**PHASE 5**

**PROJECT DOCUMENTATION & SUBMISSION**

**Project’s Objective:**

* In this project involves analyzing water quality data regulatory standards and determine water potability based on various parameters.
* This project includes defining analysis objectives, collecting water quality data, designing relevant visualizations, and building a predictive model.

**Design Thinking:**

* We continue to build the analysis by creating visualizations and building a predictive model.
* Using Visualization libraries to create histograms, scatter plots, and correlation matrices.
* Here we Build a predictive model such as logistic regression and random forest.

**Development Phases:**

* In this part we will begin building our project by loading and preprocessing the dataset.
* Preprocessing the data and performing exploratory data analysis for the given water quality analysis test.
* Handling missing values and outliers by isnull methods in python.
* Using these models to determine water potability based on water quality parameters.

**DATA PREPROCESSING:**

* We selected Support Vector Machine algorithm.
* In this part we will begin building our project by loading and preprocessing the dataset.
* Preprocessing the data and performing exploratory data analysis for the given water quality analysis test.
* Handling missing values and outliers by isnull methods in python.

**EXPLORATORY DATA ANALYSIS:**

Using Visualization libraries to create histograms, scatter plots, and correlation matrices.

**# IMPORTING PACKAGES**

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler

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

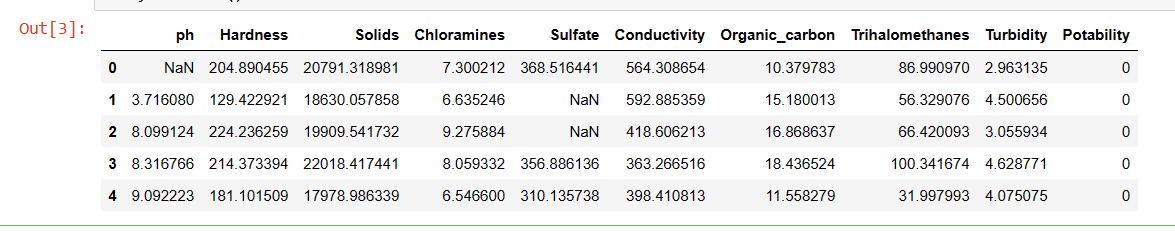
from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

**#Displaying Datasets**

analysis = pd.read\_csv('water\_potability[1].csv')

analysis.head()

**#Output**



**Data Visualization**

**#Heatmap visualization**

plt.figure(figsize = (12,8))

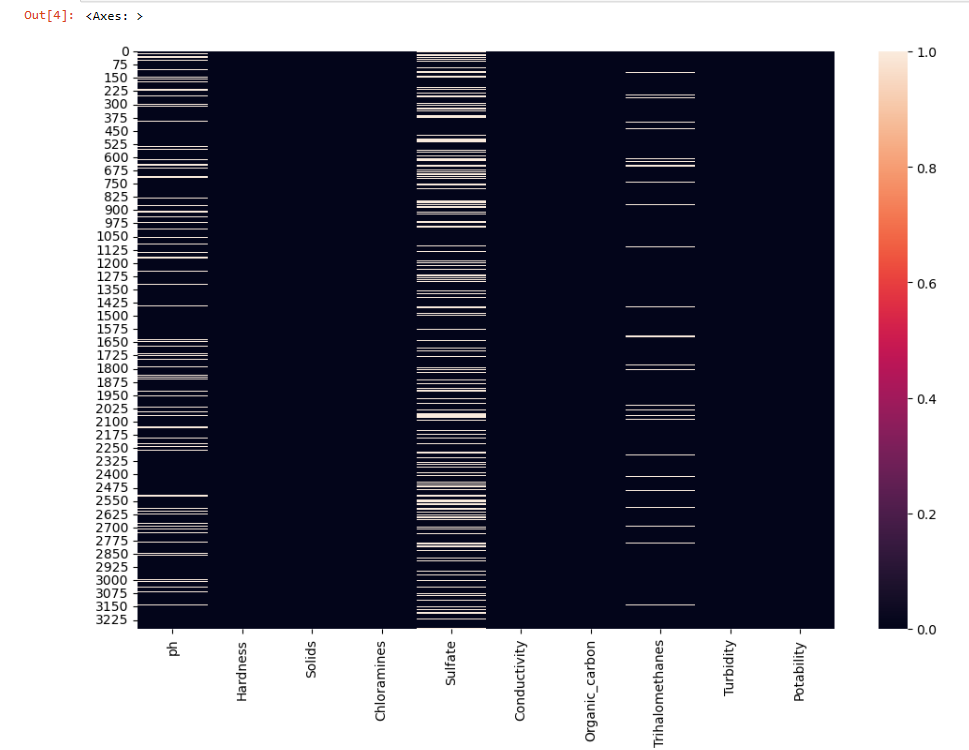
sns.heatmap(analysis.isnull())

sns.heatmap(analysis.isnull())

