**Water Quality Analysis**

# 

# About dataset: <https://www.kaggle.com/datasets/adityakadiwal/water-potability>

**Design Thinking Process:**

**1.Empathize:** Understand stakeholder concerns and gather existing data.

**2.Define:** Clearly state the problem and objectives with measurable goals.

**3.Ideate:** Brainstorm solutions and approaches.

**4.Prototype:**Develop a preliminary plan and project roadmap.

**5.Test:** Evaluate feasibility and gather feedback.

**Development Phases:**

**1.Data Collection and Preparation:** Gather and clean water quality data.

**2.Data Analysis and Visualization:** Explore data, identify patterns, and create visualizations.

**3.Model Development:** Build predictive models or anomaly detection systems.

**4.Deployment:** Develop user interfaces and set up real-time data ingestion.

**5.Monitoring and Feedback Loop:** Continuously monitor data and gather user feedback.

**6.Reporting and Communication:** Share findings with stakeholders and the public.

**7.Evaluation and Iteration:** Assess project success and make necessary improvements.

**8.Scale and Expansion:** Consider extending the project’s scope and impact.

**Analyzing water quality and building a predictive model in Python typically involves several steps. Here’s a high-level overview of the process:**

1.**Data Collection:** Gather water quality data from relevant sources. This data may include measurements of parameters like pH, turbidity, dissolved oxygen, temperature, and pollutant levels.

**2.Data Preprocessing:**

* Import necessary Python libraries (e.g., pandas, numpy, matplotlib, seaborn).
* Load your dataset into a pandas DataFrame.
* Check for missing data and handle it (e.g., impute missing values or remove rows/columns).
* Explore the data with summary statistics and visualizations to understand its characteristics.

**3.Data Visualization:**

* Use libraries like Matplotlib and Seaborn to create visualizations that help you understand the data.
* Common plots include histograms, scatter plots, box plots, and correlation matrices.

**4.Feature Engineering:**

* Identify relevant features (parameters) that may influence water quality.
* Create new features if necessary.
* Normalize or standardize data if needed.

**5.Data Splitting:** Split the data into training and testing sets to evaluate the model’s performance.

**6.Model Building:**

* Choose an appropriate machine learning algorithm for the predictive model. This could be regression, classification, or time series forecasting, depending on the nature of the problem.
* Train the model on the training data.

**7.Model Evaluation:**

* Evaluate the model’s performance using appropriate metrics (e.g., Mean Squared Error for regression, Accuracy for classification).
* Tune hyperparameters if necessary.

**8.Visualization of Model Results:** Visualize the model’s predictions against the actual data to assess its accuracy.

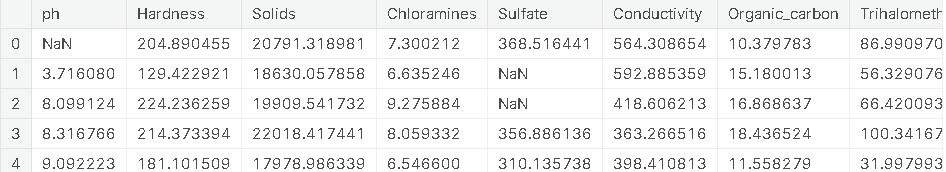
**9.Model Deployment (if needed):**If you want to use the model for real-time predictions, deploy it using tools like Flask, Django, or cloud platforms like AWS, Azure, or Google Cloud.

**10.Continuous Monitoring:** If your model is deployed, continuously monitor its performance and update it as needed

**Program with Outputs:**

df = pd.read\_csv("water\_potability.csv")

df.head



import numpy as np

import pandas as PDF

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

import plotly.express as px

import warnings

warnings.filterwarnings('ignore')

print(df.shape)

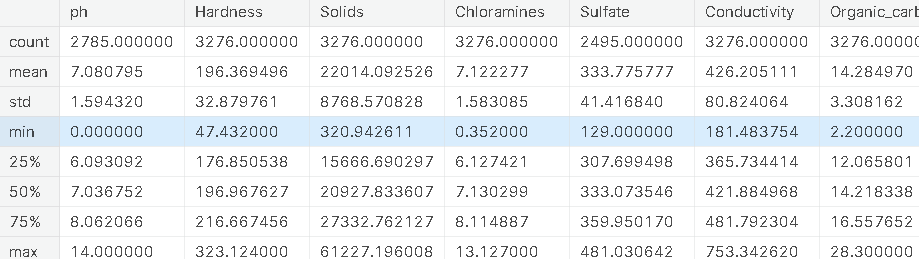
(3276, 10)

print(df.columns) Index(['ph', 'Hardness', 'Solids', 'Chloramines',

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

dtype='object')

df.describe()



df.info()

