B.P.H.E. SOCIETY'S

# AHMEDNAGAR COLLEGE, AHMEDNAGAR, 414001

T.Y.B.Sc. – FINAL YEAR 2022-23

**DEPARTMENT OF STATISTICS**

A PROJECT ON

# "STATISTICAL ANALYSIS OF WATER QUALITY USING PREDICTIVE MODELING AND MACHINE LEARNING TECHNIQUES"

**Submitted by:**

Samruddhi Rohalke

Tejashri Tapare

Rajesh Rasane

Zahra Merchant

Zoha Bahlooli

Monika Borude

**Guided by:**

Prof. Yogesh Yewale

AHMEDNAGAR COLLEGE

DEPARTMENT OF STATISTICS

2022-23

B.P.H.E SOCIETY'S

AHMEDNAGAR COLLEGE,

AHMEDNAGAR – 414001

# CERTIFICATE

This is to certify that

Ms. Samruddhi Rohakale

Ms. Tejashri Tapare

Mr. Rajesh Rasane

Ms. Zahra Merchant

Ms. Zoha Bahlooli

Ms. Monika Borude

Of class T.Y.B.Sc. Statistics has

Satisfactorily completed project on

**"Statistical Analysis of Water Quality using Predictive Modeling and Machine Learning Techniques"**

For the academic year 2022-23

| | |
|---|---|
| Prof. Yogesh Yewale | Dr. Malti Yeola |
| Project Guide | Head of the Department |

# TABLE OF CONTENTS

# ACKNOWLEDGEMENT

Any Project completed successfully offers a great sense of achievement and satisfaction. The project would remain incomplete if the people who made it possible and whose guidance and encouragement go without mention.

We wish to express our cofound gratitude and indebtedness to our estimated guide Yogesh Yewale Sir, Department of Statistics AHMEDNAGAR COLLEGE, AHMEDNAGAR for his expert guidance and scholarly supervision, endless motivation, encouragement throughout the execution of this work.

We are also grateful to the faculties of the department of statistics AHMEDNAGAR COLLEGE, AHMEDNAGAR, those who have directly and indirectly rendered their valuable contribution while preparing our project. We are honored to express our deep sense of gratitude for principal **DR. R. J. Barnabas**, who has always sent an example of hard work and destination to all students.

Finally, we would like to extend a deep appreciation to all those associated with this project for having shared a genuine desire to make a positive contribution to address the challenges associated with every element of this project.

# MOTIVATION

Water plays the very crucial role in our life. Also, quality of drinkable water is very important. The polluted water affects environment greatly and leads to many waterborne diseases such as Cholera, Diarrhea, Typhoid, Hepatitis and various skin diseases. In village we need to check the water quality every time when the water is supplied to villagers.

Motivation behind the project was to check the water quality. Main motive of this project is to classify whether the water is drinkable or not. So that we provide quality of water to the people. The sample of water is analyzed in the lab or any small place before it is supply, for that we check the parameters like Ph, Hardness, Solids, Chloramines, Sulfate, Conductivity, Organic_Carbon, Trihalomethanes, Turbidity which have standard set of values. For these values we are testing a hypothesis for all parameters so that we can conclude the given water sample (population) is potable or not and lastly, we are doing model fitting in which we are searching for best fitting model with which we can estimate (classify) potability for given data.

# INTRODUCTION

Water is one of the most fundamental necessities of life. Life would not be nearly possible without water. Quality of drinkable water is very important. We can't decide water quality by our naked eyes. To check the quality of water we use an automated tool using machine learning based classification techniques is proposed in the paper. Different classifiers are used to check whether the given sample of water is drinkable or not and the results of those classifiers are decided for water quality. There are lot of classification model like Naïve bayes, Logistic regression, KNN, SVM, etc. However, experimental results indicate that ensemble classifiers are the best classification to check the water quality. This project uses data provided from Kaggle. This data contains components which are in the water.

This project follows 5 stages. The 5 stages adopted for this project are –

1. **Problem Definition (Project Overview, Project Statement and Metrics)**
2. **Data Collection**
3. **Data Cleaning, exploring and pre-processing**
4. **Modeling**
5. **Evaluating**

# TERMINOLOGY

**pH value:** pH is an important parameter in evaluating the acid-base balance of water. It is also the indicator of acidic or alkaline condition of water status. WHO has recommended maximum permissible limit of pH from 6.5 to 8.5. The current investigation ranges were 6.52 – 6.83 which are in the range of WHO standards.

**Hardness:** Hardness is mainly caused by calcium and magnesium salts. These salts are dissolved from geologic deposits through which water travels. The length of time water is in contact with hardness producing material helps determine how much hardness there is in raw water. Hardness was originally defined as the capacity of water to precipitate soap caused by Calcium and Magnesium.

**Solids (Total dissolved solids – TDS):** Water has the ability to dissolve a wide range of inorganic and some organic minerals or salts as potassium, calcium, sodium, bicarbonates, chlorides, magnesium, Sulfates etc. These minerals produced un-wanted taste and diluted color in appearance of water. This is the importance parameter for the use of water. The water with high TDS value indicates that water is highly mineralized. Desirable limit for TDS is 500 mg/l and maximum limit is 1000 mg/l which prescribed for drinking purpose.

**Chloramines:** Chlorine and chloramine are the major disinfectants used in public water systems. Chloramines are most commonly formed when ammonia is added to chlorine to treat drinking water. Chlorine levels up to 4 milligrams per liter (mg/L or 4 parts per million (ppm)) are considered safe in drinking water.

**Sulfate:** Sulfates are naturally occurring substances that are found in minerals, soil, and rocks. They are present in ambient air, groundwater, plants, and food. The principal commercial use of Sulfate is in the chemical industry. Sulfate concentration in seawater is about 2,700 milligrams per liter (mg/L). It ranges from 3 to 30 mg/L in most freshwater supplies, although much higher concentrations (1000 mg/L) are found in some geographical locations.

**Conductivity:** Pure water is not a good conductor of electric current rather it's a good insulator. Increase in ions concentration enhances the electrical conductivity of water. Generally, the number of dissolved solids in water determines the electrical conductivity. Electrical conductivity (EC) actually measures the ionic process of a solution that enables it to transmit current. According to WHO standards, EC value should not exceed 400 μS/cm.

**Organic_carbon:** Total Organic Carbon (TOC) in source water comes from decaying natural organic matter (NOM) as well as synthetic sources. TOC is a measure of the total amount of carbon in organic compounds in pure water. According to US EPA < 2 mg/L as TOC in treated/drinking water, and < 4 mg/L in source water which is use for treatment.

**Trihalomethanes:** THMs are chemicals which may be found in water treated with chlorine. The concentration of THMs in drinking water varies according to the level of organic material in the water, the amount of chlorine required to treat the water, and the temperature of the water that is being treated. THM levels up to 80 ppm is considered safe in drinking water.

**Turbidity:** The turbidity of water depends on the quantity of solid matter present in the suspended state. It is a measure of light emitting properties of water and the test is used to indicate the quality of waste discharge with respect to colloidal matter. The mean turbidity value obtained for Wondo Genet Campus (0.98 NTU) is lower than the WHO recommended of 5.00 NTU.

**Potability:** Indicates if water is safe for human consumption where 1 means Potable and 0 means Not Potable.

# HYPOTHESIS TESTING

Hypothesis testing is one of the most important concepts in Statistics which is heavily used by statisticians, Machine Learning Engineers, and Data Scientists. In hypothesis testing, statistical tests are used to check whether the null hypothesis is rejected or not rejected. These statistical tests assume a null hypothesis of no relationship or no difference between groups.

## Parametric and Non-Parametric Test:

**Parametric** tests are those tests for which we have prior knowledge of the population distribution (i.e., normal), or if not then we can easily approximate it to a normal distribution which is possible with the help of the Central Limit Theorem.

In **Non-Parametric tests**, we don't make any assumption about the parameters for the given population or the population we are studying. In fact, these tests don't depend on the population.

As our project aim is to check whether water is drinkable or not i.e., the given water is potable or not potable. So, for testing this claim we have to perform the hypothesis testing for each parameter (pH, Hardness, Solids, Chloramines, Sulfate, Conductivity, Organic_carbon, Trihalomethanes, Turbidity).

| | |
|---|---|
| **pH** | **6.5 – 8.5** |
| **Hardness** | **60mg/L – 180mg/L (or ppm) of $CaCO_3$** |
| **Solids** | **less than 500 mg/L – 2000 mg/L (or ppm) of TDS** |
| **Chloramines** | **less than 4 mg/L (or ppm)** |
| **Sulfate** | **less than 250 mg/L (or ppm)** |
| **Conductivity** | **50 – 250 µS/cm** |
| **Organic_carbon** | **less than 2 mg/L (or ppm)** |
| **Trihalomethanes** | **less than 80 µg/L (or ppb)** |
| **Turbidity** | **less than 0.3 NTU** |

**CLAIM:** If all the parameters value are in the specified range, then we can conclude that the water is potable, otherwise it is not potable.

So, for checking this claim we perform the hypothesis testing for a given sample data. The first step to proceed by parameter testing we want to check whether the given sample is coming from normal population or not.

