ANAND INSTITUTE OF HIGHER TECHNOLOGY
OLD MAHABALIPURAM ROAD,
KALASALINGAM NAGAR,
KAZHIPATTUR,

WATER QUALITY ANALYSIS
BY
DATA ANALYTICS WITH COGNOS

PHASE – 4

NAME   : RICHARD NICHOLES M

REG NO.   : 310121104085

BRANCH   : COMPUTER SCIENCE & ENGINEERING

YEAR/SEM : III / V

Data Analytics With COGNOS

PROJECT : Water Quality Analysis

PROJECT ID : Proj_229794_Team_1

TEAM MEMBERS :
- Santhosh Kumar P
- Yokeswaran K
- Vinoth S
- Sathya Moorthy A
- Vasudevan K

TEAM LEADER : Richard Nicholes M

Water Quality Analysis

Introduction

- Water is a fundamental resource essential for the sustenance of life, and its quality is a critical factor in ensuring the well-being of both humans and the environment.

- Water quality analysis is crucial for ensuring safe drinking water, protecting aquatic ecosystems, managing water resources, and complying with environmental regulations.

- Water quality analysis encompasses a range of parameters and measurements to evaluate the condition of water bodies, including rivers, lakes, oceans, groundwater, and even treated tap water. These parameters may include temperature, pH, turbidity, dissolved oxygen,

- It involves collecting water samples from various sources, conducting laboratory tests, and interpreting the results to make informed decisions about water treatment, pollution control, and resource management.

- Continuous monitoring and analysis help safeguard human health and the environment while ensuring sustainable access to clean water.

Phase – 4 : { Development Part – 2 }

Continue building the Analysis by creating visualizations and a building predictive model.

- In this phase, we will continue building upon the foundation established in the earlier phases.
- We've already collected and preprocessed water quality data in Phase 1, performed exploratory data analysis in Phase 2, and in Phase 3, we laid the groundwork for data visualization and predictive modeling.
- Phase 4 will encompass two major components: data visualization and the development of a predictive model.

Data Collection :

Water Quality Analysis is done by using the Dataset of **"Water_Potability"** provided by the dataset site www.Kaggle.com

DATASET: https://www.kaggle.com/datasets/adityakadiwal/water-potability

Data Observation :

     The water_potability.csv file contains water quality metrics for 3276 different water bodies. It contains Water Quality Parameters such as pH, Hardness, Solids, Chloramines, Sulphate, Conductivity, Organic_Carbon, Trihalomethanes, Turbidity. The Potability value defines the water quality based on the parameters given.

