# WATER QUALITY ANALYSIS

Project’s objectives:

The objective of this study is the assessment of physical, chemical and bacteriological quality of the drinking water. This document provides the procedures to be used to analyse the quality of water. This involves setting analysis objectives, gathering relevant chemicals data, creating informative visualizations, and extracting valuable insights from the data. Ultimately, the project seeks to provide a comprehensive understanding of side effects of poor water quality and provide effective solution for the same.

Design Thinking:

Monitoring of drinking water quality is an important component of water management, while data analysis is necessary for the identification and characterization of water quality problems. Assessment is the process by which water quality data is transformed into information. The information gained from monitoring is essential for analysing water quality.

Development Phases:

OVERVIEW OF THE PROCESS:

DATA COLLECTION: Collect water quality data from various sources. This data can include parameters like pH, turbidity, temperature, dissolved oxygen, chemical concentrations, and more. Data can be collected through sensors, manual measurements, or public datasets.

DATA CLEANING: Before loading the data into Cognos, you need to clean it. This involves handling missing values, correcting errors, and ensuring data consistency.

DATA LOADING: You can load the cleaned data into IBM Cognos by connecting to a database or importing data from a file. IBM Cognos typically supports various data sources and formats.

DATA PREPROCESSING: Data preprocessing is essential to prepare the data for analysis. Some common preprocessing steps for water quality data includes

**Normalization/Scaling:** Scale numerical features to a common range, e.g., between 0 and 1, to ensure that they have equal weight in the analysis

**Feature Engineering**: Create new features from existing data that may be more informative for your analysis, like calculating averages, sums, or ratios of certain parameters.

**Outlier Detection**: Identify and handle outliers in the data, as they can skew the analysis.

**Categorization**: Convert categorical data, like water source types or locations, into numerical values for analysis.

## DATA TRANSFORMATION: Depending on your analysis goals, you may need to transform the data. For water quality analysis, this might involve time-series aggregation, spatial aggregation, or any other transformation that helps in deriving meaningful insights.

## EXPLORATORY DATA ANALYSIS: EDA involves visualizing the data and running basic statistical analyses to gain a better understanding of the dataset. Use Cognos to create various charts, graphs, and dashboards to explore the data.

## FEATURE SECTION: Identify which features are most relevant for your water quality analysis. This can help reduce dimensionality and improve the efficiency of your analytics.

## MODEL BUILDING AND ANALYSIS: Use Cognos to build models for water quality prediction, classification, or clustering, depending on your specific objectives. You can choose from a variety of data analysis techniques, including regression, decision trees, clustering algorithms, and more.

## MODEL EVALUATION: Evaluate the performance of your models using appropriate metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), accuracy, precision, recall, or F1-score, depending on the type of analysis.

## DEPLOYMENT AND AUTOMATION: If the analysis needs to be automated for real-time monitoring, you can deploy the models and dashboards within IBM Cognos to ensure ongoing data analysis.

## MAINTENANCE: Regularly update and maintain your data analysis process, including refreshing data, retraining models, and adapting to changes in data quality or data sources.

DOCUMENTATION: Maintain comprehensive documentation of your work, including data sources, preprocessing steps, model details, and results.

PROGRAMMING:

import numpy as np *# linear algebra*

import pandas as pd *# data processing, CSV file I/O (e.g. pd.read\_csv)*

import os

for dirname, \_, filenames **in** os.walk('/kaggle/input'):

for filename **in** filenames:

print(os.path.join(dirname, filename))

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import warnings

warnings.filterwarnings('ignore')

data = pd.read\_csv('/kaggle/input/water-quality-testing/Water Quality Testing.csv')

data.head()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sample ID | pH | Temperature (°C) | Turbidity (NTU) | Dissolved Oxygen (mg/L) | Conductivity (µS/cm) |
| 1 | 7.25 | 23.1 | 4.5 | 7.8 | 342 |  |
| 2 | 7.11 | 22.3 | 5.1 | 6.2 | 335 |  |
| 3 | 7.03 | 21.5 | 3.9 | 8.3 | 356 |  |
| 4 | 7.38 | 22.9 | 3.2 | 9.5 | 327 |  |
| 5 | 7.45 | 20.7 | 3.8 | 8.1 | 352 |  |

data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 500 entries, 0 to 499

Data columns (total 6 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Sample ID 500 non-null int64