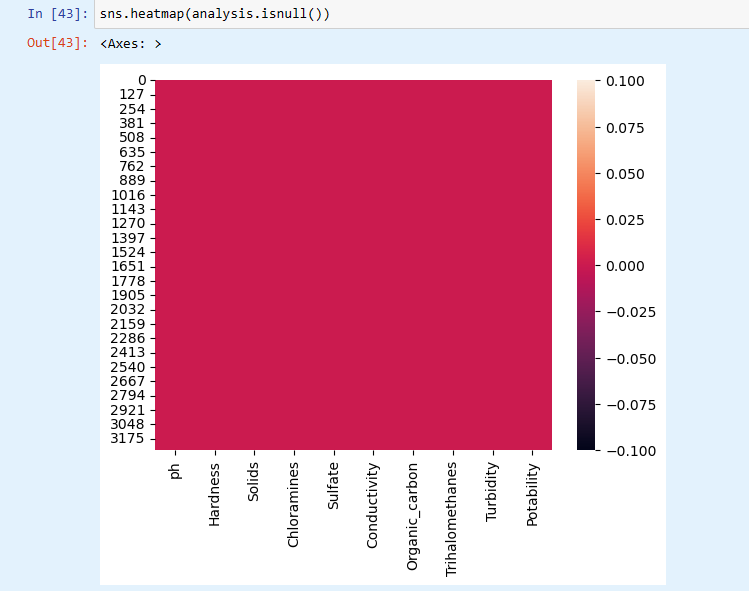
plt.figure(figsize = (12,8))

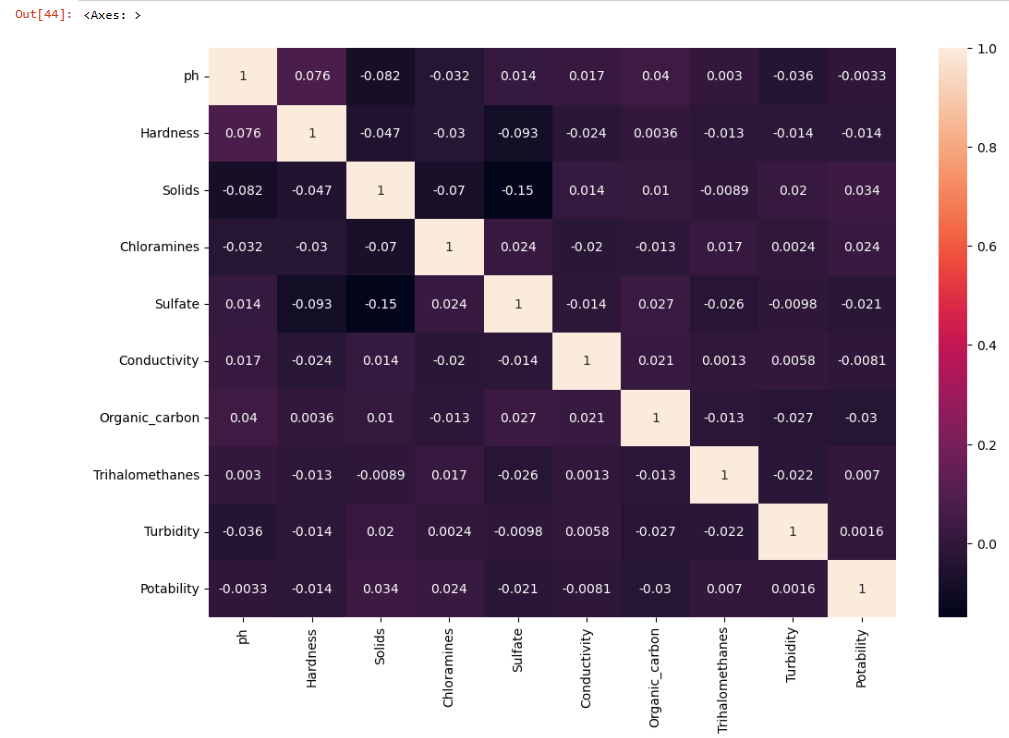
sns.heatmap(analysis.corr(),annot= True)

