**Phase:4 DEVELOPMENT 2**

**INTRODUCTION**

Data Collection: Gather water quality data from relevant sources. This data may include parameters like pH, turbidity, dissolved oxygen, nutrient levels, and contaminants.

Data Preprocessing: Clean and preprocess the data. This involves handling missing values, outliers, and ensuring data is in a suitable format for analysis.

Feature Selection: Identify which water quality parameters are most relevant to your analysis. Feature selection helps in reducing dimensionality and improving model performance.

Model Selection: Choose an appropriate machine learning or statistical model for water quality analysis. Common models include regression, decision trees, random forests, and neural networks.

Data Split: Split your dataset into training and testing sets to train and evaluate the model's performance.

Model Training: Train your selected model using the training dataset. Tune hyperparameters to optimize model performance if necessary.

Model Evaluation: Evaluate your model using the testing dataset. Common evaluation metrics for regression tasks include Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared.

Visualization: Visualize the results to gain insights and interpret the model's performance. Create graphs, charts, or maps to display water quality trends and predictions.

Model Deployment: If the model performs well, you can deploy it for real-time water quality monitoring. This might involve setting up data pipelines to feed real-time data into the model.

Continuous Monitoring: Continuously monitor the model's performance and retrain it as new data becomes available to keep it up to date.

**Program:**

# Import necessary libraries

import pandas as pd

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

from sklearn.linear\_model import LinearRegression # Example model, choose the appropriate one

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

import matplotlib.pyplot as plt

# Load and preprocess your water quality data

data = pd.read\_csv("water\_quality\_data.csv") # Load your dataset

# Data preprocessing (e.g., handling missing values, feature selection)

# Split the data into training and testing sets

X = data[['Feature1', 'Feature2', ...]] # Select relevant features

y = data['Target'] # Target variable

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

# Choose and train a model (e.g., Linear Regression)

model = LinearRegression()

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

# Make predictions on the test data

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

# Evaluate the model

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f"Mean Absolute Error: {mae}")

print(f"Mean Squared Error: {mse}")

print(f"R-squared: {r2}")

# Visualization (e.g., scatter plot of observed vs. predicted values)

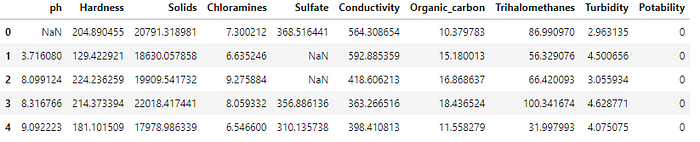
plt.scatter(y\_test, y\_pred)

plt.xlabel("Observed")

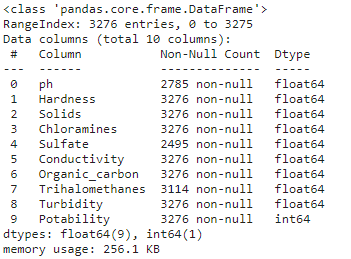
plt.ylabel("Predicted")

plt.show()

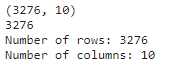
Output:



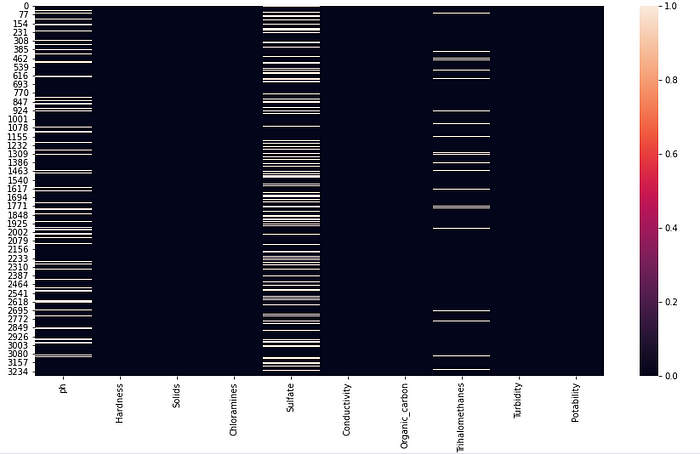
df.info(memory\_usage="deep")



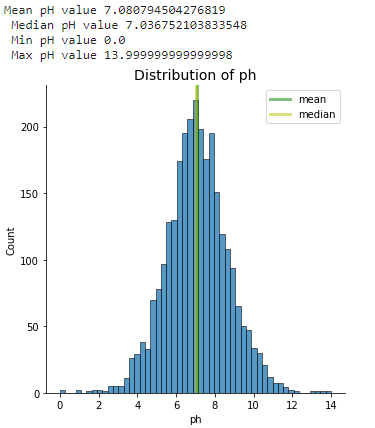
# Shape of the DataFrame - shows tuple of (#Rows, #Columns)  
print(df.shape)  
# Find the number of rows within a DataFrame  
print(len(df))  
# Extracting information from the shape tuple  
print(f'Number of rows: {df.shape[0]} \nNumber of columns: {df.shape[1]}')



plt.figure(figsize=(15,8))  
sns.heatmap(df.isnull());



sns.displot(df["ph"], kde=False)  
plt.axvline(x=df.ph.mean(), linewidth=3, color='g', label="mean", alpha=0.5)  
plt.axvline(x=df.ph.median(), linewidth=3, color='y', label="median", alpha=0.5)  
  
# set title, legends and labels  
plt.xlabel("ph")  
plt.ylabel("Count")  
plt.title("Distribution of ph", size=14)  
plt.legend(["mean", "median"]);  
  
print(f'Mean pH value {df.ph.mean()}   
 \n Median pH value {df.ph.median()}   
 \n Min pH value {df.ph.min()}   
 \n Max pH value {df.ph.max()}')



**Conclusion**

Finally, we were able to review the histogram of the pH variable to ensure that the variable followed external expectations.