**Phase 3: Development part 1**

**Project Title: Water Quality Analysis**

**Problem :**

This 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 preprocessing and loading the dataset.

**Definition:**

 The ph regulatory standards and determine water potability based on various parameters. This project includes defining analysis objectives, collecting water quality data, designing relevant visualizations, and import numpy as np .

**Preprocessing and loading**:

Preprocessing and loading a dataset is a crucial step in many data science and machine learning tasks. The specific steps can vary depending on the type of data and your goals, but I'll provide a general outline of the process. Here are the steps for preprocessing and loading a dataset:

**1.Data Collection**:

• Gather the dataset from various sources, such as databases, APIs, online repositories, or manual data entry.

**2. Import Libraries**:

• Import the necessary libraries for data manipulation and analysis, such as NumPy, pandas, and any specific libraries for your task.

**3. Load the Dataset:**

• Use library-specific functions to load the dataset into your programming environment. For example:

• For CSV files: pd.read\_csv("data.csv")

• For Excel files: pd.read\_excel("data.xlsx")

• For SQL databases: Use database connectors or libraries like SQLAlchemy.

**4. Exploratory Data Analysis (EDA):**

• Perform initial data exploration to understand the dataset's structure, variables, and relationships.

* This might include using functions like head(), info(), and describe() in pandas.

**5. Data Cleaning:**

* Identify and handle missing data. You can use methods like isna(), fillna(), or dropna() in pandas.

• Deal with duplicates, outliers, and inconsistent data.

**6. Data Transformation**:

• Convert data types if needed (e.g., converting a column from a string to a numeric type).

• Encode categorical variables (e.g., one-hot encoding or label encoding).

• Normalize or scale numeric features if necessary.

**7. Feature Engineering (optional):**

* Create new features based on existing ones to improve the performance of your machine learning models.

**8. Data Splitting:**

• Split the dataset into training, validation, and testing sets.

* The typical split is something like 70% for training, 15% for validation, and 15% for testing.

**9. Data Loading for Machine Learning:**

* Depending on the machine learning framework you're using (e.g., scikit-learn, TensorFlow, PyTorch), you may need to convert your data into the appropriate format.
* For scikit-learn, this typically involves separating the features and labels.

**10. Data Scaling and Normalization**:

* Depending on the algorithm you plan to use, you might need to scale or normalize the features.
* For example, you can use StandardScaler in scikit-learn.

**11. Data Pipeline (optional):**

• Create a data processing pipeline to automate the preprocessing steps and ensure consistency.

**12. Save Preprocessed Data (optional):**

• If you want to save the preprocessed dataset for later use, you can save it to a file (e.g., CSV) using library-specific functions.

**13. Model Training:**

• Use the preprocessed dataset to train your machine learning model.

**14. Model Evaluation and Tuning:**

• Evaluate the model's performance, and if necessary, fine-tune it based on the evaluation results.

**15. Model Deployment (if applicable):**

• If you're building a production system, deploy your model with the necessary preprocessing steps in place.

**16. Documentation:**

* Keep track of all the preprocessing steps and transformations you've applied to the dataset.
* Documentation is crucial for reproducibility.

**Python Script:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

plt.style.use('fivethirtyeight')

plt.style.use('dark\_background')

import seaborn as sns

color = sns.color\_palette()

import plotly.express as ex

import plotly.graph\_objs as go

import plotly.offline as pyo

import scipy.stats as stats

import pymc3 as pm

import theano.tensor as tt

from matplotlib.colors import ListedColormap

from scipy.stats import norm, boxcox

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

from collections import Counter

from scipy import stats

from tqdm import tqdm\_notebook

from sklearn import metrics

from sklearn.preprocessing import StandardScaler

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

from sklearn.model\_selection import (GridSearchCV, KFold, train\_test\_split, cross\_val\_score)

from imblearn.over\_sampling import SMOTE

from collections import Counter

from sklearn.linear\_model import LogisticRegression

from sklearn.naive\_bayes import GaussianNB

from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier

from sklearn import svm

from xgboost.sklearn import XGBClassifier

from sklearn.tree import DecisionTreeClassifier

# from catboost import CatBoostClassifier

# Importing The Dataset:

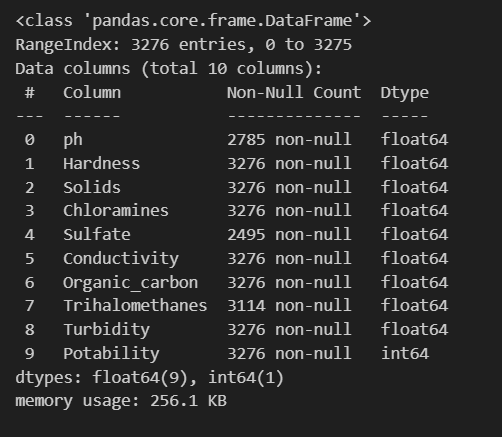
path = "water\_potability.csv"

df = pd.read\_csv(path)