**#Output 1**



**#Output 2**

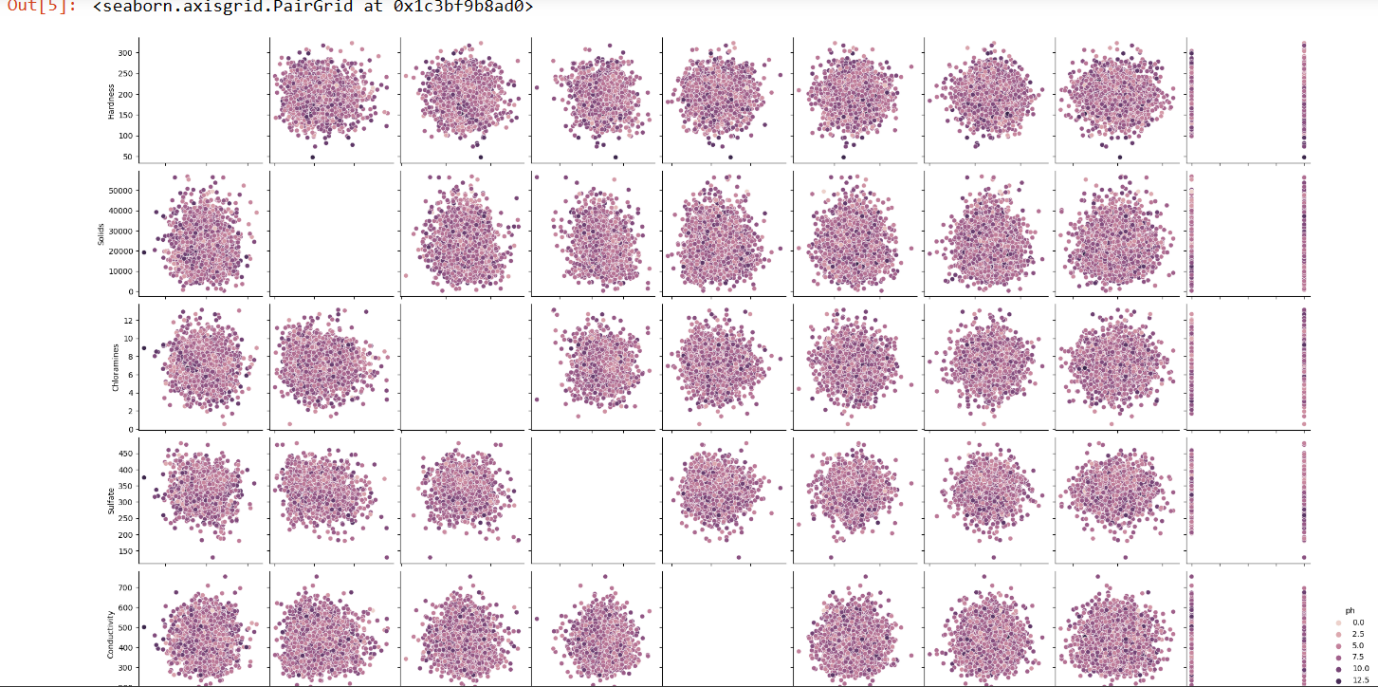


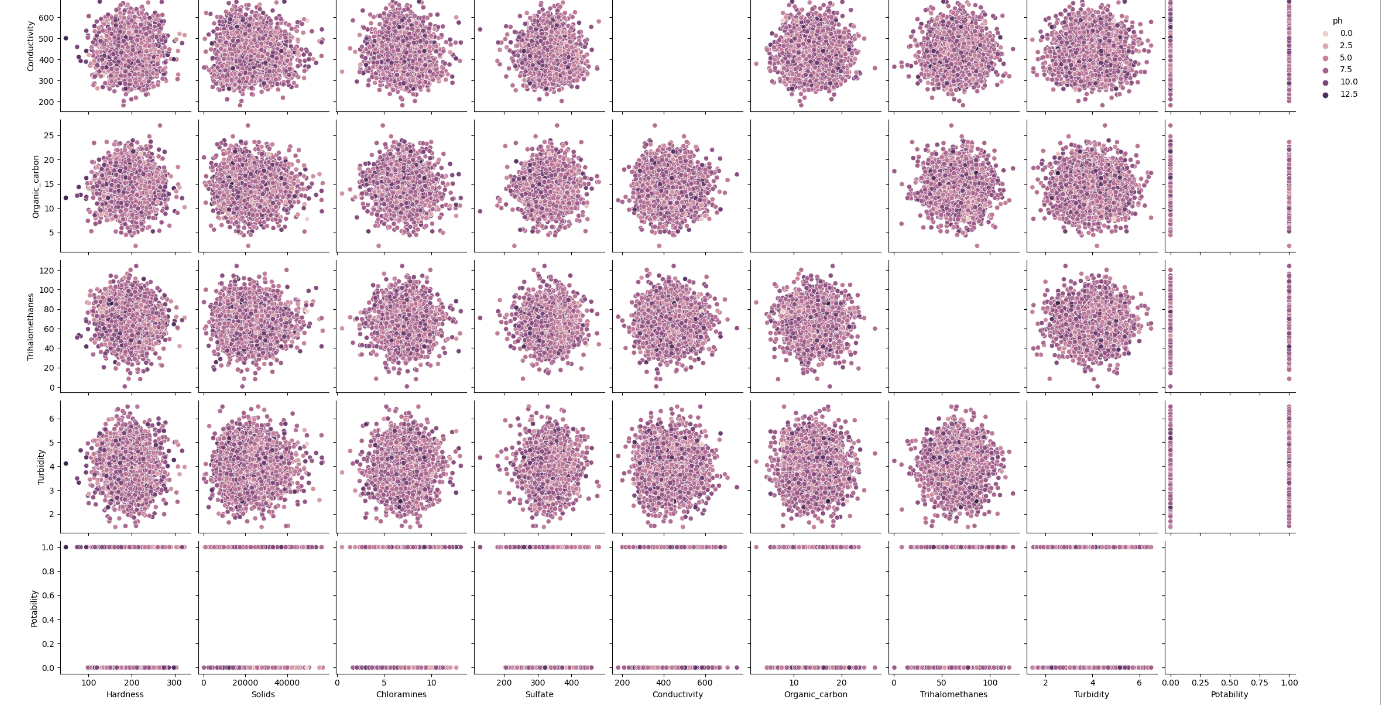


**#Pairplots**

sns.pairplot(analysis, hue="ph")