If Potability value is 0 then the water is Potable or the value is 1 then the water is Not Potable.

TITLE : Water Quality Analysis

DATASET ID : adityakadiwal/water-potability

SOURCE : The dataset was created by a Kaggle user named Aditya Kadiwal, collected from various sources about the water parameters.

DESCRIPTION :

1. The dataset provides information about the Water Quality Analysis from various types of water around the world.
2. It includes data on common water parameter and the potability result.

Importance of Visualizing the Dataset :

The importance of visualizing datasets cannot be overstated in the realm of data analysis. Visualization serves as a powerful bridge between raw data and human comprehension. It enhances our ability to understand, interpret, and extract meaningful insights from complex datasets.

By transforming numbers and statistics into charts, graphs, and interactive displays, visualization offers several key advantages. Firstly, it enables us to detect patterns, trends, and outliers that might remain hidden in tabular data, facilitating more accurate and timely decision-making. Moreover, it supports data exploration by allowing users to interact with the data, making it easier to uncover specific details and refine analysis.

This feature is particularly valuable in the age of big data, where sifting through vast datasets can be a formidable challenge. It is a time-saving tool that provides a rapid overview of data, streamlining the analysis process.

Visualizing the dataset :

- Visualizing a dataset in Python is the process of using data visualization libraries like Matplotlib, Seaborn, or Plotly to create graphical representations of data. The goal is to gain insights, identify patterns, and present data in a visual format that is easy to understand.

- Visualizing a dataset in IBM Cognos Analytics is a fundamental aspect of data analysis and interpretation. It involves creating graphical representations of data to uncover patterns, relationships, and insights, making it an essential tool in data-driven decision-making and storytelling.

Identify The Dataset:

The first step is to identify the dataset that you want to load. This dataset may be stored in a local file, in a database, or in a cloud storage service.

Load The Dataset:

Once you have identified the dataset, you need to load it into the IBM Cognos Analytics. This may involve using a built-in function in the machine learning library, or it may involve writing our own code.
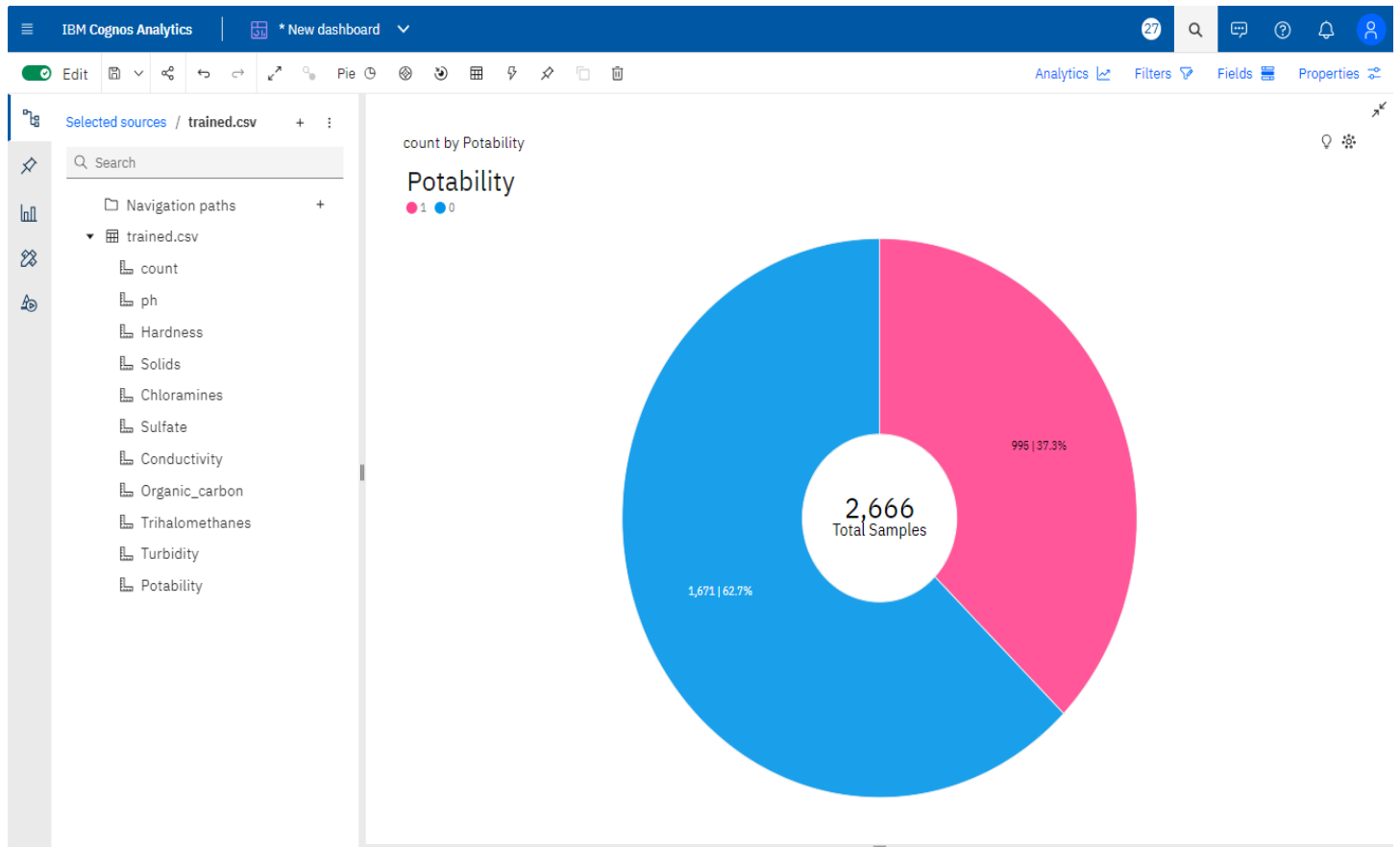
Preprocess The Dataset:

Once the dataset is loaded into the IBM Cognos Analytics Environment, you may need to think a logic before you can start visualizing the dataset.

Visualization of the dataset Using IBM Cognos Analytics

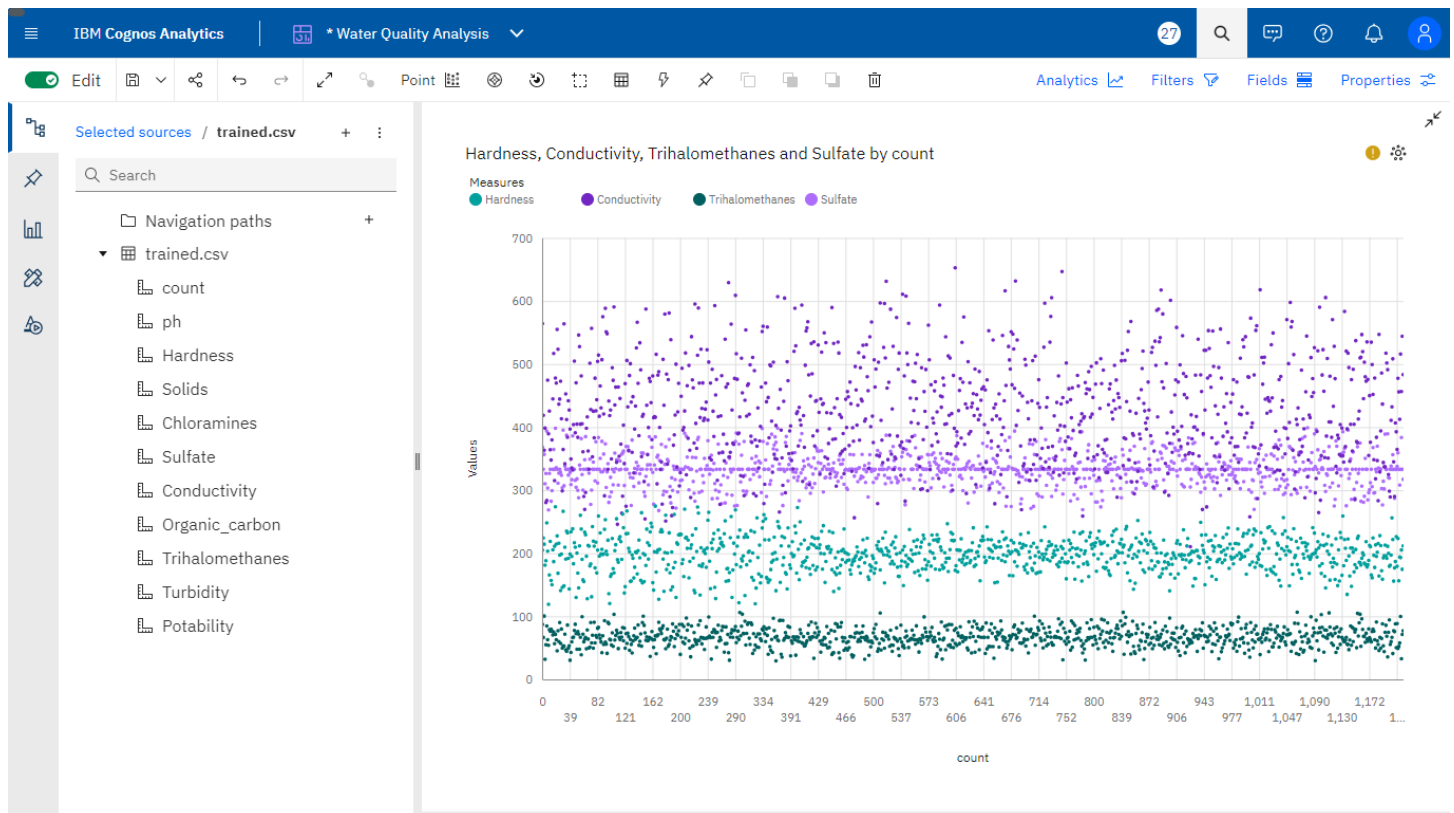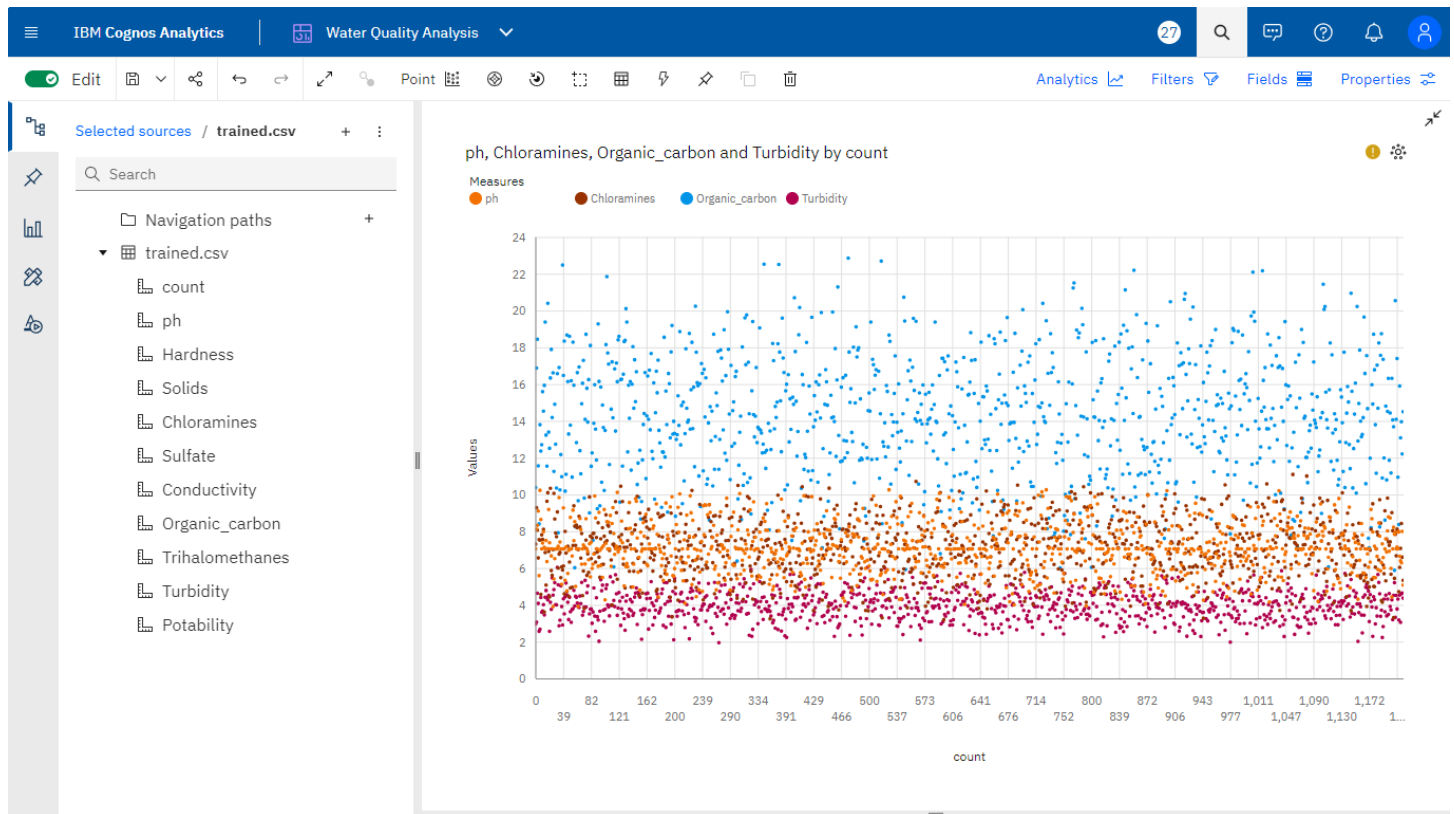## Visualization using Pie Chart :

