DATA ANALYSIS WITH COGNOS

Group 2

Project 11 – WATER ANALYSIS



COLLEGE CODE:5113

TEAM 10

TEAM LEADER: INIYAN P – au511321104032

TEAM MATES:

KAMALESH L - au511321104039

DHANUSH M – au511321104021

MUKESH C - au511321104057

KUGAN K - au511321104044

WATER QUALITY ANALYSIS

Water quality analysis involves testing and evaluating the characteristics of water to determine its safety and suitability for various purposes. It includes assessing parameters such as pH, dissolved oxygen, turbidity, and the presence of contaminants like bacteria, chemicals, and heavy metals. This analysis helps in ensuring that water meets the required standards for drinking, industrial use, and environmental protection.

**Introduction:**

Being able to provide enough fresh drinking water is a core requirement. Within the climate change debate, one of the largest challenges is ensuring enough freshwater to survive.

Water quality is a big concern that impacts all the specifies. Only about three percent of Earth’s water is freshwater.

Of that, only 1.2 percent can be used as drinking water, with the remainder locked up in glaciers, ice caps, and permafrost, or

buried deep in the ground. Using a data-driven approach to assess the features that impact the water quality could greatly improve our understanding of what makes water drinkable.

We will seek to find hidden insights with data analysis techniques using pandas and numpy. For the data visualizations, the matplotlib and seaborn libraries will be used.

**Goal:**

The goal of water quality analysis using data analytics is to assess and monitor the quality of water in various environmental settings, such as natural bodies of water, industrial processes, drinking water supplies, and wastewater treatment systems. This analysis aims to ensure the safety of water for human consumption, protect aquatic ecosystems, and maintain overall environmental health. Data analytics plays a crucial role in achieving this goal by providing insights, patterns, and predictions from large datasets related to water quality.

**Necessary step to follow:**

**1.Import Libraries:**

Start by importing the necessary python libraries:

import numpy as np *# linear algebra*

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

import seaborn as sns

import matplotlib.pyplot as plt

import plotly.express as px

import missingno as msno

**2.Load the Dataset:**

Load your dataset into a Pandas DataFrame. Ensure that the data is in a format that Pandas can work with, such as CSV or Excel.

data = pd.read\_csv('water\_quality\_data.csv')

**Preprocessing the dataset**

Data preprocessing transforms the data into a format that is more easily and effectively processed in data mining, machine learning and other data science tasks. The techniques are generally used at the earliest stages of the machine learning and AI development pipeline to ensure accurate result

**Importance of preprocessing the dataset:**

Preprocessing of datasets in water analysis is of paramount importance as it plays a pivotal role in ensuring the accuracy and reliability of the results obtained from various water quality assessments. This critical step involves a series of data cleaning, transformation, and organization processes that help researchers and scientists eliminate errors, outliers, and inconsistencies in the data.

By carefully handling and preparing the data, analysts can enhance the precision of their measurements, leading to more meaningful interpretations of water quality indicators.

**VISUALISATION LIBRAIRES :**

Visualization libraries are tools that enable the creation of graphical representations of data for better understanding and interpretation. Here are explanations of some commonly used visualization libraries in Python:

**MATPLOTLIB**:

Primary Purpose: Matplotlib is one of the foundational libraries for creating static, interactive, and publication-quality visualizations in Python.

**Features:**

Provides a wide range of plots: line plots, scatter plots, bar charts, histograms, etc.

Highly customizable, allowing fine control over every aspect of a plot.

Offers subplots, axes, and figure objects for creating complex layouts.

Use Cases: Widely used for basic to intermediate-level plotting in scientific research, data analysis, and publication-quality figures.

**Seaborn:**

Primary Purpose: Seaborn is built on top of Matplotlib and offers a higher-level interface for statistical data visualization.

Features:

Simplifies the creation of more complex plots compared to Matplotlib.

Built-in themes and color palettes.

Specialized plots for statistical estimation and exploring data distributions.

Use Cases: Commonly used in data analysis and exploration, providing quick and easy creation of complex visualizations for statistical data.

**Plotly:**

Primary Purpose: Plotly is a web-based interactive visualization library known for creating interactive plots and dashboards.

Features:

Capable of generating interactive, web-based visualizations that can be embedded in web applications.

Offers a wide range of chart types, from basic to 3D visualizations.

Plotly Express, a high-level API, simplifies the creation of a variety of plot types.

**Use Cases:**

Ideal for creating interactive and visually appealing dashboards, especially for web applications and presentations.

**Primary Purpose**: Bokeh is another interactive visualization library designed for modern web browsers.

Features:

Focuses on interactivity and scalability, especially for big data.