**#Output**





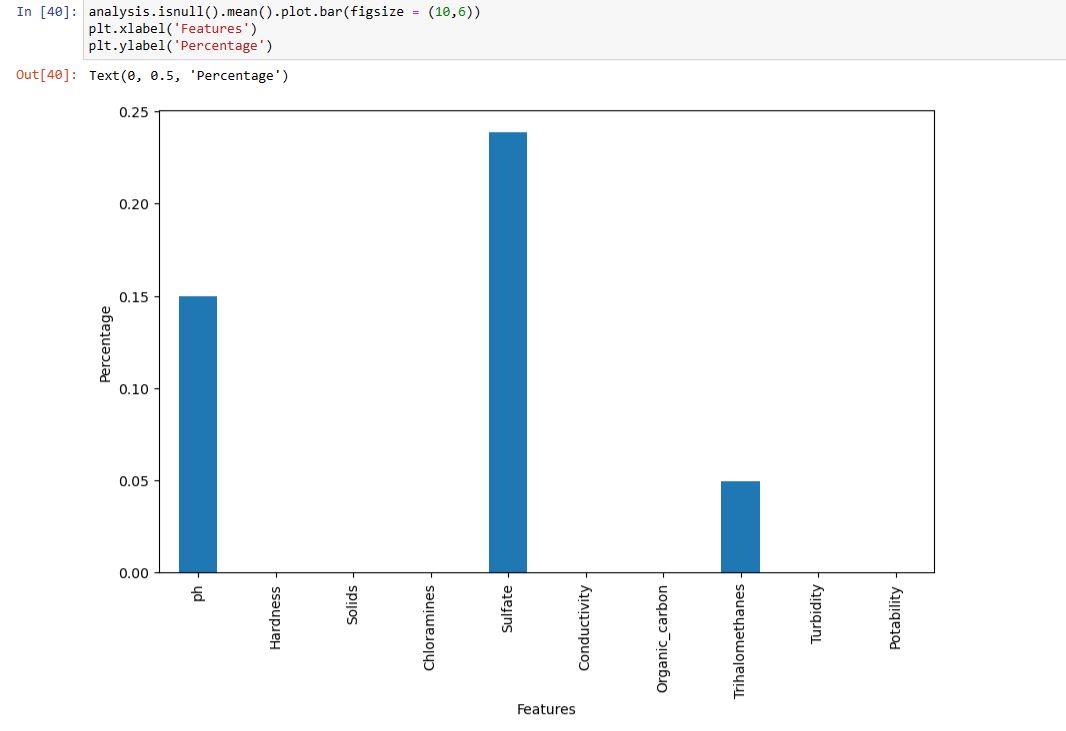
**#Barplots**

analysis.isnull().mean().plot.bar(figsize = (10,6))

plt.xlabel('Features')

plt.ylabel('Percentage')

