Access to Safe drinking water is eassently a global issue. The World Health Organization (WHO) estimates that half of all people in the world are affected by the lack of safe drinking water. With this assesment, we will explore the data and look for patterns in the data to analyze if the given data is a good indicator of safe drinking water.

In [20]:

*# Import all required libraries*

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

**import** **seaborn** **as** **sns**

**from** **scipy** **import** stats

np.warnings.filterwarnings('ignore', category=np.VisibleDeprecationWarning)

*#sns.set\_context('notebook')*

**Data Set**

The dataset is downloaded from kaggle.com and is available for download at:

<https://www.kaggle.com/adityakadiwal/water-potability>

**EDA - Exploratory Data Analysis**

In this section we will explore the data and look for patterns in the data to analyze if the given data is a good indicator of safe drinking water.

1) Describe the data

2) Visualize the data

3) Identify the missing values and fill them

4) Identify the outliers and remove them

5) Identify the categorical variables and encode them (if any)

6) Identify the numerical variables and perform basic statistical analysis

In [2]:

*# File is stored in github repository for easiness of access*

INPUT\_FILE\_PATH = “https://raw.githubusercontent.com/Mitalnmt/Lab1\_AIL303m/main/water\_potability.csv”

In [3]:

*# Read the csv file from the url*

df = pd.read\_csv(INPUT\_FILE\_PATH)

In [4]:

*# Print the first 5 rows of the dataframe*

display(df.head())

|  | **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 |

**More information about the data**

ph - 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.

Hardness - Hardness is a measure of the physical properties of the water. It is a measure of the ability of the water to support the roots and the leaves. The lower the hardness, the more support the roots and leaves can have.

Solids (Total dissolved solids - TDS) - TDS is a measure of the solids in the 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.

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.

Sulfate - Sulfate is a common disinfectant used in public water systems. Sulfate levels up to 2 milligrams per liter (mg/L or 2 parts per million (ppm)) are considered safe in drinking water.

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.

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.

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.

Turbidity - Turbidity is a measure of the water’s ability to absorb particulate matter. The lower the turbidity, the more it can absorb particulate matter.

Potability (Target variable) - Indicates if water is safe for human consumption where 1 means Potable and 0 means Not potable.

In [5]:

*# datatypes of the columns*

print(df.shape)

print(df.dtypes)

ph float64

Hardness float64

Solids float64

Chloramines float64

Sulfate float64

Conductivity float64

Organic\_carbon float64

Trihalomethanes float64

Turbidity float64

Potability int64

dtype: object

In [6]:

*# Describe the data*

display(df.describe(include='all'))

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]:

*# Check if there are any null columns*

df.isnull().sum()

Out[7]:

ph 491

Hardness 0

Solids 0

Chloramines 0

Sulfate 781

Conductivity 0

Organic\_carbon 0

Trihalomethanes 162

Turbidity 0

Potability 0

dtype: int64

In [8]:

*# Lets try to plot misisng values*

missing = df.isnull().sum()

missing = missing[missing > 0]

plt.figure(figsize=(8,4))

sns.barplot(x=missing.index, y=missing.values)

plt.title("Missing values per column")

plt.xticks(rotation=45)

plt.show()

**Analyze ph column**

In [9]:

*# for ph column*

*# set the histogram, mean and median*

col = "ph"

series = df[col].dropna()

plt.figure(figsize=(6,3.2))

# Giả sử bạn đang dùng Seaborn

# Nếu không, code sẽ chạy vào nhánh else của Matplotlib

USE\_SEABORN = True

if USE\_SEABORN:

sns.histplot(series, bins=40, kde=False)

else:

plt.hist(series, bins=40)

plt.axvline(series.mean(), color='red', linestyle='--', label=f"Mean = {series.mean():.2f}")

plt.axvline(series.median(), color='green', linestyle=':', label=f"Median = {series.median():.2f}")

plt.title("Distribution of pH")

plt.legend()

plt.show()

print(f"Skewness of {col}:", series.skew())

Based on the above data, we can impute ph with either mean or median. There is no skweness in the data.