*Let's check the normality of each parameter by Shapiro test.*

**Import Data Set:**

```
library(readxl)

waterqualitydata <-
read_excel ("D:/Users/Zahra/Desktop/water_potability.xlsx")

view(waterqualitydata)
```

**Shapiro-Wilk normality test (l.o.s = 5%)**

**To test: $H_0$ = pH is normally distributed**

**$H_1$ = pH is not normally distributed.**

```
> shapiro.test(waterqualitydata$ph)

	Shapiro-Wilk normality test

data:  waterqualitydata$ph
W = 0.99587, p-value = 5.727e-07
> shapiro.test(waterqualitydata$Hardness)

	Shapiro-Wilk normality test

data:  waterqualitydata$Hardness
W = 0.99597, p-value = 9.61e-08

> shapiro.test(waterqualitydata$Solids)

	Shapiro-Wilk normality test

data:  waterqualitydata$Solids
W = 0.97773, p-value < 2.2e-16
```

```
> shapiro.test(waterqualitydata$Chloramines)

	Shapiro-Wilk normality test

data:  waterqualitydata$Chloramines
W = 0.99677, p-value = 1.818e-06

> shapiro.test(waterqualitydata$Sulfate)

	Shapiro-Wilk normality test

data:  waterqualitydata$Sulfate
W = 0.99602, p-value = 3.467e-06

> shapiro.test(waterqualitydata$Conductivity)

	Shapiro-Wilk normality test

data:  waterqualitydata$Conductivity
W = 0.99297, p-value = 1.494e-11

> shapiro.test(waterqualitydata$Organic_carbon)

	Shapiro-Wilk normality test

data:  waterqualitydata$Organic_carbon
W = 0.99952, p-value = 2.094e-11

> shapiro.test(waterqualitydata$Trihalomethanes)

	Shapiro-Wilk normality test

data:  waterqualitydata$Trihalomethanes
W = 0.99886, p-value = 0.03479
```

```
> shapiro.test(waterqualitydata$Turbidity)

        Shapiro-Wilk normality test

data:  waterqualitydata$Turbidity
W = 0.9997, p-value = 0.0236
```

## Conclusion:

*As p-value for each test is less than 0.05 so we reject null hypothesis at 5% l.o.s*

By seeing the output of Shapiro test, we can easily conclude that the data does not follow normal distribution. So, we should go with corresponding non parametric test.

There exists a suitable non parametric test for checking median (median is measure of central tendency for non-parametric test) value which is known as one sample Wilcoxon signed rank test.

The following is the information about the test.

## Wilcoxon's Signed Rank Test:

It is one of the most non-parametric tests used to test the location of a population based on a sample of data or to compare the locations of two populations using two samples. The sign test for location utilizes only the signs of difference of observations from hypothesized median (or the difference of observations in the pairs) without considering the magnitude of the difference. If the information regarding magnitude is available then a test procedure that takes into account the size and the relative magnitude of the differences as well, is expected to give a better performance. Wilcoxon's signed rank test is based on this consideration. However, the better performance is obtained at the cost of additional assumption of symmetry of the population about true median.

### Testing Problem:

Suppose $X_1,.., X_n$ is a random sample of size n from the distribution of random variable X. Let $F_x(.)$ be the distribution function and M be the median of X. It is required to test the hypothesis.

$H_0: M = M_0$ against one of the alternatives,

1) $H_1: M > M_0$
2) $H_1: M < M_0$
3) $H_1: M \neq M_0$

**Assumptions:**

1. $F_x(.)$ is continuous
2. $F_x(.)$ is symmetric about M.

**Test Statistic:**

Let $T^+$ = sum of positive ranks

$T^-$ = sum of negative ranks.

Note that, $T^+$ and $T^-$ both are non-negative numbers and

$$T^+ + T^- = \sum_{i=1}^{n} \frac{n(n+1)}{2}$$

Under $H_0$, the distribution of $T^+$ and $T^-$ are identical and each distribution is symmetric about the common mean n(n+1)/4. So, any one of the $T^+$ or $T^-$ can be used as the test statistic.

**R Code:**

```
> a=wilcox.test(waterqualitydata$ph,mu=7.04,alternative ="greater")
> a

	Wilcoxon signed rank test with continuity correction

data:  waterqualitydata$ph
V = 1985957, p-value = 4.407e-08
alternative hypothesis: true location is greater than 7.04

> a=wilcox.test(waterqualitydata$Hardness,mu=196.98,alternative ="less")
> a

	Wilcoxon signed rank test with continuity correction

data:  waterqualitydata$Hardness
```

```
V = 2640692, p-value = 1.2e-07
alternative hypothesis: true location is less than 196.98

> a=wilcox.test(waterqualitydata$Solids,mu=20927.83,alternative ="less")
> a

        Wilcoxon signed rank test with continuity correction

data:  waterqualitydata$Solids
V = 2894362, p-value = 3.2e-04
alternative hypothesis: true location is less than 20927.83

> a=wilcox.test(waterqualitydata$Chloramines,mu=7.13,alternative ="less")
> a

        Wilcoxon signed rank test with continuity correction

data:  waterqualitydata$Chloramines
V = 2668379, p-value = 1.062e-05
alternative hypothesis: true location is less than 7.13

> a=wilcox.test(waterqualitydata$Sulfate,mu=333.07,alternative ="less")
> a

        Wilcoxon signed rank test with continuity correction

data:  waterqualitydata$Sulfate
V = 1580798, p-value = 4.52e-2
alternative hypothesis: true location is less than 333.07

> a=wilcox.test(waterqualitydata$Conductivity,mu=421.88,alternative ="less")
> a

        Wilcoxon signed rank test with continuity correction
```

```
data:  waterqualitydata$Conductivity
V = 2773451, p-value = 2.342e-03
alternative hypothesis: true location is less than 421.88

> a=wilcox.test(waterqualitydata$Organic_carbon,mu=14.22,alternative
="less")
> a

	Wilcoxon signed rank test with continuity correction

data:  waterqualitydata$Organic_carbon
V = 2737315, p-value = 3.122e-09
alternative hypothesis: true location is less than 14.22

>a=wilcox.test(waterqualitydata$Trihalomethanes,mu=66.62,alternative="less
")
> a

	Wilcoxon signed rank test with continuity correction

data:  waterqualitydata$Trihalomethanes
V = 2407313, p-value = 4.122e-11
alternative hypothesis: true location is less than 66.62


> a=wilcox.test(waterqualitydata$Turbidity,mu=3.96,alternative ="less")
> a

	Wilcoxon signed rank test with continuity correction

data:  waterqualitydata$Turbidity
V = 2712644, p-value = 3.12e-04
alternative hypothesis: true location is less than 3.96
```

As all the null hypothesis is rejected for the given parameters, we can conclude that all the parameters are in suitable range. But in future the decision may or may not be same as it is in the present because it depends on the given sample. **Note:** Suppose in the future if we get similar data and we want to check the given water is potable or not we can use appropriate machine learning model for checking purpose.

**Benefits of using model over the hypothesis:**

The hypothesis is possible if and only if the given sample is considerably large. Sometimes it is very costly to et the large sample but if you have given only one data point, we can't use the hypothesis but we can use the model to get idea about the census.

**Scope of using the machine learning models in day-to-day life:**

We can easily see that in summer season some villages face the water problem. Sometimes water provided to them maybe collected from river or lake or well which is not tested chemically whether it is potable or not because by the naked eyes we can't figure out the water as potable or not.

So, if we have provided the parameters value it will be very difficult for human being to check each value in parameter space and give conclusion about the sample. Sometimes it will reject the sample even it satisfies all the require conditions. So, to increase the efficiency of work we use the machine learning models.

# MACHINE LEARNING

**What is Machine Learning?**

Machine Learning (ML) is basically the study of computer algorithms that can improve automatically through experience and by the use of past data. It is seen as a part of Artificial Intelligence (AI). Machine Learning algorithms build a model based on sample data, known as training data, in order to make decisions and test its accuracy with the help of test data. Machine Learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, computer vision, etc.

Nowadays, the demand of statistics in ML is increasing day by day. In models, statistical methods are required in the preparation of train data and test data and also to check the accuracy of the models.

This includes:

- Outlier detection
- Missing value imputation
- Data sampling
- Data scaling
- Variable encoding.

This all can be done in machine learning by applying the proper statistical tools.

**Why we use it?**

The response variable of our data was in the form of classification type. So, we classify our data in two groups namely potable water or non-potable water as like a binary variable.

**Potable water = 1**

**Non-Potable water = 0**

There are also some classification models that are used in machine learning. Examples of those models are:

1. Logistic Regression
2. K-Nearest Neighbor
3. Support Vector Machines
4. Kernel SVM
5. Naïve Bayes
6. Decision Tree Classification
7. Random Forest Classification
8. ANN
9. CNN

We want to develop a model that can predict the values of potability. Our focus is on both accuracy of the predictions and interpretability of the model.

Therefore, we have chosen the models that suits our data best. We will evaluate three different models covering the complexity spectrum.

1. Logistic Regression
2. K-Nearest Neighbors
3. Decision Tree.

To head start the ML process, the cleaning of data is must.

**Why data cleaning is important?**

To reduce the errors and to increase the efficiency of model we need to clean our data.

