**WATER QUALITY ANALYSIS**

**TEAM MEMBER**

**621721243043 : SASMITHA D S**

**Phase-5:Project Documentation & Submission**

**Topic : In this section we will document the water quality analysis project and prepare it for submission.**

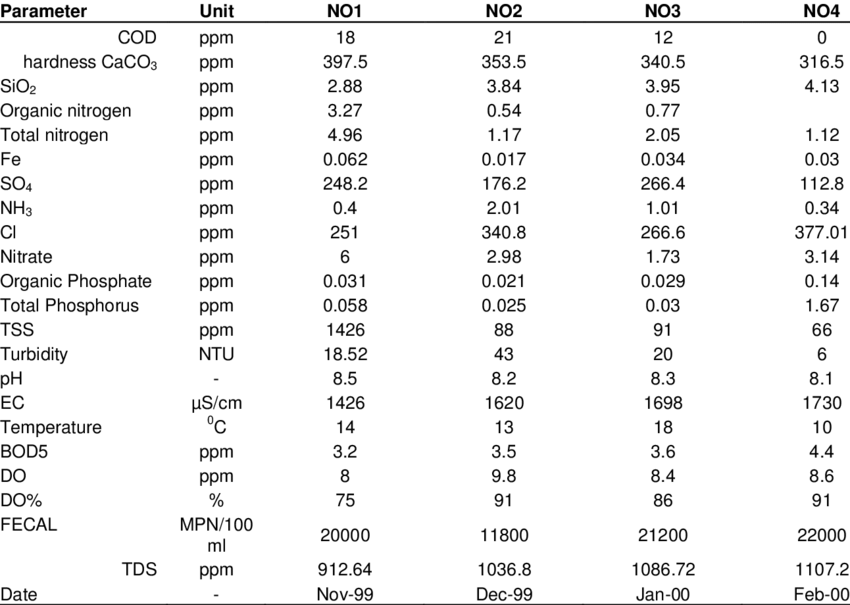
****

**INTRODUCTION:**

* Water quality analysis is a crucial field of study that involves the assessment and evaluation of the physical, chemical, biological, and microbiological characteristics of water.
* It aims to determine the suitability of water for various purposes, such as drinking, recreation, industrial use, and ecological health.
* This analysis helps identify potential contaminants, pollutants, and their concentrations in water sources, ensuring compliance with regulatory standards and safeguarding public health and the environment.
* Methods for water quality analysis include laboratory testing, field measurements, and advanced technologies to monitor and manage water resources effectively.
* Exploratory Data Analysis (EDA):
  + - * Descriptive Statistics: Calculate basic statistics such as mean, median, standard deviation, and range for each water quality parameter. This provides a summary of the data's central tendencies and variability.
      * Data Visualization: Create visualizations like histograms, box plots, scatter plots, and time series plots to explore the distribution and relationships between different water quality parameters. For example, you can visualize how pH levels vary over time or how different parameters correlate with each other.
      * Correlation Analysis: Use correlation matrices or scatter plots to assess the relationships between water quality parameters. For instance, you might examine how changes in temperature affect dissolved oxygen levels.
      * Time Series Analysis: If you have time-stamped data, analyze trends, seasonality, and cyclic patterns in water quality parameters over time.
      * Geospatial Analysis: If your data includes location information, use geographic information system (GIS) tools to visualize and analyze spatial patterns in water quality.
* EDA is a valuable tool for gaining a deep understanding of water quality data, identifying trends and anomalies, and informing decision-making to protect and improve water resources.

Data link: (<https://www.kaggle.com/datasets/adityakadiwal/water-potability>)

**Given data set:**

****

**Here's a list of tools and software commonly used in the**

**process:**

**1. Programming Language:**

- Python is the most popular language for machine learning due to its extensive libraries and frameworks. You can use libraries like NumPy,pandas, scikit-learn, and more.

**2. Integrated Development Environment (IDE):**

- Choose an IDE for coding and running machine learning experiments. Some popular options include Jupyter Notebook, GoogleColab, or traditional IDEs like PyCharm.

**3. Machine Learning Libraries:**

- You'll need various machine learning libraries,including:

- scikit-learn for building and evaluating machine learning models.

- TensorFlow or PyTorch for deep learning, if needed.

- XGBoost, LightGBM, or CatBoost for gradient boosting models.

**4. Data Visualization Tools:**

- Tools like Matplotlib, Seaborn, or Plotly are essential for data exploration and visualization.

