WATER QUALITY ANALYSIS

PHASE- 4 :

VISUALIZATION AND ALGORITHMS

Visualization libraries are essential tools in data analysis and interpretation. They allow you to represent complex data sets in graphical or pictorial formats, making it easier to identify patterns, trends, and insights. Here are some advantages of using visualization libraries.

Advantages of Visualization Libraries:

1. Data Comprehension:

Visualizations provide a clear and concise way to understand large and complex datasets. Patterns and trends that might not be apparent in raw data can be easily identified through visual representations.

2. Communication:

Visualizations are powerful tools for communicating findings to non-technical stakeholders. A well-designed graph or chart can convey insights more effectively than raw data, making it easier for others to understand the information.

3. Decision Making:

Visualizations enable better decision-making by providing a visual summary of data. Decision-makers can quickly grasp the situation and make informed choices based on the visualized information.

4. Identification of Outliers:

Visualizations can highlight outliers or anomalies in the data, making it easier to detect errors or unusual patterns that might require further investigation.

5. Exploratory Data Analysis (EDA):

Visualization libraries are crucial for EDA, allowing data scientists to explore the data, generate hypotheses, and identify relationships between variables.

6. Storytelling:

Visualizations can be used to tell a compelling data-driven story. By creating a sequence of visualizations, you can guide your audience through a narrative, leading to a better understanding of the data and its implications.

Types of Visualization Libraries

HISTOGRAM

A histogram is a graphical representation of the distribution of a dataset. It is an estimate of the probability distribution of a continuous variable. To construct a histogram, the first step is to "bin" the range of values—that is, divide the

entire range of values into a series of intervals—and then count how many values fall into each interval. The bins are usually specified as consecutive, non-overlapping intervals of a variable. The bins (intervals) must be adjacent and are often (but not necessarily) of equal size.

1. **Data Distribution:**

   Histograms provide a clear visual summary of the distribution of the data. You can easily see whether the data is symmetric, skewed, unimodal, bimodal, or multimodal.

2. **Central Tendency:**

   Histograms can help you identify the central tendency of the data, such as the mean, median, and mode. For example, in a symmetric distribution, these measures tend to be around the center of the histogram.

3. **Variability:**

   You can assess the variability or spread of the data. A wide histogram indicates high variability, while a narrow histogram indicates low variability.