We will clean our data set using **Python**:

Codes for cleaning data:

In [1]:

```
import pandas as pd #To import and analyze data
import numpy as np #To work with array - mathematical operations
import matplotlib.pyplot as plt # Data visualiztion and graphical plotting
import seaborn as sns;sns.set() # Data visualization and exploratory data
analysis
import math # Mathematical calculations
```

In [2]:

```
data = pd.read_csv('water_potability.csv') #Importing data from csv format
```

```
data
```

Out[2]:

| | ph | Hardness | Solids | Chloramines | Sulfate | Conductivity | Organic_carbon | Trihalomethanes | Turbidity | Potability |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | NaN | 204.890455 | 20791.318981 | 7.300212 | 368.516441 | 564.308654 | 10.379783 | 86.990970 | 2.963135 | 0 |
| **1** | 3.716080 | 129.422921 | 18630.057858 | 6.635246 | NaN | 592.885359 | 15.180013 | 56.329076 | 4.500656 | 0 |
| **2** | 8.099124 | 224.236259 | 19909.541732 | 9.275884 | NaN | 418.606213 | 16.868637 | 66.420093 | 3.055934 | 0 |
| **3** | 8.316766 | 214.373394 | 22018.417441 | 8.059332 | 356.886136 | 363.266516 | 18.436524 | 100.341674 | 4.628771 | 0 |
| **4** | 9.092223 | 181.101509 | 17978.986339 | 6.546600 | 310.135738 | 398.410813 | 11.558279 | 31.997993 | 4.075075 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **3271** | 4.668102 | 193.681735 | 47580.991603 | 7.166639 | 359.948574 | 526.424171 | 13.894419 | 66.687695 | 4.435821 | 1 |
| **3272** | 7.808856 | 193.553212 | 17329.802160 | 8.061362 | NaN | 392.449580 | 19.903225 | NaN | 2.798243 | 1 |
| **3273** | 9.419510 | 175.762646 | 33155.578218 | 7.350233 | NaN | 432.044783 | 11.039070 | 69.845400 | 3.298875 | 1 |
| **3274** | 5.126763 | 230.603758 | 11983.869376 | 6.303357 | NaN | 402.883113 | 11.168946 | 77.488213 | 4.708658 | 1 |
| **3275** | 7.874671 | 195.102299 | 17404.177061 | 7.509306 | NaN | 327.459760 | 16.140368 | 78.698446 | 2.309149 | 1 |

3276 rows × 10 columns

In [3]:

```
data.shape #Rows, columns
```

Out[3]:

```
(3276, 10)
```

## Data Cleaning

In [4]:

```
data.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
```

In [5]:

```
data.isnull().sum()
```

Out[5]:

```
ph                 491
Hardness             0
Solids               0
Chloramines          0
Sulfate            781
Conductivity         0
Organic_carbon       0
Trihalomethanes    162
Turbidity            0
Potability           0
dtype: int64
```

In [6]:

```
data.fillna(data.mean(),inplace = True) #Fill null values by it's average value
data
```

Out[6]:

| | ph | Hardness | Solids | Chloramines | Sulfate | Conductivity | Organic_carbon | Trihalomethanes | Turbidity | Potability |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 7.080795 | 204.890455 | 20791.318981 | 7.300212 | 368.516441 | 564.308654 | 10.379783 | 86.990970 | 2.963135 | 0 |
| **1** | 3.716080 | 129.422921 | 18630.057858 | 6.635246 | 333.775777 | 592.885359 | 15.180013 | 56.329076 | 4.500656 | 0 |
| **2** | 8.099124 | 224.236259 | 19909.541732 | 9.275884 | 333.775777 | 418.606213 | 16.868637 | 66.420093 | 3.055934 | 0 |
| **3** | 8.316766 | 214.373394 | 22018.417441 | 8.059332 | 356.886136 | 363.266516 | 18.436524 | 100.341674 | 4.628771 | 0 |

| | ph | Hardness | Solids | Chloramines | Sulfate | Conductivity | Organic_carbon | Trihalomethanes | Turbidity | Potability |
|---|---|---|---|---|---|---|---|---|---|---|
| **4** | 9.092223 | 181.101509 | 17978.986339 | 6.546600 | 310.135738 | 398.410813 | 11.558279 | 31.997993 | 4.075075 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **3271** | 4.668102 | 193.681735 | 47580.991603 | 7.166639 | 359.948574 | 526.424171 | 13.894419 | 66.687695 | 4.435821 | 1 |
| **3272** | 7.808856 | 193.553212 | 17329.802160 | 8.061362 | 333.775777 | 392.449580 | 19.903225 | 66.396293 | 2.798243 | 1 |
| **3273** | 9.419510 | 175.762646 | 33155.578218 | 7.350233 | 333.775777 | 432.044783 | 11.039070 | 69.845400 | 3.298875 | 1 |
| **3274** | 5.126763 | 230.603758 | 11983.869376 | 6.303357 | 333.775777 | 402.883113 | 11.168946 | 77.488213 | 4.708658 | 1 |
| **3275** | 7.874671 | 195.102299 | 17404.177061 | 7.509306 | 333.775777 | 327.459760 | 16.140368 | 78.698446 | 2.309149 | 1 |

3276 rows × 10 columns

In [7]:

```
data.isnull().sum() #Checking null values
```

Out[7]:

```
ph                 0
Hardness           0
Solids             0
Chloramines        0
Sulfate            0
Conductivity       0
Organic_carbon     0
Trihalomethanes    0
Turbidity          0
Potability         0
dtype: int64
```

**As data cleaning is done, we can move further.**

To use the machine learning model the basic assumption is that there should be no multicollinearity between the regressor. So, in our data type let $X_1$, $X_2$, $X_3$, $X_4$, $X_5$, $X_6$, $X_7$, $X_8$, $X_9$ be pH, Hardness, Solids, Chloramines, Sulfate, Conductivity, Organic_Carbon, Trihalomeathanes, Turbidity respectively. These are the regressor in our data which affects the value of the response variable. So to check the multicollinearity between the regressor we use the Heat Map as statistical tool.

## HEAT MAP

**What is heat map?**

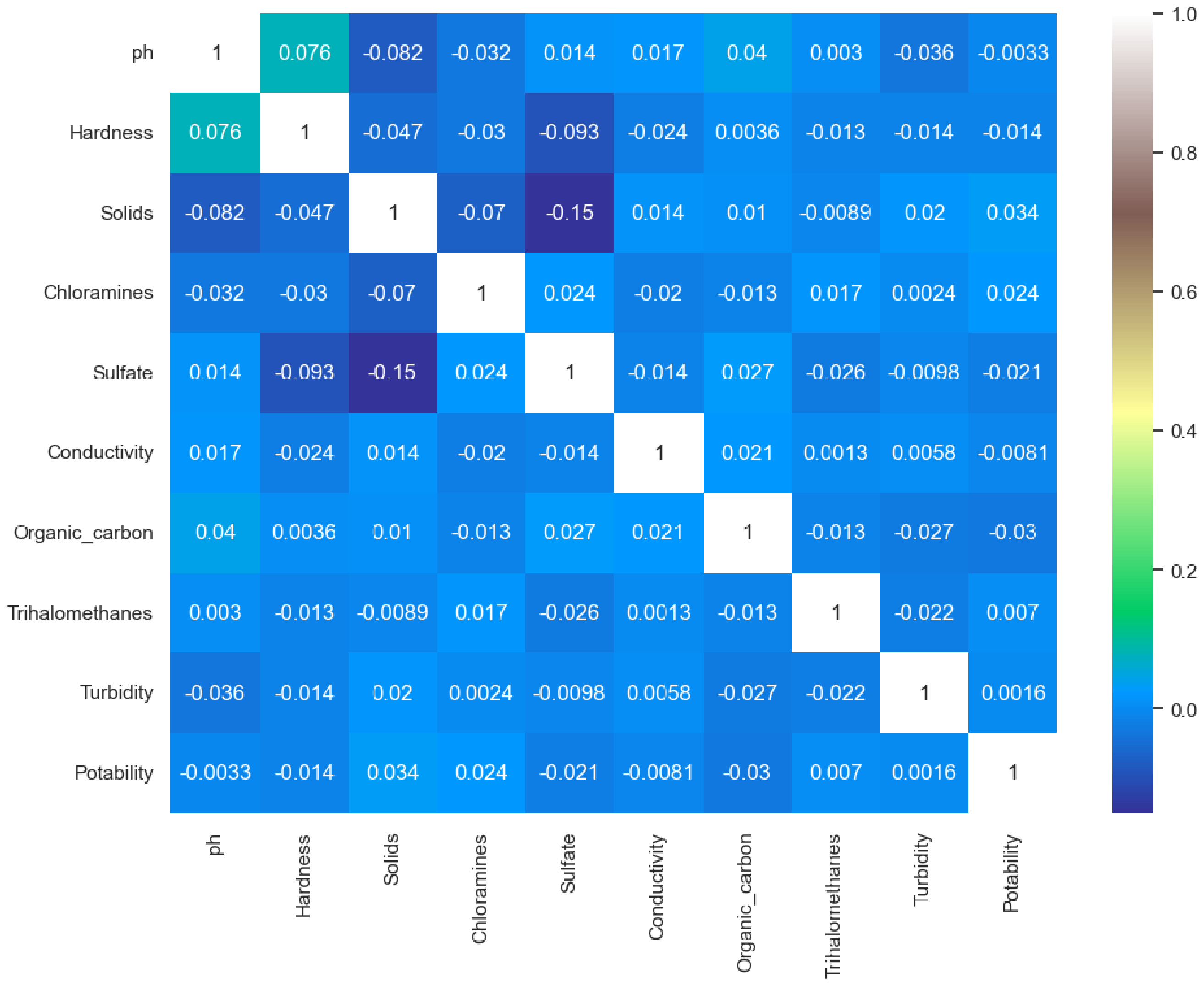
A heat map is basically the representation of two-dimensional information (data) with the help of colors. It gives warm-to-cool spectrum to show which parts of a data has the most attention. We use the heatmap as a correlation matrix. In heatmap correlation matrix, both the axis has same variables and we check the correlation between them by using it. The dark color represents the positive correlation and the medium light color gives no correlation between the variable.