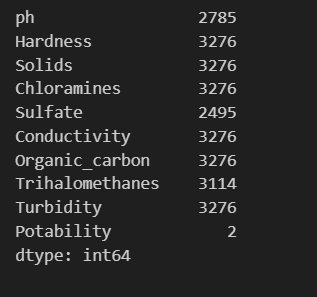
# Initial Analysis

df.shape

df.info()



df.nunique()



df['Potability']=df['Potability'].astype('category')

#Statistical Analysis

df.describe().T.style.background\_gradient(subset=['mean','std','50%','count'], cmap='PuBu')

#Portability is 1 - means good for Human

df[df['Potability']==1].describe().T.style.background\_gradient(subset=['mean','std','50%','count'], cmap='PuBu')

# Portability is 0 - means not good for Human

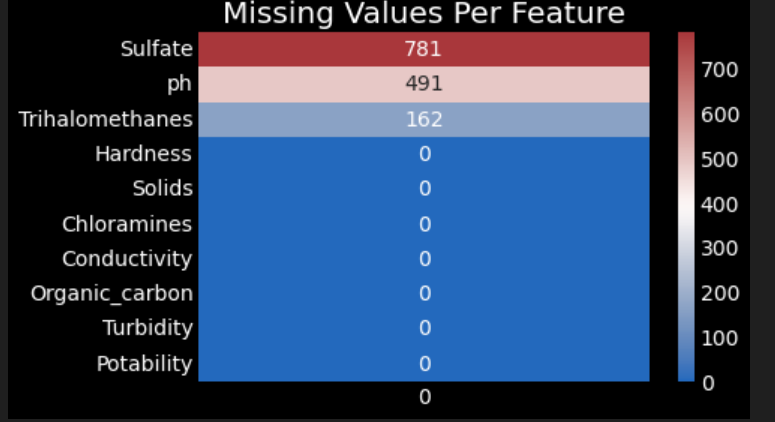
df[df['Potability']==0].describe().T.style.background\_gradient(subset=['mean','std','50%','count'], cmap='RdBu')

# Check for missing values

plt.title('Missing Values Per Feature')

nans = df.isna().sum().sort\_values(ascending=False).to\_frame()

sns.heatmap(nans,annot=True,fmt='d',cmap='vlag')



df[df['Sulfate'].isnull()]

df[df['ph'].isnull()]

df[df['Trihalomethanes'].isnull()]

Imputing the missing values with the mean

#Inputing ph value

phMean\_0 = df[df['Potability'] == 0]['ph'].mean(skipna=True)

df.loc[(df['Potability'] == 0) & (df['ph'].isna()), 'ph'] = phMean\_0

phMean\_1 = df[df['Potability'] == 1]['ph'].mean(skipna=True)

df.loc[(df['Potability'] == 1) & (df['ph'].isna()), 'ph'] = phMean\_1

#Imputing 'Sulfate' value

SulfateMean\_0 = df[df['Potability'] == 0]['Sulfate'].mean(skipna=True)

df.loc[(df['Potability'] == 0) & (df['Sulfate'].isna()), 'Sulfate'] = SulfateMean\_0

SulfateMean\_1 = df[df['Potability'] == 1]['Sulfate'].mean(skipna=True)

df.loc[(df['Potability'] == 1) & (df['Sulfate'].isna()), 'Sulfate'] = SulfateMean\_1

#Imputing 'Trihalomethanes' value

TrihalomethanesMean\_0 = df[df['Potability'] == 0]['Trihalomethanes'].mean(skipna=True)

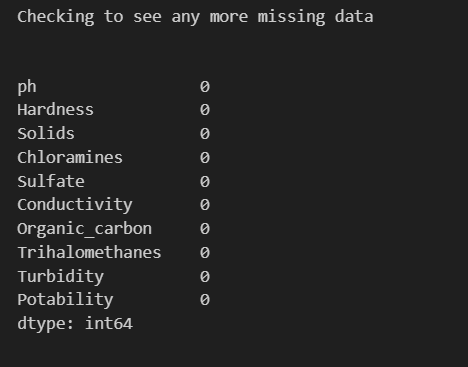
df.loc[(df['Potability'] == 0) & (df['Trihalomethanes'].isna()), 'Trihalomethanes'] = TrihalomethanesMean\_0

TrihalomethanesMean\_1 = df[df['Potability'] == 1]['Trihalomethanes'].mean(skipna=True)

df.loc[(df['Potability'] == 1) & (df['Trihalomethanes'].isna()), 'Trihalomethanes'] = TrihalomethanesMean\_1

print('Checking to see any more missing data \n')

df.isna().sum()

****

**Conclusion:**

* In conclusion, preprocessing and loading a dataset are fundamental steps in any data science or machine learning project.
* These steps ensure that your data is in a usable format and that it meets the requirements of your analysis or modeling tasks.
* By following the outlined steps, you can clean, transform, and prepare your data for further analysis or model training, ultimately leading to more accurate and meaningful insights..