Allows the creation of interactive and real-time plots with high-performance capabilities.

Provides tools for building complex dashboards and applications.

**Use Cases:**

Particularly useful for creating interactive visualizations that require smooth interactions and are meant to be presented in web applications.

**Primary Purpose:**

Altair is a declarative statistical visualization library based on Vega and Vega-Lite visualization grammars.

**Features:**

Emphasizes a concise and friendly API for creating high-level, statistically accurate visualizations.

Generates JSON specifications for visualizations compatible with various front-end tools.

**Use Cases:**

Ideal for creating simple and clear visualizations for data exploration and presentation.

Each library has its strengths and is suited to different use cases. The choice of library often depends on the nature of the data, the type of visualization required, interactivity needs, and the platform where the visualization will be presented.

**HISTOGRAM :**

**CODE :**

*import pandas as pd*

*import numpy as np*

*import matplotlib.pyplot as plt*

*import seaborn as sns*

*# Generating sample water quality data*

*np.random.seed(42)*

*data = {*

*"pH": np.random.normal(7, 0.5, 100),*

*"Chloride": np.random.uniform(5, 20, 100),*

*"Nitrates": np.random.uniform(0, 10, 100),*

*"Sulfates": np.random.normal(8, 2, 100),*

*"Quality": np.random.uniform(0, 100, 100),*

*}*

*water\_quality = pd.DataFrame(data)*

*# Histograms for each water quality parameter*

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

*plt.subplot(2, 3, 1)*

*sns.histplot(water\_quality["pH"], kde=True)*

*plt.title("pH Distribution")*

*plt.subplot(2, 3, 2)*

*sns.histplot(water\_quality["Chloride"], kde=True)*

*plt.title("Chloride Distribution")*

*plt.subplot(2, 3, 3)*

*sns.histplot(water\_quality["Nitrates"], kde=True)*

*plt.title("Nitrates Distribution")*

plt.subplot(2, 3, 4)

sns.histplot(water\_quality["Sulfates"], kde=True)

plt.title("Sulfates Distribution")

plt.subplot(2, 3, 5)

sns.histplot(water\_quality["Quality"], kde=True)

plt.title("Quality Distribution")

plt.tight\_layout()

# Scatter plot matrix for correlation visualization

sns.pairplot(water\_quality)

plt.suptitle("Pairwise Relationships", y=1.02)

plt.show()

# Correlation matrix

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

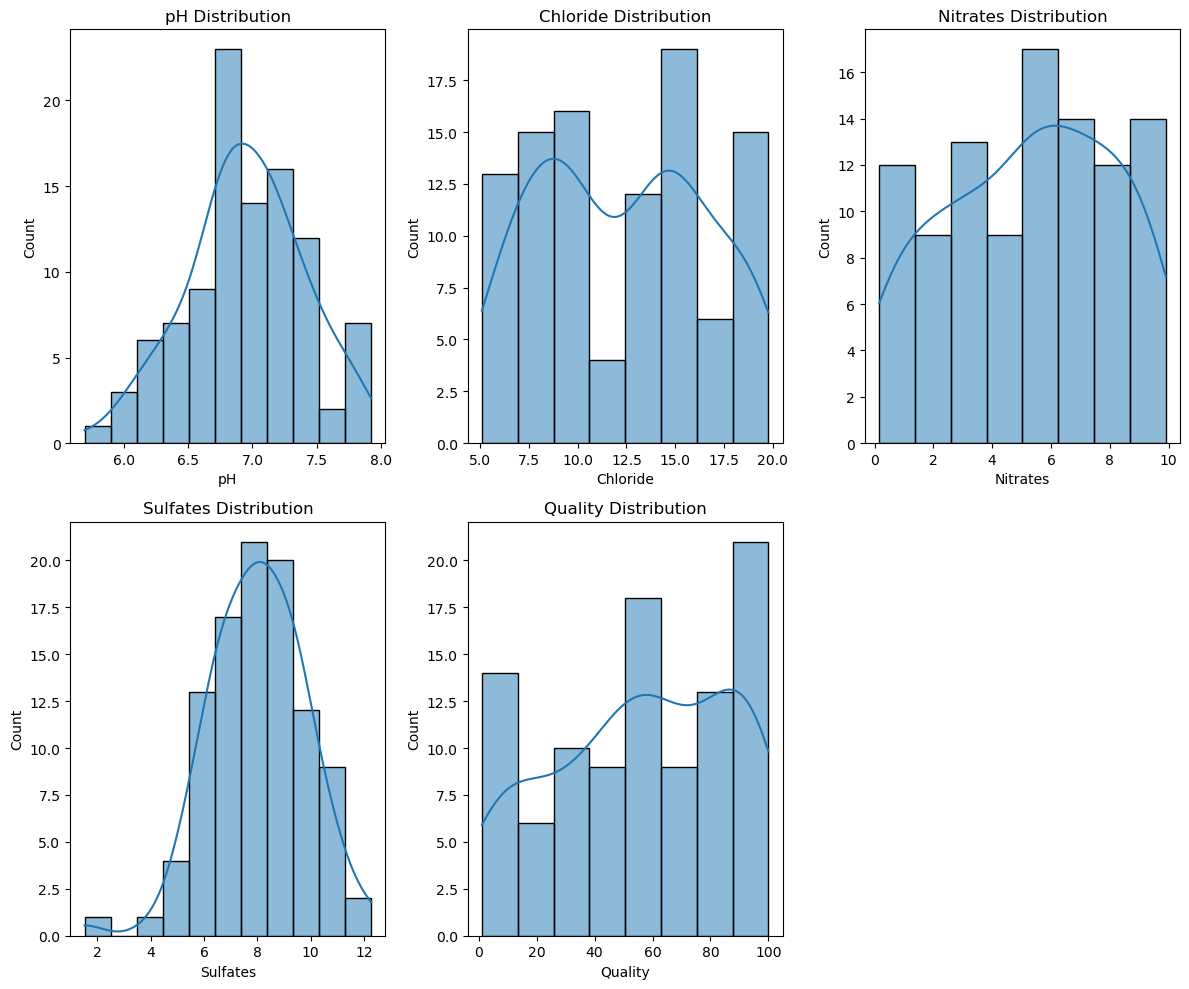
correlation\_matrix = water\_quality.corr()