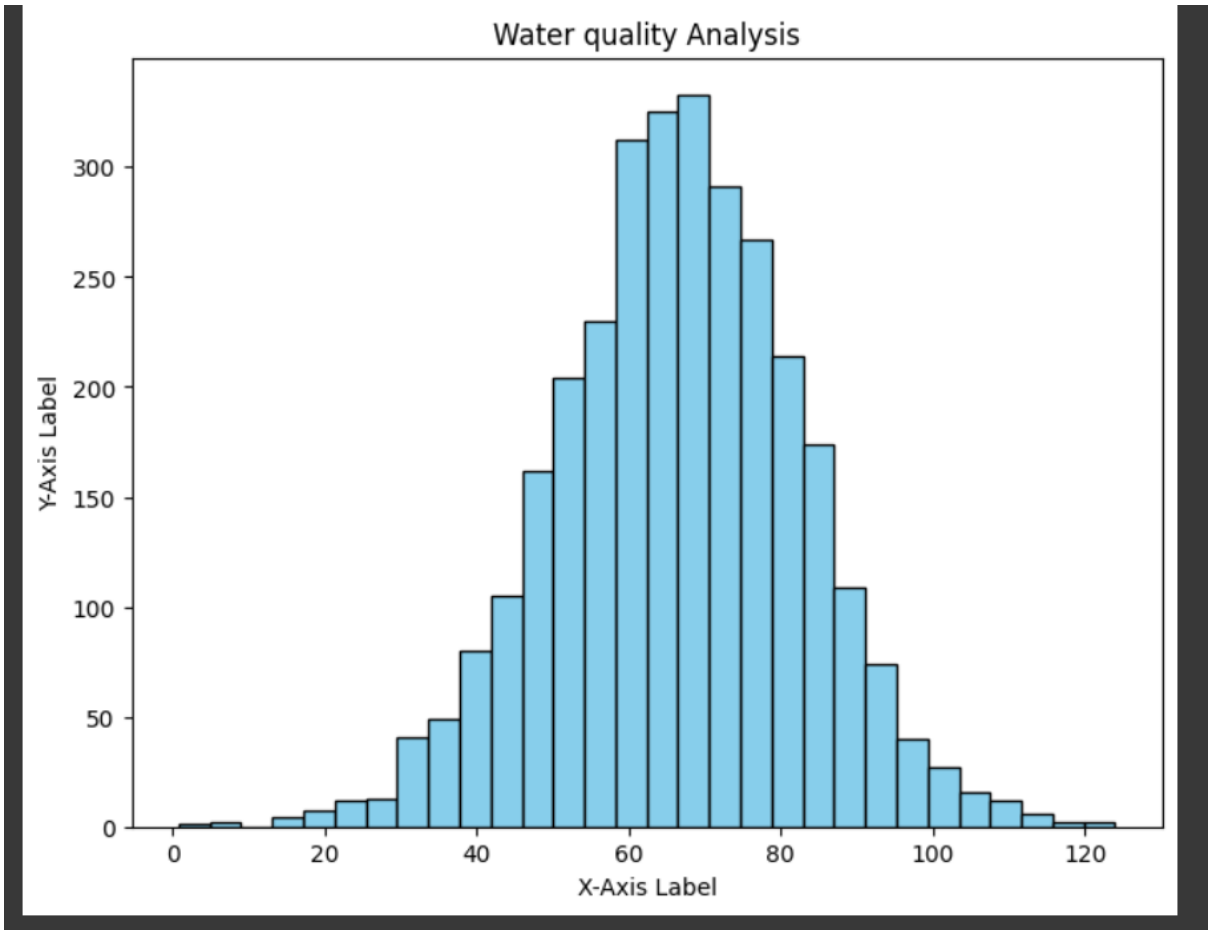
4. **Shape:**

   Histograms can reveal the shape of the data distribution. Common shapes include normal (bell-shaped)

CODE:

```
import pandas as pd import matplotlib.pyplot as plt

import seaborn as sns

data = pd.read_csv("water.zip")

plt.figure(figsize=(8, 6))

plt.hist(data['Turbidity'], bins=30, color='green',
edgecolor='blue')

 plt.title('Turbidity')

plt.xlabel('X-axis label')

 plt.ylabel('Y-axis label')

 plt.show()
```

OUTPUT:



SCATTER PLOTS

It is a popular type of data visualization that uses Cartesian coordinates to display values for two variables. Each data point is represented as a dot, allowing you to observe the relationship between the two variables. Here are some advantages of using scatter plots in data analysis:

1. Visualizing Relationships:

Scatter plots help you understand the relationship between two variables. You can quickly identify patterns such as linear or non-linear correlations, clusters, or outliers.

2. Identifying Correlations:

Scatter plots are excellent for identifying correlations between variables. Positive correlations (variables increase together), negative correlations (one variable increases while the other decreases), or the absence of correlation can be easily spotted.

