PURE ML Water Quality Checker

FDM Mini Project- Final Report

Group Number - G32

Registration Number	Student Name
IT21207822	Senarathne H. A. T. S.
IT21177514	Ransika M. R. T.
IT21268762	Wijewardhana T.W. P. P.
IT21462320	IT21462320 – Perera K.R.M.

Background

What is Potable Water?

Potable water, often referred to simply as "drinking water," is water that is safe and suitable for human consumption. It is a vital resource for sustaining life and is essential for various everyday activities, including drinking, cooking, bathing, and sanitation.

Characteristics of Potable Water?

1. **Safety:** Potable water must meet strict safety standards to ensure it is free from harmful contaminants. Common contaminants that must be removed or reduced to safe levels include bacteria, viruses, parasites, heavy metals (such as lead and arsenic), chemical pollutants, and organic matter.
2. **Clarity and Color:** Potable water should be clear and colorless, without any visible particles or discoloration. Turbidity (cloudiness) can indicate the presence of suspended solids or microorganisms.
3. **Taste and Odor:** Potable water should be free from unpleasant tastes and odors. Chemicals, algae, or bacteria can cause water to taste or smell bad.
4. **Chemical Composition:** Potable water typically contains dissolved minerals and salts, which contribute to its taste and overall quality. However, the concentration of certain minerals like calcium, magnesium, and sodium should be within acceptable limits to prevent adverse health effects.
5. **pH Level:** The pH level of potable water should fall within a safe and neutral range, usually around 6.5 to 8.5. Extreme pH levels can be harmful and affect the water's taste.
6. **Disinfection:** Potable water is often treated with disinfectants such as chlorine, chloramine, or ozone to kill or inactivate harmful microorganisms like bacteria and viruses.

How Water Becomes Unpotable?

Water can become unpotable, or unfit for human consumption, due to various factors and contaminants. Here are some common ways water can become unpotable,

1. **Microbial Contamination:** Water can become unpotable when it is contaminated with harmful microorganisms such as bacteria, viruses, and parasites. These contaminants can cause waterborne diseases, making it unsafe to drink. Sources of microbial contamination can include sewage runoff, improperly treated wastewater, or animal waste entering water sources.
2. **Chemical Contamination:** Chemical pollutants can render water unpotable. These pollutants may include heavy metals (e.g., lead, arsenic, mercury), industrial chemicals, pesticides, fertilizers, and pharmaceutical residues. Chemical contamination can result from industrial discharges, agricultural runoff, or improper disposal of hazardous substances.
3. **Sediment and Turbidity:** Excessive sediment and turbidity in water can make it unpotable by clouding the water and reducing its clarity. Sediment can carry pathogens and other contaminants and can clog water treatment systems. Sources of sediment and turbidity include erosion, construction activities, and natural events like landslides.
4. **High Mineral Content:** While minerals are naturally present in water, high concentrations of certain minerals, such as calcium, magnesium, and sodium, can make water taste unpleasant and have adverse health effects. This condition is often referred to as "hard water."
5. **Excess Salinity:** Water with a high salt content, known as saline or brackish water, is not suitable for drinking or irrigation without proper treatment. Saline water can result from the intrusion of saltwater into freshwater sources, such as coastal aquifers, or from natural geological processes.
6. **Algae Blooms:** Algae blooms, often caused by nutrient pollution (e.g., excess nitrogen and phosphorus), can lead to the growth of harmful algal species. These algae can produce toxins that contaminate water and pose health risks if ingested.

7. **Acidic or Alkaline Conditions:** Extreme pH levels in water can make it unpotable. Highly acidic or alkaline water can cause health issues and affect the taste and quality of water.
8. **Radioactive Contaminants:** Radioactive substances in water, such as radium, uranium, and radon, can pose health risks if consumed. These contaminants may enter water sources through geological processes or human activities like mining.

What are the Outcomes of Drinking Polluted Water?

Drinking polluted water can have serious and potentially life-threatening health consequences. The outcomes of consuming contaminated water depend on the type and level of pollutants present, as well as the duration of exposure. Here are some of the common outcomes and health risks associated with drinking polluted water:

1. **Waterborne Diseases:** Contaminated water is a major source of waterborne diseases caused by microorganisms such as bacteria, viruses, and parasites. Common waterborne diseases include cholera, typhoid, dysentery, giardiasis, and hepatitis A. These illnesses can lead to symptoms like diarrhea, vomiting, dehydration, and, in severe cases, can be fatal, particularly in vulnerable populations like children and the elderly.
2. **Gastrointestinal Problems:** Consumption of water contaminated with pathogens or fecal matter can lead to gastrointestinal problems, including stomach cramps, nausea, and diarrhea. Chronic exposure to contaminated water can result in long-term health issues.
3. **Chemical Toxicity:** Drinking water contaminated with industrial chemicals, pesticides, or pharmaceutical residues can cause a range of health issues depending on the specific chemical involved. These can include organ damage, hormonal disruptions, and increased cancer risk.
4. **Skin and Respiratory Issues:** Exposure to water contaminated with certain chemicals or pollutants can lead to skin irritations, rashes, and respiratory problems, particularly when water is used for bathing or showering.

5. **Algae Toxins:** Harmful algal blooms in water bodies can produce toxins that, when consumed, can lead to symptoms such as nausea, vomiting, abdominal pain, and in severe cases, liver or nerve damage.
6. **Radioactive Contaminants:** Exposure to radioactive substances in drinking water can increase the risk of cancer and other radiation-related health problems.
7. **Long-term Health Effects:** Chronic exposure to low levels of contaminants in drinking water over an extended period may result in cumulative health effects, including cancer, organ damage, and developmental issues.

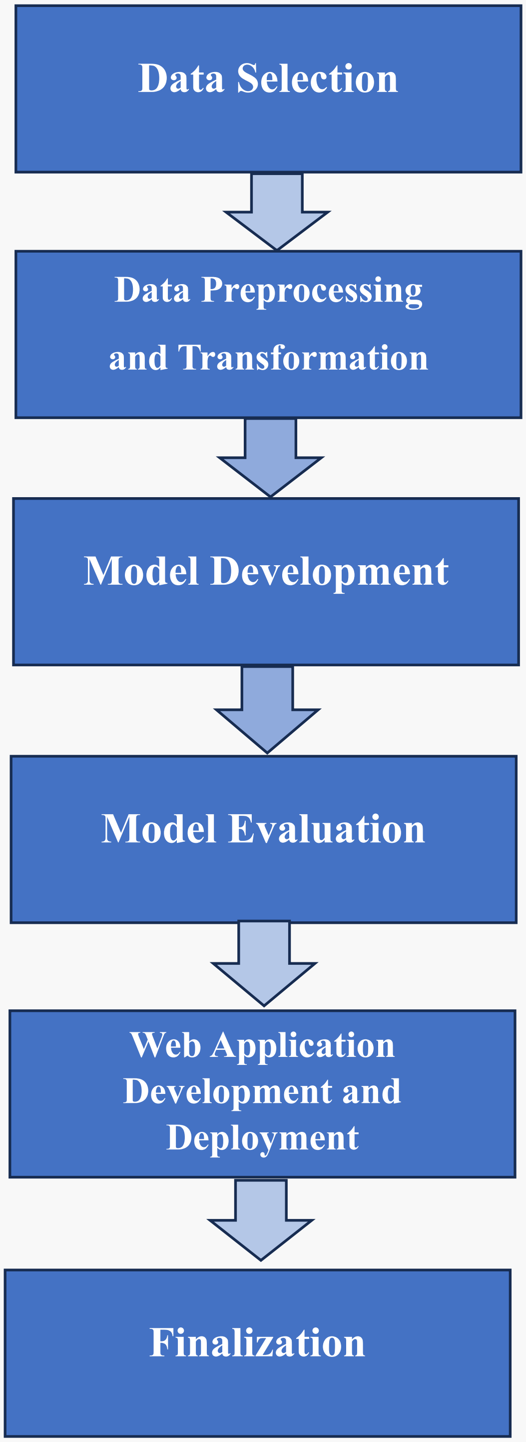
Our Project

According to the mentioned information, the consumption of polluted water has become a major obstacle to the existence of mankind. Accordingly, predicting the potability of water resources and recommending potable water for the consumption of human beings is a significant task. Because it ensures **improved health, proper hydration levels in the body, good digestive, skin, and hair health, prevention of dental issues, maintenance of cognitive, and kidney functions,** and **free from long-term health effects like cancers**.

Practically, determining the potability of water resources is happening in laboratory environments using a number of biochemical reactions and specific instruments. They do have so many disadvantages and limitations like **expensiveness, high time consumption, complexity, limited scope,** and **resource intensiveness**.

Our aim is to develop a software solution that can predict the potability of water resources based on **pH Value, Hardness, Solids, Chloramines, Sulphate, Conductivity, Organic Carbon, Trihalomethanes,** and **Turbidity** parameters. We expect to develop a Classification Model using a suitable algorithm (Artificial Neural Network, Decision Tree, Support Vector Machine, K-Nearest Neighbor, Random Forest, Logistic Regression, and AdaBoost Algorithm). Based on given parameters, which can classify whether a given water sample is likely to potable or not. To train and test the model we use Water Quality dataset, published on Kaggle.

The Project Workflow



01). Data Selection

Dataset Name: Water Quality Dataset

Link to Dataset: https://www.kaggle.com/datasets/adityakadiwal/water-potability

Author: Aditya Kadiwal

Number of Rows: 3276

Number of Columns: 10

Number of Class Attributes: 1

Names, Brief Description, and Standard Units of each Attribute:

Attribute Name	Brief Description	Standard Unit
pH	PH is an important parameter in evaluating the acid–base balance of water. 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. Hardness was originally defined as the capacity of water to precipitate soap caused by Calcium and Magnesium.	mg/L
Solids	Water has the ability to dissolve a wide range of inorganic and some organic minerals or salts.	ppm

	These minerals produced unwanted taste and diluted color in appearance of water. The water with high Solids 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. Chlorine levels up to 4 mg/L or 4 ppm are considered safe in drinking water.	ppm
Sulfate	Sulfates are naturally occurring substances that are found in minerals, soil, and rocks. It ranges from 3 to 30 mg/L in most freshwater supplies.	mg/L
Conductivity	Pure water is not a good conductor of electric current rather is a good insulator. An increase in ion concentration enhances the electrical conductivity of water. Generally, the amount of dissolved solids in water determines the electrical conductivity. According	µS/cm

	to WHO standards, the EC value should not exceed 400 μS/cm.	
Organic Carbon	Total Organic Carbon (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/Lit in source water which is used for treatment.	ppm
Trihalomethanes	THMs are chemicals that may be found in water treated with chlorine. THM levels up to 80 ppm are considered safe in drinking water.	μg/L
Turbidity	The turbidity of water depends on the quantity of solid matter present in the suspended state. It is a measure of the light-emitting properties of water. WHO recommended a maximum value of 5.00 NTU for drinking water.	NTU (Nephelometric Turbidity Unit)
Potability	Indicates if water is safe for human consumption where 1 means Potable and 0 means Not potable.	