1 pH 500 non-null float64

2 Temperature (°C) 500 non-null float64

3 Turbidity (NTU) 500 non-null float64

4 Dissolved Oxygen (mg/L) 500 non-null float64

5 Conductivity (µS/cm) 500 non-null int64

dtypes: float64(4), int64(2)

memory usage: 23.6 KB

data.shape

(500, 6)

data.isnull().sum()

Sample ID 0

pH 0

Temperature (°C) 0

Turbidity (NTU) 0

Dissolved Oxygen (mg/L) 0

Conductivity (µS/cm) 0

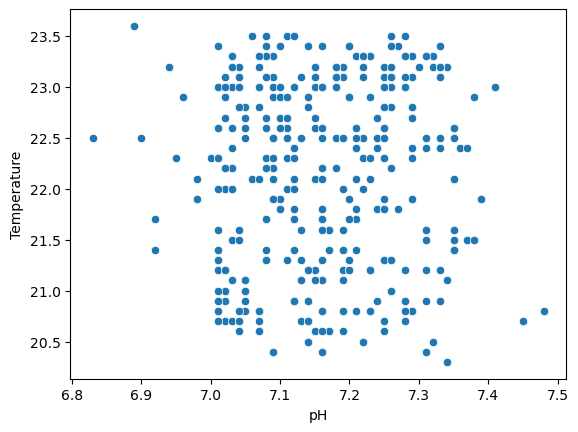
dtype: int64

data.head()

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sample ID | pH | Temperature (°C) | Turbidity (NTU) | Dissolved Oxygen (mg/L) | Conductivity (µS/cm) | s |
| 1 | 7.25 | 23.1 | 4.5 | 7.8 | 342 |  |
| 2 | 7.11 | 22.3 | 5.1 | 6.2 | 335 |  |
| 3 | 7.03 | 21.5 | 3.9 | 8.3 | 356 |  |
| 4 | 7.38 | 22.9 | 3.2 | 9.5 | 327 |  |
| 5 | 7.45 | 20.7 | 3.8 | 8.1 | 352 |  |

sns.scatterplot(data=data,x='pH',y='Temperature')

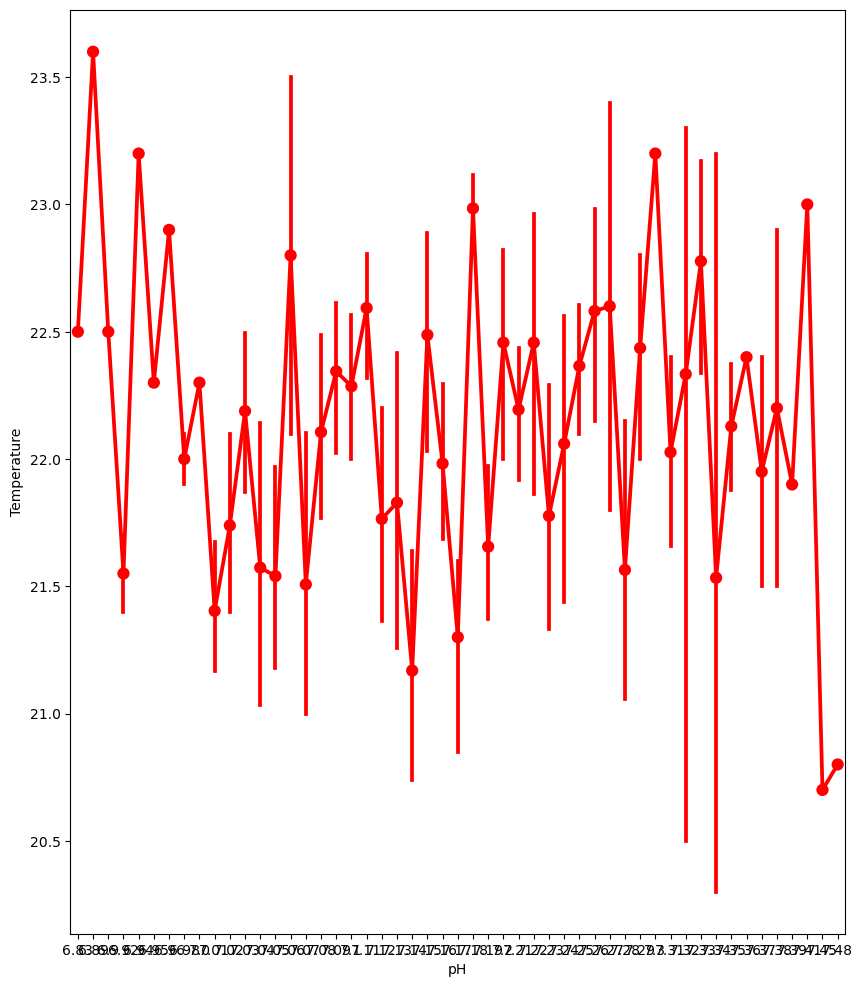
plt.show()