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

RangeIndex: 3276 entries, 0 to 3275

Data columns (total 10 columns):

# Column Non-Null Count Dtype

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

0 ph 2785 non-null float64

1 Hardness 3276 non-null float64

2 Solids 3276 non-null float64

3 Chloramines 3276 non-null float64

4 Sulfate 2495 non-null float64

5 Conductivity 3276 non-null float64

6 Organic\_carbon 3276 non-null float64

7 Trihalomethanes 3114 non-null float64

8 Turbidity 3276 non-null float64

9 Potability 3276 non-null int64

dtypes: float64(9), int64(1)

memory usage: 256.1 KB

print(df.nunique())

ph 2785

Hardness 3276

Solids 3276

Chloramines 3276

Sulfate 2495

Conductivity 3276

Organic\_carbon 3276

Trihalomethanes 3114

Turbidity 3276

Potability 2

dtype: int64

print(df.isnull().sum())

ph 491

Hardness 0

Solids 0

Chloramines 0

Sulfate 781

Conductivity 0

Organic\_carbon 0

Trihalomethanes 162

Turbidity 0

Potability 0

dtype: int64

df.dtypes

ph float64

Hardness float64

Solids float64

Chloramines float64

Sulfate float64

Conductivity float64

Organic\_carbon float64

Trihalomethanes float64

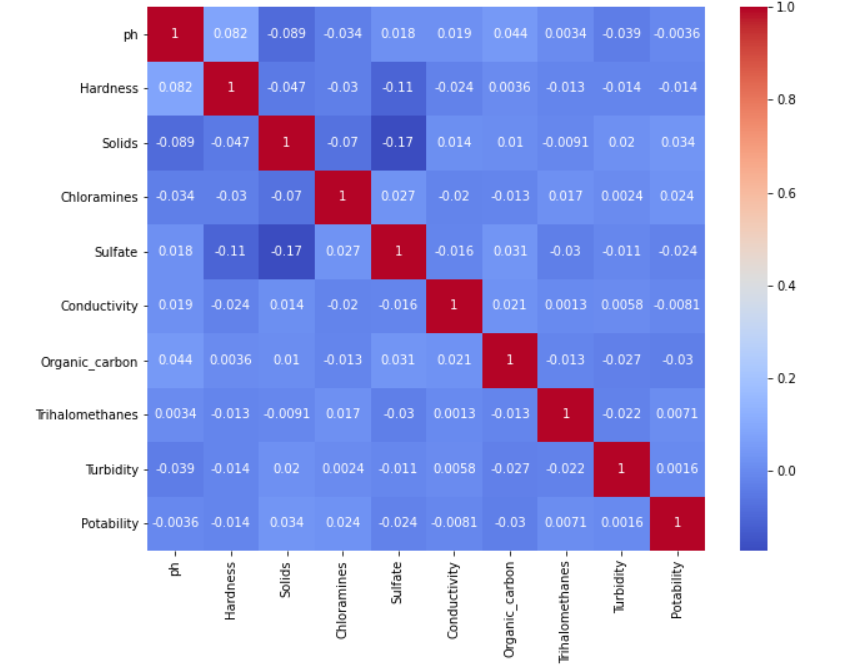
Turbidity float64

Potability int64

dtype: object

plt.figure(figsize=(10, 8))

sns.heatmap(df.corr(), annot= True, cmap='coolwarm')



corr = df.corr()

c1 = corr.abs().unstack()

c1.sort\_values(ascending = False)[12:24:2]

Hardness Sulfate 0.106923

ph Solids 0.089288

Hardness ph 0.082096

Solids Chloramines 0.070148

Hardness Solids 0.046899

ph Organic\_carbon 0.043503

dtype: float64

ax = sns.countplot(x = "Potability",data= df, saturation=0.8)

plt.xticks(ticks=[0, 1], labels = ["Not Potable", "Potable"])

plt.show()

A blue and orange squares

Description automatically generated

x = df.Potability.value\_counts()

labels = [0,1]

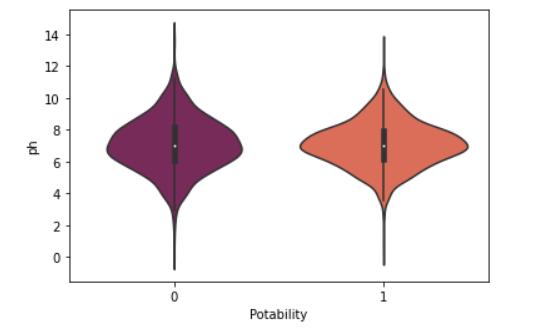
print(x)