**#Output**



**#Handling Missing Values**

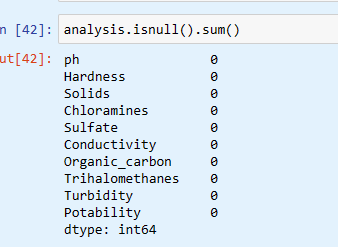
analysis['ph'] = analysis['ph'].fillna(analysis['ph'].mean())

analysis['Sulfate'] = analysis['Sulfate'].fillna(analysis['Sulfate'].mean())

analysis['Trihalomethanes'] = analysis['Trihalomethanes'].fillna(analysis['Trihalomethanes'].mean())

analysis.isnull().sum()

**#Output**



x = analysis.drop("Potability",axis = 1)

y = analysis['Potability']

x.shape, y.shape

scaler = StandardScaler()

x = scaler.fit\_transform(x)

x

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size = 0.2)

x\_train.shape, x\_test.shape

**#Support Vector Machine**

from sklearn.svm import SVC

model\_svm = SVC(kernel='rbf')

model\_svm.fit(x\_train,y\_train)

pred\_svm = model\_svm.predict(x\_test)

accuracy\_score\_svm = accuracy\_score(y\_test,pred\_svm)

accuracy\_score\_svm\*100

**#Output**

**68.14024390243902**

**#Histogram**

Value = analysis['Hardness'].mean()

plt.figure(figsize=(12,6))

plt.subplot(1,2,1)

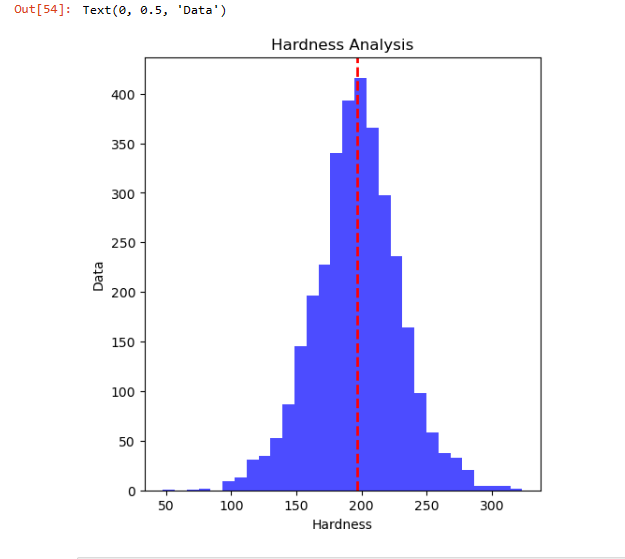
plt.hist(analysis['Hardness'],bins=30,color='blue',alpha=0.7)

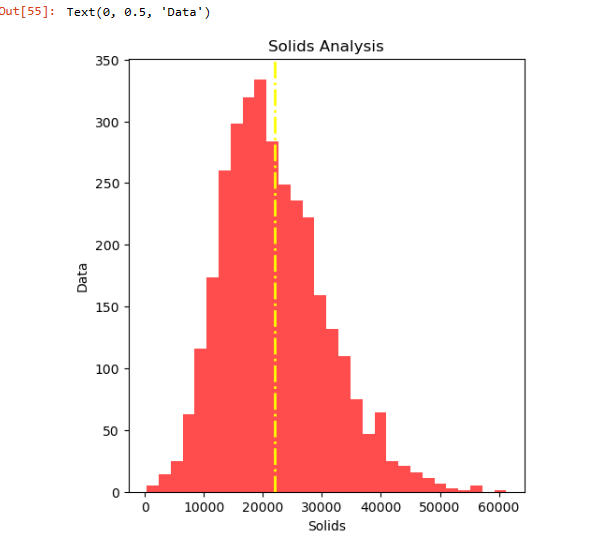
plt.axvline(Value, color='red', linestyle='dashed', linewidth=2)

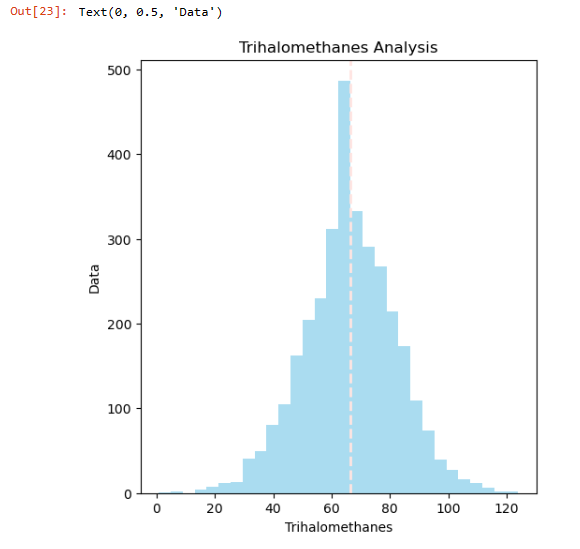
plt.title('Hardness Analysis')