sns.heatmap(correlation\_matrix, annot=True, cmap="coolwarm", fmt=".2f")

plt.title("Correlation Matrix")

plt.show()

### SAMPLE OUTPUT :



### CORRELATION :

#### CODE :

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

# Generating synthetic water quality data

np.random.seed(42)

data = {

'pH': np.random.uniform(6, 9, 100),

'Dissolved Oxygen': np.random.uniform(4, 12, 100),

'Turbidity': np.random.uniform(0.1, 5, 100),

'Conductivity': np.random.uniform(50, 500, 100),

'Chloride': np.random.uniform(10, 100, 100)

}

df = pd.DataFrame(data)

# Calculating the correlation matrix

corr = df.corr()

# Plotting the correlation matrix using Seaborn

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

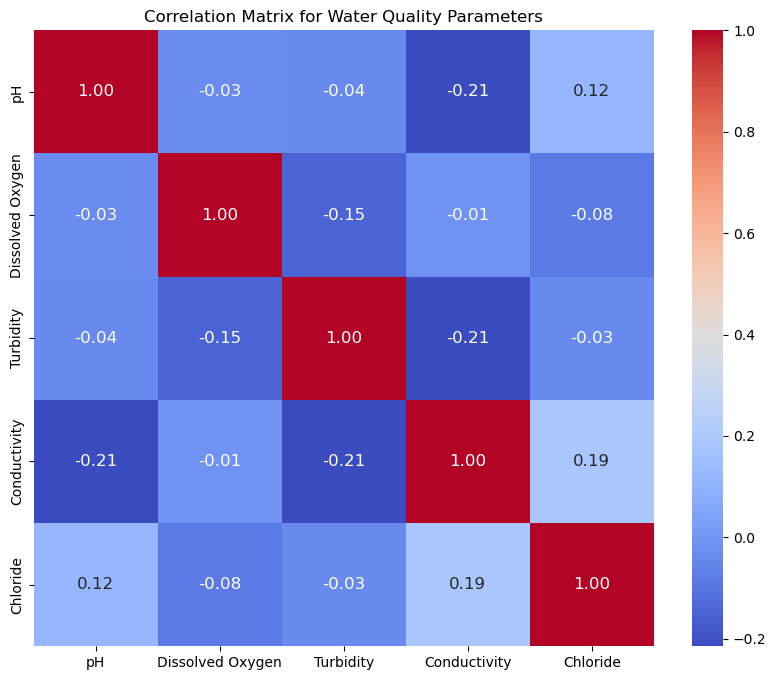
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f', annot\_kws={"size": 12})

plt.title('Correlation Matrix for Water Quality Parameters')

Plt.show()