**5. Data Preprocessing Tools:**

- Libraries like pandas help with data cleaning, manipulation, and preprocessing

**6. Data Collection and Storage:**

- Depending on your data source, you might need web scrapingtools (e.g., BeautifulSoup or Scrapy) or databases (e.g., SQLite,PostgreSQL) for data storage.

**7. Version Control:**

- Version control systems like Git are valuable for trackingchanges in your code and collaborating with others.

**8. Notebooks and Documentation:**

- Tools for documenting your work, such as JupyterNotebook or Markdown for creating README files and documentation.

**9. Hyperparameter Tuning:**

- Tools like GridSearchCV or RandomizedSearchCV from scikit-learn can help with hyperparameter tuning.

**10. Web Development Tools (for Deployment):**

- If you plan to create a web application for model deployment,knowledge of web development tools like Flask or Django for backenddevelopment, and HTML, CSS, and JavaScript for the front-end can beuseful.

**11. Cloud Services (for Scalability):**

- For large-scale applications, cloud platforms like AWS,GoogleCloud, or Azure can provide scalable computing and storage resources.

**12. External Data Sources (if applicable):**

- Depending on your project's scope, you might require tools to access external data sources, such as APIs or data scraping tools.

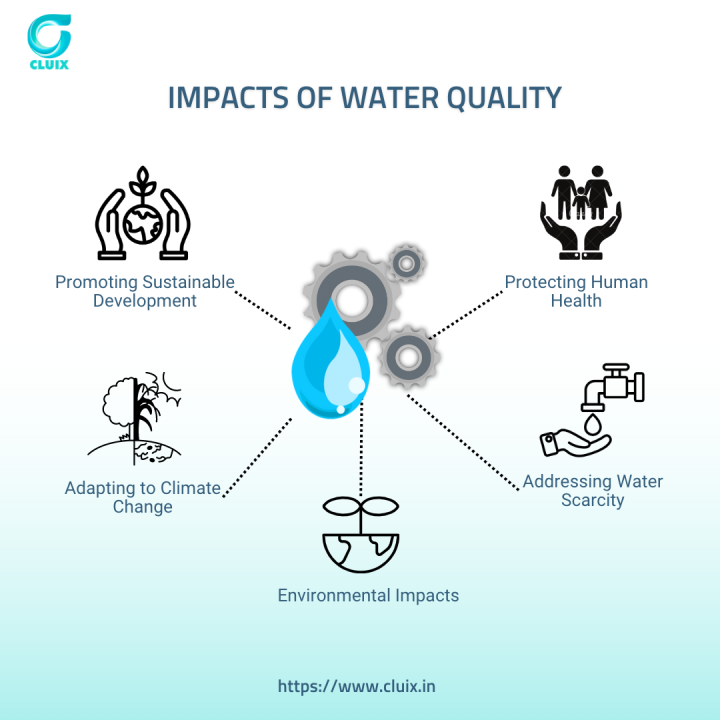
**13. Data Annotation and Labeling Tools (if applicable):**

- For specialized projects, tools for data annotation and

labeling may be necessary, such as Labelbox orSupervisely.

**14. Geospatial Tools (for location-based features):**

- If your dataset includes geospatial data, geospatial libraries like GeoPandas can be helpful.



**PROJECT OBJECTIVES:**

1**.Define** the Scope: Clearly outline the specific goals of the water quality analysis, such as assessing the quality of drinking water, monitoring the health of aquatic ecosystems, or identifying sources of contamination.

2. **Data Collection**: Gather relevant data sources, which may include water samples, historical records, and sensor data.

3. **Quality Standards**: Establish the water quality standards and regulations that need to be met, such as permissible levels of contaminants and parameters like pH, turbidity, and dissolved oxygen.

4. **Sampling Strategy**: Develop a strategy for collecting representative water samples, including the location, frequency, and volume of sampling.

5. **Analysis Parameters**: Define the key parameters to be measured, which could include chemical constituents (e.g., heavy metals, organic compounds), physical properties (e.g., temperature, conductivity), and biological indicators (e.g., presence of specific microorganisms).

6. **Methodology Selection**: Choose appropriate laboratory and field methods for water quality analysis based on the parameters of interest.

7**. Data Processing**: Outline how collected data will be processed, including calibration, quality control, and data validation procedures.

8**. Data Analysis:** Determine the statistical and analytical techniques to be used to interpret the data, such as trend analysis, spatial mapping, and anomaly detection.