3. Outlier Detection:

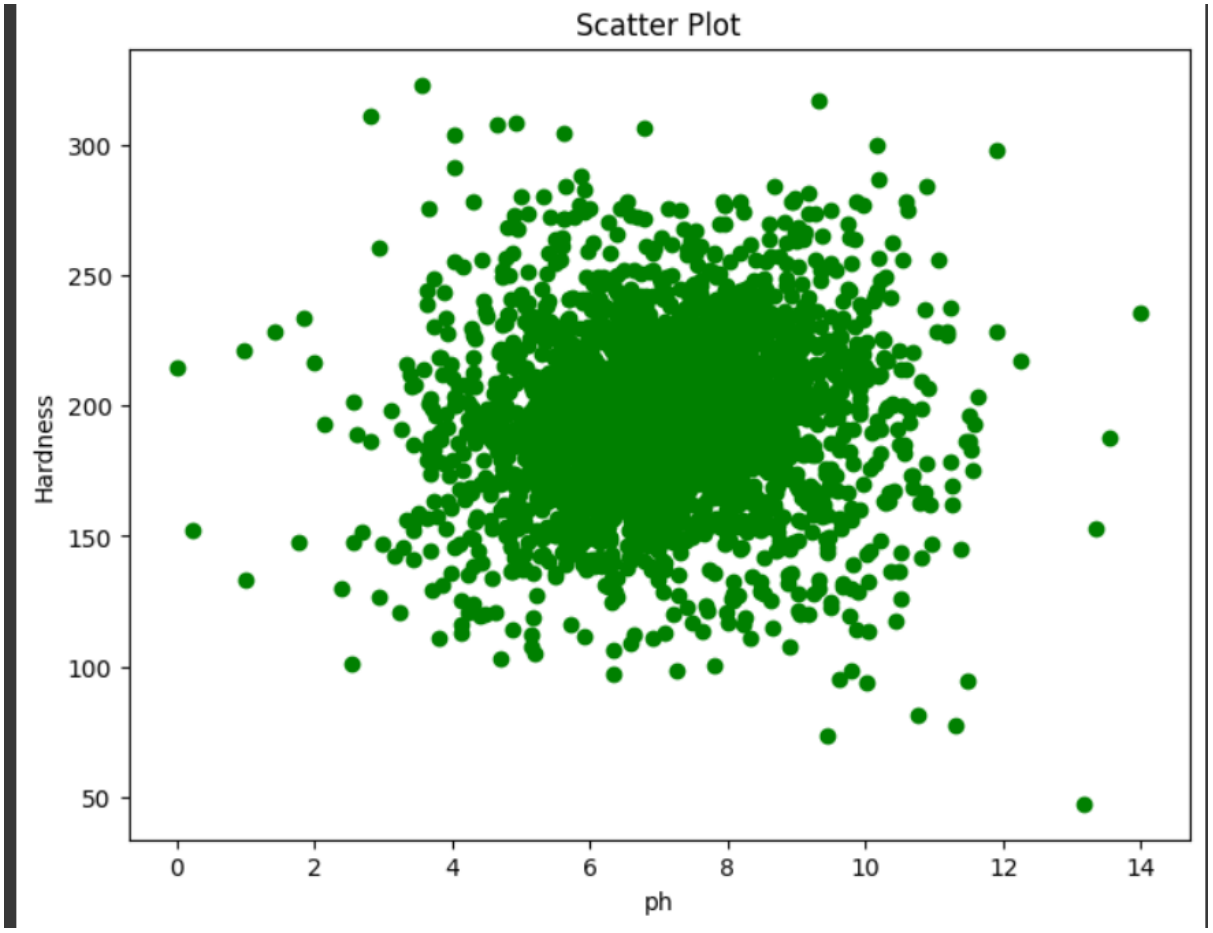
Outliers, or data points that significantly differ from the rest of the data, can be visually identified on a scatter plot. Outliers might be errors in the data or represent significant events that need further investigation.