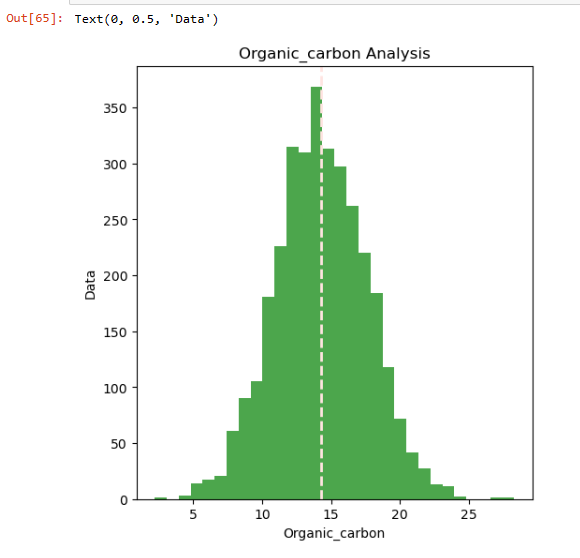
plt.xlabel('Hardness')

plt.ylabel('Data')

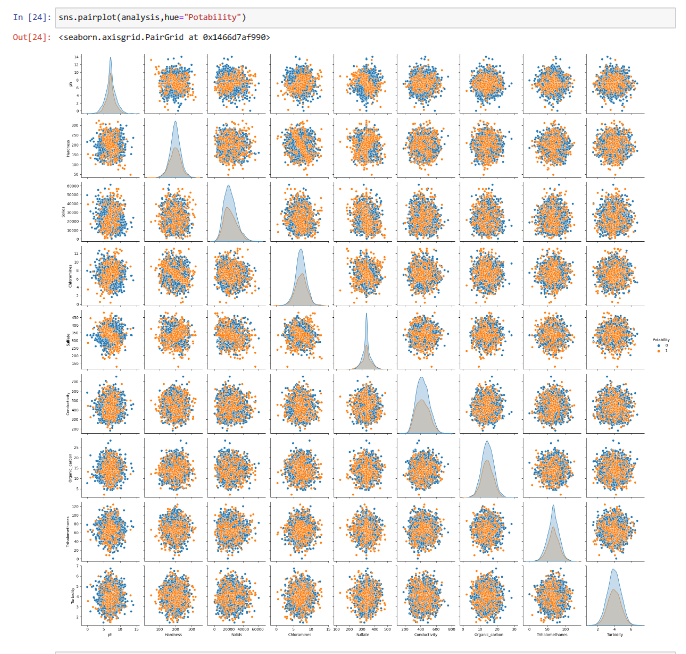








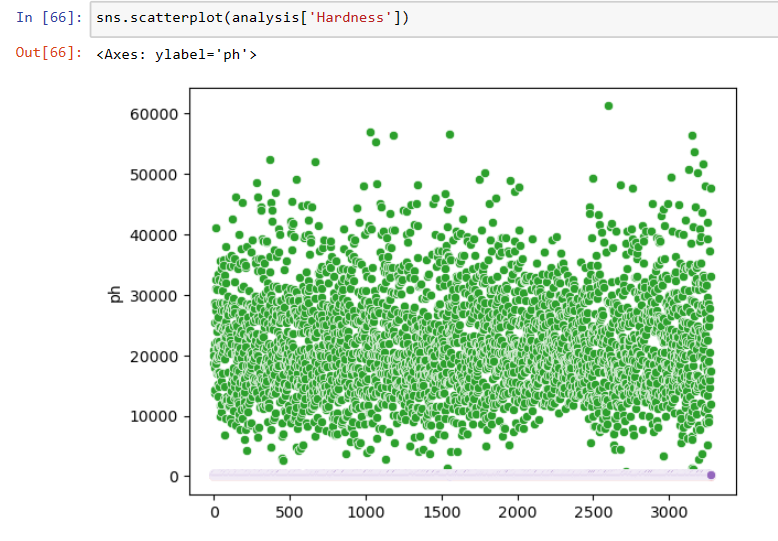
sns.pairplot(analysis,hue="Potability")