plt.figure(figsize=(10,12))

sns.pointplot(data=data,x='pH',y='Temperature',color='red')

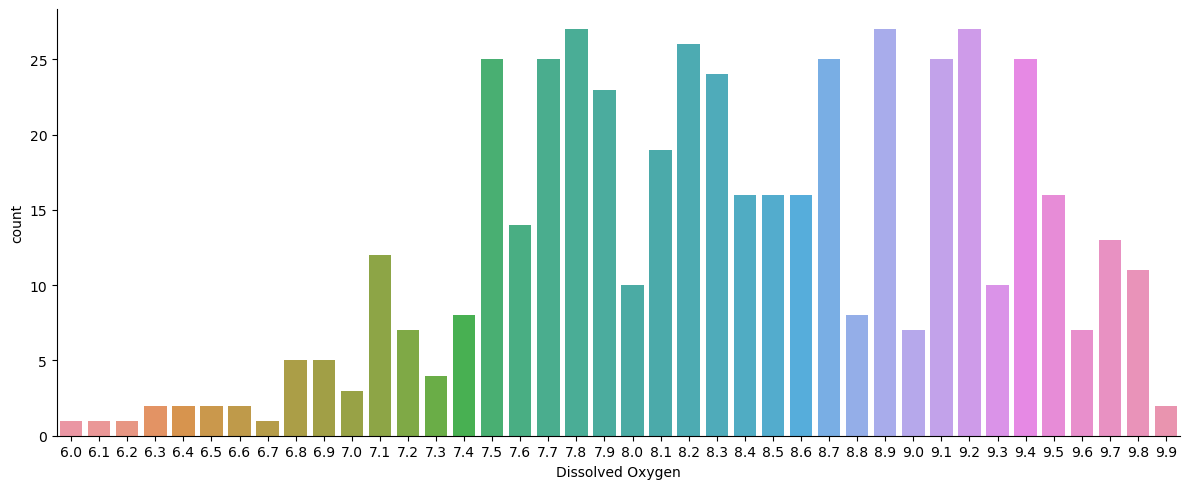
plt.show()



data['Dissolved Oxygen'].value\_counts()