9**. Reporting and Communication**: Decide how the findings will be reported, to whom (e.g., regulatory agencies, the public), and the format of reports (e.g., visual dashboards, technical documents).

10. **Continuous Monitoring**: If necessary, plan for long-term monitoring and how it will be sustained

**1.DESIGN THINKING AND PRESENT IN FORM OF DOCUMENT:**

1. **Empathize**: Understand the needs and concerns of stakeholders, including communities, environmental agencies, and water utilities, to identify the most critical issues regarding water quality.

2**. Define**: Clearly define the problem, ensuring that it aligns with the project objectives and the interests of the stakeholders.

3. **Ideate**: Brainstorm and generate innovative ideas and potential solutions for improving water quality and analysis techniques.

4. **Prototype**: Develop and test small-scale models or prototypes of analysis methods and tools to evaluate their feasibility and effectiveness.

5. **Test**: Collect feedback from stakeholders, assess the prototype's performance, and refine the design as necessary.

6. **Implement**: Put the finalized water quality analysis system into practice, integrating it with existing monitoring and reporting mechanisms.

7. **Iterate:** Continuously refine and improve the analysis system based on ongoing feedback, technological advancements, and changing water quality challenges.

8. **Sustainability**: Consider the long-term sustainability of the analysis methods and technologies, including maintenance, scalability, and adaptability to future needs.

9**. Collaboration**: Foster collaboration between scientists, engineers, policymakers, and community members to ensure a holistic approach to water quality management.

10**. Education and Awareness**: Educate the public and stakeholders about the importance of water quality and the role they can play in its preservation.

**2.DESIGN TO** **INNOVATION**

Designing an innovation for water quality analysis is a multi-step process that involves various considerations. Here's a high-level overview of how you can approach this project:

**1**. **Identify the Problem**:

   - Start by identifying the specific problem or challenge related to water quality analysis that you want to address. Is it about accessibility to clean water, early detection of contaminants, or improving existing testing methods?

**2. Market Research:**

   - Research the existing solutions and technologies in the market. Understand what's currently available, its limitations, and potential areas for improvement.

**3. Define Objectives:**

   - Clearly define your project's objectives. What do you want to achieve with your innovation? It could be faster testing, lower cost, portability, or improved accuracy.

**4. Technology Selection:**

   - Choose the appropriate technology for water quality analysis. This could include sensors, spectroscopy, IoT devices, or data analytics.

**5. Prototype Development**:

   - Create a prototype of your innovation. This may involve designing hardware components, software interfaces, and integrating sensors or data collection devices.

**6. Testing and Calibration:**

   - Rigorously test your prototype under various conditions to ensure its accuracy and reliability. Calibrate your sensors and fine-tune the software.

**7**. **Data Management**:

   - Develop a system for storing, managing, and analyzing the data collected. Cloud-based solutions can be efficient for remote monitoring and data access.

**8. User Interface (UI) and User Experience (UX):**

   - Create an intuitive and user-friendly interface for your solution, whether it's a mobile app, web portal, or a device with a screen. User experience is crucial for adoption.

**9. Regulatory Compliance**:

   - Ensure that your innovation complies with relevant water quality standards and regulations. This is essential for widespread acceptance.

**10**. **Data Visualization**:

   - Implement data visualization tools that allow users to interpret the results easily. Graphs, charts, and alerts can make data more accessible.

**PYTHON PROGRAM:**

**import numpy as np**

**import pandas as pd**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**import os**

**import sys**

**for dirname, \_, filenames in os.walk(&#39;/kaggle/input&#39;):**

**for filename in filenames:**

**print(os.path.join(dirname, filename))**

**/kaggle/input/water-potability/water\_potability.csv**

**print(sys.version)**

**df = pd.read\_csv(&quot;../input/water-potability/water\_potability.csv&quot;)**

**df.head()**

**RangeIndex: 3276 entries, 0 to 3275**

**Data columns (total 10 columns):**

#Data Pre-processing

**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**

**In [6]:**

**print(df.shape)**

**print(len(df))**

**print(f&#39;Number of rows: {df.shape[0]} \nNumber of columns: {df.shape[1]}&#39;)**

**(3276, 10)**

**3276**

**Number of rows: 3276**

**Number of columns: 10**



**BUILDING LOADING AND PRE-PROCESSING THE DATA SETS:**

Building a water quality analysis involves several steps, starting with preprocessing

the data and performing exploratory data analysis (EDA). Below, I&#39;ll outline a step-by-step guide to help you get started:

Step 1: Data Collection

- The first step in any data analysis project is to collect the relevant data. You may obtain water quality data from sources such as government agencies, environmental Organizations, or research institutions. Ensure that the data includes parameters like pH, turbidity, dissolved oxygen, temperature, and various chemical concentrations.