02). Data Preprocessing and Transformation

The data preprocessing and transformation phase includes 3 major preprocessing steps. These are Data Cleaning, Feature Selection, and Data Transformation. There are several advantages of using preprocessed datasets for machine learning model development. Those are increasing the accuracy of the model, reducing the time and resources required to train the model, preventing overfitting, improving the model's ability to generalize to new data, etc.

02.01. Data Cleaning

Under data cleaning, handling of missing values and removal of outliers steps are happened.

02.01.01. Handling Missing Values (Fill in missing values automatically with the attribute mean for all samples belonging to the same class.)

Nature of the Dataset Before Handling Missing Values,

ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
	204.89046	20791.319	7.3002119	368.516441	564.3086542	10.37978308	86.99097046	2.963135381	0
3.71608	129.42292	18630.058	6.6352459		592.8853591	15.18001312	56.32907628	4.500656275	0
8.099124	224.23626	19909.542	9.2758836		418.6062131	16.86863693	66.42009251	3.05593375	0
8.316766	214.37339	22018.417	8.0593324	356.886136	363.2665162	18.4365245	100.3416744	4.628770537	0
9.092223	181.10151	17978.986	6.5466	310.135738	398.4108134	11.55827944	31.99799273	4.075075425	0
5.584087	188.31332	28748.688	7.5448688	326.678363	280.4679159	8.39973464	54.91786184	2.559708228	0
10.22386	248.07174	28749.717	7.5134085	393.663396	283.6516335	13.78969532	84.60355617	2.672988737	0
8.635849	203.36152	13672.092	4.5630087	303.309771	474.6076449	12.3638167	62.79830896	4.401424715	0
	118.98858	14285.584	7.8041736	268.646941	389.3755659	12.70604897	53.92884577	3.595017181	0
11.18028	227.23147	25484.508	9.0772	404.041635	563.8854815	17.92780641	71.97660103	4.370561937	0
7.36064	165.5208	32452.614	7.5507009	326.624354	425.3834195	15.58681044	78.74001566	3.662291783	0
7.974522	218.6933	18767.657	8.1103845		364.0982305	14.5257457	76.48591118	4.011718108	0
7.119824	156.70499	18730.814	3.6060361	282.344051	347.7150273	15.92953591	79.50077834	3.445756223	0
	150.17492	27331.362	6.8382235	299.415781	379.7618348	19.37080718	76.50999553	4.413974183	0
7.496232	205.34498	28388.005	5.0725578		444.6453523	13.2283111	70.30021265	4.777382337	0
6.347272	186.73288	41065.235	9.6295963	364.487687	516.7432819	11.53978119	75.07161729	4.376348291	0
7.051786	211.04941	30980.601	10.094796		315.1412672	20.39702184	56.65160379	4.268428858	0
9.18156	273.81381	24041.326	6.9049897	398.350517	477.9746419	13.38734078	71.45736221	4.503660796	0
8.975464	279.35717	19460.398	6.2043209		431.44399	12.88875905	63.8212371	2.43608559	0
7.37105	214.49661	25630.32	4.4326693	335.754439	469.9145515	12.50916394	62.79727715	2.560299148	0
	227.43505	22305.567	10.333918		554.8200865	16.33169328	45.38281518	4.133422644	0
6.660212	168.28375	30944.364	5.8587691	310.930858	523.6712975	17.88423519	77.04231805	3.749701241	0
	215.97786	17107.224	5.6070605	326.943978	436.256194	14.18906221	59.85547583	5.459250956	0

In [12]:
```python
# total sum of null values in each column
raw_data.isnull().sum()
```

Out[12]:
```
ph                 491
Hardness             0
Solids               0
Chloramines          0
Sulfate            781
Conductivity         0
Organic_carbon       0
Trihalomethanes    162
Turbidity            0
Potability           0
dtype: int64
```

In [13]:
```python
# total number of null values in entire dataframe
print(f"Total Number of NULL Values in Entire Dataset: {raw_data.isnull().sum().sum()}")
```

```
Total Number of NULL Values in Entire Dataset: 1434
```

Python code for handling missing values,

(The following code segments belong to the removal of missing values of the "pH" attribute only.)

Fill in Missing Values in "ph" column

In [73]:
```python
#calculate class 0 and 1 total ph
class0TotalPh=0
class1TotalPh=0

for i in range(0,3276):

    if((raw_data["ph"][i]>0) and (raw_data["Potability"][i] == 0)):
        class0TotalPh+=float(raw_data["ph"][i])
    elif((raw_data["ph"][i]>0) and (raw_data["Potability"][i] == 1)):
        class1TotalPh+=float(raw_data["ph"][i])


print(class1TotalPh)
print(class0TotalPh)
```

```
7788.235408179008
11931.777286193996
```

In [74]:
```python
# calculate class 0 and 1 average ph values
class0AvgPh=class0TotalPh/class0
class1AvgPh=class1TotalPh/class1

print(class0AvgPh)
print(class1AvgPh)
```

```
5.971860503600598
6.094080914068082
```

```
In [75]: # fill in missing values by average ph values
for i in range(0,3276):

    if((not raw_data["ph"][i]>0) and (raw_data["Potability"][i] == 0)):
        raw_data["ph"][i]=class0AvgPh

    elif((not raw_data["ph"][i]>0) and (raw_data["Potability"][i] == 1)):
        raw_data["ph"][i]=class1AvgPh
```

```
In [77]: #checking fill in missing values in "ph" column successfull or not
raw_data.isnull().sum()

Out[77]: ph                   0
Hardness             0
Solids               0
Chloramines          0
Sulfate            781
Conductivity         0
Organic_carbon       0
Trihalomethanes    162
Turbidity            0
Potability           0
dtype: int64
```

Nature of the Dataset After Handling Missing Values,

ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
5.9718605	204.89046	20791.319	7.30021187	368.516	564.3086542	10.37978308	86.99097046	2.9631354	0
3.7160801	129.42292	18630.058	6.63524588	252.849	592.8853591	15.18001312	56.32907628	4.5006563	0
8.0991242	224.23626	19909.542	9.2758836	252.849	418.6062131	16.86863693	66.42009251	3.0559338	0
8.3167659	214.37339	22018.417	8.05933238	356.886	363.2665162	18.4365245	100.3416744	4.6287705	0
9.0922235	181.10151	17978.986	6.54659997	310.136	398.4108134	11.55827944	31.99799273	4.0750754	0
5.5840866	188.31332	28748.688	7.54486879	326.678	280.4679159	8.39973464	54.91786184	2.5597082	0
10.223862	248.07174	28749.717	7.51340847	393.663	283.6516335	13.78969532	84.60355617	2.6729887	0
8.6358487	203.36152	13672.092	4.56300869	303.31	474.6076449	12.3638167	62.79830896	4.4014247	0
5.9718605	118.98858	14285.584	7.80417355	268.647	389.3755659	12.70604897	53.92884577	3.5950172	0
11.180284	227.23147	25484.508	9.07720002	404.042	563.8854815	17.92780641	71.97660103	4.3705619	0
7.3606401	165.5208	32452.614	7.55070091	326.624	425.3834195	15.58681044	78.74001566	3.6622918	0
7.9745216	218.6933	18767.657	8.1103845	252.849	364.0982305	14.5257457	76.48591118	4.0117181	0
7.1198244	156.70499	18730.814	3.60603609	282.344	347.7150273	15.92953591	79.50077834	3.4457562	0
5.9718605	150.17492	27331.362	6.83822347	299.416	379.7618348	19.37080718	76.50999553	4.4139742	0
7.4962322	205.34498	28388.005	5.07255777	252.849	444.6453523	13.2283111	70.30021265	4.7773823	0
6.3472718	186.73288	41065.235	9.62959628	364.488	516.7432819	11.53978119	75.07161729	4.3763483	0
7.0517858	211.04941	30980.601	10.094796	252.849	315.1412672	20.39702184	56.65160379	4.2684289	0
9.18156	273.81381	24041.326	6.90498973	398.351	477.9746419	13.38734078	71.45736221	4.5036608	0
8.9754643	279.35717	19460.398	6.20432086	252.849	431.44399	12.88875905	63.8212371	2.4360856	0
7.3710503	214.49661	25630.32	4.43266929	335.754	469.9145515	12.50916394	62.79727715	2.5602991	0
5.9718605	227.43505	22305.567	10.3339179	252.849	554.8200865	16.33169328	45.38281518	4.1334226	0

In [85]:
```
#checking fill in missing values in "Trihalomethanes" column successfull or not
raw_data.isnull().sum()
```

Out[85]:
```
ph                 0
Hardness           0
Solids             0
Chloramines        0
Sulfate            0
Conductivity       0
Organic_carbon     0
Trihalomethanes    0
Turbidity          0
Potability         0
dtype: int64
```

02.01.02. Removing Outliers

Python Code for Removal of Outliers,

(The following code segments belong to the removal of missing values of the "pH" attribute only.)

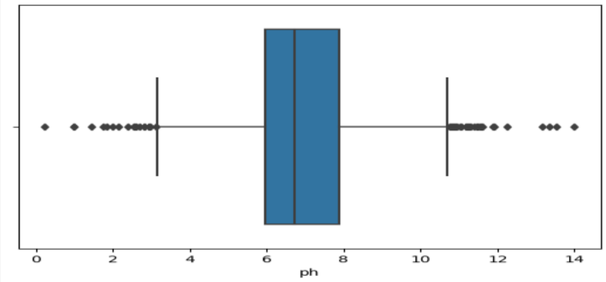
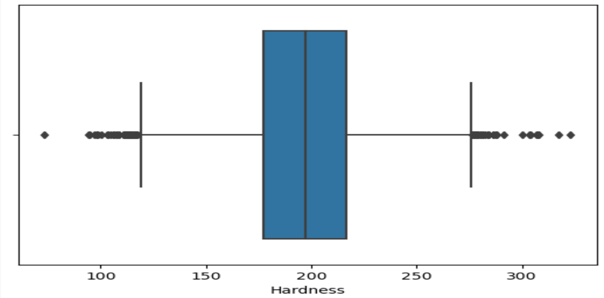
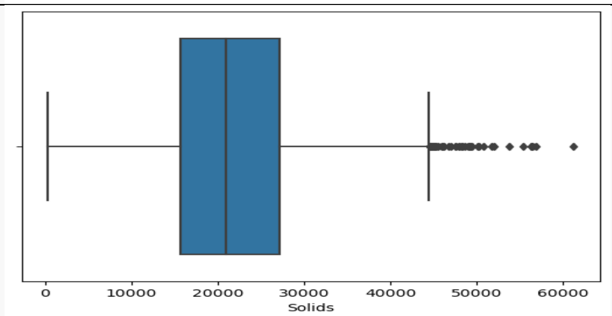
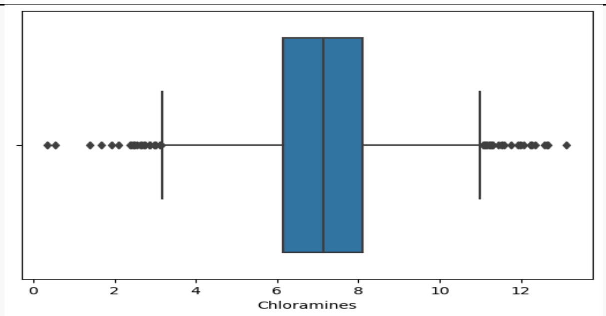
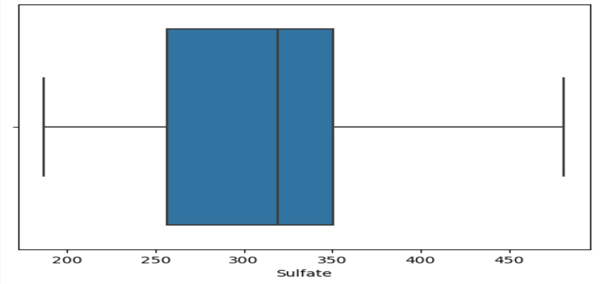
In [480]:
```
# calculate IQR value
phQ1=raw_data.ph.quantile(0.25)
phQ3=raw_data.ph.quantile(0.75)
phIQR=phQ3-phQ1
print(phIQR)
```