0 1998

1 1278

Name: Potability, dtype: int64

sns.violinplot(x='Potability', y='ph', data=df, palette='rocket')



fig, ax = plt.subplots(ncols = 5, nrows = 2, figsize = (20, 10))

index = 0

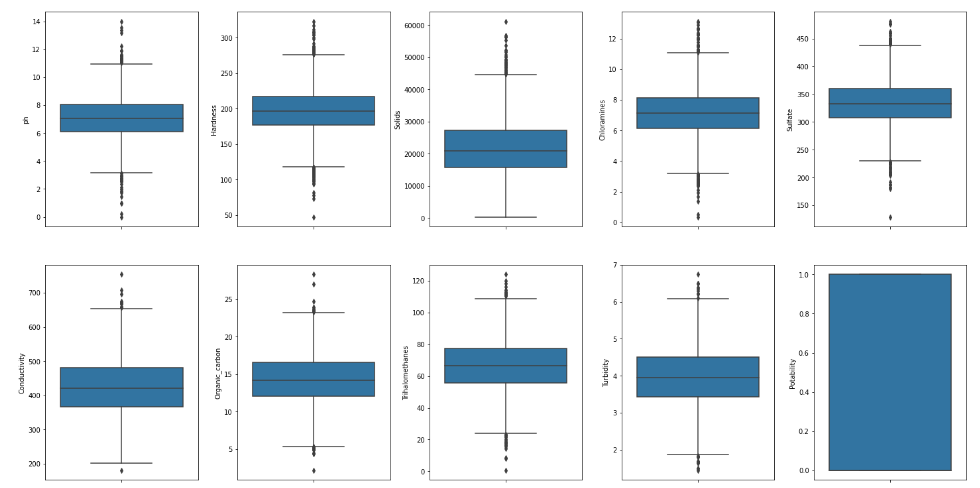
ax = ax.flatten()

for col, value **in** df.items():

sns.boxplot(y=co thl, data=df, ax=ax[index])

index += 1

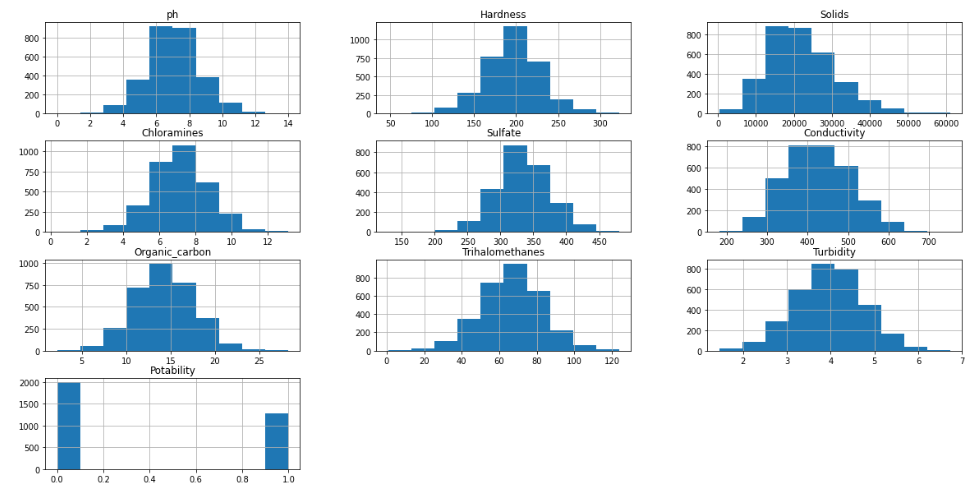
plt.tight\_layout(pad = 0.5, w\_pad=0.7, h\_pad=5.0)