**Analyze Sulfate column**

In [10]:

col = "ph"

series = df[col].dropna()

plt.figure(figsize=(6,3.2))

# Giả sử bạn đang dùng Seaborn

# Nếu không, code sẽ chạy vào nhánh else của Matplotlib

USE\_SEABORN = True

if USE\_SEABORN:

sns.histplot(series, bins=40, kde=False)

else:

plt.hist(series, bins=40)

plt.axvline(series.mean(), color='red', linestyle='--', label=f"Mean = {series.mean():.2f}")

plt.axvline(series.median(), color='green', linestyle=':', label=f"Median = {series.median():.2f}")

plt.title("Distribution of pH")

plt.legend()

plt.show()

print(f"Skewness of {col}:", series.skew())

Based on the above data, we can impute Sulphate with either mean or median.

**Analyze Trihalomethanes column**

In [11]:

col = "Trihalomethanes"

series = df[col].dropna()

plt.figure(figsize=(6,3.2))

# Giả sử bạn đang dùng Seaborn

USE\_SEABORN = True

if USE\_SEABORN:

sns.histplot(series, bins=40, kde=False)

else:

plt.hist(series, bins=40)

plt.axvline(series.mean(), color='red', linestyle='--', label=f"Mean = {series.mean():.2f}")

plt.axvline(series.median(), color='green', linestyle=':', label=f"Median = {series.median():.2f}")

plt.title("Distribution of Trihalomethanes")

plt.legend()

plt.show()

print(f"Skewness of {col}: {series.skew():.3f}")

Based on the above data, we can impute Trihalomethanes with either mean or median.

**Missing Value imputation**

**Missing values in ph column**

In [12]:

*# impute missing values with mean*

df\_imputed = df.copy()

for col in numeric\_cols:

df\_imputed[col] = df\_imputed[col].fillna(df\_imputed[col].mean()) # dùng mean

# nếu muốn dùng median: df\_imputed[col] = df\_imputed[col].fillna(df\_imputed[col].median())

print("Total NA after imputation:", df\_imputed.isna().sum().sum())

**Identify outliers in the data**

In [13]:

*# check outliers*

for col in numeric\_cols:

q1 = df\_imputed[col].quantile(0.25)

q3 = df\_imputed[col].quantile(0.75)

iqr = q3 - q1

lower, upper = q1 - 1.5 \* iqr, q3 + 1.5 \* iqr

outliers = df\_imputed[(df\_imputed[col] < lower) | (df\_imputed[col] > upper)]

print(f"{col}: {len(outliers)} outliers")

**Identify corrleation between variables**

In [14]:

# Compute correlation matrix

corr = df\_imputed.corr(numeric\_only=True)

# Plot heatmap

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

sns.heatmap(corr, cmap="coolwarm", center=0)

plt.title("Correlation Heatmap")

plt.show()

# Check categorical variables (if any)

categorical\_cols = [c for c in df\_imputed.columns if df\_imputed[c].dtype == 'O']

if not categorical\_cols:

print("There are no categorical variables in the dataset.")

else:

print("Categorical variables:", categorical\_cols)

There are no categorical variables in the dataset.

**Identify skweness in the data**

In [15]:

*# identify skewness*

skewness = df\_imputed[numeric\_cols].skew()

print(skewness)

*# Showing the skewed columns*

skewed\_cols = skewness[skewness.abs() > 1]

print("\nNumber of skewed columns:", len(skewed\_cols))

print("Skewed columns:", list(skewed\_cols.index))

Number of skewed columns : 0

Out[15]:

|  | **Skew** |
| --- | --- |

There are no skew in our data :)

**Lets see the distribution of Potability**

In [31]:

# distribution of Potability

print(df\_imputed["Potability"].value\_counts())

# optional: visualize

plt.figure(figsize=(5,3))

sns.countplot(x="Potability", data=df\_imputed)

plt.title("Distribution of Potability")

plt.xlabel("Potability (0 = Not drinkable, 1 = Drinkable)")

plt.ylabel("Count")

plt.show()

Out[31]:

0 1998

1 1278

Name: Potability, dtype: int64

**Feature Transformation**

In [16]:

*# print the dataframe head*

display(df\_imputed.head())

Out[16]:

|  | **ph** | **Hardness** | **Solids** | **Chloramines** | **Sulfate** | **Conductivity** | **Organic\_carbon** | **Trihalomethanes** | **Turbidity** | **Potability** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 7.080795 | 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 | 333.775777 | 592.885359 | 15.180013 | 56.329076 | 4.500656 | 0 |
| **2** | 8.099124 | 224.236259 | 19909.541732 | 9.275884 | 333.775777 | 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 |