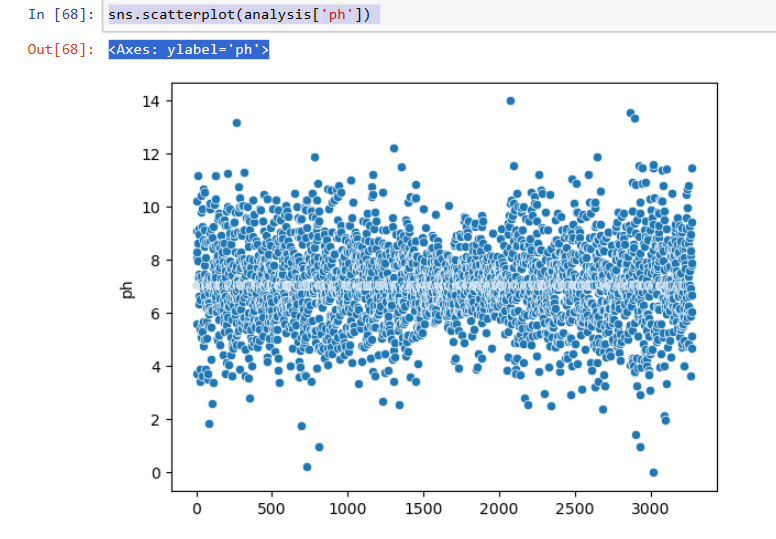


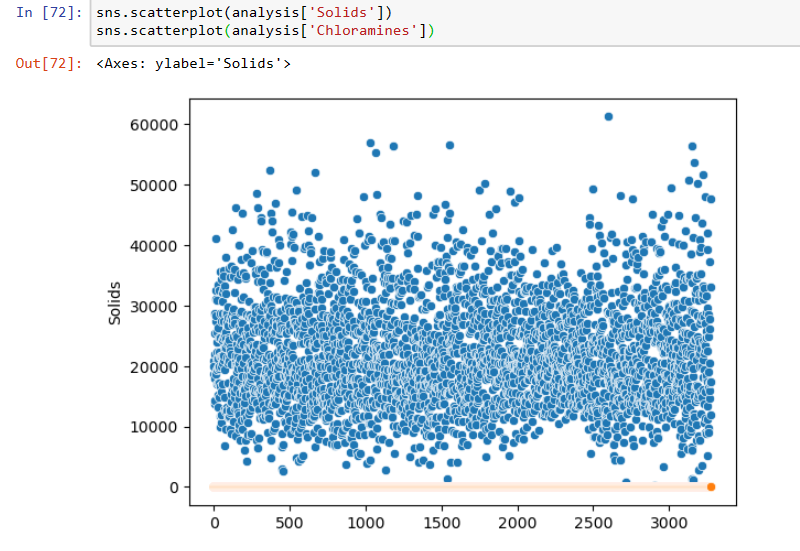
**#ScatterPlots**

sns.scatterplot(analysis['Hardness'])

sns.scatterplot(analysis['ph'])

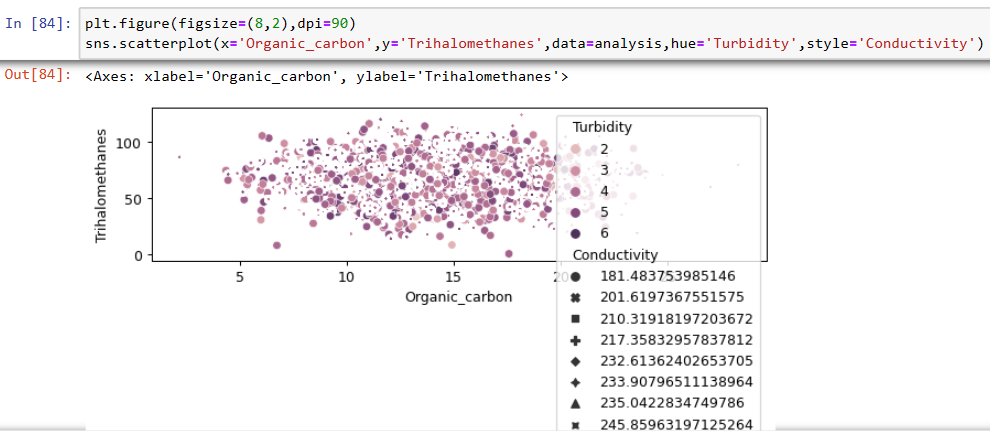






plt.figure(figsize=(8,2),dpi=90)

sns.scatterplot(x='Organic\_carbon',y='Trihalomethanes',data=analysis,hue='Turbidity',style='Conductivity')