As it gives visual as well as numerical value to check the correlation. The values in the cell indicate the strength of the relationship, with positive values indicating a positive relationship and negative values indicating a negative relationship. In addition, correlation plots can be used to identify outliers and to detect linear and non-linear relationships. The color-coding of the cells makes it easy to identify relationships between variables at a glance.

**Check the multicollinearity between the regressors:**

By using **Python,** we plot the heatmap for our data. The respective commands are as follows:

```
#Correlation using heatmap
sns.heatmap(data.corr(),annot=True,cmap='terrain')
fig=plt.gcf()
fig.set_size_inches(11,8)
plt.show()
```



| | ph | Hardness | Solids | Chloramines | Sulfate | Conductivity | Organic_carbon | Trihalomethanes | Turbidity | Potability |
|---|---|---|---|---|---|---|---|---|---|---|
| ph | 1 | 0.076 | -0.082 | -0.032 | 0.014 | 0.017 | 0.04 | 0.003 | -0.036 | -0.0033 |
| Hardness | 0.076 | 1 | -0.047 | -0.03 | -0.093 | -0.024 | 0.0036 | -0.013 | -0.014 | -0.014 |
| Solids | -0.082 | -0.047 | 1 | -0.07 | -0.15 | 0.014 | 0.01 | -0.0089 | 0.02 | 0.034 |
| Chloramines | -0.032 | -0.03 | -0.07 | 1 | 0.024 | -0.02 | -0.013 | 0.017 | 0.0024 | 0.024 |
| Sulfate | 0.014 | -0.093 | -0.15 | 0.024 | 1 | -0.014 | 0.027 | -0.026 | -0.0098 | -0.021 |
| Conductivity | 0.017 | -0.024 | 0.014 | -0.02 | -0.014 | 1 | 0.021 | 0.0013 | 0.0058 | -0.0081 |
| Organic_carbon | 0.04 | 0.0036 | 0.01 | -0.013 | 0.027 | 0.021 | 1 | -0.013 | -0.027 | -0.03 |
| Trihalomethanes | 0.003 | -0.013 | -0.0089 | 0.017 | -0.026 | 0.0013 | -0.013 | 1 | -0.022 | 0.007 |
| Turbidity | -0.036 | -0.014 | 0.02 | 0.0024 | -0.0098 | 0.0058 | -0.027 | -0.022 | 1 | 0.0016 |
| Potability | -0.0033 | -0.014 | 0.034 | 0.024 | -0.021 | -0.0081 | -0.03 | 0.007 | 0.0016 | 1 |

**Conclusion of heat map:**

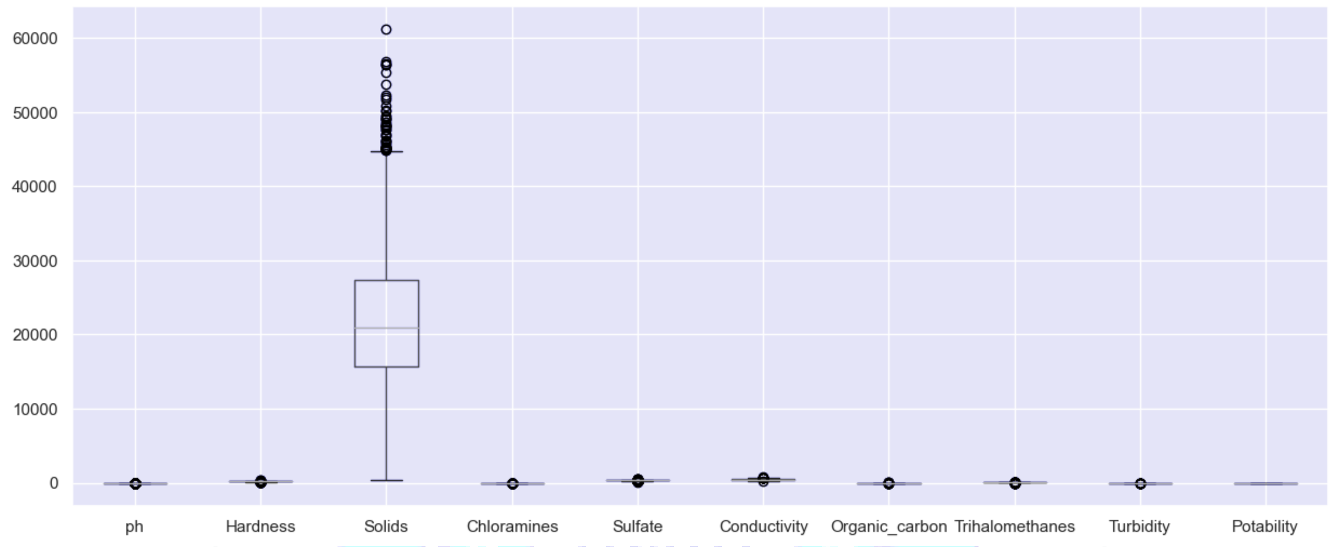
As the correlation coefficient are negligible, we can conclude that the parameters are uncorrelated.

## BOX PLOT

#Checking outliers using boxplot:

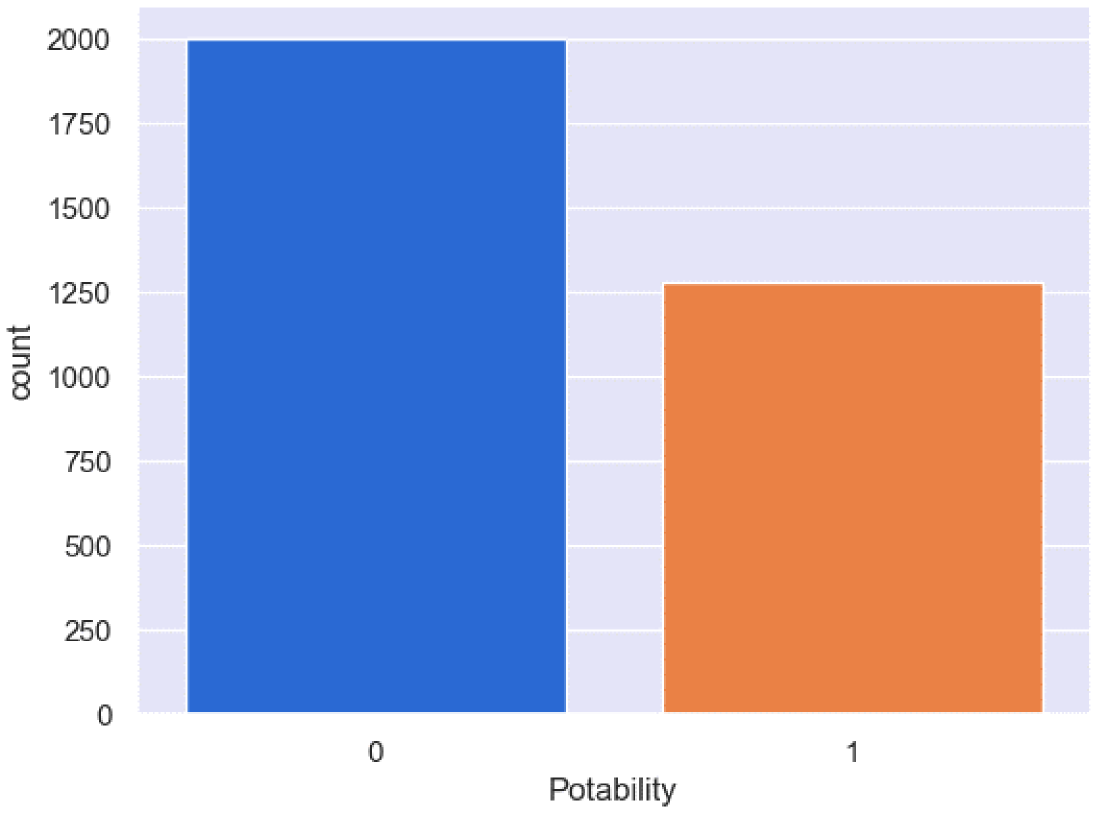
data.boxplot(figsize=(15,6))

plt.show()