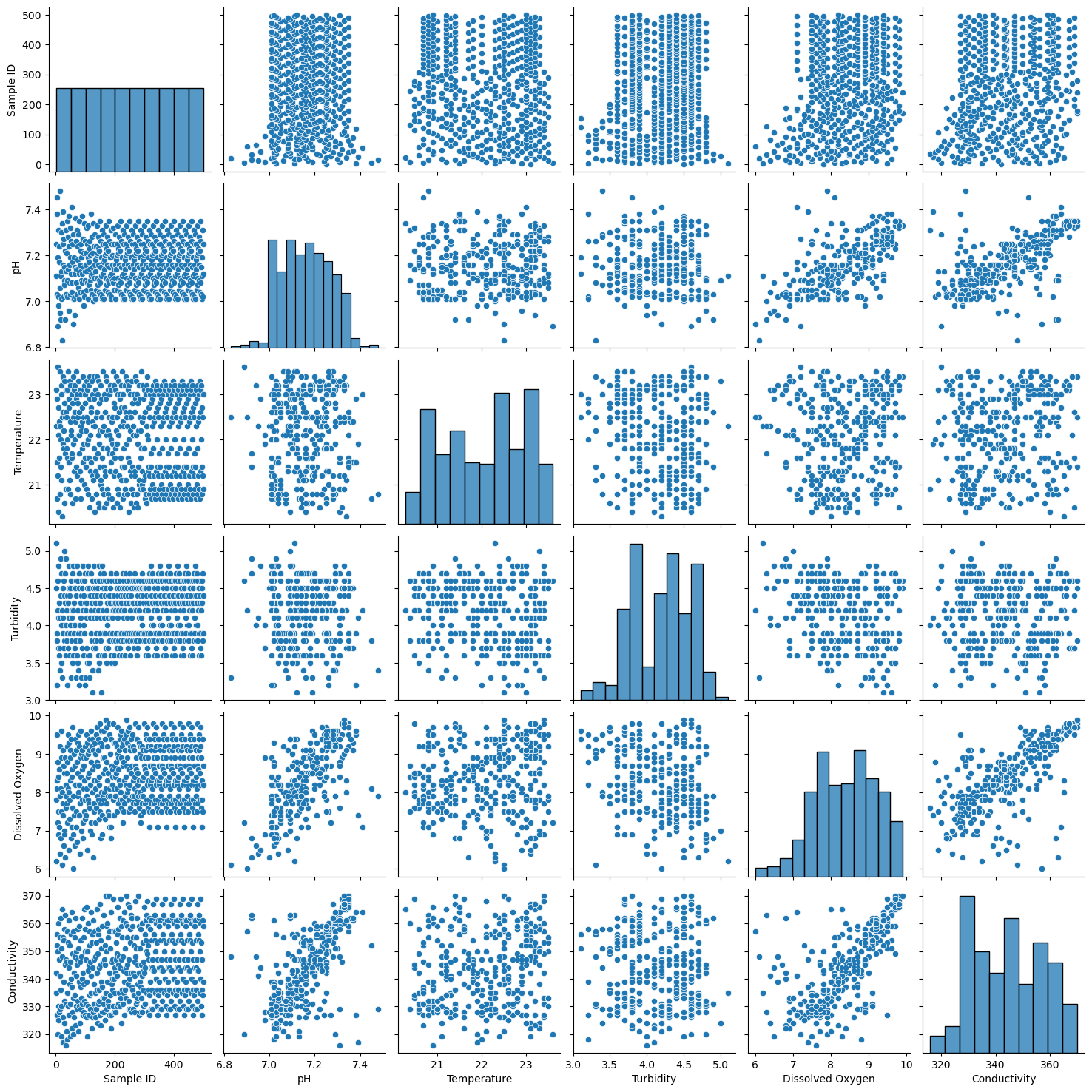
sns.catplot(data=data,x='Dissolved Oxygen',kind='count',aspect=2.4)

plt.show()



sns.pairplot(data)