      Visualizing the Potability data using Pie Chart provided in the Cognos Analytics.

      Here the insights of the Pie chart is the dataset contains 62.7% Not Potable and 37.3% Potable datas among the 2666 Water Samples.
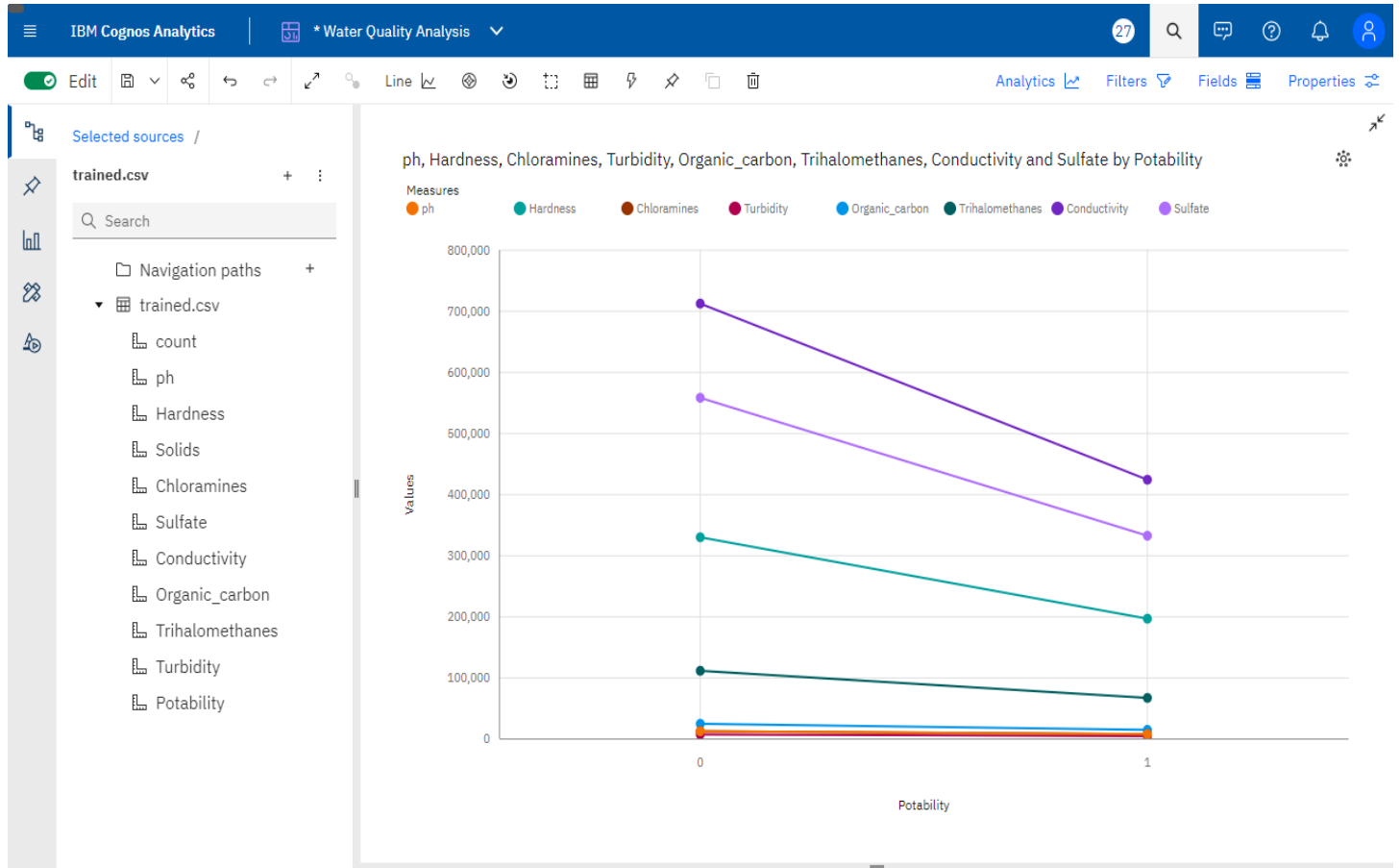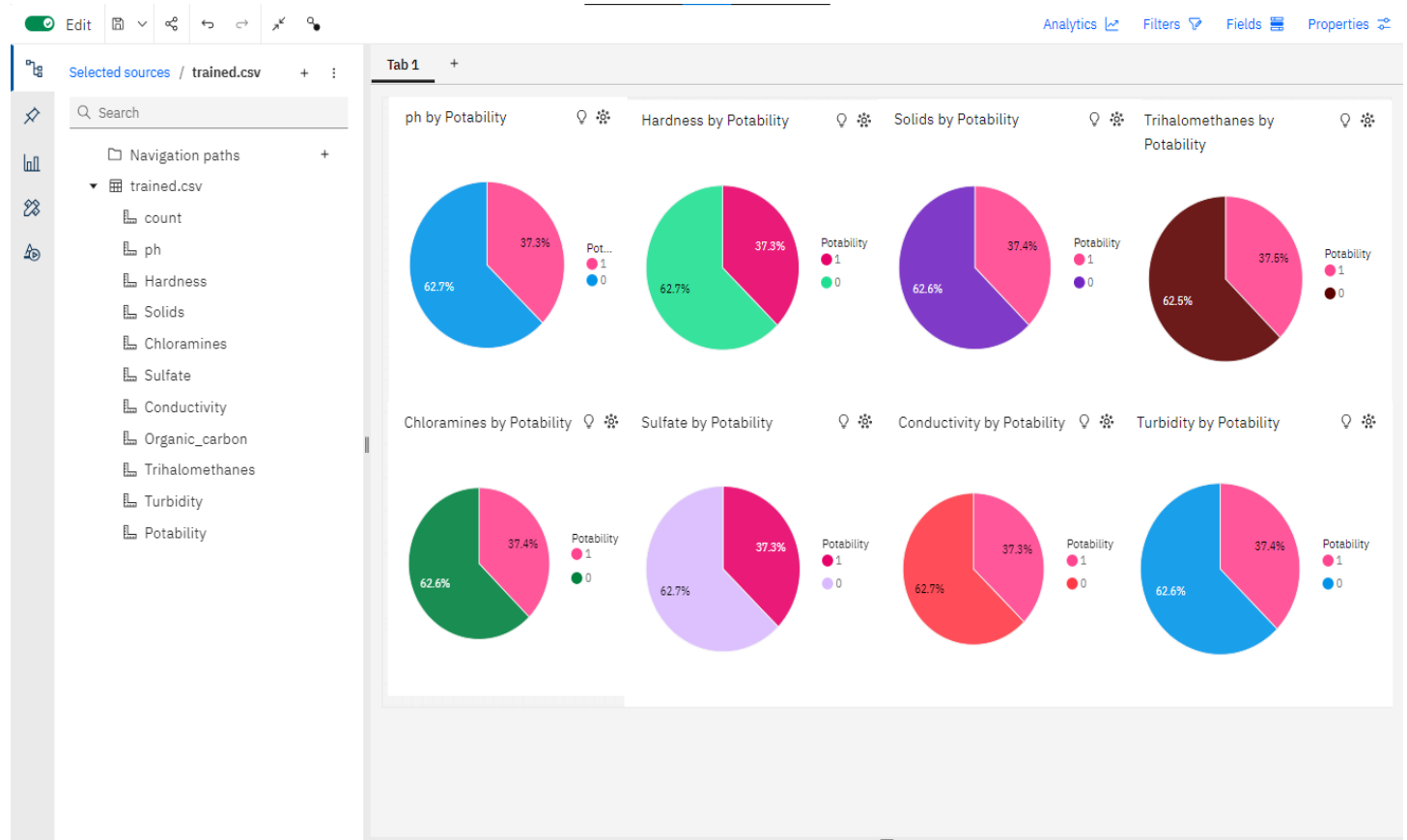
Visualizing using the Points :