### 

**SAMPLE OUTPUT :**



**SCATTER PLOT :**

**CODE :**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Sample water quality data (replace this with your own dataset)

data = {

'pH': [7.0, 7.2, 7.5, 7.8, 8.0, 6.5, 7.1, 7.3],

'Chlorine (ppm)': [0.5, 0.7, 0.4, 0.8, 0.6, 0.3, 0.9, 0.55],

'Dissolved Oxygen (ppm)': [8, 7, 9, 6, 8.5, 7.5, 9.5, 6.5],

'Temperature (C)': [22, 24, 20, 26, 25, 23, 21, 27]

}

df = pd.DataFrame(data)

# Create a scatterplot using Matplotlib

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

# Using Matplotlib

plt.scatter(df['pH'], df['Chlorine (ppm)'], label='pH vs Chlorine')

plt.xlabel('pH')

plt.ylabel('Chlorine (ppm)')

plt.title('Scatterplot of pH vs Chlorine')

plt.legend()

plt.grid(True)

plt.show()

# Using Seaborn for another scatterplot

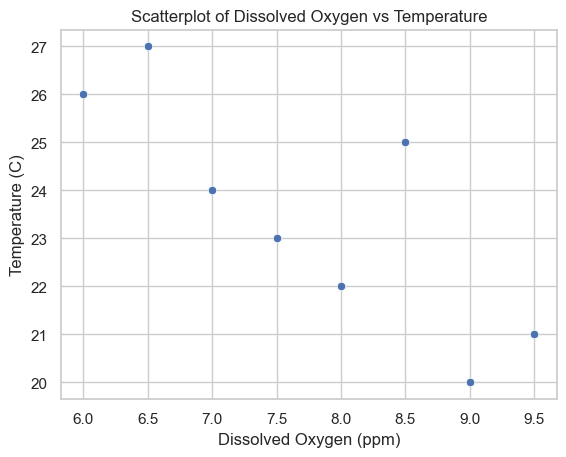
sns.set(style='whitegrid')

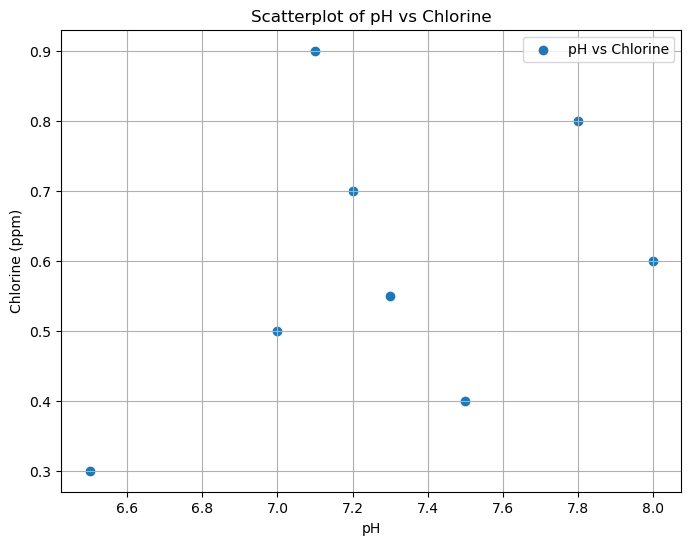
sns.scatterplot(x='Dissolved Oxygen (ppm)', y='Temperature (C)', data=df)

plt.title('Scatterplot of Dissolved Oxygen vs Temperature')

plt.show()

***SAMPLE OUTPUT :***





**PREDICTIVE MODEL :**

Logistic regression :

import pandas as pd

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

from sklearn.linear\_model import LogisticRegression

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

import matplotlib.pyplot as plt

import seaborn as sns

# Load your dataset (replace 'your\_dataset.csv' with the actual file path)

data = pd.read\_csv('water\_potability.csv')

# Check the first few rows of the dataset

print(data.head())

# Separate features (water quality parameters) and the target variable

X = data.drop('Potability', axis=1)

y = data['Potability']

# Split the data into training and testing sets (80% train, 20% test)

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

# Create a logistic regression model

model = LogisticRegression(max\_iter=10000) # Increase max\_iter for convergence

# Train the model

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

# Make predictions

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

# Calculate accuracy

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

print(f"Accuracy: {accuracy:.2f}")

# Create a confusion matrix

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

sns.heatmap(conf\_matrix, annot=True, cmap='Blues', fmt='d')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()

# Generate a classification report

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

print("Classification Report:")

print(class\_report)