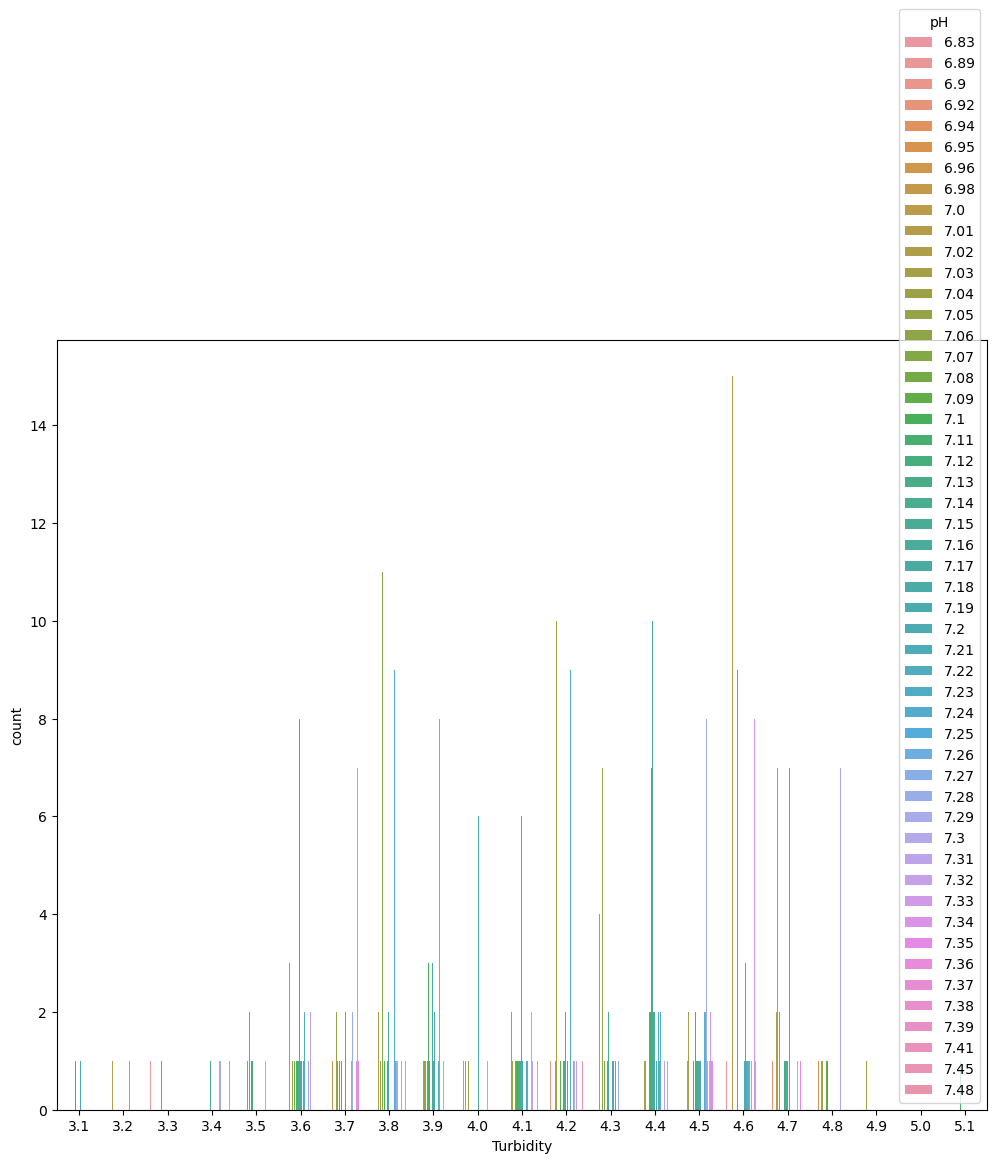
plt.show()



plt.figure(figsize=(12,10))

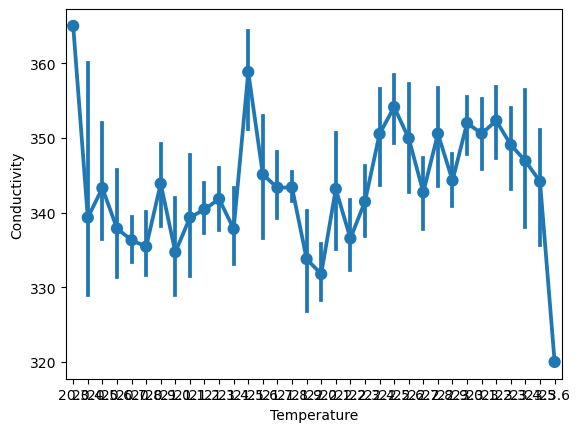
sns.countplot(data=data,x='Turbidity',hue='pH')

plt.show()



sns.pointplot(data=data,x='Temperature',y='Conductivity')

plt.show()



sns.lineplot(data=data,x='Temperature',y='Conductivity')

plt.show()

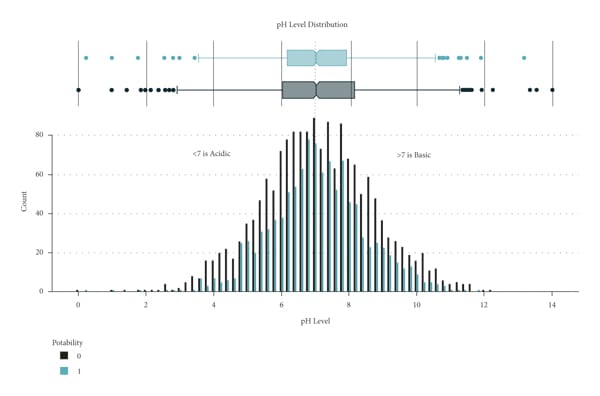
A graph showing a temperature

Description automatically generated

Predicting Water Potability Using a Machine Learning Model

Algorithm: Machine learning approaches were used to estimate the water potability in order to meet this aim. We used algorithms for both regression and classification. We employed the following algorithms in our research.