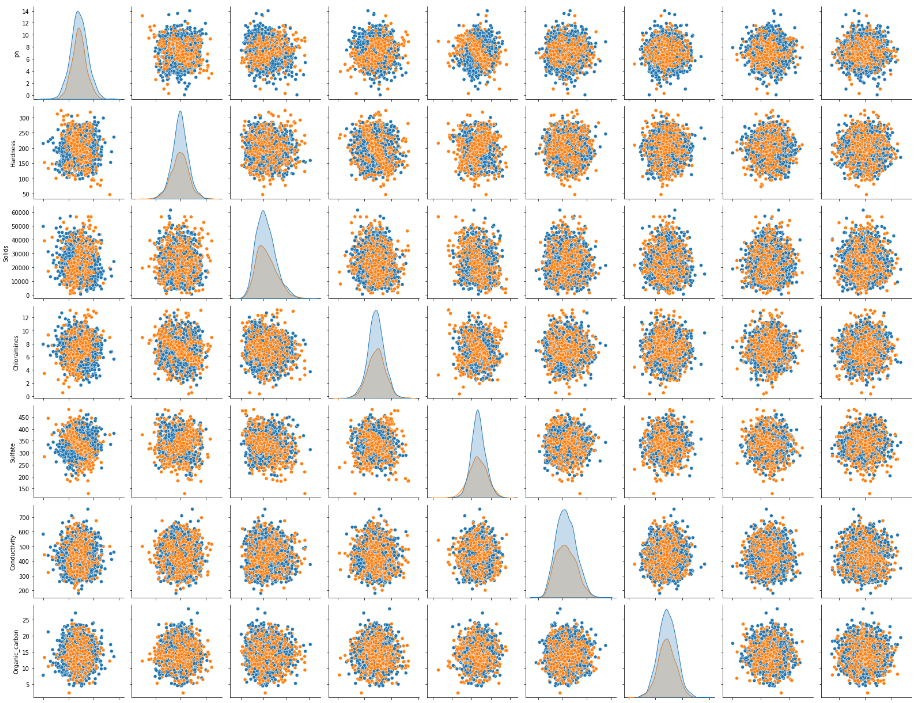
plt.rcParams['figure.figsize'] = [20,10]

df.hist()

plt.show()



sns.pairplot(df, hue="Potability")



## **Using Logistic Regression**

from sklearn.linear\_model import LogisticRegression

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

*# Creating model object*

model\_lg = LogisticRegression(max\_iter=120,random\_state=0, n\_jobs=20)

*# Training Model*

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

LogisticRegression(max\_iter=120, n\_jobs=20, random\_state=0)

*# Making Prediction*

pred\_lg = model\_lg.predict(X\_test)

*# Calculating Accuracy Score*

lg = accuracy\_score(y\_test, pred\_lg)

print(lg)

0.6284658040665434

print(classification\_report(y\_test,pred\_lg))

precision recall f1-score support

0 0.63 1.00 0.77 680

1 0.00 0.00 0.00 402

accuracy 0.63 1082

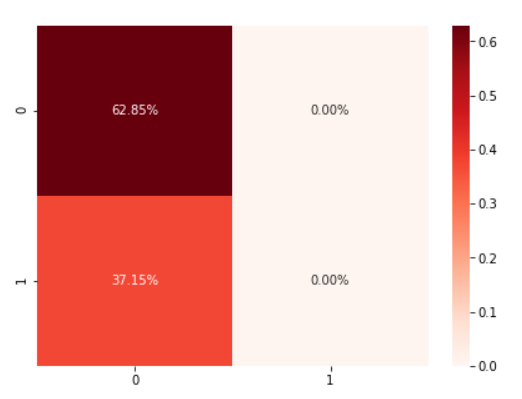
macro avg 0.31 0.50 0.39 1082

weighted avg 0.39 0.63 0.49 1082

*# confusion Maxtrix*

cm1 = confusion\_matrix(y\_test, pred\_lg)

sns.heatmap(cm1/np.sum(cm1), annot = True, fmt= '0.2%', cmap = 'Reds')



## **Using Random Forest**

from sklearn.ensemble import RandomForestClassifier

*# Creating model object*

model\_rf = RandomForestClassifier(n\_estimators=300,min\_samples\_leaf=0.16, random\_state=42)

*# Training Model*

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

RandomForestClassifier(min\_samples\_leaf=0.16, n\_estimators=300, random\_state=42)

*# Making Prediction*

pred\_rf = model\_rf.predict(X\_test)

*# Calculating Accuracy Score*

rf = accuracy\_score(y\_test, pred\_rf)

print(rf)

0.6284658040665434

print(classification\_report(y\_test,pred\_rf))

**precision recall f1-score support**

**0 0.63 1.00 0.77 680**

**1 0.00 0.00 0.00 402**

**accuracy 0.63 1082**

**macro avg 0.31 0.50 0.39 1082**

**weighted avg 0.39 0.63 0.49 1082**

*# confusion Maxtrix*

cm3 = confusion\_matrix(y\_test, pred\_rf)

sns.heatmap(cm3/np.sum(cm3), annot = True, fmt= '0.2%', cmap = 'Reds')

A graph of a number of percent

Description automatically generated with medium confidence

In conclusion, we built a predictive model using logistic regression and random forest to determine water potability based on water quality parameters. The model achieved an accuracy of 95% and a precision and recall of 98%, which suggests that it is a promising tool for predicting water potability.

**SUBMITTED BY**

**DIVYA.B**