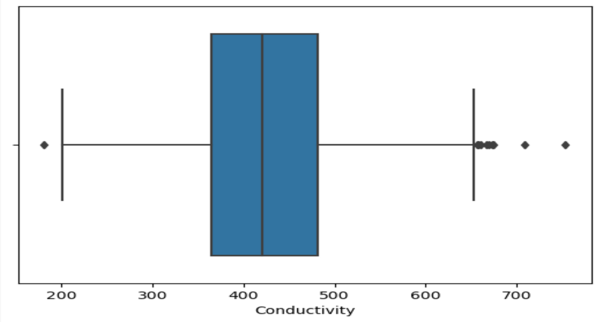
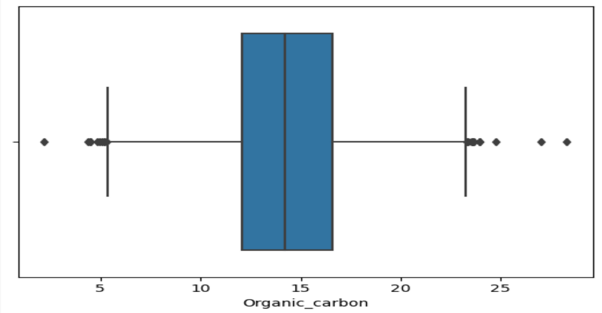
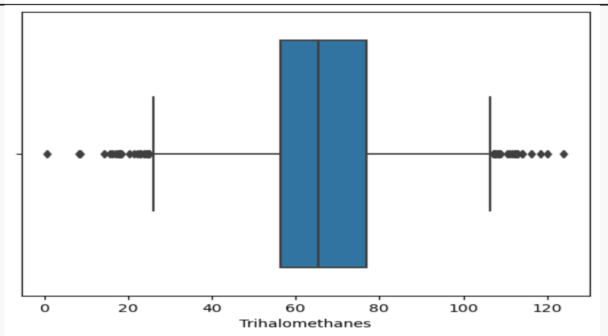
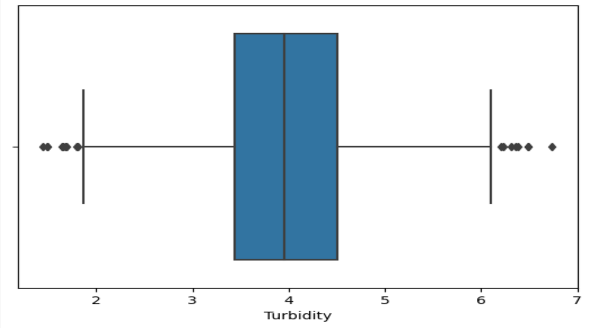
```
1.8981892518994021
```

In [481]:
```
#removal of outliers
raw_data=raw_data[~((raw_data.ph < (phQ1-1.5*phIQR)) | (raw_data.ph > (phQ3+1.5*phIQR)))]
```

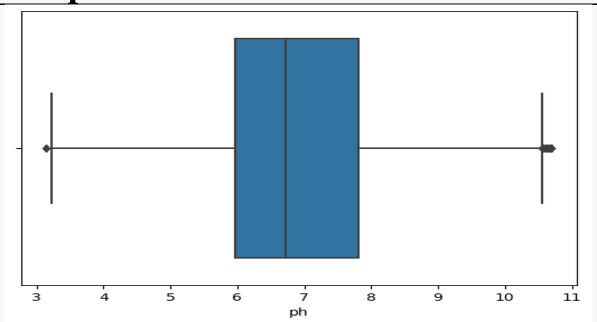
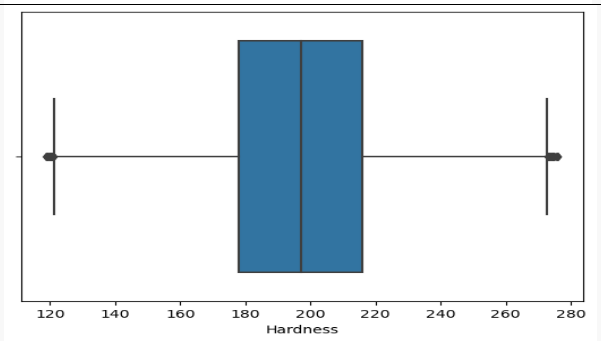
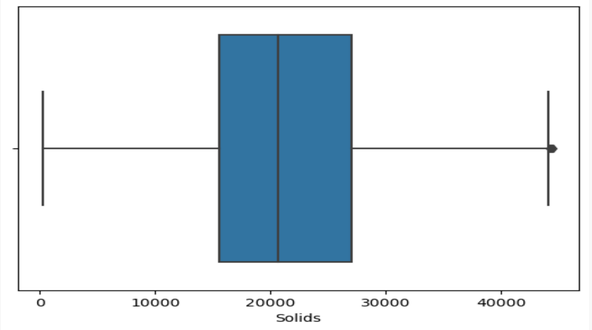
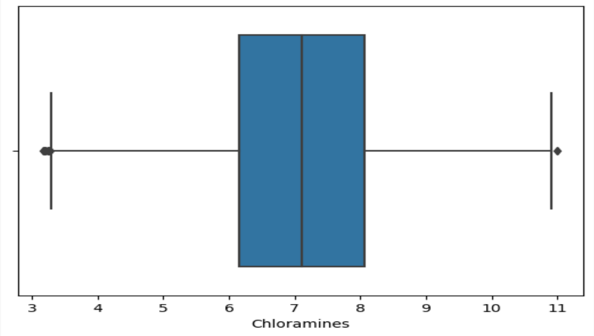
Outlier Removing Process,

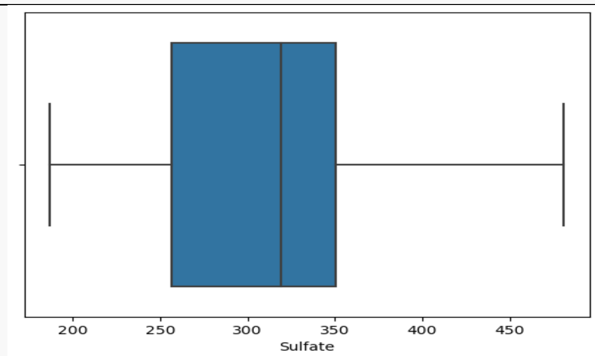
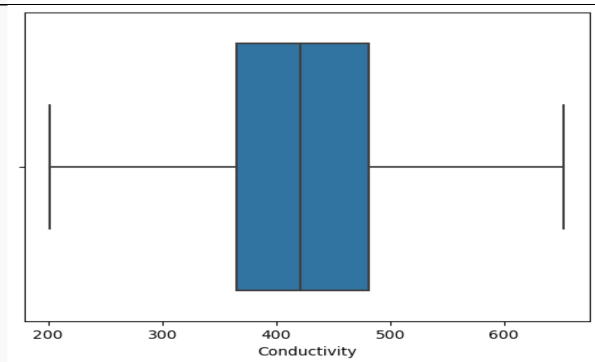
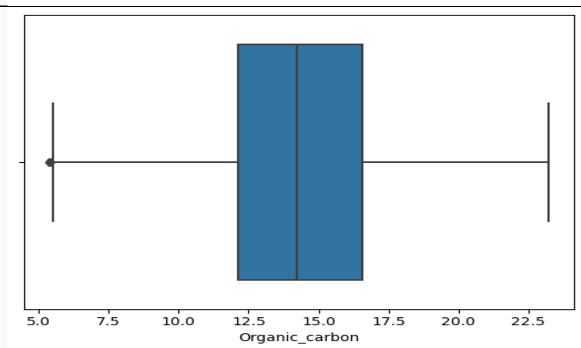
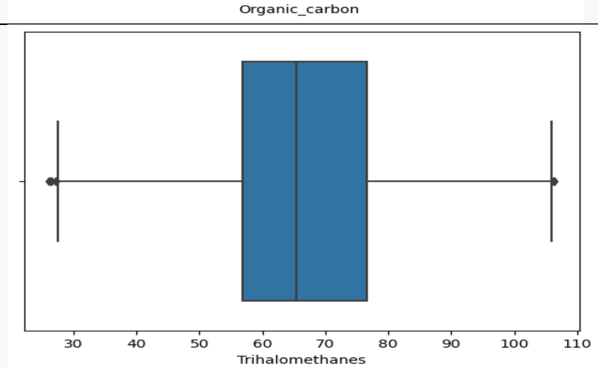
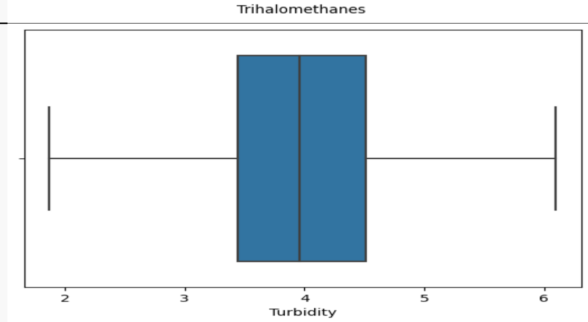
Iteration 01,