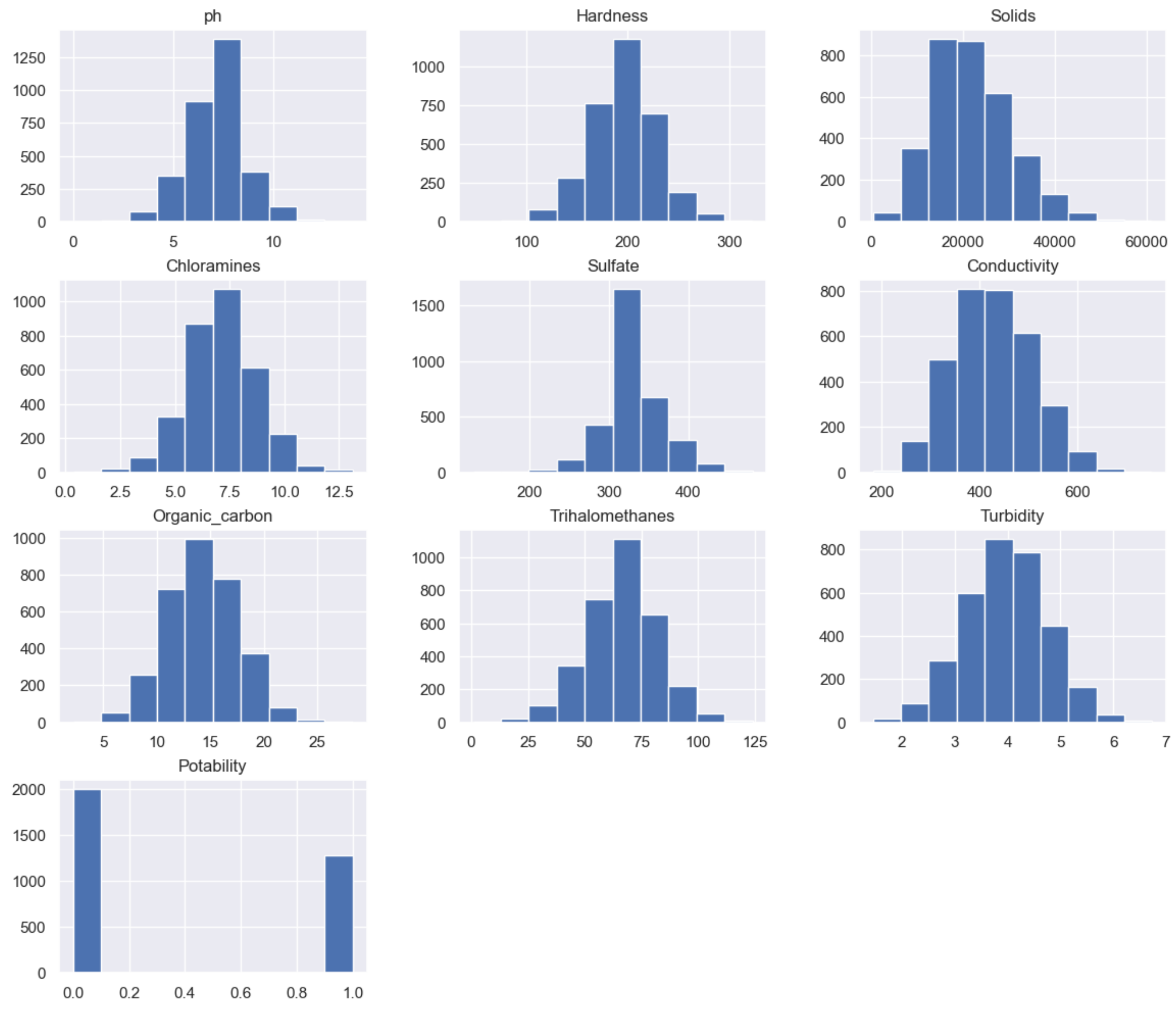
We are not removing outliers because they may decide the quality of water.

```
#Countplot for potability
sns.countplot(x='Potability', data = data)
plt.show()
```



**We just want to see the count of potability in data set.**

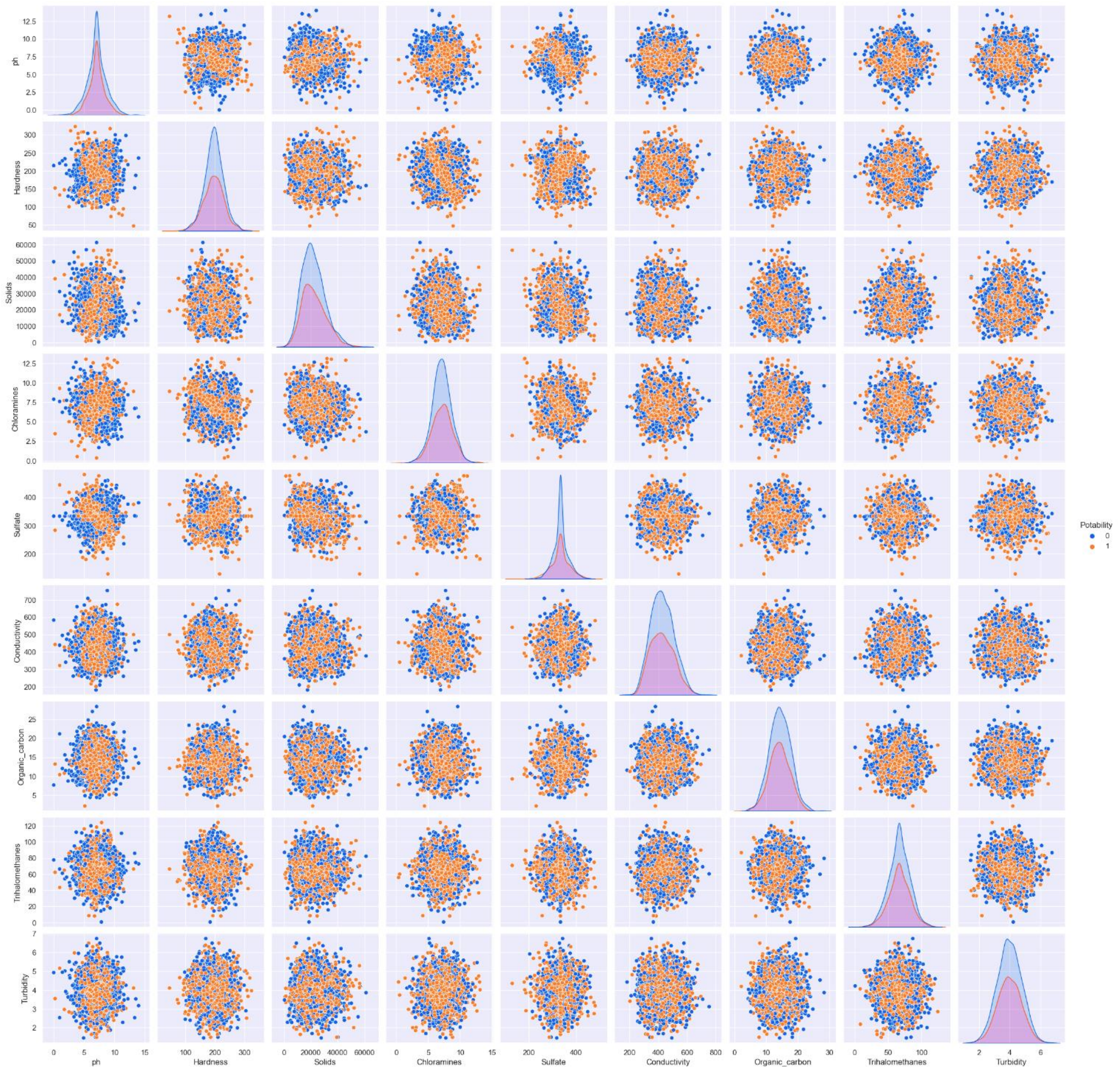
```
#Graphical representation of parameters using histogram
data.hist(figsize =(14,12))
plt.show()
```



**Conclusion:**

From the above graph, we conclude that our data don't follow normal distribution. By using Shapiro-Wilk test also it was confirmed.

```
#graphical representation of relationship between the parameters using
pairplots.

sns.pairplot(data,hue='Potability')

plt.show()
```



# DECISION TREE

**What is decision tree?**

Decision tree is a decision support tool that uses a tree-like method of decisions and their possible consequences, including chance event outcomes, resource. It is Supervised Machine Learning algorithm which uses set of rules to make decisions. It is one of the classification algorithms which uses rule-based approach.

For example:

Planning the next vacation which depends on various factors such as time, no. of members, budget.

It can perform both classification and regression tasks so referred as CART algorithm (Classification And Regression Tree).