CODE:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv("water.zip")
plt.figure(figsize=(8, 6))
```

```
plt.scatter(data['ph'], data['Hardness'], color='green', marker='o')
plt.title('Scatter Plot')
 plt.xlabel('ph')
plt.ylabel('Hardness')
 plt.show()
```

OUTPUT:



CORRELATION MATRIX

Correlation matrices are used in data analysis and statistics for several important reasons

1. Understanding Relationships:

Correlation matrices show how strongly pairs of variables are related. A correlation close to 1 indicates a strong positive relationship, while a correlation close to -1 indicates a strong negative relationship. A correlation near 0 suggests a weak or no linear relationship between variables.

2. Feature Selection:

In machine learning and statistics, understanding correlations can help in feature selection. Highly correlated variables might provide redundant information. Identifying and removing highly correlated features can improve the performance of some machine learning algorithms and simplify the model interpretation.

3. Multicollinearity Detection:

Correlation matrices are used to identify multicollinearity in regression analysis. Multicollinearity occurs when two or more independent variables in a regression model are highly correlated, leading to unstable coefficient estimates. Identifying multicollinearity helps in making necessary adjustments to the regression model.

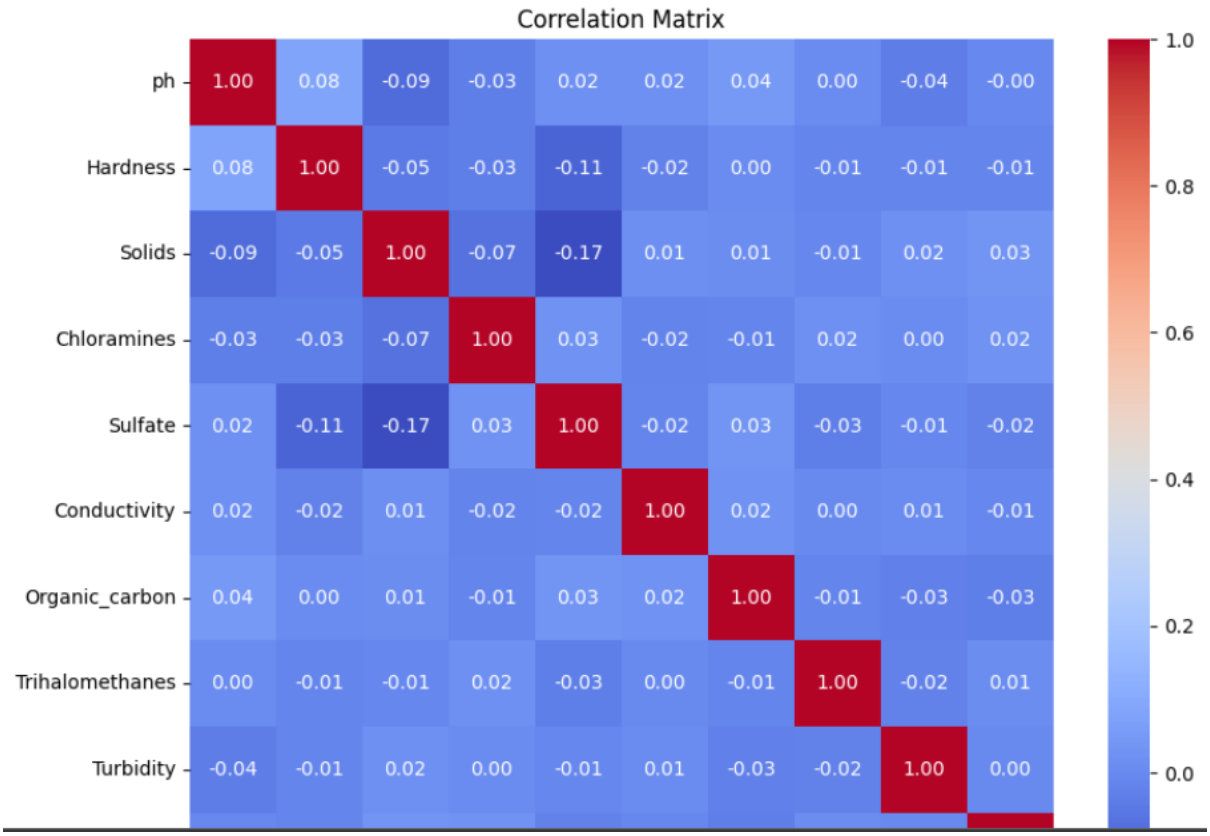
4.Data Cleaning:

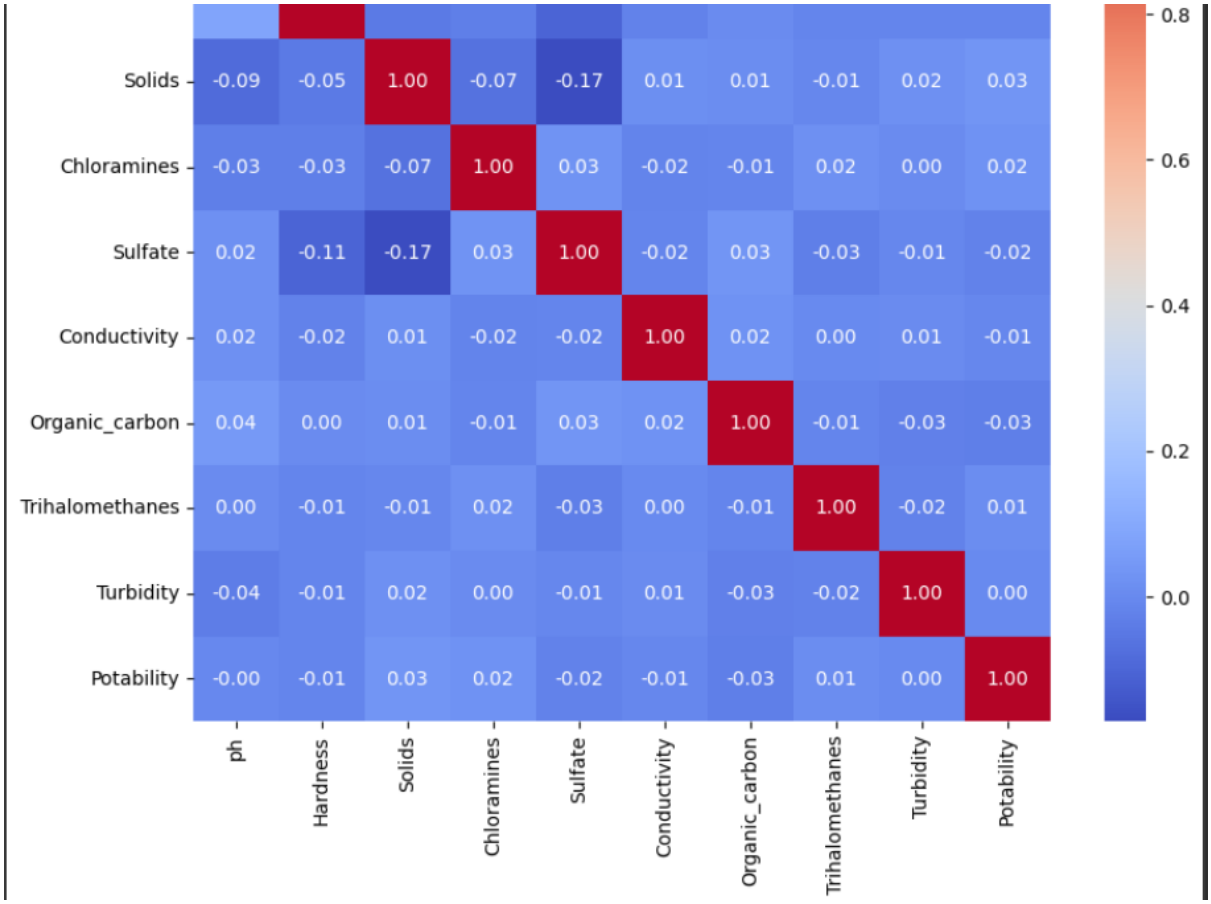
Correlation matrices can highlight potential data errors.

CODE:

```
 import pandas as pd
 import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv("water.zip")
plt.figure(figsize=(10, 8))
correlation_matrix = data.corr()
sns.heatmap(correlation_matrix, annot=True,
cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```

OUTPUT:





LOGISTIC REGRESSION

Logistic Regression is a statistical method used for binary classification problems, where the outcome variable is categorical and has two classes. It predicts the probability of an instance belonging to a particular category. Unlike Linear Regression, which predicts a continuous outcome, Logistic Regression models the probability that the given input belongs to a specific category.

How Logistic Regression Works:

1. Sigmoid Function (Logistic Function):

Logistic Regression uses the sigmoid function to map any real-valued number to the range of 0 and 1. The sigmoid function is defined as:

[sigma(z) = \frac{1}{1 + e^{-z}\]

where

(z = b_0 + b_1x_1 + b_2x_2 + ldots + b_nx_n)

(a linear combination of input features and their corresponding weights).

2. Probability Prediction:

The sigmoid function converts the linear combination of input features into a probability score between 0 and 1. If the probability is greater than or equal to 0.5, the instance is classified as the positive class; otherwise, it is classified as the negative class.

CODE:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
data=pd.read_csv("water.zip")
```

```
data_cleaned = data.dropna(axis=1)

X = data_cleaned.drop(['Potability'], axis=1)

y = data_cleaned['Potability']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

logistic_regression_model =
LogisticRegression(random_state=42)

logistic_regression_model.fit(X_train, y_train)

logistic_regression_predictions =
logistic_regression_model.predict(X_test)

logistic_regression_accuracy = accuracy_score(y_test,
logistic_regression_predictions)

print('Logistic Regression Accuracy:',
logistic_regression_accuracy)

print('Logistic Regression Classification Report:')

print(classification_report(y_test,
logistic_regression_predictions))# Initialize the Random
Forest Classifier model

random_forest_model =
RandomForestClassifier(random_state=42)

random_forest_model.fit(X_train, y_train)

random_forest_predictions =
random_forest_model.predict(X_test)

random_forest_accuracy = accuracy_score(y_test,
random_forest_predictions)

print('Random Forest Classifier Accuracy:',
random_forest_accuracy)

print('Random Forest Classifier Classification Report:')
```

```
print(classification_report(y_test,
random_forest_predictions))
```

OUTPUT:

```
 Logistic Regression Accuracy: 0.6280487804878049
Logistic Regression 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


  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-
packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this
behavior.
```

RANDOM FOREST CLASSFIER

Random Forest is an ensemble learning method used for both classification and regression tasks. It creates a "forest" of decision trees, where each tree is trained on a random subset of the dataset and makes its own prediction. The final prediction in classification tasks is determined by a majority vote from all the individual trees.

How Random Forest Works

1. Bootstrapping:

Random Forest builds multiple decision trees through a process called bootstrapping. It creates random subsets of the original dataset with replacement. Each subset is used to train a decision tree.

2. Feature Selection:

For each split in the decision tree, a random subset of features is considered. This randomness ensures that the individual trees are diverse and not highly correlated.

3. Decision Trees:

Each subset of data is used to train a decision tree. Decision trees in a Random Forest can grow deep and capture complex patterns in the data.

CODE:

```
from sklearn.ensemble importRandomForestClassifier
random_forest_model = RandomForestClassifier(random_state=42)
random_forest_model.fit(X_train, y_train)
random_forest_predictions = random_forest_model.predict(X_test)
random_forest_accuracy = accuracy_score(y_test, random_forest_predictions)
print('Random Forest Classifier Accuracy:', random_forest_accuracy)
print('Random Forest Classifier Classification Report:')
print(classification_report(y_test, random_forest_predictions))
```

OUTPUT:

```
Random Forest Classifier Accuracy: 0.6173780487804879
Random Forest Classifier Classification Report:
              precision    recall  f1-score   support

           0       0.65      0.83      0.73       412
           1       0.47      0.26      0.33       244

    accuracy                           0.62       656
   macro avg       0.56      0.54      0.53       656
weighted avg       0.59      0.62      0.58       656
```