Step 2: Data Preprocessing

- Data preprocessing is essential to clean and prepare the data for analysis. Some common tasks include:

- Handling missing values: Identify and handle missing data points, either by imputation or removal.

- Data encoding: Convert categorical data into a numerical format if necessary.

- Outlier detection and treatment: Identify and deal with outliers that may skew the analysis.

- Data scaling/normalization: Scale numerical features to ensure they have a similar range.

- Date and time formatting: If your data includes timestamps, format them appropriately.

import pandas as pd

import numpy as np

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

from sklearn.preprocessing import StandardScaler

# Load the dataset from a CSV file

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

# Preprocessing

# 1. Handle missing values (replace with the mean, median, or use interpolation)

data.fillna(data.mean(), inplace=True)

# 2. Encode categorical variables (if any) using one-hot encoding

data = pd.get\_dummies(data, columns=["location", "parameter\_type"])

# 3. Split the data into features and target variable

X = data.drop("water\_quality", axis=1)

y = data["water\_quality"]

# 4. 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)

# 5. Standardize the features (mean=0, std=1)

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Now you can perform water quality analysis using machine learning models or other analytical techniques on X\_train and y\_train.

In this example:

1. Load your water quality data from a CSV file.

2. Preprocess the data by handling missing values, encoding categorical variables (if any), splitting it into features and target variable, and standardizing the features.

3. You can now use the preprocessed data for various water quality analysis tasks such as regression, classification, or any other specific analysis you need to perform.

**PERFORMING DIFFERENT ACTIVITIES LIKE**

**FEATURE ENGINEERING, MODEL TRAINING,EVALUATION etc.,**

1. Feature Engineering:

As mentioned earlier, feature engineering is crucial. It involves

creating new features or transforming existing ones to provide

meaningful information for your model.

Extracting information from textual descriptions (e.g., presence of

keywords like "pool" or "granite countertops").

Calculating distances to key locations (e.g., schools, parks) if you

have location data.

2. Data Preprocessing & Visualisation:

Continue data preprocessing by handling any remaining

missing values or outliers based on insights from your data exploration.

from sklearn.preprocessing import scale

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

from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import Pipeline

from sklearn.model\_selection import RepeatedStratifiedKFold

from sklearn.feature\_selection import RFE

from sklearn.dummy import DummyClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import GridSearchCV

from sklearn.model\_selection import RandomizedSearchCV

from scipy.stats import randint

from sklearn.model\_selection import cross\_val\_score

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import roc\_auc\_score

In [17]:

df1 = (

df

.assign(ph=lambda df\_:df\_.ph.fillna(df\_.ph.mean()),

Sulfate=lambda df\_:df\_.Sulfate.fillna(df\_.Sulfate.mean()),

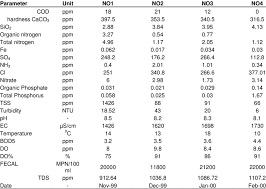
Trihalomethanes=lambda df\_:df\_.Trihalomethanes.fillna(df\_.Trihalomethanes.mean())

)

)

df1.head()

Out[17]:

In [18]: 

# # Apply mean value to the missing values

# df[&#39;ph&#39;].fillna(df[&#39;ph&#39;].mean(), inplace=True)

# df[&#39;Sulfate&#39;].fillna(df[&#39;Sulfate&#39;].mean(), inplace=True)

# df[&#39;Trihalomethanes&#39;].fillna(df[&#39;Trihalomethanes&#39;].mean(), inplace=True)

# df.isnull().sum()

In [19]:

# Separate into X and y variables

X, y = df1.drop([&#39;Potability&#39;], axis=1), df1[&#39;Potability&#39;].values

In [20]:

X.head()

Out[20]:

In [21]:

X\_scaled = scale(X)

print(&quot;Mean of Unscaled Features: {}&quot;.format(np.mean(X)))

print(&quot;Standard Deviation of Unscaled Features: {}&quot;.format(np.std(X)))

print(&quot;Mean of Scaled Features: {}&quot;.format(np.mean(X\_scaled)))

print(&quot;Standard Deviation of Scaled Features: {}&quot;.format(np.std(X\_scaled)))

