 'gini', 'min_samples_split': 10, 'splitter': 'best'}
0.581 (0.028) with: {'criterion': 'gini', 'min_samples_split': 10, 'splitter': 'random'}
0.588 (0.028) with: {'criterion': 'gini', 'min_samples_split': 12, 'splitter': 'best'}
0.581 (0.029) with: {'criterion': 'gini', 'min_samples_split': 12, 'splitter': 'random'}
0.590 (0.025) with: {'criterion': 'gini', 'min_samples_split': 14, 'splitter': 'best'}
0.591 (0.030) with: {'criterion': 'gini', 'min_samples_split': 14, 'splitter': 'random'}
0.583 (0.032) with: {'criterion': 'entropy', 'min_samples_split': 2, 'splitter': 'best'}
0.571 (0.027) with: {'criterion': 'entropy', 'min_samples_split': 2, 'splitter': 'random'}
0.584 (0.029) with: {'criterion': 'entropy', 'min_samples_split': 4, 'splitter': 'best'}
0.578 (0.033) with: {'criterion': 'entropy', 'min_samples_split': 4, 'splitter': 'random'}
0.587 (0.028) with: {'criterion': 'entropy', 'min_samples_split': 6, 'splitter': 'best'}
0.589 (0.024) with: {'criterion': 'entropy', 'min_samples_split': 6, 'splitter': 'random'}
0.587 (0.028) with: {'criterion': 'entropy', 'min_samples_split': 8, 'splitter': 'best'}
0.580 (0.031) with: {'criterion': 'entropy', 'min_samples_split': 8, 'splitter': 'random'}
0.588 (0.026) with: {'criterion': 'entropy', 'min_samples_split': 10, 'splitter': 'best'}
0.586 (0.027) with: {'criterion': 'entropy', 'min_samples_split': 10, 'splitter': 'random'}
0.588 (0.032) with: {'criterion': 'entropy', 'min_samples_split': 12, 'splitter': 'best'}
0.593 (0.026) with: {'criterion': 'entropy', 'min_samples_split': 12, 'splitter': 'random'}
0.587 (0.031) with: {'criterion': 'entropy', 'min_samples_split': 14, 'splitter': 'best'}
0.590 (0.023) with: {'criterion': 'entropy', 'min_samples_split': 14, 'splitter': 'random'}
Training Score: 90.64885496183206
Testing Score: 56.40243902439024
```

SCATTER PLOT:

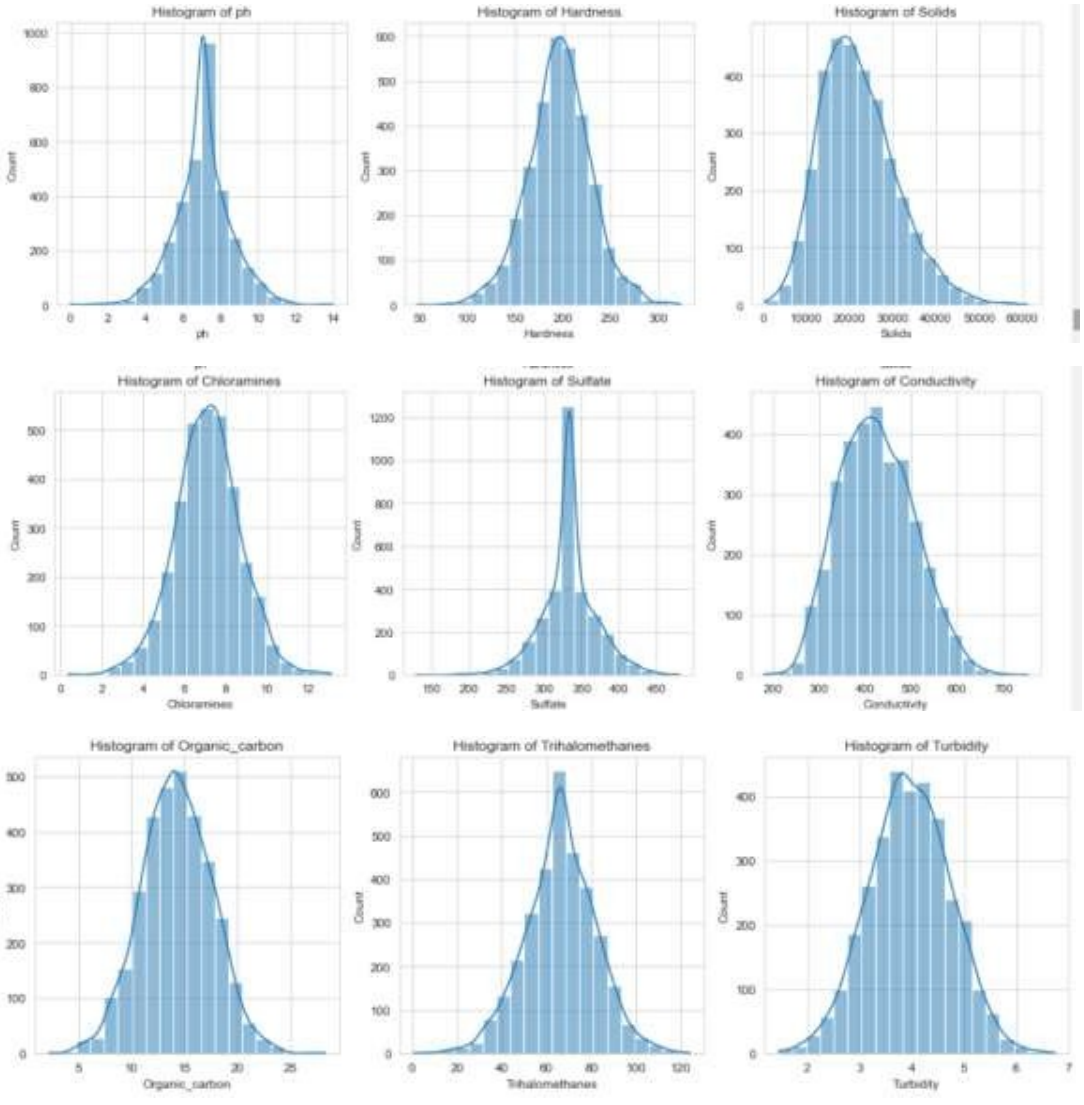
```
[32]: sns.pairplot(df, hue='Potability')
      plt.show()
```



HISTOGRAMS:

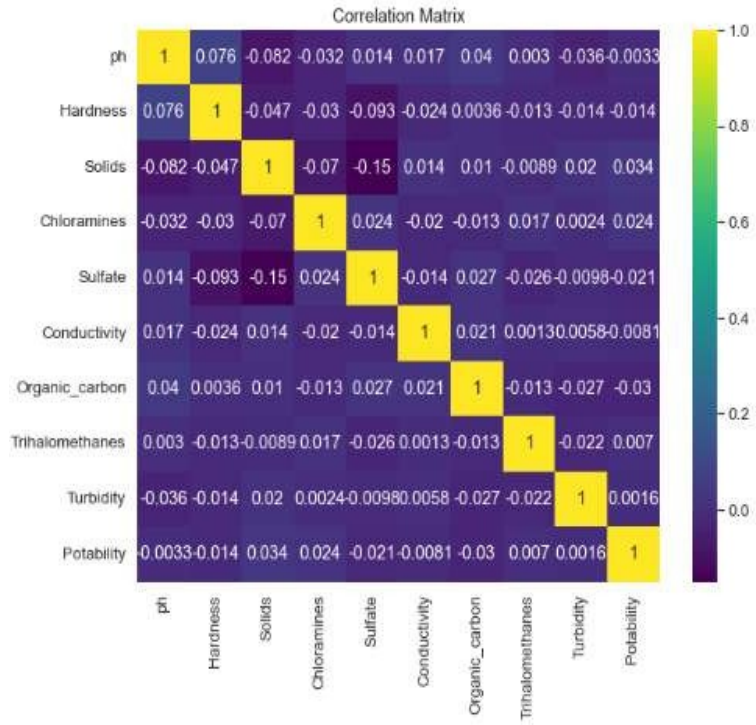
[33]:

```
plt.figure(figsize=(15, 10))
sns.set_style("whitegrid")

for column in df.columns:
    plt.subplot(3, 3, df.columns.get_loc(column) + 1)
    sns.histplot(df[column], bins=20, kde=True)
    plt.title(f'Histogram of {column}')

plt.tight_layout()
plt.show()
```



CORRELATION MATRIX:

```
[38]: plt.figure(figsize=(10, 8))
      sns.set(font_scale=1.2)
      sns.heatmap(df.corr(), annot=True, cmap='viridis')
      plt.title('Correlation Matrix')
      plt.show()
```



PREDICTIVE MODEL:

1. Using Logistic Regression:

```
[39]: from sklearn.model_selection import train_test_split
      X = df.drop('Potability', axis=1)
      y = df['Potability']
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[40]: from sklearn.linear_model import LogisticRegression
      logistic_model = LogisticRegression(random_state=42)
      logistic_model.fit(X_train, y_train)
```

```
[40]: LogisticRegression(random_state=42)
```

```
[41]: y_pred = logistic_model.predict(X_test)
```

```
[42]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

      accuracy = accuracy_score(y_test, y_pred)
      confusion_mat = confusion_matrix(y_test, y_pred)
      classification_rep = classification_report(y_test, y_pred)

      print(f'Accuracy: {accuracy:.2f}')
      print('Confusion Matrix:\n', confusion_mat)
      print('Classification Report:\n', classification_rep)
```