Visualizing the Distribution of each Water Quality Parameters provided in the Dataset using Point Chart provided in the Cognos.
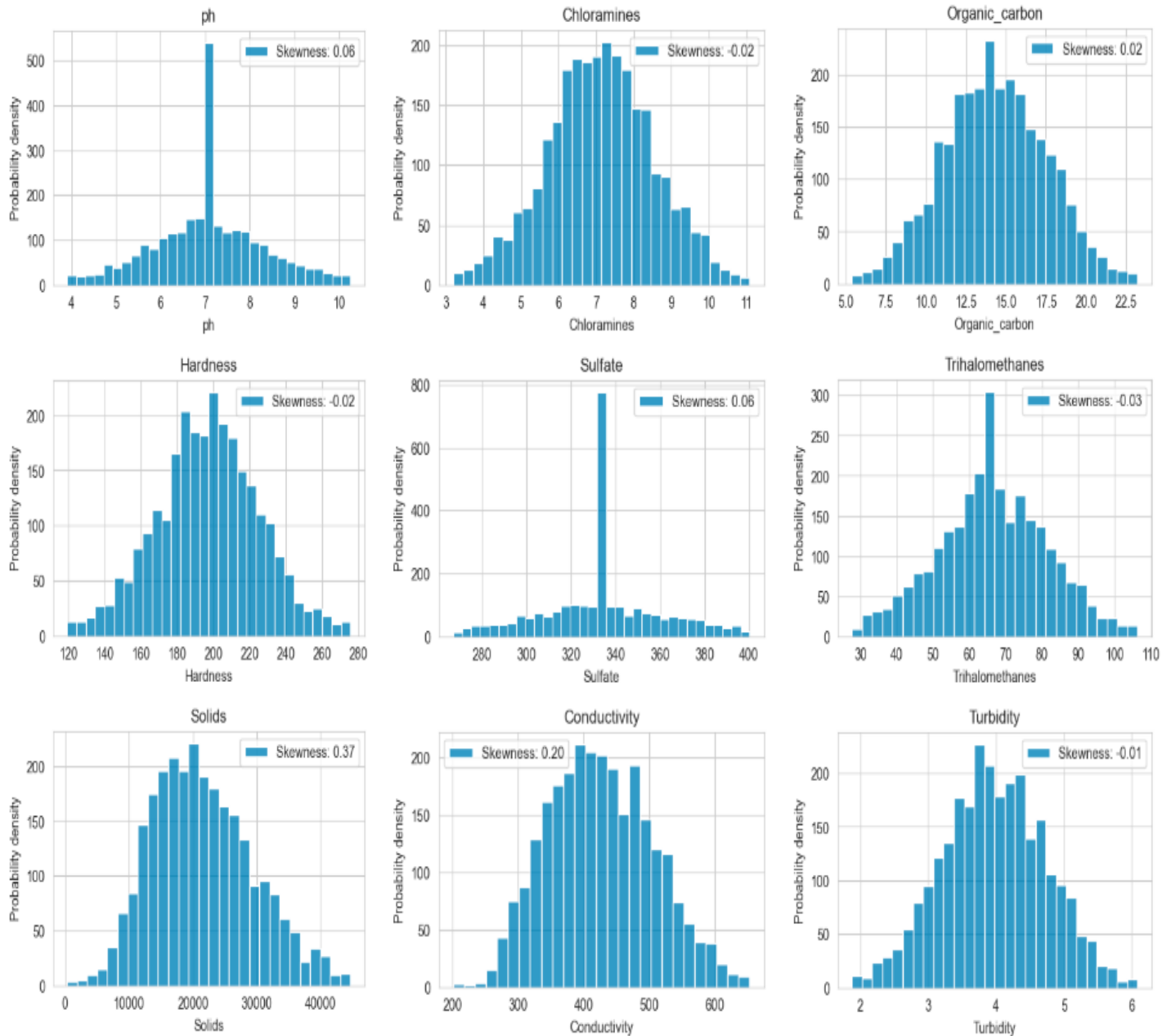
Visualization using Line Chart :

  Visualizing the Potability of each water quality parameters using Line Chart provided in the Cognos Analytics.
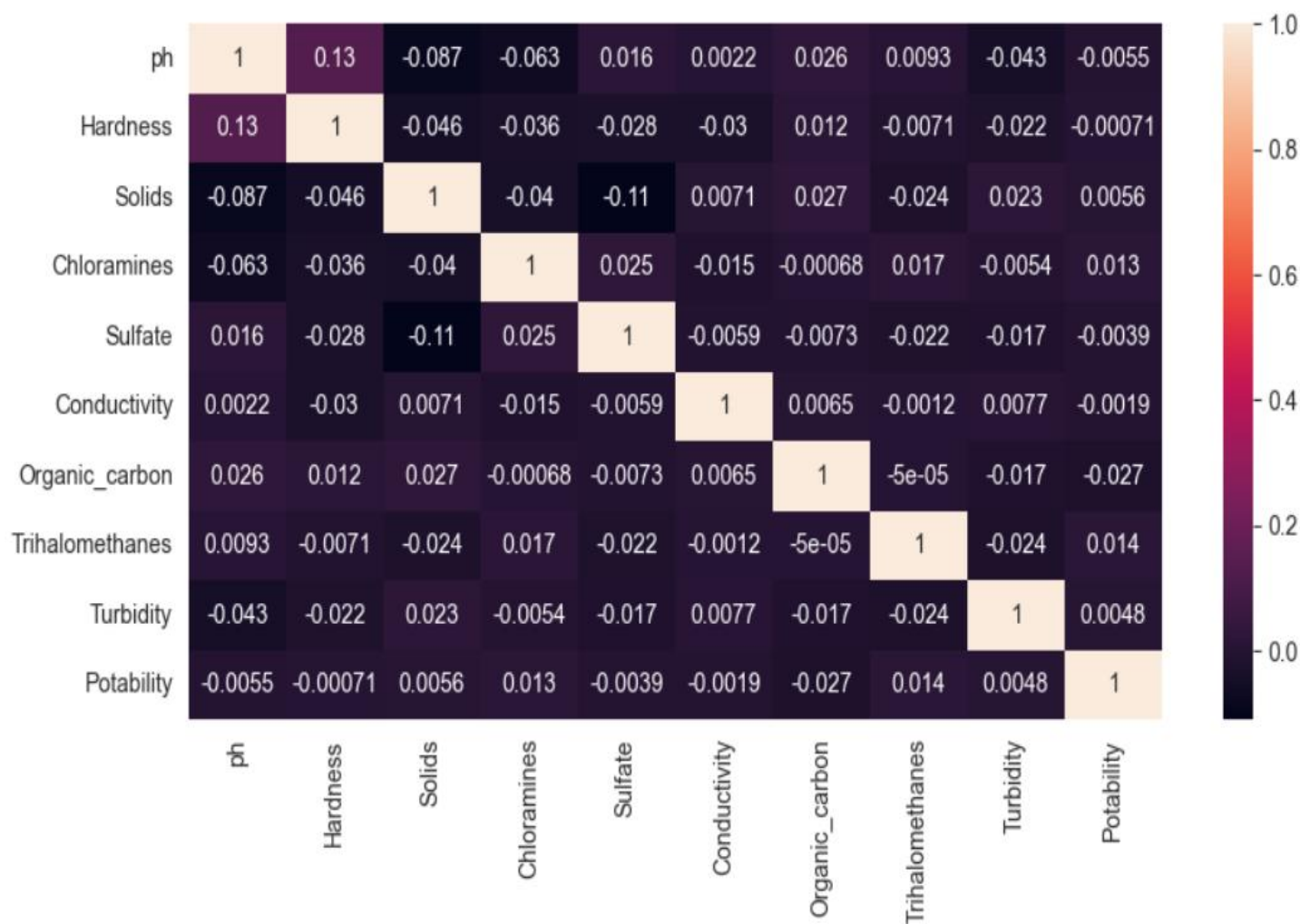
Visualization using the Pie Chart :

Visualizing the Potability of each water quality parameters using Pie Chart provided in the Cognos Analytics.