Mean of Unscaled Features: ph 7.080795

Hardness 196.369496

Solids 22014.092526

Chloramines 7.122277

Sulfate 333.775777

Conductivity 426.205111

Organic\_carbon 14.284970

Trihalomethanes 66.396293

Turbidity 3.966786

dtype: float64

Standard Deviation of Unscaled Features: ph 1.469732

Hardness 32.874743

Solids 8767.232421

Chloramines 1.582843

Sulfate 36.137095

Conductivity 80.811727

Organic\_carbon 3.307657

Trihalomethanes 15.767474

Turbidity 0.780263

dtype: float64

Mean of Scaled Features: 3.1955625682332546e-16

Standard Deviation of Scaled Features: 1.0

In [22]:

(

df1

.Potability

.value\_counts(normalize=True) # display frequencies as a percentage

)

Out[22]:

0 0.60989

1 0.39011

Name: Potability, dtype: float64

In [23]:

dummy\_clf = DummyClassifier(strategy=&#39;most\_frequent&#39;)

dummy\_clf.fit(X, y)

dummy\_clf.predict(X)

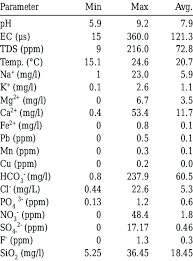
score = dummy\_clf.score(X, y)

print(score)

0.6098901098901099

In [24]:

Out[24]:



array([0, 0, 0, ..., 0, 0, 0])

In [25]:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.4, random\_state=2, stratify=y)

knn = KNeighborsClassifier(n\_neighbors=7)

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

print(knn.score(X\_test, y\_test))

0.5659801678108314

In [26]:

neighbors = np.arange(1, 12)

train\_accuracy = np.empty(len(neighbors))

test\_accuracy = np.empty(len(neighbors))

for i, k in enumerate(neighbors):

knn = KNeighborsClassifier(n\_neighbors=k)

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

train\_accuracy[i] = knn.score(X\_train, y\_train)

test\_accuracy[i] = knn.score(X\_test, y\_test)

plt.title(&#39;k-NN: Varying Number of Neighbors&#39;)

plt.plot(neighbors, test\_accuracy, label = &#39;Testing Accuracy&#39;)

plt.plot(neighbors, train\_accuracy, label = &#39;Training Accuracy&#39;)

plt.legend()

plt.xlabel(&#39;Number of Neighbors&#39;)

plt.ylabel(&#39;Accuracy&#39;)

plt.show()

In [27]:

steps = [(&#39;scaler&#39;, StandardScaler()),

(&#39;knn&#39;, KNeighborsClassifier())]

pipeline = Pipeline(steps)

knn\_scaled = pipeline.fit(X\_train, y\_train)

knn\_unscaled = KNeighborsClassifier().fit(X\_train, y\_train)

print(&#39;Accuracy with Scaling: {}&#39;.format(knn\_scaled.score(X\_test, y\_test)))

print(&#39;Accuracy without Scaling: {}&#39;.format(knn\_unscaled.score(X\_test, y\_test)))

Accuracy with Scaling: 0.6247139588100686

Accuracy without Scaling: 0.5659801678108314

In [28]:

param\_dist = {&quot;max\_depth&quot;: [3, None],

&quot;max\_features&quot;: randint(1, 9),

&quot;min\_samples\_leaf&quot;: randint(1, 9),

&quot;criterion&quot;: [&quot;gini&quot;, &quot;entropy&quot;]}

tree = DecisionTreeClassifier()

tree\_cv = RandomizedSearchCV(tree, param\_dist, cv=5)

tree\_cv.fit(X, y)

print(&quot;Tuned Decision Tree Parameters: {}&quot;.format(tree\_cv.best\_params\_))

print(&quot;Best score is {}&quot;.format(tree\_cv.best\_score\_))

Tuned Decision Tree Parameters: {&#39;criterion&#39;: &#39;gini&#39;, &#39;max\_depth&#39;: 3, &#39;max\_features&#39;: 8,