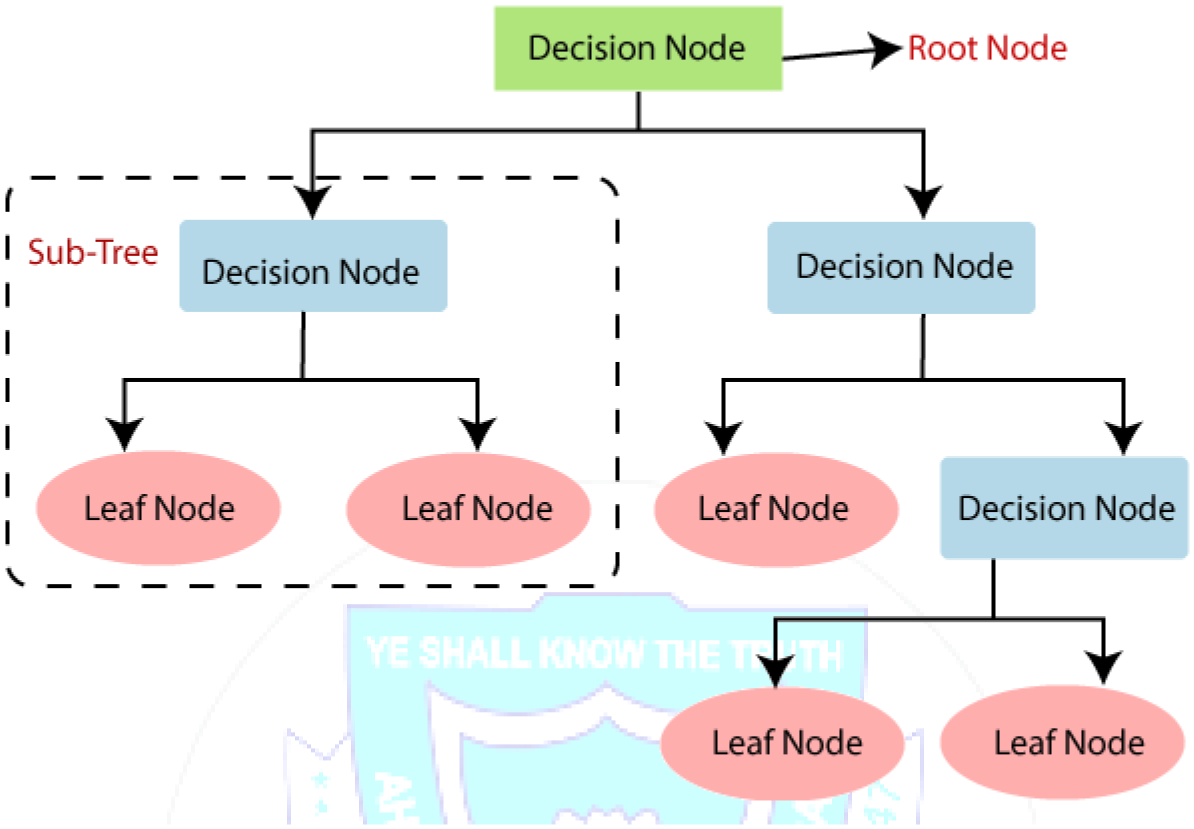
**Intuition:** Need of use of dataset features to create YES/NO type questions (In our case water potability) until we isolate all data points belonging to each class.

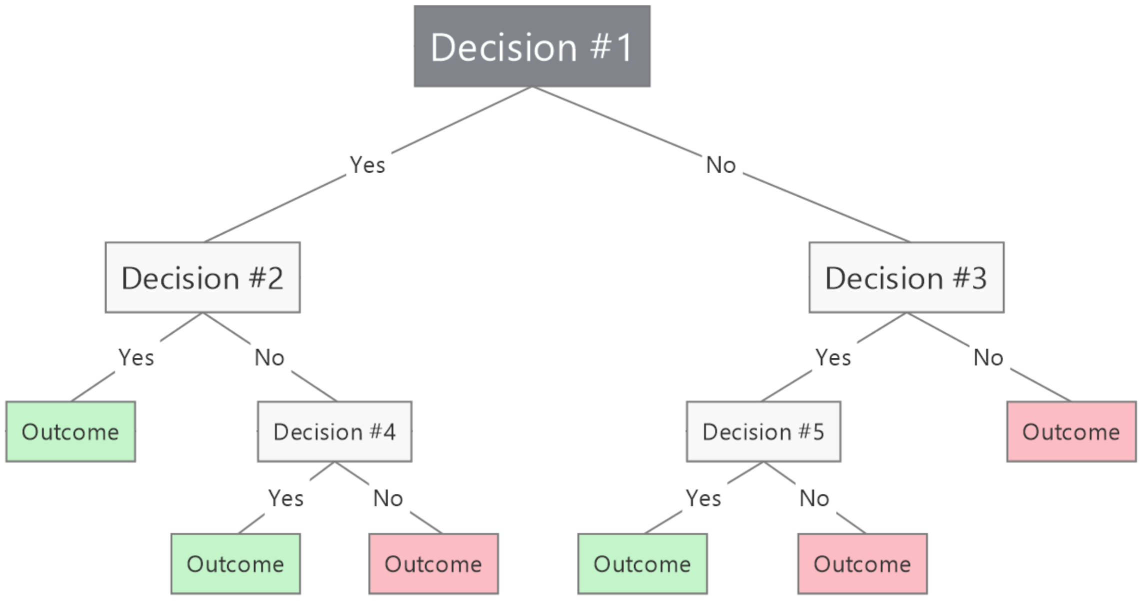
**Model characteristics:**

1. Fewer the splits more the accuracy.
2. Algorithms assigns only one class to each leaf node.
3. It picks best split to minimize loss function on basis of purity – "GINI Impurity"

$$G = 1 - \sum_{k=1}^{c} (P_k^2)$$

4. Uses greedy approach
5. It can be linearized into decision rules
6. It should be parallel by a probability model as a choice model
7. Descriptive means for calculating conditional probabilities
8. Categorical variable decision tree


Decision Node
Root Node
Sub-Tree
Decision Node
Decision Node
Leaf Node
Leaf Node
Leaf Node
Decision Node
Leaf Node
Leaf Node


Decision #1
Yes
No
Decision #2
Decision #3
Yes
No
Yes
No
Outcome
Decision #4
Decision #5
Outcome
Yes
No
Yes
No
Outcome
Outcome
Outcome
Outcome

## Python – Code

### #Split data set into train and test set:

```
X = data.drop('Potability', axis=1) #Input variable

X

Y = data['Potability']

Y

from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size =0.2, shuffle = True, random_state =0)

X_train
```

| | ph | Hardness | Solids | Chloramines | Sulfate | Conductivity | Organic_carbon | Trihalomethanes | Turbidity |
|---|---|---|---|---|---|---|---|---|---|
| **2128** | 5.514748 | 228.735924 | 35343.628580 | 4.346608 | 333.775777 | 526.112381 | 14.930982 | 46.780508 | 2.798158 |
| **1519** | 7.080795 | 210.732854 | 13671.416030 | 8.546187 | 418.470551 | 352.252328 | 10.353659 | 45.304007 | 3.364891 |
| **40** | 7.080795 | 233.858996 | 11703.923907 | 4.599388 | 309.039320 | 349.399633 | 18.338893 | 42.677465 | 3.510004 |
| **1151** | 4.303575 | 227.007086 | 7323.302301 | 7.490508 | 326.695199 | 412.896404 | 12.906730 | 68.748918 | 2.010537 |
| **2404** | 9.624727 | 217.372780 | 25175.754158 | 9.883946 | 329.174454 | 394.054835 | 20.277571 | 85.840258 | 2.615257 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **835** | 7.440825 | 183.362713 | 17259.852302 | 4.610245 | 335.626443 | 452.995293 | 9.700906 | 80.537065 | 2.496343 |
| **3264** | 5.893103 | 239.269481 | 20526.666156 | 6.349561 | 341.256362 | 403.617560 | 18.963707 | 63.846319 | 4.390702 |
| **1653** | 6.648005 | 191.841801 | 15176.290678 | 5.661663 | 333.775777 | 471.047129 | 15.438287 | 56.532387 | 3.829784 |
| **2607** | 7.675914 | 233.300759 | 23673.100606 | 8.407497 | 333.775777 | 232.613624 | 18.459408 | 60.993590 | 5.040461 |
| **2732** | 7.080795 | 160.915815 | 13943.244974 | 8.399730 | 380.768478 | 344.154228 | 15.208691 | 75.575056 | 4.141552 |

## #Model fitting decision tree:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,confusion_matrix, precision_score
data = DecisionTreeClassifier(criterion='gini', min_samples_split=10, splitter='best')
data.fit(X_train,Y_train)
```

```
☑ DecisionTreeClassifier
DecisionTreeClassifier(min_samples_split=10)
```

## #Prediction for test dataset:

```
prediction = data.predict(X_test)
prediction
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1,
       0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1,
       1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0,
       1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0], dtype=int64)
```

## #Accuracy

```
accuracy_score(prediction, Y_test)
print('accuracy_score:', accuracy_score(prediction,Y_test)*100,'%')
print("feature importance:\n{}".format(data.feature_importances_))
accuracy_score: 58.536585365853654 %
feature importance:
[0.15702817 0.13455843 0.11400517 0.14687985 0.12228679 0.09538537
 0.08244871 0.0785982  0.0688093 ]
```

## #Confusion matrix

```
confusion_matrix(prediction,Y_test)
  array([[272, 132],
         [140, 112]], dtype=int64)
```

## #Model Evaluation

### #Confusion Matrix