Decision Tree Classifier: The tree is a monitored form of learning which could be used to counteract obstacles, albeit it is most extensively adopted towards categorization.



In a pine classifier, nodes in the network carry collection traits, routes symbolize prior information, and then each node afords the inference.

AdaBoost Classifier: By turning a number of poor learners into strong learners, these methods boost prediction power. Boosting algorithms work on the idea of first building a model on the training dataset and then building a second model to correct the faults in the first model.

Gradient Boost Classifier: In contradiction with AdaBoost, the training context loads are not improved; however, every estimator is prepared by using presidency’s errors as symbols. Gradient Boost is a technique that includes Classification and Regression Tree (CART) as the concealer trainee.

Random Forest Classifier: Random Forest is a predictor that estimates the statistics of too many selections applied on discrete clades to optimize a set’s anticipated performance. Unlike the decision tree, which is prone to overfitting due to the biasing in the number of nodes in each branch, random forest uses bagging and boosting to combat overfitting and achieve higher accuracy.

A graph of different colored bars

Description automatically generated with medium confidence

Visualization Outputs:

Here are some common examples of visualization outputs using python libraries like matplotlib and seaborn:

import matplotlib.pyplot as plt

import seaborn as sns

# Example 1: Line Chart

x = [1, 2, 3, 4, 5]

y = [10, 15, 13, 17, 20]

plt.plot(x, y)

plt.xlabel('X-axis')

plt.ylabel('Y-axis')

plt.title('Line Chart Example')

plt.show()

# Example 2: Bar Chart

categories = ['A', 'B', 'C', 'D']

values = [30, 50, 25, 45]

sns.barplot(x=categories, y=values)

plt.xlabel('Categories')

plt.ylabel('Values')

plt.title('Bar Chart Example')

plt.show()

# Example 3: Scatter Plot

import numpy as np

x = np.random.rand(50)

y = np.random.rand(50)

plt.scatter(x, y)

plt.xlabel('X-axis')

plt.ylabel('Y-axis')

plt.title('Scatter Plot Example')

plt.show()

Correlation Matrix:

You can create a correlation matrix using the pandas and seaborn libraries to visualize the relationships between variables in a dataset:

import pandas as pd

import seaborn as sns

# Example Data

data = pd.DataFrame({

'A': [1, 2, 3, 4, 5],

'B': [2, 3, 4, 5, 6],

'C': [3, 4, 5, 6, 7]

})

# Calculate the correlation matrix

correlation\_matrix = data.corr()

# Create a heatmap

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm')

plt.title('Correlation Matrix')

plt.show()

Model Evaluation:

Here’s an example of model evaluation using a classification model in python, specifically with scikit-learn:

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

# Example Data

# Assume you have a dataset with features X and target variable y

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train a Random Forest Classifier

clf = RandomForestClassifier(n\_estimators=100, random\_state=42)

clf.fit(X\_train, y\_train)

# Make predictions

y\_pred = clf.predict(X\_test)

# Model Evaluation

accuracy = accuracy\_score(y\_test, y\_pred)

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

classification\_rep = classification\_report(y\_test, y\_pred)

print(f'Accuracy: {accuracy}')

print('Confusion Matrix:\n', conf\_matrix)

print('Classification Report:\n', classification\_rep)

This code example shows how to evaluate a classification model by calculating accuracy, generating a confusion matrix, and obtaining a classification report with precision, recall, and F1-score.

Conclusion:

Water quality analysis through data analytics is a crucial tool for monitoring, assessing, and improving water resources. By harnessing the power of data, we can identify patterns, detect contaminants, and make informed decisions to safeguard both environmental sustainability and public health. This approach enables us to better understand the complexities of water quality, and with ongoing advancements in data analytics, we are better equipped to address water quality challenges and ensure the availability of clean and safe water for our communities and the planet.