Attribute	**Boxplot**
pH	
Hardness	
Solids	
Chloramines	
Sulfate	

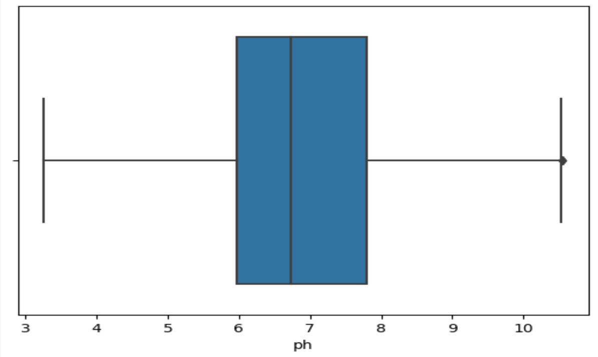
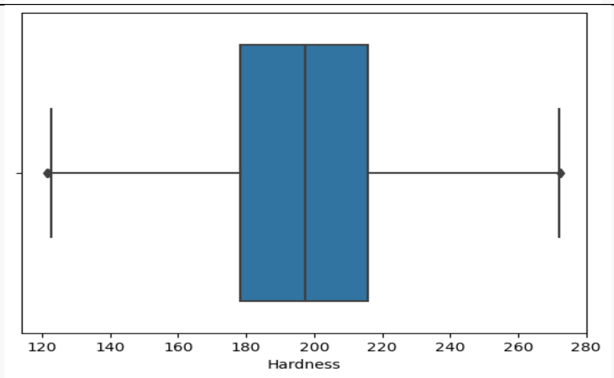
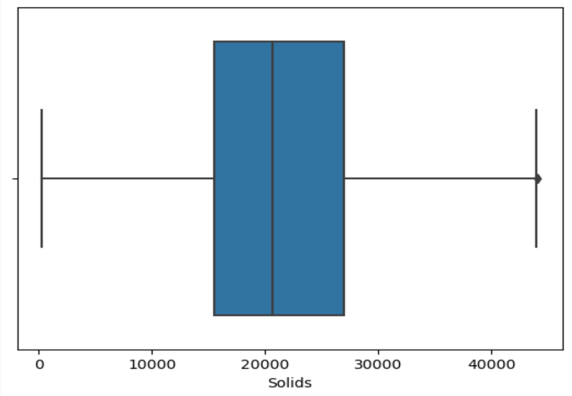
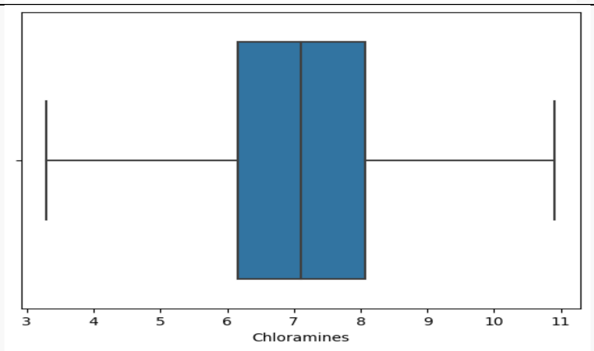
Conductivity		
Organic Carbon		
Trihalomethanes		
Turbidity		

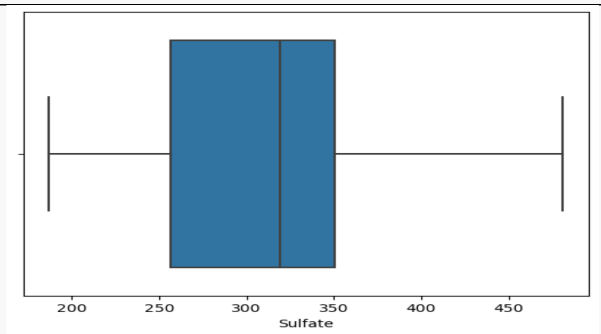
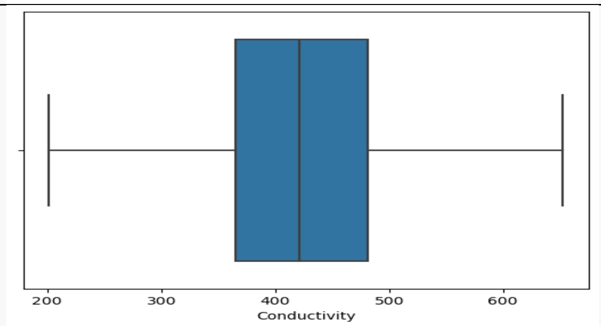
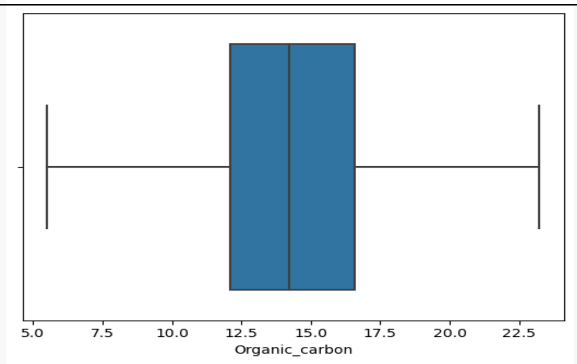
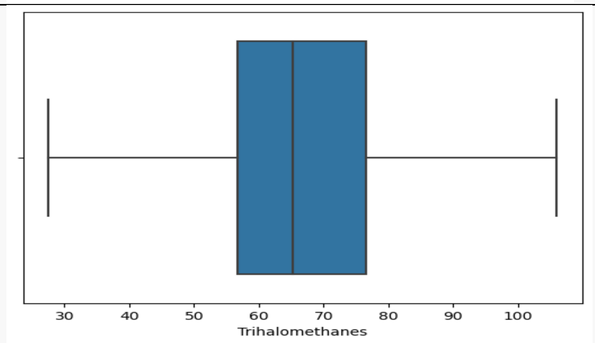
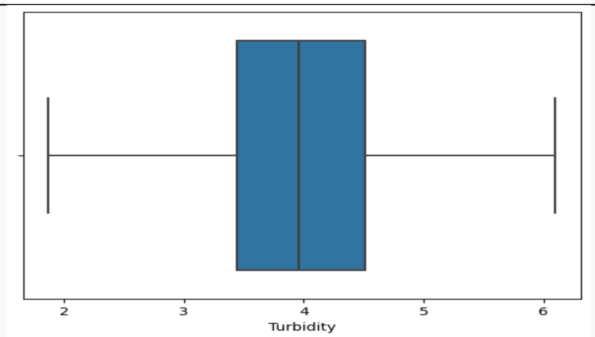
Iteration 02,

Attribute	Boxplot
pH	
Hardness	
Solids	
Chloramines	

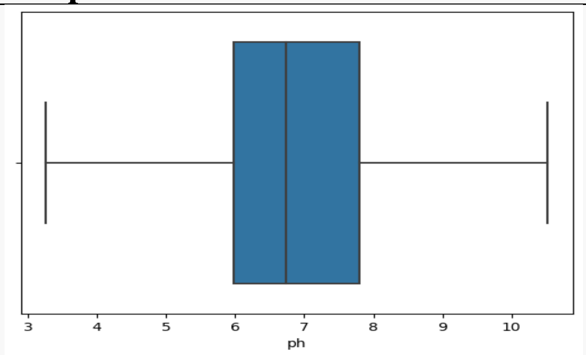
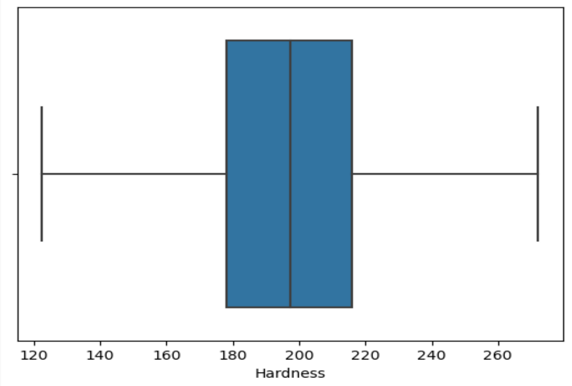
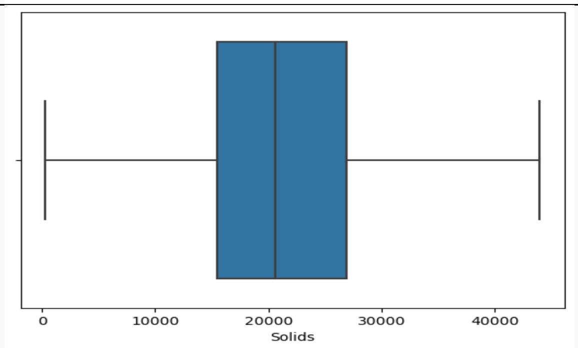
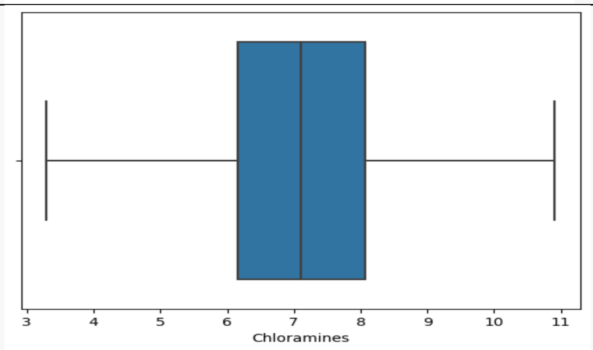
Sulfate	
Conductivity	
Organic Carbon	
Trihalomethanes	
Turbidity	

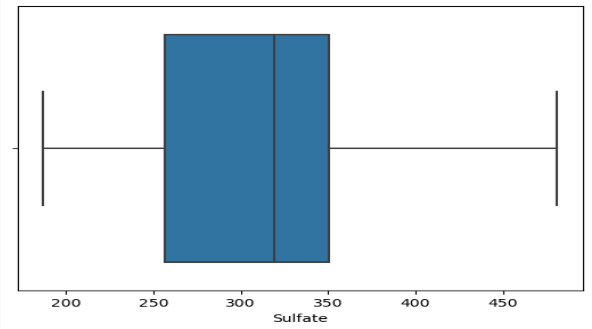
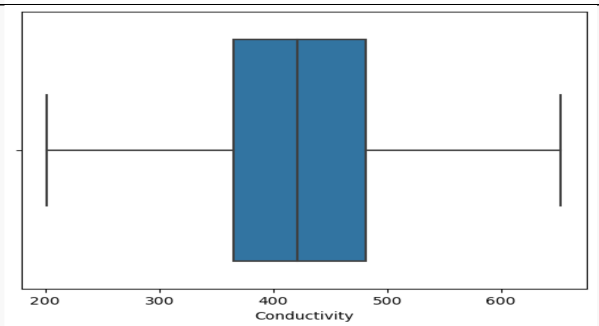
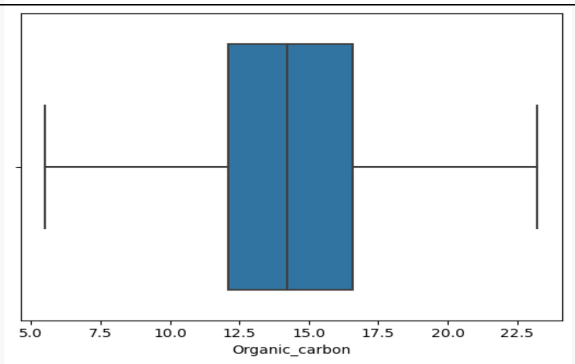
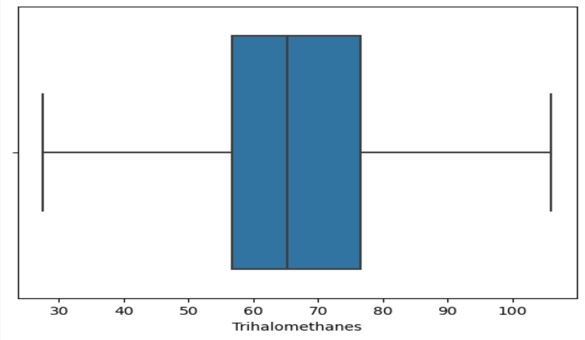
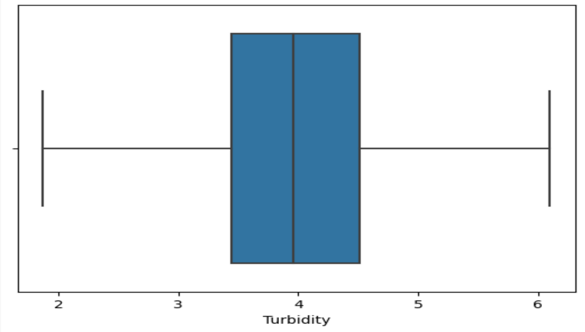
Iteration 03,

Attribute	Boxplot
pH	
Hardness	
Solids	
Chloramines	

<table>
<tr><td>Sulfate</td><td>
</td></tr>
<tr><td>Conductivity</td><td>
</td></tr>
<tr><td>Organic Carbon</td><td>
</td></tr>
<tr><td>Trihalomethanes</td><td>
</td></tr>
<tr><td>Turbidity</td><td>
</td></tr>
</table>

Iteration 03,

Attribute	Boxplot
pH	
Hardness	
Solids	
Chloramines	

Sulfate	
Conductivity	
Organic Carbon	
Trihalomethanes	
Turbidity	

02.01.03. Feature Selection

02.01.03.01. Dropping Constant Features

Removing low variance features which do not affect to final output.