Visualization using the Histplot :

Visualizing the Potability Density of each water quality parameters using Histplot provided in the Cognos Analytics.

Visualizing Correaltion of the parameters :

RECALL...

In the Development Part 1, We have came across data preprocessing which includes Loading and Cleaning the dataset, Handling the Null values and so on.

In this phase we explore different types of machine learning algorithms to Train and Test the data includes Logistic Regression, K-Nearest Neighbor, Support Vector Machine, Decision Tree Classifier etc.

After the analysis and comparison of different algorithms we work with the Machine Learning Algorithm which gives us more Accuracy than others to make a Water Quality Predictive Model.

SPLITTING THE DATASET INTO TRAINING AND TEST SETS :

```python
### splitting data into x and y
X = df.iloc[: , : -1]
y = df.iloc[ : , -1]
```

```python
# split dataset into train and test
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size = 0.3, random_state= 5)
```

Splitting a dataset into training and test sets is essential for machine learning. It allows us to evaluate a model's performance on unseen data, detect overfitting, fine-tune hyperparameters, and select the best model. This practice ensures reliable, generalizable results and helps make informed decisions when deploying machine learning models in real-world applications.

FEATURE SCALING :

Feature scaling is an important preprocessing step in many machine learning algorithms. You can use the StandardScaler from scikit-learn to scale your features so that they have a mean of 0 and a standard deviation of 1.

```python
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train_final = sc.fit_transform(X_train)
X_test_final = sc.transform(X_test)
```

In this code, we first create a StandardScaler instance. We then fit the scaler on the training data using the fit_transform method, and apply the same transformation to both the training and test data using the transform method. This ensures that the scaling is consistent between the two sets.

MACHINE LEARNING ALGORITHMS :

Machine learning algorithms play a crucial role in data analysis by enabling automated data modeling, pattern recognition, and predictive analytics. Machine learning algorithms enhance data analysis by automating complex tasks, uncovering hidden patterns, and providing data-driven insights.

LIBRARY :

```python
from sklearn.metrics import classification_report, accuracy_score
```

LOGISTIC REGRESSION :

Logistic Regression is a commonly used algorithm for binary and multiclass classification problems. It is a fundamental classification algorithm that models the probability of class membership using the sigmoid function. It estimates parameters to create a decision boundary that separates data points into different classes. Accuracy is used to evaluate the model's performance by comparing its predictions to actual class labels.

```python
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression()
log_reg.fit(X_train_final, y_train)
y_pred = log_reg.predict(X_test_final)
log_acc=accuracy_score(y_test, y_pred)
print(classification_report(y_test, y_pred))
print("Test Set Accuracy : ",log_acc)
```

	precision	recall	f1-score	support
0	0.62	1.00	0.77	497
1	0.00	0.00	0.00	303
accuracy			0.62	800
macro avg	0.31	0.50	0.38	800
weighted avg	0.39	0.62	0.48	800

```
Test Set Accuracy :  0.62125
```

The Test Accuracy of Logistic Regression is 62%

K-Nearest Neighbor Classifier :

     KNN is a simple yet effective supervised machine learning algorithm used for both classification and regression tasks. It operates on the principle of similarity and is based on the idea that data points with similar features are more likely to belong to the same class or have similar target values. KNN is a straightforward algorithm that relies on the concept of similarity to classify or predict data points. It is non-parametric and lazy (as it doesn't build a model during training), making it suitable for various tasks.

```python
from sklearn.neighbors import KNeighborsClassifier


knn = KNeighborsClassifier(n_neighbors=9)
knn.fit(X_train_final, y_train)
y_pred = knn.predict(X_test_final)
knn_acc = accuracy_score(y_test,y_pred)


print(classification_report(y_test, y_pred))
print("Test Set Accuracy : ", knn_acc)
```

	precision	recall	f1-score	support
0	0.65	0.86	0.74	497
1	0.51	0.24	0.33	303
accuracy			0.62	800
macro avg	0.58	0.55	0.53	800
weighted avg	0.60	0.62	0.58	800

Test Set Accuracy : 0.62375

The Test Accuracy of K-Nearest Neighbor is 62%

SUPPORT VECTOR CLASSIFIER :

    SVM is a powerful algorithm for classification and regression tasks that aims to find an optimal hyperplane to separate different classes while maximizing the margin between them. It can handle both linear and nonlinear data, and its performance is evaluated using accuracy metrics on training and test datasets.

```python
from sklearn.svm import SVC

svc_classifier = SVC(class_weight = "balanced"  , C=100, gamma=0.01)
svc_classifier.fit(X_train_final, y_train)
y_pred_scv = svc_classifier.predict(X_test_final)
svm_acc = accuracy_score(y_test, y_pred_scv)

print(classification_report(y_test, y_pred))
print("The Test Accuracy is : ",svm_acc)
```

```
              precision    recall  f1-score   support

           0       0.66      0.87      0.75       497
           1       0.55      0.26      0.35       303

    accuracy                           0.64       800
   macro avg       0.60      0.56      0.55       800
weighted avg       0.62      0.64      0.60       800

The Test Accuracy is :  0.6325
```

The Test Accuracy of Support Vector Classifier is 63%

Decision Tree Classifier :

A Decision Tree is a machine learning algorithm used for both classification and regression tasks. It builds a tree-like structure of decisions based on feature values to classify data. It makes use of impurity measures like entropy to determine the best feature splits. By controlling the tree's depth and evaluating its accuracy, one can create a model that balances between fitting the training data well and generalizing to new data.

```python
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier(criterion='entropy',max_depth=5)
dtc.fit(X_train_final, y_train)
y_pred = dtc.predict(X_test_final)
dtc_acc= accuracy_score(y_test,dtc.predict(X_test_final))

print(classification_report(y_test, y_pred))
print("Test Set Accuracy : ", dtc_acc)
```

	precision	recall	f1-score	support
0	0.63	0.92	0.75	497
1	0.46	0.11	0.17	303
accuracy			0.61	800
macro avg	0.54	0.51	0.46	800
weighted avg	0.56	0.61	0.53	800

Test Set Accuracy : 0.61375

The Test Accuracy of Decision Tree Classifier is 61%

Algorithm Analysis :

     This code facilitates the comparison of different machine learning models by recording and presenting their training and test accuracy in a structured table, sorted by test accuracy for easy assessment.

```python
models = pd.DataFrame({
    'Model': ['Logistic','KNN', 'SVM',  'Decision Tree Classifier'],
    'Test': [ log_acc,knn_acc, svm_acc, dtc_acc]
})

models.sort_values(by = 'Test', ascending = False)
```

	Model	Test
2	SVM	0.63250
1	KNN	0.62375
0	Logistic	0.62125
3	Decision Tree Classifier	0.61375

     This code organizes the performance metrics of different machine learning models in a structured tabular format, sorts the models based on their test accuracy, and provides a clear comparison of how well each model performed on the test dataset. This is a common practice to help data analysts and machine learning practitioners choose the best model for a given task.