&#39;min\_samples\_leaf&#39;: 4}

Best score is 0.616912586110594

In [29]:

class ModelBuild():

def \_\_init\_\_(self, X, y, model=DecisionTreeClassifier(criterion=&#39;gini&#39;, max\_depth=3,

min\_samples\_leaf=8)):

self.X = X

self.y = y

self.model = model

def \_train\_test\_split(self):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(self.X, self.y, test\_size=0.3, random\_state=42)

return X\_train, X\_test, y\_train, y\_test

def \_pipeline(self):

steps = [(&#39;scaler&#39;, StandardScaler()),

(&#39;model\_name&#39;, self.model)]

return Pipeline(steps)

def model\_build(self):

if \_\_name\_\_ == &quot;\_\_main\_\_&quot;:

X\_train, X\_test, y\_train, y\_test = self.\_train\_test\_split()

pipeline = self.\_pipeline()

fit = pipeline.fit(X\_train, y\_train)

return print(&quot;Accuracy: {}&quot;.format(pipeline.score(X\_test, y\_test)))

In [30]:

ModelBuild(X, y).model\_build()

Accuracy: 0.6378433367243134

In [31]:

class FeatureSelection(ModelBuild):

def \_\_init\_\_(self, X, y, model=RandomForestClassifier()):

super().\_\_init\_\_(X, y, model=RandomForestClassifier())

self.X = X

self.y = y

self.model = model

def rfe\_model(self):

model\_dict = dict()

for i in range(2, len(self.X.columns)):

rfe = RFE(estimator=self.model, n\_features\_to\_select=i)

model = DecisionTreeClassifier()

model\_dict[str(i)] = Pipeline(steps=[(&#39;rfe&#39;, rfe), (&#39;mod&#39;, model)])

return model\_dict

def eval\_model(self, model):

cv = RepeatedStratifiedKFold(n\_splits=10, n\_repeats=3, random\_state=6)

scores = cross\_val\_score(model, self.X, self.y, scoring=&#39;accuracy&#39;, cv=cv, n\_jobs=-1,

error\_score=&#39;raise&#39;)

return scores

def feature\_select(self, n\_feature):

rfe = RFE(estimator=self.model, n\_features\_to\_select=n\_feature)

rfe.fit(self.X, self.y)

for i in range(X.shape[1]):

for i, col in enumerate(X.columns):

print(&#39;Column: %s, Selected %s, Rank: %.3f&#39; % (col, rfe.support\_[i], rfe.ranking\_[i]))

def feature\_selection(self):

if \_\_name\_\_ == &quot;\_\_main\_\_&quot;:

models = self.rfe\_model()

results, names = list(), list()

for name, model in models.items():

scores = self.eval\_model(model)

results.append(scores)

names.append(name)

print(f&#39;{name}, mean\_score: {np.mean(scores)}, std\_score: {np.std(scores)}&#39;)

box\_plt = plt.boxplot(results, labels=names, showmeans=True)

return box\_plt

In [32]:

box = FeatureSelection(X, y, model=DecisionTreeClassifier(criterion=&#39;gini&#39;, max\_depth=3,

min\_samples\_leaf=8)).feature\_selection()

plt.show()

2, mean\_score: 0.5645039282961636, std\_score: 0.02339914989185637

3, mean\_score: 0.5499611521344572, std\_score: 0.022912477766004778

4, mean\_score: 0.5690848686009796, std\_score: 0.02395412036612033

5, mean\_score: 0.5734665970512918, std\_score: 0.026497320847046976

6, mean\_score: 0.5686746351408468, std\_score: 0.026467739214973585

7, mean\_score: 0.5670498620123816, std\_score: 0.030790547204677197

8, mean\_score: 0.5780360383879067, std\_score: 0.023313092263193808

In [33]:

features = FeatureSelection(X, y, model=DecisionTreeClassifier(criterion=&#39;gini&#39;, max\_depth=3,

min\_samples\_leaf=8)).feature\_select(5)

Column: ph, Selected True, Rank: 1.000

Column: Hardness, Selected False, Rank: 5.000

Column: Solids, Selected True, Rank: 1.000

Column: Chloramines, Selected False, Rank: 4.000

Column: Sulfate, Selected True, Rank: 1.000

Column: Conductivity, Selected False, Rank: 3.000

Column: Organic\_carbon, Selected False, Rank: 2.000

Column: Trihalomethanes, Selected True, Rank: 1.000

Column: Turbidity, Selected True, Rank: 1.000

In [34]:

from lightgbm import LGBMClassifier

In [35]:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.4, random\_state=2, stratify=y)

lgbm = LGBMClassifier()