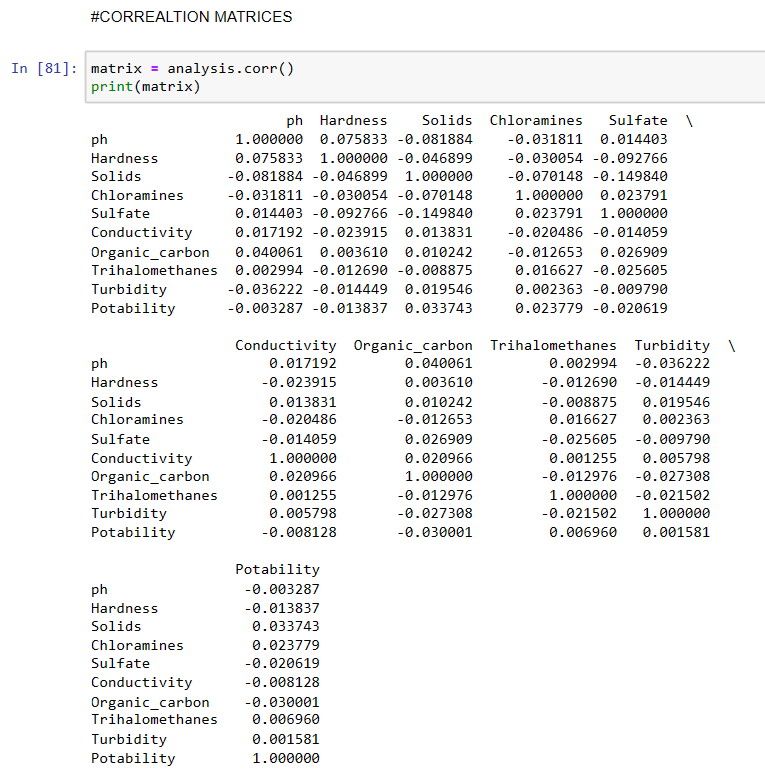


**#Corelation Matrices**

matrix = analysis.corr()

print(matrix)

**#Output**



**PREDICTIVE MODELING**

**#Logistic Regression**

from sklearn.linear\_model import LogisticRegression

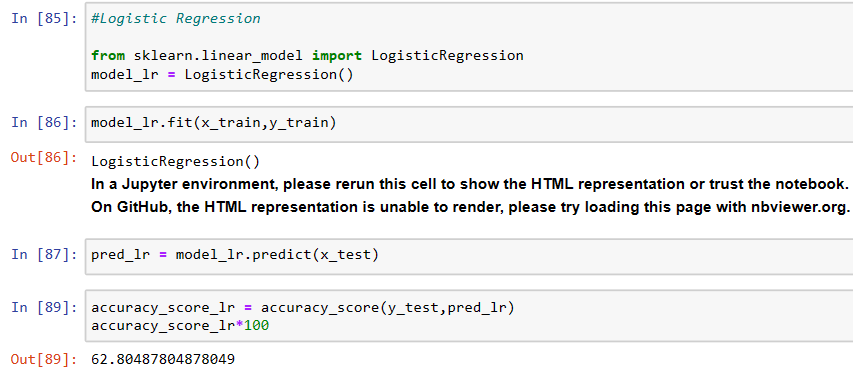
model\_lr = LogisticRegression()

model\_lr.fit(x\_train,y\_train)

pred\_lr = model\_lr.predict(x\_test)

accuracy\_score\_lr = accuracy\_score(y\_test,pred\_lr)

accuracy\_score\_lr\*100



**#Predictive Analysis Logistic Regression**

**62.80487804878049**

**#Random Forest**

from sklearn.ensemble import RandomForestClassifier

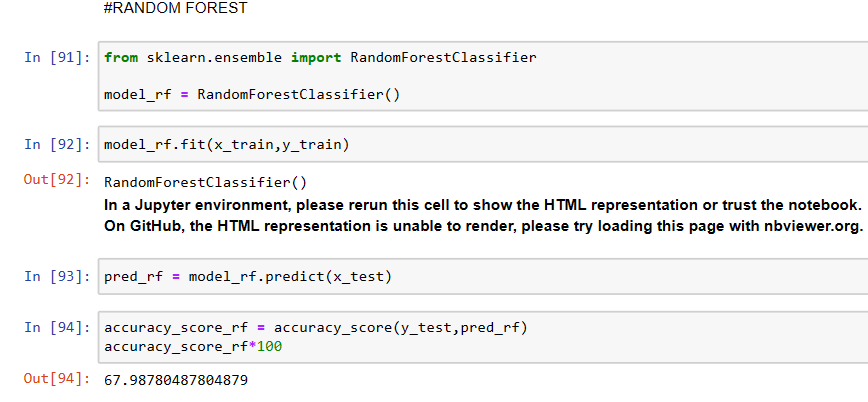
model\_rf = RandomForestClassifier()

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

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

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

accuracy\_score\_rf\*100



**#Predictive Analysis Random Forest**

**67.98780487804879**