Algorithm Visualization :

This code generates a bar plot using Seaborn to visually compare and present the test accuracy of different machine learning algorithms. The choice of colors, style, and figure size enhances the readability and presentation of the plot.

```python
colors = ["green","yellow", "black", "cyan"]
sns.set_style("whitegrid")
plt.figure(figsize=(8,3))
sns.barplot(x=models['Model'],y=models['Test'], palette = colors )
plt.ylabel("Accuracy")
plt.xlabel("Algorithms")
plt.show()
```
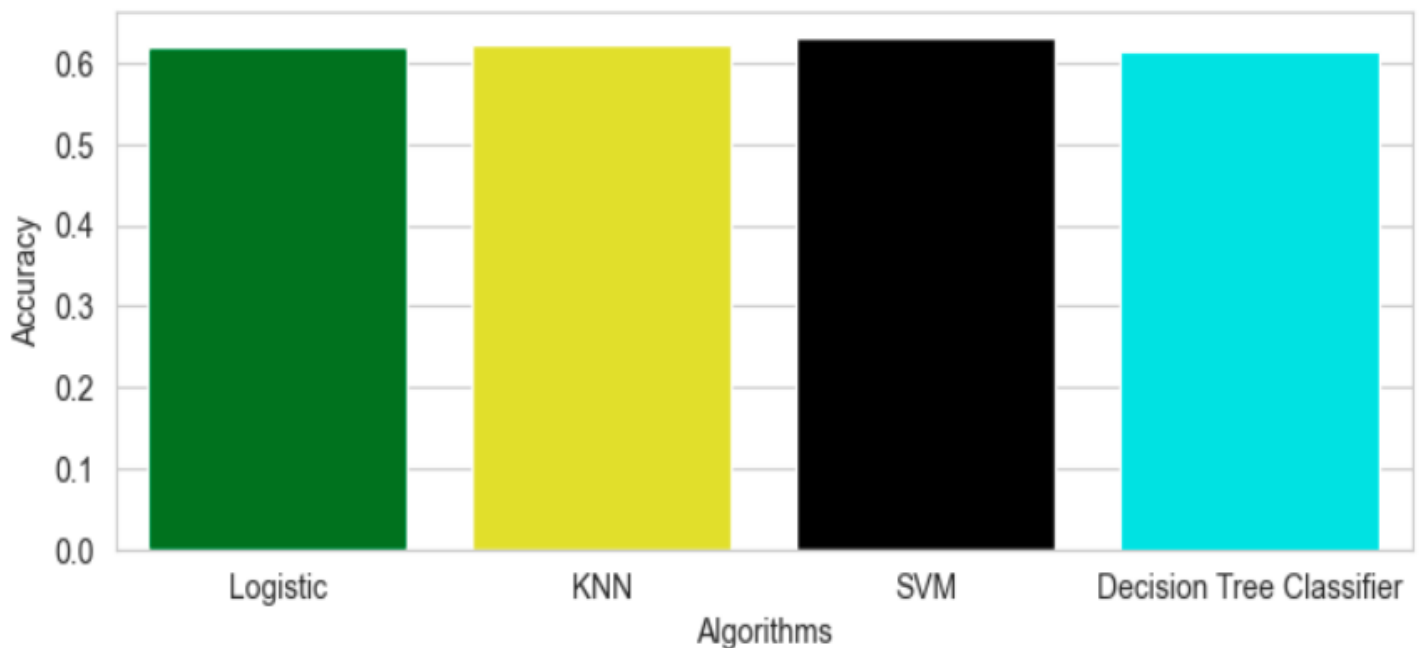

The accuracy score for support vector machine is 63%. As compare with other models the accuracy score is much higher in support vector machine.

Water Quality Predictive Model

In this project, we developed a water quality predictive model using the Support Vector Machine (SVM) algorithm within a Tkinter-based graphical user interface in Python. The model leverages historical water quality data to predict water quality levels based on various parameters, such as pH, Hardness, Solids, Chloramines, Sulphate, Conductivity, Organic_Carbon, Trihalomethanes and Turbidity. The Tkinter GUI allows users to input these parameters and receive real-time predictions, making it a valuable tool for environmental monitoring and ensuring the safety of water resources.