```
Accuracy: 0.63
Confusion Matrix:
 [[412   0]
 [244   0]]
Classification Report:
              precision    recall  f1-score   support

           0       0.63      1.00      0.77       412
           1       0.00      0.00      0.00       244

    accuracy                           0.63       656
   macro avg       0.31      0.50      0.39       656
weighted avg       0.39      0.63      0.48       656
```

2. Using Random forest:

[43]:
```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
confusion_mat_rf = confusion_matrix(y_test, y_pred_rf)
classification_rep_rf = classification_report(y_test, y_pred_rf)
print("Random Forest Model:")
print(f'Accuracy: {accuracy_rf:.2f}')
print('Confusion Matrix:\n', confusion_mat_rf)
print('Classification Report:\n', classification_rep_rf)
```

```
Random Forest Model:
Accuracy: 0.68
Confusion Matrix:
 [[353  59]
 [152  92]]
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.86      0.77       412
           1       0.61      0.38      0.47       244

    accuracy                           0.68       656
   macro avg       0.65      0.62      0.62       656
weighted avg       0.67      0.68      0.66       656
```

3. Using AdaBoost Model:

[44]:
```
from sklearn.ensemble import AdaBoostClassifier
ada_model = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=1), n_estimators=100, random_state=42)
ada_model.fit(X_train, y_train)
y_pred_ada = ada_model.predict(X_test)
accuracy_ada = accuracy_score(y_test, y_pred_ada)
confusion_mat_ada = confusion_matrix(y_test, y_pred_ada)
classification_rep_ada = classification_report(y_test, y_pred_ada)
print("AdaBoost Model:")
print(f'Accuracy: {accuracy_ada:.2f}')
print('Confusion Matrix:\n', confusion_mat_ada)
print('Classification Report:\n', classification_rep_ada)
```

```
AdaBoost Model:
Accuracy: 0.62
Confusion Matrix:
 [[351  61]
 [189  55]]
Classification Report:
              precision    recall  f1-score   support

           0       0.65      0.85      0.74       412
           1       0.47      0.23      0.31       244

    accuracy                           0.62       656
   macro avg       0.56      0.54      0.52       656
weighted avg       0.58      0.62      0.58       656
```

4. Using Decision Tree:

[45]:
```
from sklearn.tree import DecisionTreeClassifier
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)
y_pred_dt = dt_model.predict(X_test)
accuracy_dt = accuracy_score(y_test, y_pred_dt)
confusion_mat_dt = confusion_matrix(y_test, y_pred_dt)
classification_rep_dt = classification_report(y_test, y_pred_dt)
print("Decision Tree Model:")
print(f'Accuracy: {accuracy_dt:.2f}')
print('Confusion Matrix:\n', confusion_mat_dt)
print('Classification Report:\n', classification_rep_dt)
```

```
Decision Tree Model:
Accuracy: 0.58
Confusion Matrix:
 [[255 157]
 [120 124]]
Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.62      0.65       412
           1       0.44      0.51      0.47       244

    accuracy                           0.58       656
   macro avg       0.56      0.56      0.56       656
weighted avg       0.59      0.58      0.58       656
```

[46]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(15, 10))
plot_tree(dt_model, feature_names=X.columns, class_names=["Not Potable", "Potable"], filled=True)
plt.title("Decision Tree Visualization")
plt.show()
```

Insights on the analysis

Analyzing water quality is a crucial step in assessing its safety for human consumption. Insights gained from a comprehensive analysis can provide information about various physical, chemical, and biological parameters that influence water quality. Here's how these insights can help assess water quality and determine potability:

1. Chemical Composition:

- pH Levels: The pH of water indicates its acidity or alkalinity. Potable water typically has a pH between 6.5 and 8.5. Extreme values can indicate contamination and affect taste.

- Chemical Contaminants: Analysis can identify the presence of harmful substances such as heavy metals, pesticides, fertilizers, and industrial pollutants. Regulatory limits for these contaminants are established to ensure safe drinking water.

2. Microbiological Analysis:

    - Bacterial and Viral Presence: Testing for coliform bacteria, E. coli, and other pathogens helps assess microbial contamination. Potable water should be free from these harmful microorganisms to prevent waterborne diseases.

3. Physical Characteristics:

    - Turbidity: High turbidity levels indicate the presence of suspended particles. Excessive turbidity can affect water clarity and may harbor harmful microorganisms. Clear water is generally an indicator of good quality.