In [8]:
```
#removing class attribute
raw_data2=raw_data.drop(["Potability"],axis=1)
```

In [9]:
```
raw_data2.head()
```

Out[9]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity
0	5.971861	204.890456	20791.31898	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135
1	3.716080	129.422921	18630.05786	6.635246	252.848888	592.885359	15.180013	56.329076	4.500656
2	8.099124	224.236259	19909.54173	9.275884	252.848888	418.606213	16.868637	66.420093	3.055934
3	8.316766	214.373394	22018.41744	8.059332	356.886136	363.266516	18.436525	100.341674	4.628771
4	9.092223	181.101509	17978.98634	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075

In [10]:
```
#import Variance Threshold class
from sklearn.feature_selection import VarianceThreshold
```

In [11]:
```
#create variance threshold object
var_thres=VarianceThreshold(threshold=0)
```

In [12]:
```
var_thres.fit(raw_data2)
```

Out[12]:
```
VarianceThreshold
VarianceThreshold(threshold=0)
```

In [13]:
```
#print low variance features
var_thres.get_support()
```

Out[13]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True])
```

Conclusion: All attributes' variance doesn’t equal to 0. Therefore, no need of removing attributes.

02.01.03.02. Pearson Correlation

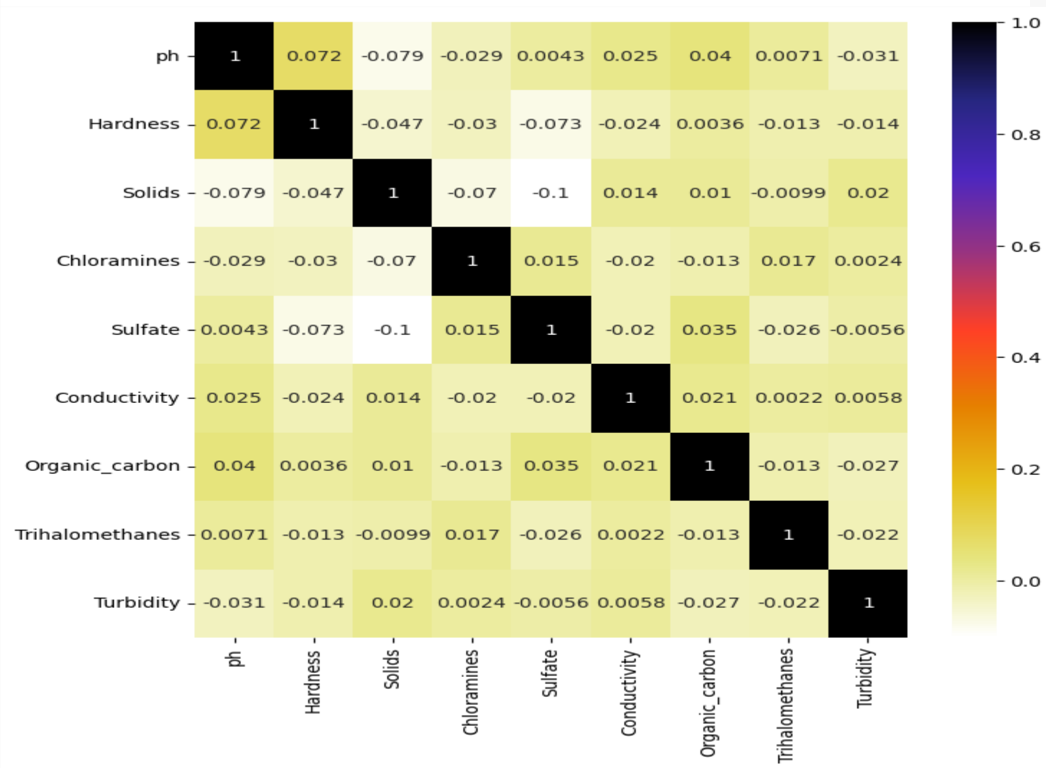
Removing independent features which are highly correlated.

```
In [18]: #calculate correlation
         corr=raw_data2.corr()
         corr
```

Out[18]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity
ph	1.000000	0.072258	-0.078639	-0.028537	0.004309	0.025330	0.040428	0.007124	-0.031166
Hardness	0.072258	1.000000	-0.046899	-0.030054	-0.073472	-0.023915	0.003610	-0.012541	-0.014449
Solids	-0.078639	-0.046899	1.000000	-0.070148	-0.102030	0.013831	0.010242	-0.009866	0.019546
Chloramines	-0.028537	-0.030054	-0.070148	1.000000	0.015360	-0.020486	-0.012653	0.016743	0.002363
Sulfate	0.004309	-0.073472	-0.102030	0.015360	1.000000	-0.020290	0.035004	-0.026357	-0.005601
Conductivity	0.025330	-0.023915	0.013831	-0.020486	-0.020290	1.000000	0.020966	0.002216	0.005798
Organic_carbon	0.040428	0.003610	0.010242	-0.012653	0.035004	0.020966	1.000000	-0.013090	-0.027308
Trihalomethanes	0.007124	-0.012541	-0.009866	0.016743	-0.026357	0.002216	-0.013090	1.000000	-0.021843
Turbidity	-0.031166	-0.014449	0.019546	0.002363	-0.005601	0.005798	-0.027308	-0.021843	1.000000

```
In [25]: # display pearson correlation matrix
         plt.figure(figsize=(8,8))
         sns.heatmap(corr,annot=True,cmap=plt.cm.CMRmap_r)
         plt.show()
```



Conclusion: There is not any highly correlated independent features. Therefore, no need of removing attributes.

02.02. Data Transformation

Under data transformation, converting and scaling data into a format that can be supportive to machine learning model is happened. To transform data Min-Max Normalizing Method is used. All the values are scaled into 0 – 1 range.

Nature of the Dataset Before Data Transformation,

(Following picture shows training input values only)

```
In [21]: print(x_train)

[[6.09408091e+00 2.08303833e+02 2.34953075e+04 ... 5.98033921e+00
  5.72030892e+01 3.21075266e+00]
 [6.50506581e+00 2.26419609e+02 1.69821320e+04 ... 1.85271048e+01
  8.04628104e+01 2.89099852e+00]
 [9.63066548e+00 1.52862434e+02 2.36417026e+04 ... 1.66701839e+01
  8.99975742e+01 4.92009116e+00]
 ...
 [7.44518929e+00 2.25397787e+02 2.47415350e+04 ... 1.55359790e+01
  8.70839193e+01 3.63789458e+00]
 [5.69447555e+00 1.93432130e+02 1.87429252e+04 ... 1.36233080e+01
  5.63264966e+01 4.10746712e+00]
 [6.81046652e+00 2.09735559e+02 3.26023401e+04 ... 1.67367486e+01
  4.23494608e+01 4.40233951e+00]]
```

Python code for Data Transformation,

03). Normalization (Min-Max Scalling)

```
In [16]: #import MinMaxScaler class
         from sklearn.preprocessing import MinMaxScaler

In [17]: # create MinMaxScaler Object which can rescale values into 0-1 range
         scaler=MinMaxScaler(feature_range=(0,1))

In [18]: # Min-Max Scaling
         x_train_scaled=scaler.fit_transform(x_train)
         x_test_scaled=scaler.fit_transform(x_test)
```

Nature of the Dataset After Data Transformation,

(The following picture shows transformed training input values only)

```
In [22]: print(x_train_scaled)

[[0.38998177 0.5711118  0.5310625  ... 0.02657182 0.3769094  0.30919085]
 [0.4465684  0.69287802 0.38180692 ... 0.73848898 0.67368775 0.23272899]
 [0.87691796 0.19845928 0.53441728 ... 0.63312526 0.79534491 0.71793992]
 ...
 [0.57600972 0.68600978 0.55962098 ... 0.56876924 0.75816863 0.41133204]
 [0.33496193 0.4711508  0.42215716 ... 0.4602422  0.36572467 0.52361953]
 [0.48861763 0.58073523 0.73975877 ... 0.63690222 0.18738712 0.5941315 ]]
```

03). Model Development

03.01. Decision Tree Algorithm

Developer Name and ID: Perera K.R.M / IT21462320

Python code for model development,

1) Importing relevant libraries

```
import numpy as np #importing
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

✓ 0.0s

2) Scaling the data

```
scaler = MinMaxScaler() #scaler
```

✓ 0.0s

```
x_scaled = scaler.fit_transform(x)#scaling the xvalues
```

✓ 0.0s

3) Training the Model

```
clf = DecisionTreeClassifier(max_leaf_nodes=20 , random_state=0)
```

] ✓ 0.0s

```
clf.fit(x_scaled_train , y_train) #training the model
```

] ✓ 0.0s

```
DecisionTreeClassifier
DecisionTreeClassifier(max_leaf_nodes=20, random_state=0)
```

4) Evaluating the model

```
from sklearn.metrics import confusion_matrix
```

50] ✓ 0.0s

```
cm = confusion_matrix(y_test , y_pred)
```

51] ✓ 0.0s

+ Code + Markdown

```
cm
```

52] ✓ 0.0s

```
array([[492,  35],
       [151, 188]], dtype=int64)
```

```
accuracy = accuracy_score(y_test , y_pred )
```

43] ✓ 0.0s

```
print(f"Accuracy : {accuracy: .2f}") #accuracy of the model
```

44] ✓ 0.0s

```
Accuracy :  0.79
```

03.02. Support Vector Machine Algorithm

Developer Name and ID: Wijewardhana T.W.P.P / IT21268762

Python code for model development,

Importing the libraries

```
[ ]  import numpy as np
     import matplotlib.pyplot as plt
     import pandas as pd
```

Create the SVM model

```
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf',degree=3, gamma=0.7)
```

Train the SVM model

```
[31] classifier.fit(X_train, y_train)

SVC
SVC(gamma=0.7)
```

Evaluate the model

```
[29] from sklearn.metrics import confusion_matrix, accuracy_score
     y_pred = classifier.predict(X_test)
     cm = confusion_matrix(y_test, y_pred)
     print(cm)
     accuracy_score(y_test, y_pred)

[[325  31]
 [186  36]]
0.6245674740484429
```