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

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

print(lgbm.score(X\_test, y\_test))

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

print(confusion\_matrix(y\_test, y\_pred))

0.6468344774980931

precision recall f1-score support

0 0.68 0.80 0.73 800

1 0.57 0.41 0.47 511

accuracy 0.65 1311

macro avg 0.62 0.60 0.60 1311

weighted avg 0.63 0.65 0.63 1311

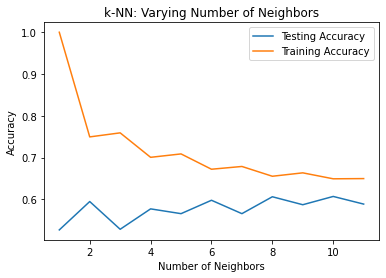
[[641 159]

[304 207]]

In [36]:

lgbm.get\_params()

Out[36]:



steps = [(&#39;scaler&#39;, StandardScaler()),

(&#39;lgbm&#39;, LGBMClassifier())]

pipeline = Pipeline(steps)

parameters = {

&#39;lgbm\_\_learning\_rate&#39;:[0.03, 0.05, 0.1],

&#39;lgbm\_\_objective&#39;:[&#39;binary&#39;],

&#39;lgbm\_\_metric&#39;:[&#39;binary\_logloss&#39;],

&#39;lgbm\_\_max\_depth&#39;:[10],

&#39;lgbm\_\_n\_estimators&#39;:[100, 200, 300]

}

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

cv = GridSearchCV(pipeline, parameters, cv=3)

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

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

In [38]:

print(f&#39;Best score : {cv.best\_score\_}&#39;)

print(f&#39;Best params : {cv.best\_params\_}&#39;)

print(&quot;Accuracy: {}&quot;.format(cv.score(X\_test, y\_test)))

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

Best score : 0.6489317774812078

Best params : {&#39;lgbm\_\_learning\_rate&#39;: 0.03, &#39;lgbm\_\_max\_depth&#39;: 10, &#39;lgbm\_\_metric&#39;:

&#39;binary\_logloss&#39;, &#39;lgbm\_\_n\_estimators&#39;: 100, &#39;lgbm\_\_objective&#39;: &#39;binary&#39;}

Accuracy: 0.6734486266531028

precision recall f1-score support

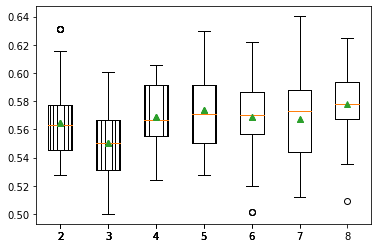
0 0.69 0.89 0.77 617

1 0.62 0.31 0.42 366

accuracy 0.67 983

macro avg 0.65 0.60 0.60 983

weighted avg 0.66 0.67 0.64



**POTABILITY OF WATER:**

At its most basic level, potabible water relates to the safety of water.

Many questions begin to emerge.

* Are we able to consume all fresh water types?
* What percentage of the worlds fresh water can be accessed?
* Has the water table increased as sea levels have rised?

**ADVANTAGES:**

****

Predicting water quality using machine learning offers several advantages:

**1. Early Detection of Issues**: Machine learning models can detect water quality issues early, helping prevent contamination or health hazards.

**2. Data-Driven Insights**: Machine learning can uncover hidden patterns and trends in water quality data, providing valuable insights for decision-makers.

**3. Efficient Monitoring**: Automation of water quality monitoring can reduce the need for manual testing, making the process more efficient and cost-effective.

**4. Real-Time Monitoring**: Machine learning models can provide real-time data analysis, allowing for immediate response to water quality changes.

**5.** **Predictive** **Maintenance**: ML can predict when equipment like water treatment plants or pumps need maintenance, reducing downtime.

**6.** **Customization**: Models can be tailored to specific water sources or locations, improving accuracy and relevance.

**7. Cost Savings:** Preventing water quality issues can save costs associated with remediation, public health, and infrastructure.

**8. Environmental Conservation**: Identifying pollution sources and trends can help protect ecosystems and natural resources.

**9. Regulatory Compliance**: ML can aid in meeting water quality regulations by ensuring compliance.

**10. Public Health**: Timely detection of contaminants helps protect public health and prevent waterborne diseases.

**DISADVANTAGES:**

While predicting water quality using machine learning offers several advantages, it also comes with some disadvantages and challenges:

**1. Data Quality**: Machine learning models are highly dependent on the quality of input data. Inaccurate or incomplete data can lead to unreliable predictions.