Code :

```python
import tkinter as tk
from tkinter import Entry, Button, Label, Frame

import pickle
import pandas as pd
import numpy as np
import joblib

scaler = joblib.load("final_scaler.save")
model = pickle.load(open('final_model.pkl', 'rb'))

# Function to make a prediction
def predict():
    input_features = [float(entry.get()) for entry in entry_widgets]
    features_value = [np.array(input_features)]
    feature_names = ["ph", "Hardness", "Solids", "Chloramines", "Sulfate", "Conductivity",
"Organic_carbon","Trihalomethanes", "Turbidity"]
    df = pd.DataFrame(features_value, columns=feature_names)
    df = scaler.transform(df)
    output = model.predict(df)

    if output[0] == 1:
        prediction = "safe"
        result_label.config(text="Water is Safe for Human Consumption", fg="#68FF00")
    else:
        prediction = "not safe"
        result_label.config(text="Water is Not Safe for Human Consumption", fg="red")

# Create a Tkinter application window
app = tk.Tk()
app.title("Water Quality Prediction")

# Set the window geometry and background color
app.geometry("470x480")
```

```python
app.configure(bg='#ABFFF1')

# Create a frame for the input fields and labels
input_frame = Frame(app, bg='#ABFFF1')
input_frame.pack(pady=10)

# Create input entry fields with labels
entry_labels = ["pH:", "Hardness:", "Solids:", "Chloramines:", "Sulfate:",
                "Conductivity:", "Organic Carbon:", "Trihalomethanes:", "Turbidity"]
entry_widgets = []

for label_text in entry_labels:
    label = Label(input_frame, text=label_text, bg='#ABFFF1', font=("copperplate gothic
bold", 14,"bold"), fg='black')
    label.grid(row=entry_labels.index(label_text), column=0, sticky='w', padx=10, pady=5)
    entry = Entry(input_frame, font=("Arial", 12))
    entry.grid(row=entry_labels.index(label_text), column=1, padx=10, pady=5)
    entry_widgets.append(entry)

# Create a prediction button with custom style
predict_button = Button(app, text="Predict", command=predict, font=("copperplate gothic
bold", 16,"bold"), bg='#63F07B', fg='black',
                        width = 12)
predict_button.pack(pady=10)

# Create a label to display the prediction with increased font size and custom style
result_label = Label(app, text="", font=("copperplate gothic bold", 14), bg="#ABFFF1")
result_label.pack()

# Start the Tkinter main loop
app.mainloop()
```
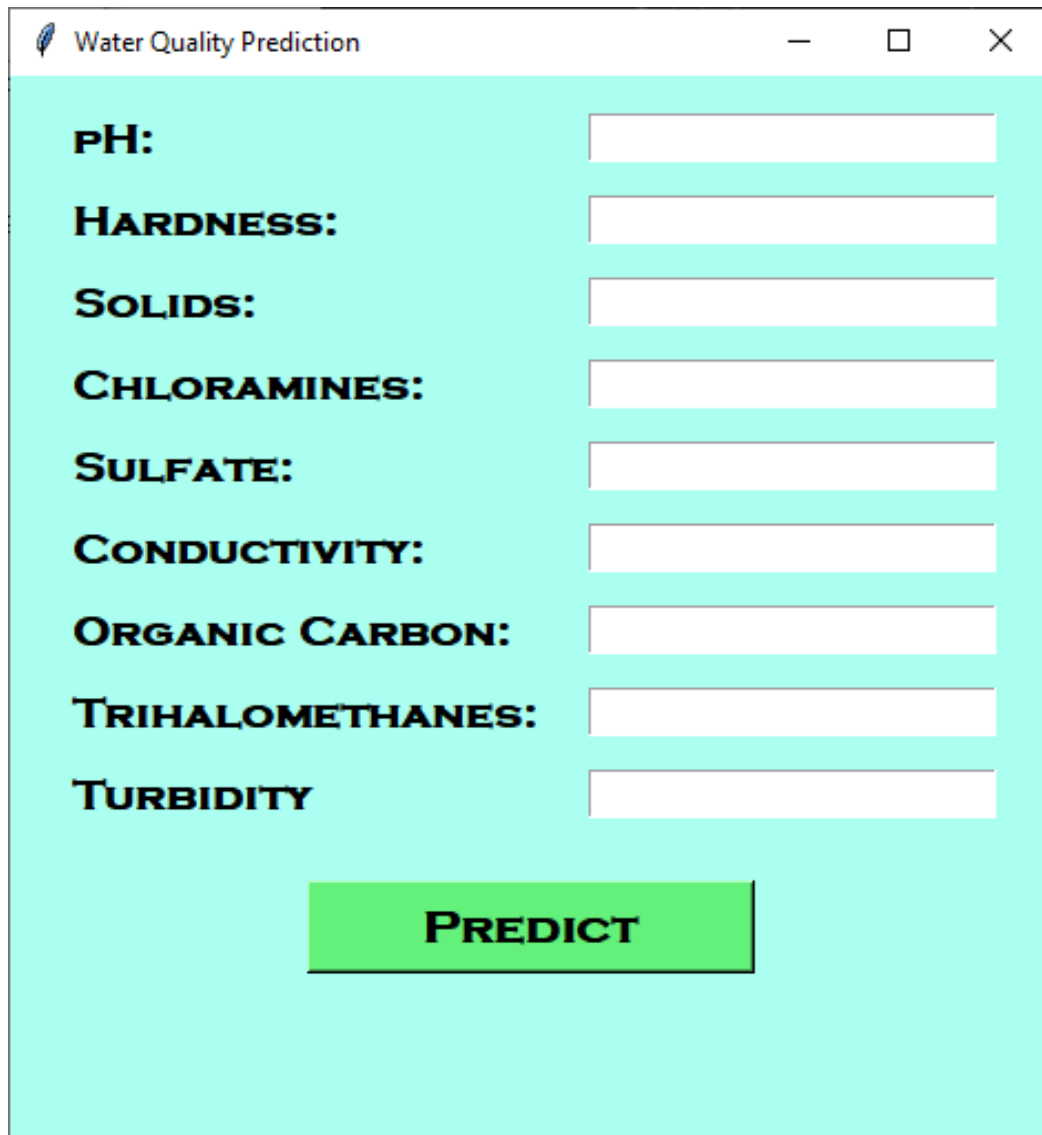
OUTPUT :

HOME INTERFACE :

Let's test the predictive model using the water parameters provided in the dataset.

Testing with Potable values :

Here, we tested the Potable values provided in the dataset. So the Predictive model shows the result i.e "Water is Safe for Human Consumption".

TESTING WITH NOT POTABLE VALUES :

Here, we tested the Not Potable values provided in the dataset. So the Predictive model shows the result i.e "Water is Not Safe for Human Consumption".

CONCLUSION :

- In the quest to build a Water Quality Prediction Model, we have embarked on a critical journey that begins with loading and preprocessing the Water_Potability dataset. We have traversed through essential steps, starting with importing the necessary libraries to facilitate data manipulation and analysis.

- Understanding the data's structure, characteristics, and any potential issues through exploratory data analysis (EDA) is essential for informed decision-making.

- Data preprocessing emerged as a pivotal aspect of this process. It involves cleaning, transforming, and refining the dataset to ensure that it aligns with the requirements of machine learning algorithms.

- With these foundational steps completed, our dataset is now primed for the subsequent stages of building and training a Water Quality Prediction Model.

- Using multiple Machine Learning Algorithms, we concluded highest accuracy algorithm to make the water quality predictive model.

- The Water Quality Predictive Model is trained using SVM algorithm with the provided water samples data.

- Then the Water Quality Predictive Model is tested with several Water Quality Parameters and it provides the result whether its Potable or Not.