**OUTPUT :**

**ph** **Hardness**  **Solids** **Chloramines**   **Sulfate**   **Conductivity**

**0** NaN 204.890455 20791.318981 7.300212 368.516441 564.308654

**1** 3.716080 129.422921 18630.057858 6.635246 NaN 592.885359

**2**  8.099124 224.236259 19909.541732 9.275884 NaN 418.606213

**3**  8.316766 214.373394 22018.417441 8.059332 356.886136 363.266516

**4**  9.092223 181.101509 17978.986339 6.546600 310.135738 398.410813

**Organic\_carbon**  **Trihalomethanes**  **Turbidity**   **Potability**

**0 10.379783 86.990970 2.963135 0**

**1 15.180013 56.329076 4.500656 0**

**2 16.868637 66.420093 3.055934 0**

**3 18.436524 100.341674 4.628771 0**

**4 11.558279 31.997993 4.075075 0**

**RANDOM FOREST :**

**CODE :**

import pandas as pd

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

from sklearn.ensemble import RandomForestClassifier

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

import seaborn as sns

import matplotlib.pyplot as plt

# Load the water quality dataset (replace with your dataset)

data = pd.read\_csv('water\_potability.csv') # Load your dataset here

# Display the first few rows of the dataset to understand its structure

print(data.head())

# Check for missing values and handle if needed

print(data.isnull().sum()) # Check for missing values

# Handle missing values (if any)

data = data.dropna() # For simplicity, dropping rows with missing values

# Define features and target variable

X = data.drop('Potability', axis=1) # Features (independent variables)

y = data['Potability'] # Target variable (dependent variable)

# 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)

# Initialize the Random Forest classifier

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

# Train the model

rf\_classifier.fit(X\_train, y\_train)

# Make predictions

predictions = rf\_classifier.predict(X\_test)

# Evaluate the model

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

print(f"Accuracy: {accuracy:.2f}")

# Generate a classification report and confusion matrix

print(classification\_report(y\_test, predictions))

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

sns.heatmap(conf\_matrix, annot=True, fmt='d')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()

**OUTPUT :**

**ph Hardness Solids Chloramines Sulfate Conductivity**

0  **NaN 204.890455 20791.318981 7.300212 368.516441 564.308654**

**1 3.716080 129.422921 18630.057858 6.635246 NaN 592.885359**

**2 8.099124 224.236259 19909.541732 9.275884 NaN 418.606213**

**3 8.316766 214.373394 22018.417441 8.059332 356.886136 363.266516**

**4 9.092223 181.101509 17978.986339 6.546600 310.135738 398.410813**

**Organic\_carbon Trihalomethanes Turbidity Potability**

**0** 10.379783 86.990970 2.963135 0

**1** 15.180013 56.329076 4.500656 0

**2** 16.868637 66.420093 3.055934 0

**3** 18.436524 100.341674 4.628771 0

**4**  11.558279 31.997993 4.075075 0

##### ph : 491

##### Hardness : 0

##### Solids : 0

##### Chloramines : 0

##### Sulfate : 781

##### Conductivity : 0

##### Organic\_carbon : 0

##### Trihalomethanes : 162

##### Turbidity : 0

##### Potability : 0

##### Dtype : int64

##### Accuracy : 0.65

###### precision recall f1-score support

0 0.65 0.84 0.73 231

1 0.65 0.40 0.50 172

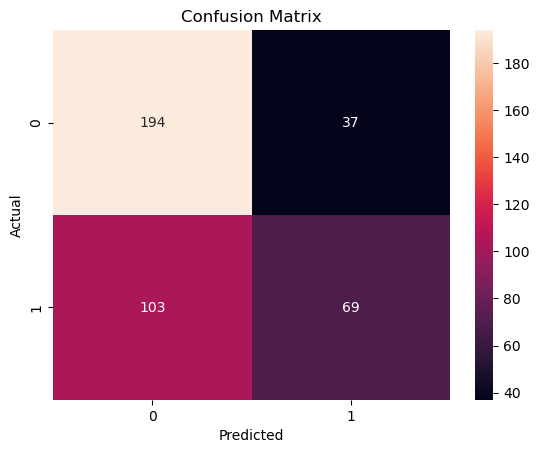
##### accuracy 0.65 403

##### macro avg 0.65 0.62 0.62 403

##### weighted avg 0.65 0.65 0.63 403

##### 

### SAMPLE OUTPUT :



### CONCLUSION :

###### In conclusion, the analysis of the water quality dataset has provided valuable insights into the

###### state of water quality in the studied area. It isevident thatwaterqualityparameterssuchaspH,

###### turbidity, dissolved oxygen, and pollutant levels play a crucial role in assessing the overall

###### health of waterbodies.Thedatasethashelped identify trends, potential issues, and areas for

###### improvement in water quality management. Further research and proactive measures may

###### be necessary to ensure the preservation and improvementofwaterquality in the region.

###### 