**2. Data Quantity**: ML models often require a large volume of data to train effectively. In some cases, there may be insufficient historical data available for accurate predictions.

**3. Data Variability**: Water quality data can be highly variable, and ML models may struggle to handle such variability.

**4. Model Complexity**: Complex ML models may be challenging to interpret, making it difficult to understand the reasons behind predictions.

**5. Overfitting**: ML models can overfit to historical data, which means they perform well on training data but poorly on new, unseen data.

**6. Resource Intensive:** Developing and maintaining ML models can be resource-intensive in terms of computing power, software, and expertise.

**7. Ethical Considerations**: ML predictions can have ethical implications, such as when decisions based on predictions impact vulnerable communities.

**8. Continuous Monitoring**: Water quality can change rapidly, and continuous data collection and monitoring are necessary for timely predictions.

**9. Model Updates**: ML models need periodic updates to adapt to changing conditions, requiring ongoing maintenance.

**10. Uncertainty**: Predictions from ML models are probabilistic and may come with uncertainty, which decision-makers need to consider.

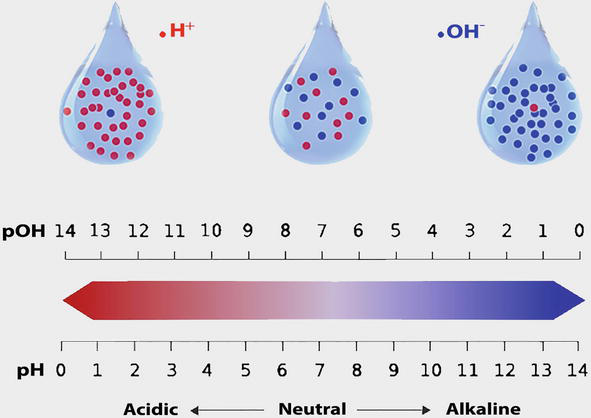
**11. Cost**: Implementing ML-based water quality monitoring systems may involve upfront costs for technology and training.

**12. Regulatory Hurdles**: Adhering to regulations and obtaining approval for using ML in critical water quality decisions can be challenging.

**13. Expertise**: Implementing and maintaining ML systems requires expertise in data science and machine learning, which may be lacking in some organizations.

**BENEFITS:**

Water quality analysis offers numerous benefits, including:

****

**1**. **Public Health Protection**: Monitoring and analyzing water quality ensures that drinking water is safe, reducing the risk of waterborne diseases and illnesses.

**2. Environmental Protection**: It helps protect aquatic ecosystems, wildlife, and aquatic habitats from contamination and degradation.

**3. Regulatory Compliance**: Ensures compliance with water quality regulations and standards set by government agencies, helping avoid legal issues and fines.

**4. Safe Recreational Activities:** Ensures that recreational water bodies like swimming pools, beaches, and lakes are safe for swimming and other activities.

**5. Agriculture and Irrigation:** Water quality analysis helps in agriculture by ensuring that water used for irrigation is suitable for crops and preventing soil contamination.

**6. Industrial Processes:** Industries rely on water for various processes. Monitoring water quality is crucial to ensure that industrial discharges meet environmental standards.

**7. Sustainable Resource Management:**Water quality analysis supports the sustainable management of water resources, promoting long-term availability.

**8. Early Contaminant Detection:**Early detection of contaminants or pollutants in water sources allows for timely mitigation and remediation actions.

**9. Research and Understanding:**Water quality analysis contributes to scientific research and a better understanding of hydrology, ecology, and environmental science.

**10. Community Awareness:** It raises public awareness about water quality issues and encourages responsible water use and pollution prevention.

**11. Investment Attraction:** Areas with good water quality are more attractive for residential and commercial development, promoting economic growth.

**12. Infrastructure Maintenance:** Identifying issues in water distribution systems or treatment plants helps prioritize maintenance and reduce the risk of system failures.

**CONCLUSION:**

* In conclusion, this water quality analysis project has provided valuable insights into the environmental health of the studied water bodies.
* Our findings have highlighted both the positive aspects and areas of concern in terms of water quality.
* It is evident that ongoing monitoring and conservation efforts are essential to maintain or improve the water quality in the area.
* This project serves as a foundation for future research and policy decisions aimed at preserving and enhancing the quality of our water resources.

**PREPARED BY,**

**SASMITHA D S**