4. Nutrient Levels:

    - Nitrates and Phosphates: Elevated levels of these nutrients may indicate contamination from agricultural runoff or sewage. Excess nutrients can lead to eutrophication, negatively impacting water quality.

5. Dissolved Oxygen (DO):

    - Oxygen Levels: DO is essential for the survival of aquatic organisms. Low DO levels can indicate pollution or other factors affecting the water's ability to support life.

6. Temperature:

    - Temperature Fluctuations: Abnormal temperature ranges can affect the solubility of gases and the growth of aquatic organisms. Extreme temperatures may indicate contamination or environmental stress.

7. Radiological Analysis:

    - Radioactive Elements: Testing for the presence of radioactive substances helps ensure water safety. Elevated levels can pose serious health risks.

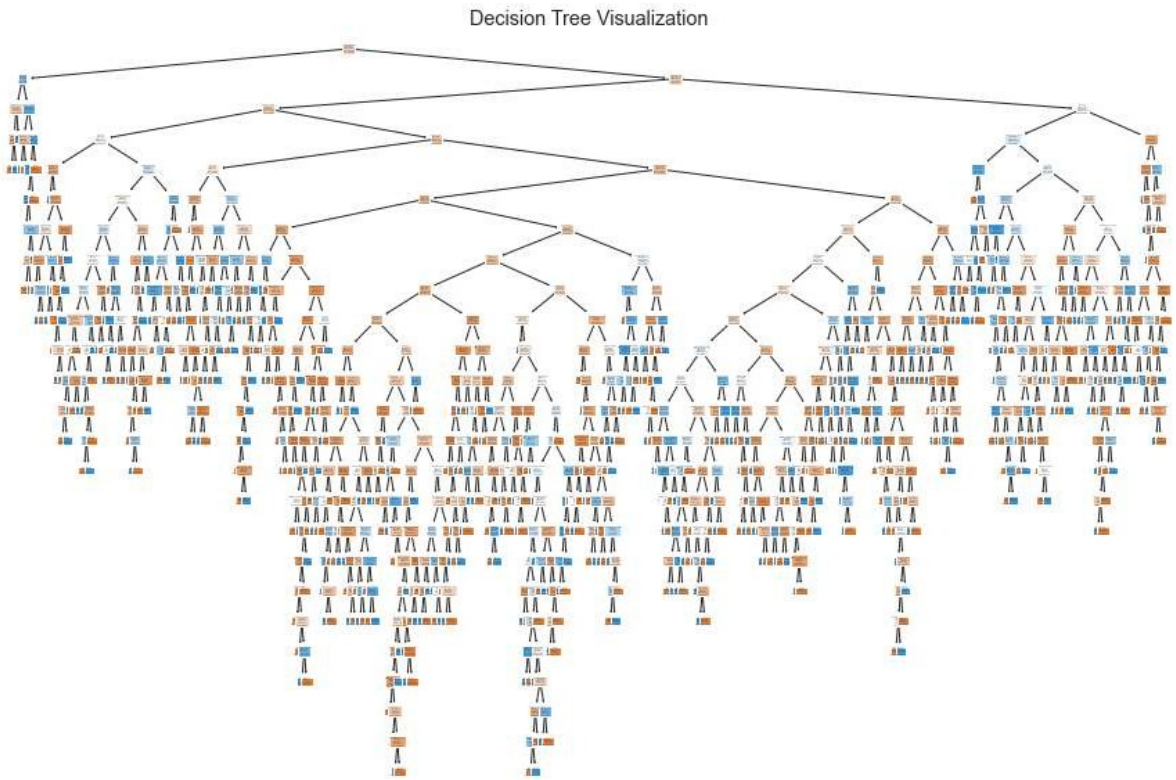
8. Conductivity and Total Dissolved Solids (TDS):

    - Concentration of Dissolved Substances: High conductivity and TDS levels may suggest the presence of minerals or salts, impacting the water's taste and safety.

9. Statistical Trend Analysis:

    - Long-Term Trends: Analyzing data over time helps identify any emerging issues or persistent problems. This is crucial for implementing preventive measures and maintaining consistent water quality.

By integrating insights from these analyses, water quality professionals and regulatory bodies can assess the overall safety and potability of water sources. Regular monitoring and prompt response to any deviations from established standards are essential to ensure a sustainable and safe water supply for communities.



VISUALIZING FEATURE IMPORTANCE

[47]:
```
import matplotlib.pyplot as plt
importances = rf_model.feature_importances_
features = X.columns
indices = importances.argsort()[::-1]
plt.figure(figsize=(10, 6))
plt.title("Feature Importances")
plt.bar(range(X.shape[1]), importances[indices], align="center")
plt.xticks(range(X.shape[1]), features[indices], rotation=90)
plt.xlabel("Feature")
plt.ylabel("Importance")
plt.show()
```