```
array([[272, 132],
       [140, 112]], dtype=int64)
```

### #Accuracy

```
accuracy_score: 58.536585365853654 %
```

## #Conclusion

Accuracy rate for fitting model given by Decision tree is 58.53% of our data.

**Advantage:**

1. Simple to understand and to interpret
2. It can handle both numerical as well as categorical data.

**Disadvantage:**

1. Unstable: change sensitive
2. Relatively inaccurate
3. Bias in favour of attributes with more level
4. Calculations can get very complex

# K-NEAREST NEIGHBOR CLASSIFIER (K-NN):

**What is K-NN?**

K-Nearest Neighbor is one is one of the simplest ML algorithms based on Supervised Learning Technique. It was first used for classification task by Fix and Hodges in 1951. K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data. It maps an input to an output based on example of input-output pairs i.e., it stores all the available data and classifies a new data point based on similarity.

- Euclidean distance - $d(x, y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$
- In K-NN classification, the output is the class membership. An object is classified by the majority votes of its neighbors, with the object being assigned to the class most common among its k nearest (K is positive integer, typically small). If k=1, then the object is simply assigned to the class of that single nearest neighbor.
- In K-NN regression, the output is the property value for the object. This value is the average of the values of its k-nearest neighbors.
- K-NN is a type of instance-based learning where the function is only approximated locally and all computation is deferred until classification. The K-NN algorithm is among the simplest of all machine learning algorithms.
- Both for classification and regression, a useful technique can be to assign weight to the neighbors, so that the nearer neighbors contribute more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight 1/d, where d is the distance to the neighbor.

- The particularity od the K-NN algorithm is that it is sensitive to the local structure of the data. The algorithm is not be confused with k-means, another popular machine learning technique.
- One of the simple procedures that can be used for classification is the Nearest Neighbor (NN) rule. It classifies as sample based on the category of its nearest neighbor. When large sample are involved, it can be shown that this rule has probability of error which is less than twice the optimum error. Hence there is less than twice the probability of error compare to any decision rule. The nearest neighbor-based classifier used some or all the patterns available in the training set to classify a test pattern. This classifier essentially involves finding the similarity between the patterns in the training set
- Among the various methods of supervised statistical pattern recognition, Nearest Neighbor rule achieves consistently high performance, without the prior assumption about the distribution from which the training examples are drawn. Sample is classified by the calculating the distance to the nearest training case K-NN classifier extends this idea by taking the K-nearest point and assigning the class points majority. It is common to select K small and odd to break ties. It can be also given by square root of sample. Larger k-value helps reduce the effects of noisy point within training data set. And the choice of k is often performed through cross validation.

## Python – Code

## #Split data set into train and test set:

```
X = data.drop('Potability', axis=1) #Input variable

X

Y = data['Potability']

Y

from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size =0.2, shuffle = True, random_state =0)

X_train
```

| | ph | Hardness | Solids | Chloramines | Sulfate | Conductivity | Organic_carbon | Trihalomethanes | Turbidity |
|---|---|---|---|---|---|---|---|---|---|
| **2128** | 5.514748 | 228.735924 | 35343.628580 | 4.346608 | 333.775777 | 526.112381 | 14.930982 | 46.780508 | 2.798158 |
| **1519** | 7.080795 | 210.732854 | 13671.416030 | 8.546187 | 418.470551 | 352.252328 | 10.353659 | 45.304007 | 3.364891 |
| **40** | 7.080795 | 233.858996 | 11703.923907 | 4.599388 | 309.039320 | 349.399633 | 18.338893 | 42.677465 | 3.510004 |
| **1151** | 4.303575 | 227.007086 | 7323.302301 | 7.490508 | 326.695199 | 412.896404 | 12.906730 | 68.748918 | 2.010537 |
| **2404** | 9.624727 | 217.372780 | 25175.754158 | 9.883946 | 329.174454 | 394.054835 | 20.277571 | 85.840258 | 2.615257 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **835** | 7.440825 | 183.362713 | 17259.852302 | 4.610245 | 335.626443 | 452.995293 | 9.700906 | 80.537065 | 2.496343 |
| **3264** | 5.893103 | 239.269481 | 20526.666156 | 6.349561 | 341.256362 | 403.617560 | 18.963707 | 63.846319 | 4.390702 |
| **1653** | 6.648005 | 191.841801 | 15176.290678 | 5.661663 | 333.775777 | 471.047129 | 15.438287 | 56.532387 | 3.829784 |
| **2607** | 7.675914 | 233.300759 | 23673.100606 | 8.407497 | 333.775777 | 232.613624 | 18.459408 | 60.993590 | 5.040461 |
| **2732** | 7.080795 | 160.915815 | 13943.244974 | 8.399730 | 380.768478 | 344.154228 | 15.208691 | 75.575056 | 4.141552 |

## #Model fitting and prediction for KNN:

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(metric='euclidean',n_neighbors=22)
knn.fit(X_train,Y_train)
```

```
☑ KNeighborsClassifier
KNeighborsClassifier(metric='euclidean', n_neighbors=22)
```

## #Prediction for test dataset:

```
prediction_knn=knn.predict(X_test)
prediction_knn
```

```
array([1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

## #Accuracy:

```
accuracy_knn=accuracy_score(Y_test,prediction_knn)*100
print('accuracy_score:',accuracy_knn,'%')
accuracy_score: 60.97560975609756 %
```

## #Confusion matrix:

```
confusion_matrix(prediction,Y_test)

array([[272, 132],
       [140, 112]], dtype=int64)
```

## #Model Evaluation:

**#Confusion Matrix**

```
array([[272, 132],
       [140, 112]], dtype=int64)
```

**#Accuracy**

```
accuracy_score: 60.97560975609756 %
```

## #Conclusion:

Accuracy rate for fitting model given by Decision tree is 60.97% of our data.

## Advantages:

1. It is easy to implement
2. It is robust to noisy training data
3. It can be more effective if the training data is large

## Disadvantages:

1. It is computationally intensive
2. Sensitive to outliers
3. Requires feature scaling
4. Needs a suitable value for k
5. Imbalanced data
6. Curse of dimensionality

# LOGISTIC REGRESSION MODEL:

**What is logistic model?**

It is a statistical method which is used to predict a ‘binary output such as Yes or No (in our case 1 or 0). Logistic regression model predicts dependent variable of data using regressors which are independent.

It is basically a supervised classification algorithm used in classification problems. As in linear regression, it is assumed that the data follows linear function similarly logistic model builds a regression model to predict the probability that given data entry belongs to Category numbered as **“1” OR “0”**

**Types of Logistic Regression:**

| Binomial | Multinomial | Ordinal |
|---|---|---|
| Target variable can have only two possible values | Target variable can have three or more possible values (not ordered) | Target variable with ordered categories. |
| e.g., 1 or 0, YES or NO, etc. | e.g., Division – A, B, C (No quantitative significance) | e.g., Temperature – High, Medium, Low |

**Assumptions:**

1. Absence of Multicollinearity – one of the most important assumptions
2. The dependent variable must be dichotomous.

**Why this model?**

As in our data, response variable is in the form of binary type and also there is no collinearity between the regressors (Using heatmap we can observe), hence we have used this model for testing quality of water i.e., whether it is potable or not.

**Model of Logistic Regression:**

1. Y = E(Y|x) + ε
2. Y = Π(x) + ε

Where, ε is Bernoulli random variable with

a. E(ε) = 0
b. Var(ε) = π(x)(1-π(x))

$$\pi(x) = \frac{e^{\beta_0+\beta_1X_1+\beta_2X_2+\beta_3X_3+\beta_4X_4+\beta_5X_5+\beta_6X_6+\beta_7X_7+\beta_8X_8+\beta_9X_9}}{1+e^{\beta_0+\beta_1X_1+\beta_2X_2+\beta_3X_3+\beta_4X_4+\beta_5X_5+\beta_6X_6+\beta_7X_7+\beta_8X_8+\beta_9X_9}}$$

Where $\beta_1$, $\beta_2$, $\beta_3$, $\beta_4$, $\beta_5$, $\beta_6$, $\beta_7$, $\beta_8$, $\beta_9$ are regression coefficients and variables are

| | |
|---|---|
| Potability | Y |
| pH | $X_1$ |
| Hardness | $X_2$ |
| Solids | $X_3$ |
| Chloramines | $X_4$ |
| Sulfate | $X_5$ |
| Conductivity | $X_6$ |
| Organic_carbon | $X_7$ |
| Trihalomethanes | $X_8$ |
| Turbidity | $X_9$ |

Logistic model considers probability using which we are going to allocate new observation to specify class. For this purpose, the threshold probability is decided and by default it is consider as P=0.5

## Python – Code

### #Split data set into train and test set:

```
X = data.drop('Potability', axis=1) #Input variable

X

Y = data['Potability']

Y

from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size =0.2, shuffle = True, random_state =0)

X_train
```

| | ph | Hardness | Solids | Chloramines | Sulfate | Conductivity | Organic_carbon | Trihalomethanes | Turbidity |
|---|---|---|---|---|---|---|---|---|---|
| **2128** | 5.514748 | 228.735924 | 35343.628580 | 4.346608 | 333.775777 | 526.112381 | 14.930982 | 46.780508 | 2.798158 |
| **1519** | 7.080795 | 210.732854 | 13671.416030 | 8.546187 | 418.470551 | 352.252328 | 10.353659 | 45.304007 | 3.364891 |
| **40** | 7.080795 | 233.858996 | 11703.923907 | 4.599388 | 309.039320 | 349.399633 | 18.338893 | 42.677465 | 3.510004 |
| **1151** | 4.303575 | 227.007086 | 7323.302301 | 7.490508 | 326.695199 | 412.896404 | 12.906730 | 68.748918 | 2.010537 |
| **2404** | 9.624727 | 217.372780 | 25175.754158 | 9.883946 | 329.174454 | 394.054835 | 20.277571 | 85.840258 | 2.615257 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **835** | 7.440825 | 183.362713 | 17259.852302 | 4.610245 | 335.626443 | 452.995293 | 9.700906 | 80.537065 | 2.496343 |
| **3264** | 5.893103 | 239.269481 | 20526.666156 | 6.349561 | 341.256362 | 403.617560 | 18.963707 | 63.846319 | 4.390702 |
| **1653** | 6.648005 | 191.841801 | 15176.290678 | 5.661663 | 333.775777 | 471.047129 | 15.438287 | 56.532387 | 3.829784 |
| **2607** | 7.675914 | 233.300759 | 23673.100606 | 8.407497 | 333.775777 | 232.613624 | 18.459408 | 60.993590 | 5.040461 |
| **2732** | 7.080795 | 160.915815 | 13943.244974 | 8.399730 | 380.768478 | 344.154228 | 15.208691 | 75.575056 | 4.141552 |

## #Model fitting and prediction for Logistic Regression Model:

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()

model.fit(X_train, Y_train)
```

```
✓ LogisticRegression

LogisticRegression()
```