03.03. Artificial Neural Network Algorithm

Developer Name and ID: Ransika M.R.T. / IT21177514

Python code for model development,

01). Import Relevant Packages, Classes and Functions

```
In [100]: import tensorflow as tf
```

```
In [101]: from tensorflow import keras
```

```
In [102]: from keras.models import Sequential
```

```
In [103]: from keras.layers import Dense
```

```
In [104]: from keras.optimizers import Adam
```

```
In [105]: import matplotlib.pyplot as plt
```

02). Create the Model

```
In [113]: model=Sequential([
              #input layer(5 columns of raw_data2 object[Temperature,Humidity,Light,CO2,HumidityRatio])
              Dense(units=9,activation="relu",input_shape=(9,)),

              #hidden layer 1
              Dense(units=100,activation="relu"),

              #hidden layer 2
              Dense(units=100,activation="relu"),

              #output layer
              Dense(units=1,activation="sigmoid")
          ])
```

03). Summary of the Model

```
In [114]: model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_36 (Dense)            (None, 9)                 90

 dense_37 (Dense)            (None, 100)               1000

 dense_38 (Dense)            (None, 100)               10100

 dense_39 (Dense)            (None, 1)                 101

=================================================================
Total params: 11291 (44.11 KB)
Trainable params: 11291 (44.11 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

04). Compile the Model

In [115]:
```
model.compile(optimizer=Adam(learning_rate=0.00001),loss="binary_crossentropy",metrics=["accuracy"])
```

05). Train the Model

In [116]:
```
history=model.fit(x=x_train_scaled,y=y_train,epochs=1000,validation_data=(x_test_scaled,y_test))
```

```
Epoch 1/1000
64/64 [==============================] - 1s 4ms/step - loss: 0.6904 - accuracy: 0.6074 - val_loss: 0.6887 - val_accuracy: 0.6
339
Epoch 2/1000
64/64 [==============================] - 0s 2ms/step - loss: 0.6896 - accuracy: 0.6074 - val_loss: 0.6877 - val_accuracy: 0.6
339
Epoch 3/1000
64/64 [==============================] - 0s 3ms/step - loss: 0.6889 - accuracy: 0.6074 - val_loss: 0.6866 - val_accuracy: 0.6
339
Epoch 4/1000
64/64 [==============================] - 0s 3ms/step - loss: 0.6882 - accuracy: 0.6074 - val_loss: 0.6857 - val_accuracy: 0.6
339
Epoch 5/1000
64/64 [==============================] - 0s 2ms/step - loss: 0.6875 - accuracy: 0.6074 - val_loss: 0.6847 - val_accuracy: 0.6
339
Epoch 6/1000
64/64 [==============================] - 0s 3ms/step - loss: 0.6867 - accuracy: 0.6074 - val_loss: 0.6838 - val_accuracy: 0.6
339
Epoch 7/1000
```

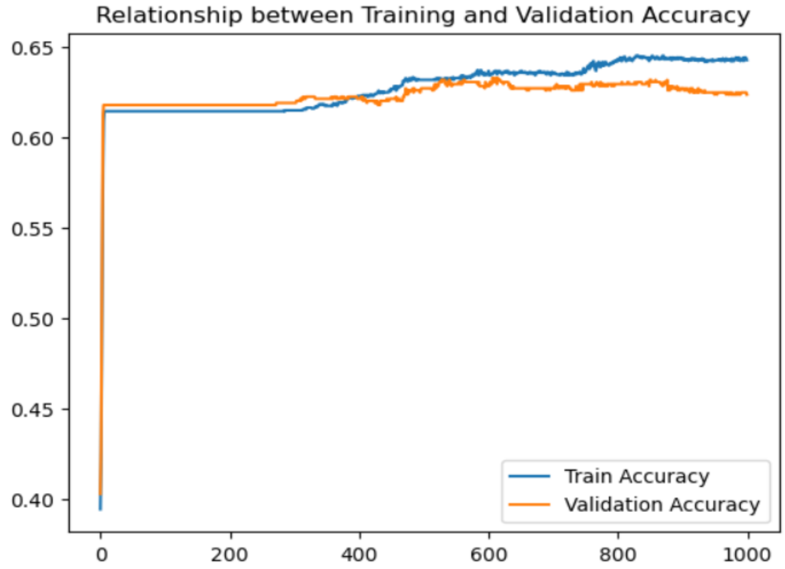
Evaluate the Model

In [35]:
```
test_loss,test_accuracy=model.evaluate(x_test_scaled,y_test)
print(f"Test Loss: {test_loss}")
print(f"Test Accuarcy: {test_accuracy}")
```

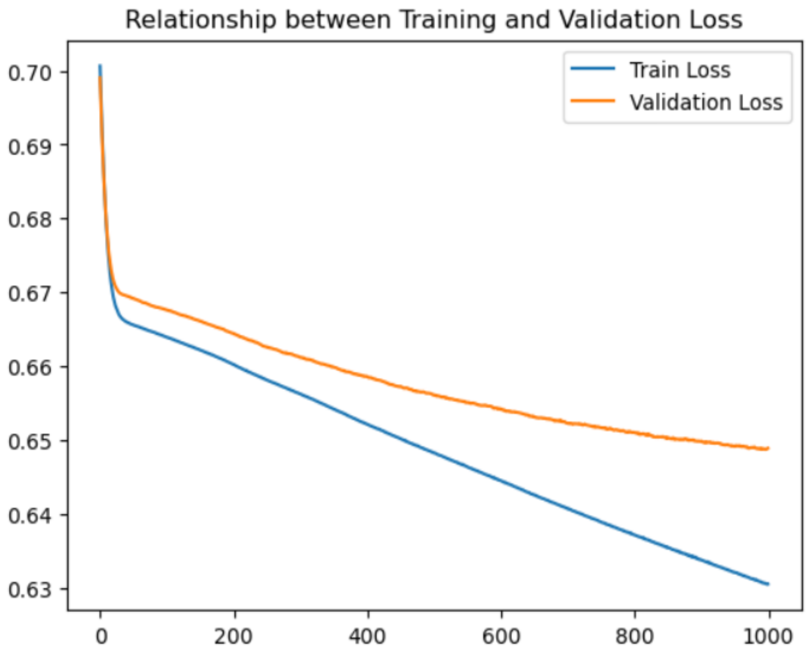
```
28/28 [==============================] - 0s 2ms/step - loss: 0.6489 - accuracy: 0.6236
Test Loss: 0.6488814353942871
Test Accuarcy: 0.6235565543174744
```

Graphs Related to the Neural Network

In [36]:
```
plt.plot(history.history["accuracy"],label="Train Accuracy")
plt.plot(history.history["val_accuracy"],label="Validation Accuracy")
plt.legend()
plt.title("Relationship between Training and Validation Accuracy")
plt.show()
```



In [37]:
```
plt.plot(history.history["loss"],label="Train Loss")
plt.plot(history.history["val_loss"],label="Validation Loss")
plt.legend()
plt.title("Relationship between Training and Validation Loss")
plt.show()
```



03.04. Logistic Regression Algorithm

Developer Name and ID: Ransika M.R.T. / IT21177514

Python code for model development,

01). Import Relevant Packages, Classes and Functions

In [18]:
```
from sklearn.linear_model import LogisticRegression
```

02). Create the Model

In [19]:
```
model=LogisticRegression()
```

03). Train the Model

In [22]:
```
model.fit(x_train_scaled,y_train)
```

Out[22]:
```
LogisticRegression()
```

Evaluate the Model

In [23]:
```
test_accuracy=model.score(x_test_scaled,y_test)

print(f"Test Accuarcy: {test_accuracy}")
```

```
Test Accuarcy: 0.6154734411085451
```

03.05. K-Nearest Neighbors Algorithm

Developer Name and ID: Senarathne H. A. T. S. / IT21207822

Import Nesessary Libraries

```
[2] import pandas as pd
    import sklearn
    from sklearn.model_selection import train_test_split
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.preprocessing import MinMaxScaler
    from sklearn.metrics import accuracy_score
```

Splitting Data

```
[7] X_train,X_test,y_train,y_test = train_test_split(X,y,train_size=0.85, random_state=42)
```

Rescaling the dataset

```
[8] scaler = MinMaxScaler(feature_range=(0,1))
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.fit_transform(X_test)
```

Model Implementation

```
[9] k=2  ##no of classes
    knn_classifier = KNeighborsClassifier(n_neighbors=k)
```

Training the model

```
[10] knn_classifier.fit(X_train,y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=2)
```

Evaluate model

```
[11] y_pred = knn_classifier.predict(X_test)
     accuracy = accuracy_score(y_test,y_pred)
     print(accuracy)
```

```
0.6327944572748267
```

Ensemble Learning Approaches

03.06. Random Forest Algorithm

Developer Name and ID: Senarathne H. A. T. S. / IT21207822

Import Necessary Libraries

```
[ ]  import pandas as pd
     import sklearn
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import accuracy_score,classification_report
```

Import Preprocessed Dataset

```
[ ]  data = pd.read_csv('/content/drive/MyDrive/Outliers Removed Dataset.csv',sep=',')
     data.head()
```

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	5.971861	204.890456	20791.31898	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135	0
1	3.716080	129.422921	18630.05786	6.635246	252.848888	592.885359	15.180013	56.329076	4.500656	0
2	8.099124	224.236259	19909.54173	9.275884	252.848888	418.606213	16.868637	66.420093	3.055934	0
3	8.316766	214.373394	22018.41744	8.059332	356.886136	363.266516	18.436525	100.341674	4.628771	0
4	9.092223	181.101509	17978.98634	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075	0

Dividing input and outputs

```
[ ]  X,y = data1[:, :-1], data1[:,-1]
     print(X)
     print(y)

[[5.97186050e+00 2.04890456e+02 2.07913190e+04 ... 1.03797831e+01
  8.69909705e+01 2.96313538e+00]
 [3.71608007e+00 1.29422921e+02 1.86300579e+04 ... 1.51800131e+01
  5.63290763e+01 4.50065627e+00]
 [8.09912419e+00 2.24236259e+02 1.99095417e+04 ... 1.68686369e+01
  6.64200925e+01 3.05593375e+00]
 ...
 [9.41951032e+00 1.75762646e+02 3.31555782e+04 ... 1.10390697e+01
  6.98454003e+01 3.29887550e+00]
 [5.12676292e+00 2.30603758e+02 1.19838694e+04 ... 1.11689462e+01
  7.74882131e+01 4.70865847e+00]
 [7.87467136e+00 1.95102299e+02 1.74041771e+04 ... 1.61403676e+01
  7.86984463e+01 2.30914906e+00]]
[0. 0. 0. ... 1. 1. 1.]
```

Splitting training and testing sets

```
[ ] X_train,X_test,y_train,y_test = train_test_split(X,y,train_size=0.75, random_state=42)
```

Model Implementation

```
[ ] from sklearn.ensemble import RandomForestClassifier
    ranfrst_model = RandomForestClassifier(n_estimators=100, max_depth=10, random_state=42)
```

Model Training

```
[ ] ranfrst_model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(max_depth=10, random_state=42)
```

Model Evaluation