In [17]:

*# Feature transformation*

*# scale the numeric columns*

# 1) Đọc lại data gốc để chắc chắn sạch

INPUT\_FILE\_PATH = "https://raw.githubusercontent.com/Mitalnmt/Lab1\_AIL303m/main/water\_potability.csv"

df\_raw = pd.read\_csv(INPUT\_FILE\_PATH)

# 2) Impute = MEAN (đề thường dùng mean)

num\_cols = ['ph','Hardness','Solids','Chloramines','Sulfate',

'Conductivity','Organic\_carbon','Trihalomethanes','Turbidity']

df\_imputed = df\_raw.copy()

for c in num\_cols:

df\_imputed[c] = df\_imputed[c].fillna(df\_imputed[c].mean()) # <-- dùng mean thay vì median

# 3) Scale CHỈ các cột numeric (theo đúng thứ tự)

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(df\_imputed[num\_cols])

# 4) Ghép lại dataframe theo đúng thứ tự cột, giữ Potability cuối

df\_transformed = pd.DataFrame(X\_scaled, columns=num\_cols, index=df\_imputed.index)

df\_transformed['Potability'] = df\_imputed['Potability'].astype(int)

In [18]:

*# After transformation print the dataframe head*

pd.set\_option("display.float\_format", "{:.6f}".format)

display(df\_transformed.head())

Out[18]:

|  | **ph** | **Hardness** | **Solids** | **Chloramines** | **Sulfate** | **Conductivity** | **Organic\_carbon** | **Trihalomethanes** | **Turbidity** | **Potability** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.000000 | 0.198981 | -0.011702 | 0.085492 | 1.043542 | 1.227178 | -0.854560 | 1.028759 | -0.935210 | 0 |
| **1** | -2.113014 | -1.696382 | -0.196962 | -0.249088 | 0.000000 | 1.473406 | 0.214093 | -0.502884 | 0.514449 | 0 |
| **2** | 0.639503 | 0.684850 | -0.087287 | 1.079558 | 0.000000 | -0.028251 | 0.590024 | 0.001189 | -0.847715 | 0 |
| **3** | 0.776180 | 0.437145 | 0.093483 | 0.467446 | 0.694190 | -0.505079 | 0.939076 | 1.695662 | 0.635242 | 0 |
| **4** | 1.263161 | -0.398477 | -0.252771 | -0.293690 | -0.710100 | -0.202262 | -0.592197 | -1.718287 | 0.113188 | 0 |

**Save the cleaned data**

In [30]:

# Tạo folder 'data' nếu chưa có

os.makedirs("data", exist\_ok=True)

# Lưu dữ liệu đã impute (chưa scale)

df\_imputed.to\_csv("data/water\_clean\_imputed.csv", index=False)

# Lưu dữ liệu đã scale

df\_transformed.to\_csv("data/water\_clean\_scaled.csv", index=False)

print("✅ Files saved into 'data' folder:")

print(" - data/water\_clean\_imputed.csv")

print(" - data/water\_clean\_scaled.csv")

**Hypothesis Testing**

We define a hypothesis to test in our data set

Hypothesis 1:

Null: Increase in pH is associated with increase in Solids

Alternate : No relataion between ph and Solids

In [29]:

# Giả sử df\_imputed là DataFrame đã được xử lý

ph = df\_imputed["ph"].dropna()

solids = df\_imputed["Solids"].dropna()

# Thực hiện Independent T-test

t\_stat, p\_val = stats.ttest\_ind(ph, solids)

print("T-test result:")

print("Statistic =", t\_stat, ", p-value =", p\_val)

# Diễn giải kết quả

alpha = 0.05

if p\_val < alpha:

print(f"Since p-value = {p\_val:.6f} < {alpha}, we reject the null hypothesis.")

else:

print(f"Since p-value = {p\_val:.6f} >= {alpha}, we fail to reject the null hypothesis.")

Out[29]:

Ttest\_indResult(statistic=-4.476932191647608, pvalue=7.705940306619221e-06)

the p value is less than 0.05 , so we are rejecting the null hypothesis at 5% significance level.

**Next Step in analyzing the data**

Write here

**Quality of data**

Write here

**Key findings**

Write here