## #Prediction for test dataset:

```
prediction=data.predict(X_test)

prediction

array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1,
       0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1,
       1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0,
       1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0], dtype=int64)
```

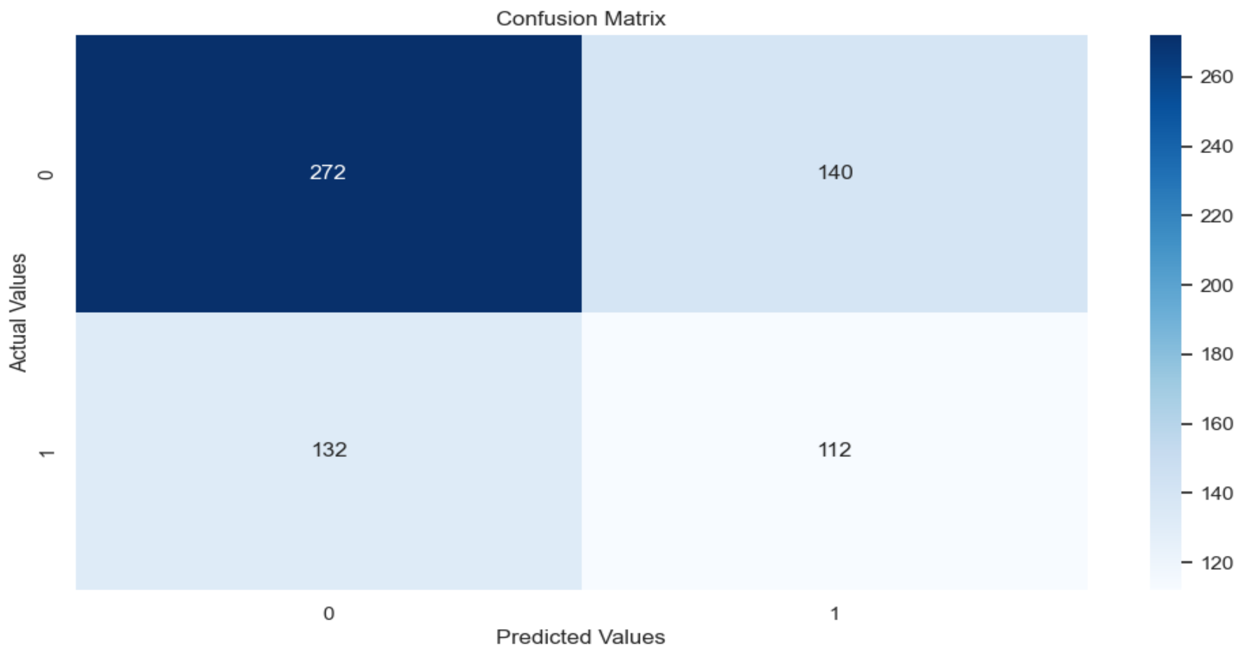
## #Accuracy:

```
test_acc = accuracy_score(Y_test,prediction)

print("The Accuracy for Test Set is {}".format(test_acc*100),'%')

The Accuracy for Test Set is 58.536585365853654 %
```

## #Confusion matrix:

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

print(classification_report(Y_test,prediction))

cm=confusion_matrix(Y_test,prediction)

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

plt.title("Confusion Matrix")

sns.heatmap(cm, annot=True,fmt='d', cmap='Blues')

plt.ylabel("Actual Values")

plt.xlabel("Predicted Values")

plt.savefig('confusion_matrix.png')
```

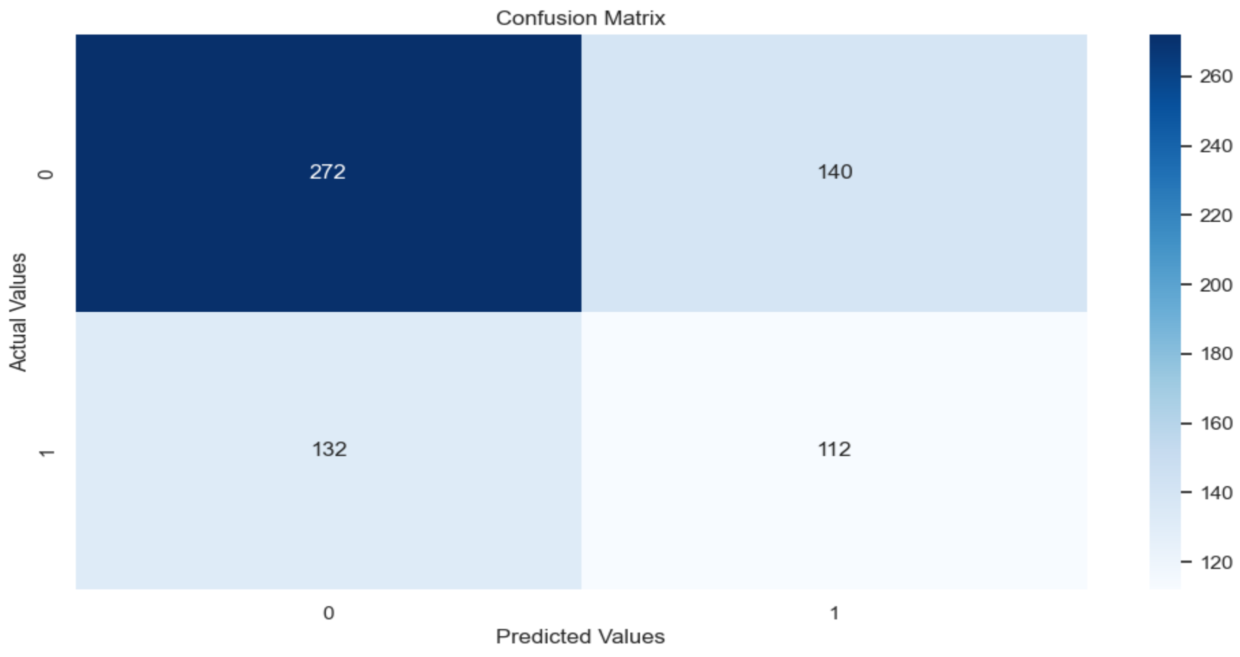
```
              precision    recall  f1-score   support

           0       0.67      0.66      0.67       412
           1       0.44      0.46      0.45       244
    accuracy                           0.59       656
   macro avg       0.56      0.56      0.56       656
weighted avg       0.59      0.59      0.59       656
```



## #Model Evaluation:

### #Confusion Matrix

```
              precision    recall  f1-score   support

           0       0.67      0.66      0.67       412
           1       0.44      0.46      0.45       244

    accuracy                           0.59       656
   macro avg       0.56      0.56      0.56       656
weighted avg       0.59      0.59      0.59       656
```



### #Accuracy

The Accuracy for Test Set is 58.536585365853654 %

## #Conclusion:

Accuracy rate for fitting model given by Decision tree is 58.53% of our data.

# CONCLUSION

Our project includes the understanding of the Machine Learning and its basic types. The classification models were used to analyze the water quality. The supervised classification models namely decision tree, KNN model and Logistic Model were fitted to our sample data of **3276** sample points. The water quality analysis is based on the parameters present in it, which pH, Hardness, Solids, Chloramines, Sulfate, Conductivity, Organic_Carbon, Trihalomethanes, Turbidity their standard ranges were provided by Who and lab reports. Of the three models that were fitted to this data, KNN model proved to be the best fit with accuracy 60.97%. With this accuracy, it concludes that our data is correct fitted.

Among all the three models' logistic regression classifier has highest accuracy due to the very less miss classification. Hence, we can say that **Logistics Regression** model is best for our data.

# REFERENCES

**BOOKS**

1. Data Mining Concepts and Techniques (Third Edition) by Jiawei Han, Micheline Kamber, Jian Pei
2. Fundamentals of Python Programming by Richard L.Halterman

**LINKS**

1. https://www.kaggle.com/datasets/adityakadiwal/water-potability
2. https://www.analyticsvidhya.com/blog/2021/06/hypothesis-testing-parametric-and-non-parametric-tests-in-statistics/
3. https://www.javatpoint.com/machine-learning
4. https://www.wikipedia.org/
5. https://github.com/python
6. https://www.analyticsvidhya.com/blog/2020/11/popular-classification-models-for-machine-learning/
7. https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html