```
from sklearn.metrics import accuracy_score,confusion_matrix

y_pred = ranfrst_model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))

conf_matrix = confusion_matrix(y_test,y_pred)
print(conf_matrix)
```

```
Accuracy: 0.7853185595567868
[[407  38]
 [117 160]]
```

03.07. AdaBoost Algorithm (Base Estimator: Decision Tree Algorithm)

Developer Name and ID: Ransika M.R.T. / IT21177514

01). Import Relevant Packages, Classes and Functions

In [110]:
```
from sklearn.ensemble import AdaBoostClassifier
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier
```

02). Create the Model

In [186]:
```
#base estimators
#create decison tree object
dt=DecisionTreeClassifier(max_leaf_nodes=20,random_state=0)
```

In [175]:
```
# Create adaboost classifer object
# n_estimators=> Number of week learners
abc = AdaBoostClassifier(n_estimators=200,estimator=dt,learning_rate=0.01)
```

03). Train the Model

In [176]:
```
model = abc.fit(x_train_scaled, y_train)
```

04). Make Predictions

In [177]:
```
y_pred = model.predict(x_test_scaled)
```

Evaluate the Model

In [178]:
```
acc=metrics.accuracy_score(y_test, y_pred)
```

In [179]:
```
print(f"Accuracy: {acc}")
```

```
Accuracy: 0.651270207852194
```

04). Model Evaluation

Algorithm	Achieved Maximum Accuracy Value
Decision Tree Algorithm	0.79
Support Vector Machine Algorithm	0.6455331412103746
Artificial Neural Network Algorithm	0.6235565543174744
Logistic Regression Algorithm	0.6154734411085451
Random Forest Algorithm	0.7853185595567868
K-Nearest Neighbors Algorithm	0.6327944572748267
AdaBoost Algorithm	0.651270207852194

The machine learning model, developed using the Decision Tree Algorithm shows maximum accuracy level. Therefore, it is used to deploy in the web application.

05). Web Application Development and Deployment

05.01. Frontend Development

05.01.01. Code

- Home.html

```html
</head>
<body>
    <div class="banner">
        <nav class="navbar">
            <ul>
                <li><a class="navbar-brand" href="Home.html">Water Portability Checker</a></li>
            </ul>
            <ul>
                <li><a href="Form.html">Check Now</a></li>
                <li><a href="Tips.html">Tips</a></li>
                <li><a href="#">Contact Us</a></li>
            </ul>
        </nav>
        <br><br><br><br><br><br><br><br><br><br>

        <div class="container-body">

            <h1 class="header">Welcome to the Water Portability Checker</h1>

            <p>This application allows you to check the portability of water based on pH level, turbidity, and temperature.</p>

            <br><br><br>
            <button type="button"><span></span> Water Portability</button>
            <button type="button"><span></span> To Preserve Water</button>

        </div>
    </div>
</body>
</html>
```

- Form.html

```
Form.html
Templates > Form.html > html > body > div.banner > div.container > form.form

            <h1 class="header">Enter Values</h1><br>
            <form class="form"action="/portability"  method="post">

                <div class="form-group">
                    <label for="ph">pH Level:</label>
                    <input type="text" class="form-control" id="ph" name="ph" required>
                </div>

                <div class="form-group">
                    <label for="hardness">Hardnss:</label>
                    <input type="text" class="form-control" id="ph" name="hardness" required>
                </div>

                <div class="form-group">
                    <label class="form-lab" for="solids">Solids:</label>
                    <input type="text" class="form-control" id="ph" name="solids" required>
                </div>

                <div class="form-group">
                    <label class="form-lab" for="chloramines">Chloramines:</label>
                    <input type="text" class="form-control" id="ph" name="chloramines" required>
                </div>

                <div class="form-group">
                    <label class="form-lab" for="sulfate">Sulfate:</label>
                    <input type="text" class="form-control" id="ph" name="sulfate" required>
                </div>

                <div class="form-group">
                    <label class="form-lab" for="conductivity">Conductivity:</label>
                    <input type="text" class="form-control" id="ph" name="conductivity" required>
                </div>

                <div class="form-group">
                    <label class="form-lab" for="organic_carbon">Organic Carbon:</label>
                    <input type="text" class="form-control" id="ph" name="organic_carbon" required>
                </div>

                <div class="form-group">
                    <label class="form-lab" for="trihalomethanes">Trihalomethanes:</label>
                    <input type="text" class="form-control" id="ph" name="trihalomethanes" required>
                </div>

                <div class="form-group">
                    <label for="turbidity">Turbidity:</label>
                    <input type="text" class="form-control" id="turbidity" name="turbidity" required>
                </div>
```

- Potablesuccess.html

```
potablesuccess.html
Templates > potablesuccess.html > ...

</head>
<body>
    <div class="banner">
        <nav class="navbar">
            <ul>
                <li><a class="navbar-brand" href="#">Water Portability Checker</a></li>
            </ul>
            <ul>
                <li><a href="Form.html">Check Now</a></li>
                <li><a href="#">Tips</a></li>
                <li><a href="#">Contact Us</a></li>
            </ul>
        </nav>
        <br><br><br><br><br><br><br><br><br><br>

        <div class="container-body">
            <br><br><br><br><br><br><br><br><br>

            <img src="../static/success.png" alt="icon-Suitable">
            <p>Suitable For Consumption</p>



        </div>
    </div>
</body>
</html>
```

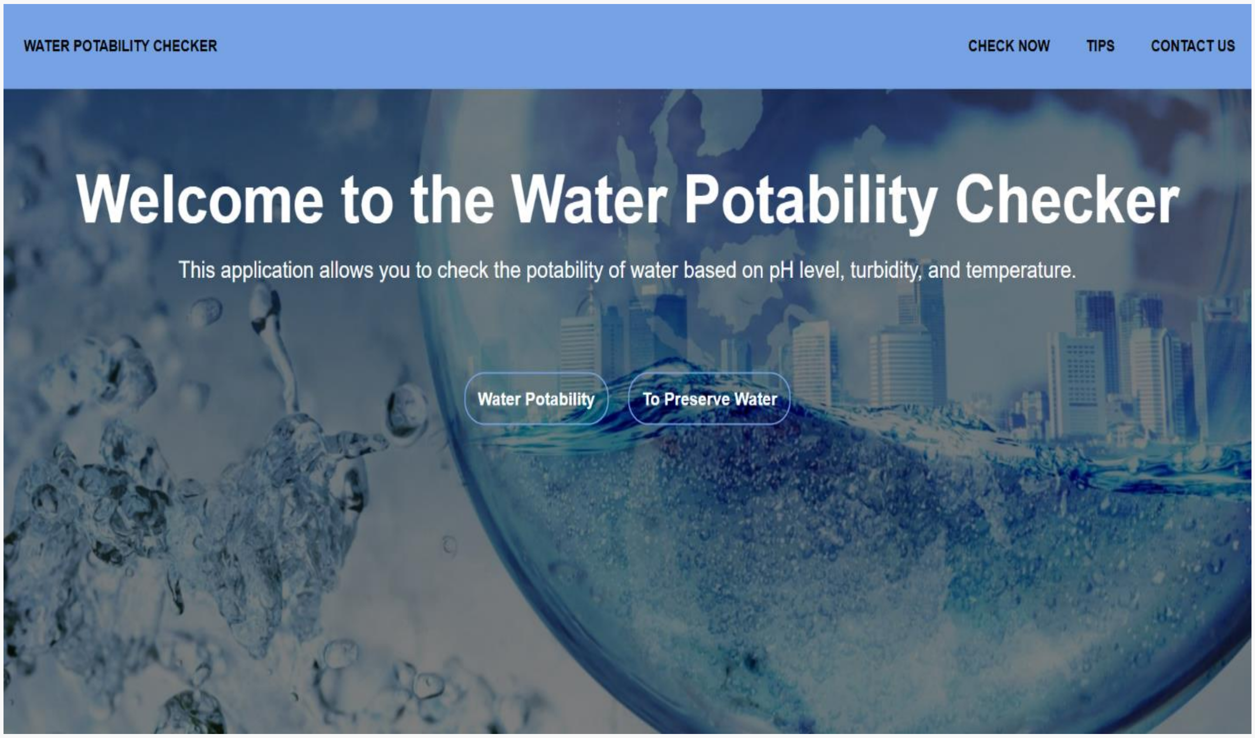
- potableunsuccess.html

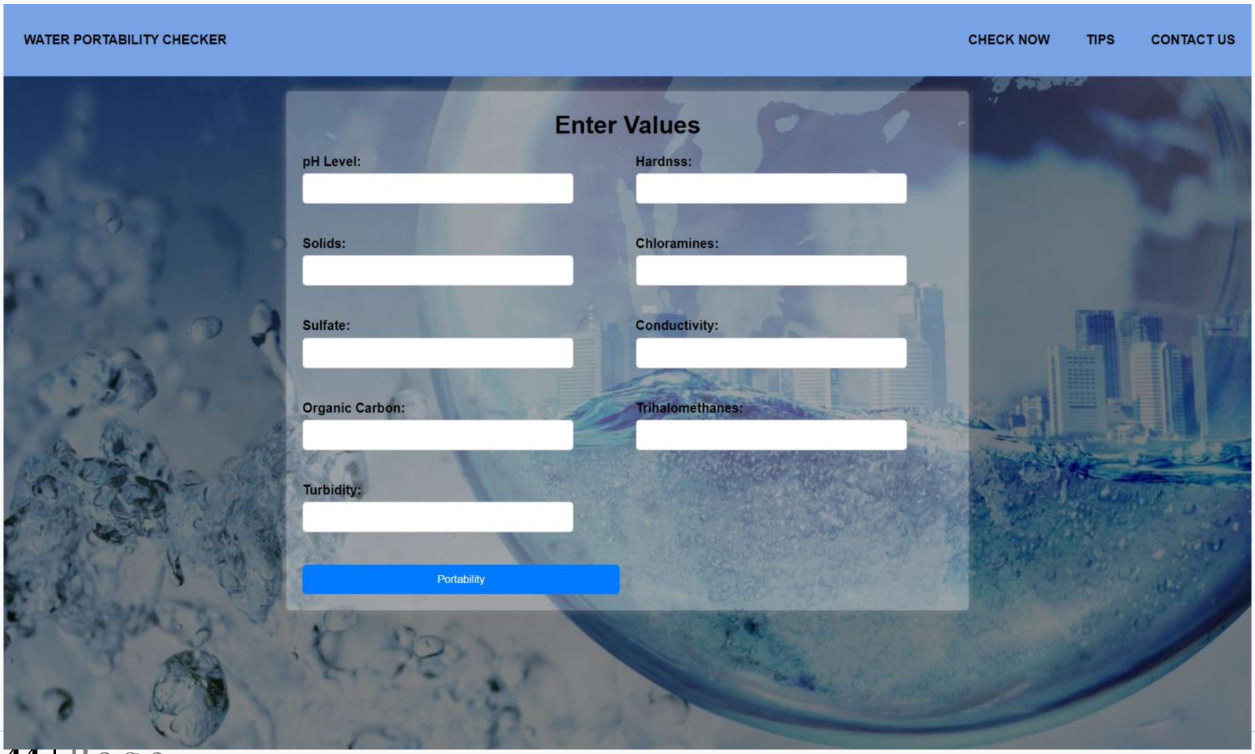
```html
</head>
<body>
    <div class="banner">
        <nav class="navbar">
            <ul>
                <li><a class="navbar-brand" href="#">Water Portability Checker</a></li>
            </ul>
            <ul>
                <li><a href="Form.html">Check Now</a></li>
                <li><a href="#">Tips</a></li>
                <li><a href="#">Contact Us</a></li>
            </ul>
        </nav>
        <br><br><br><br><br><br><br><br><br><br>

        <div class="container-body">
            <br><br><br><br><br><br><br><br><br>

            <img src="../static/unsuccess.png" alt="icon-Suitable">
            <p>Not Suitable For Consumption</p>



        </div>
    </div>
</body>
</html>
```

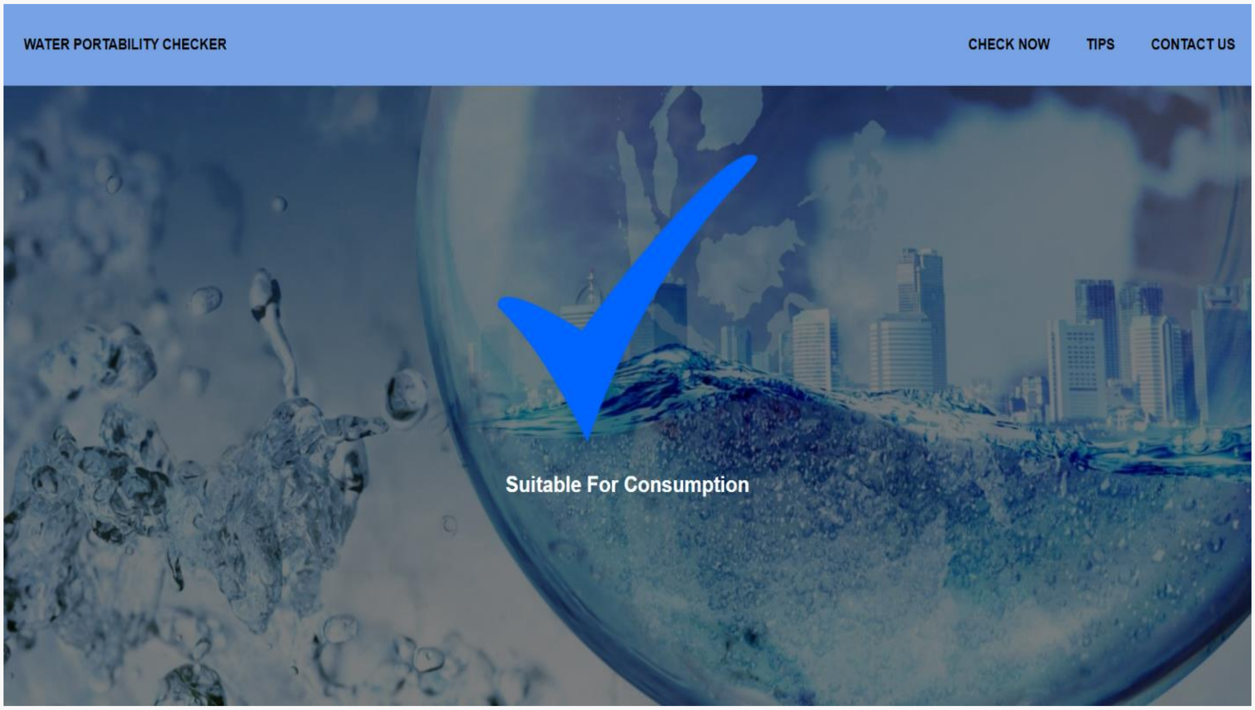
- Tips.html

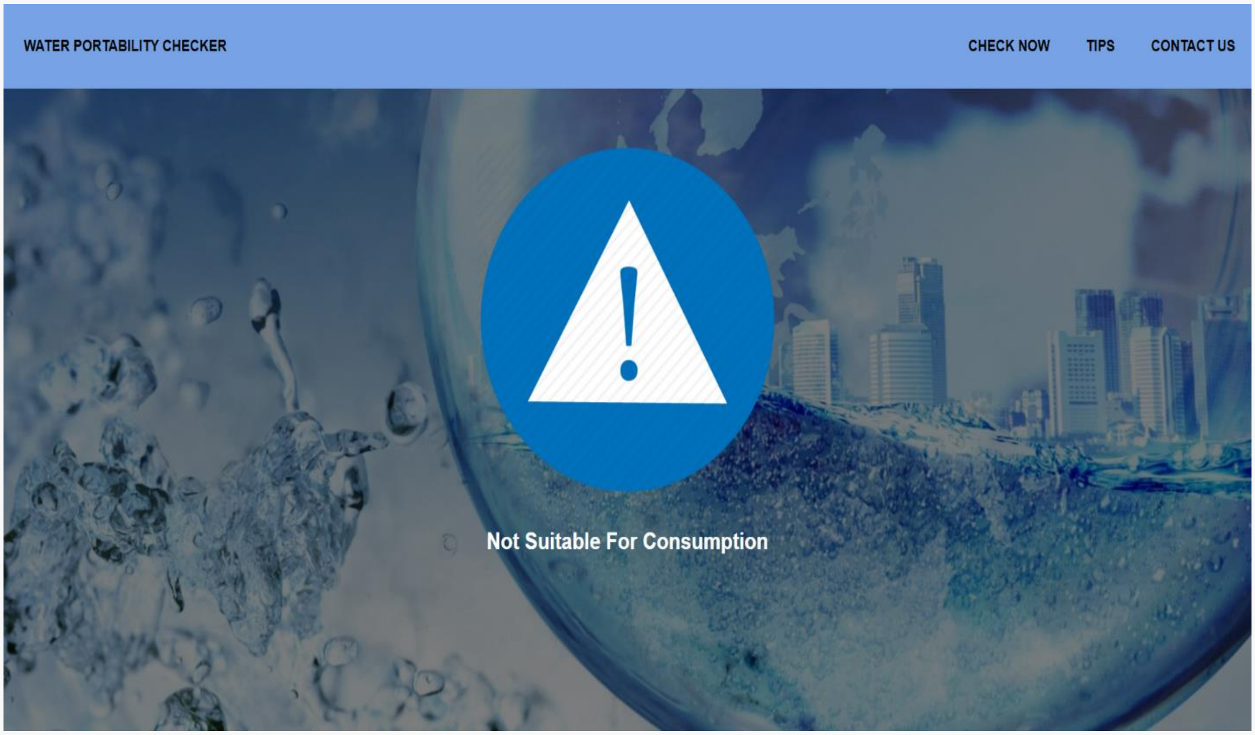
```html
</head>
<body>
    <div class="top">

    <nav class="navbar">
        <ul>
            <li><a class="navbar-brand" href="Home.html">Water Portability Checker</a></li>
        </ul>
        <ul>
            <li><a href="Form.html">Check Now</a></li>
            <li><a href="Tips.html">Tips</a></li>
            <li><a href="#">Contact Us</a></li>
        </ul>
    </nav>

        <div class="topic"><h1>TIPS</h1></div>

        <div class="content1">
        <h2>Consider harvesting rainwater for non-potable uses<br></h2></div>

        <div class="content">
            <h2>If you rely on well water, have it tested regularly for contaminants<br></h2></div>

        <div class="content1">
                <h2>If you are unsure about water quality in your area,then boil the water<br></h2></div>
        <div class="content">
                <h2> Use a water filter for your home to remove impurities and contaminants<br></h2></div>
        <div class="content1">
                <h2> Stay yourself informed about any water quality issues in your area<br></h2>
        </div>

        <div class="content">
            <h2> Store water in clean, food-grade containers to prevent cross-contamination<br></h2>
        </div>
        <div class="content1">
            <h2> Regularly maintain appliances, like water heaters or water softeners<br></h2>
        </div>

        <div class="content">
            <h2> Avoid disposing of hazardous chemicals or pollutants down the drain or toilet<br></h2>
        </div>

        <div class="content1">
            <h2> Regularly Test your water resources for contaminants<br></h2>
        </div>
</div>
</body>
</html>
```

05.01.02. User Interfaces






WATER PORTABILITY CHECKER
CHECK NOW
TIPS
CONTACT US
Suitable For Consumption


WATER PORTABILITY CHECKER
CHECK NOW
TIPS
CONTACT US
Not Suitable For Consumption


WATER PORTABILITY CHECKER
CHECK NOW
TIPS
CONTACT US
TIPS
Consider harvesting rainwater for non-potable uses
If you rely on well water, have it tested regularly for contaminants
If you are unsure about water quality in your area,then boil the water
Use a water filter for your home to remove impurities and contaminants
Stay yourself informed about any water quality issues in your area
Store water in clean, food-grade containers to prevent cross-contamination
Regularly maintain appliances, like water heaters or water softeners
Avoid disposing of hazardous chemicals or pollutants down the drain or toilet
Regularly Test your water resources for contaminants

05.02. Backend Development

05.02.01. Code

```
from flask import Flask , render_template , request
import pickle #Importing the relevant Libraries


app = Flask(__name__ ) #Create Flask application Instance



model = pickle.load(open('saved_model.sav' , 'rb')) #Loading the model
```

```
@app.route('/') #Flask route to render Home page
def home():


    return render_template('Home.html', **locals())

@app.route('/Form.html') #Flask route to render Form
def form():


    return render_template('Form.html')
```

```python
@app.route('/portability' , methods = ['POST' , 'GET']) #Flask route to render form result
def portability():

    ph = float(request.form['ph']) #converting form field data to floating point numbers
    hardness = float(request.form['hardness'])
    solids = float(request.form['solids'])
    chloramines = float(request.form['chloramines'])
    sulfate = float(request.form['sulfate'])
    conductivity = float(request.form['conductivity'])
    organic_carbon = float(request.form['organic_carbon'])
    trihalomethanes = float(request.form['trihalomethanes'])
    turbidity = float(request.form['turbidity'])

    #Making prediction based on input feature values
    result = model.predict([[ph , hardness , solids , chloramines , sulfate , conductivity , organic_carbon , trihalomethanes , turbidity]])

    if result == 1: #If result == 1 render portable success page

        return render_template('potablesuccess.html')
    else:
        return render_template('potableunsuccess.html')

@app.route('/Tips.html') #Flask route to render Tips
def tips():

    return render_template('Tips.html')
```

```python
@app.route('/Home.html') #Flask route to return to Home page
def home1():

    return render_template('Home.html')

if __name__ == '__main__': #Start the flask application with debugging

    app.run(debug=True)
```

06). Finalization

06.01. Roles and Contribution

Student Name	Registration Number	Roles and Contribution
Senarathne H.A.T.S.	IT21207822	**Group Leader** – 1. Frontend Development 2. ML Models (Random Forest Algorithm, K-Nearest Neighbors Algorithm) 3. Presentation
Ransika M.R.T.	IT21177514	1. Data Preprocessing 2. ML Models (Artificial Neural Network, Logistic Regression Algorithm) 3. Documentation 4. Presentation
Perera K.R.M.	IT21462320	1. Backend Development 2. Web Application Deployment 3. Presentation 4. ML Models (Decision Tree Algorithm)
Wijewardhana T.W.P.P.	IT21268762	1. Frontend Development 2. ML Models (Support Vector Machine Algorithm) 3. Presentation

06.02. What are We Learn through FDM Mini Project?

1. We gained knowledge about how to use theoretically taught things in the university in practice, identify the practical difficulties (Ex: Overfitting) that arise, and what methods can be used to overcome them.

2. Gaining knowledge of data preprocessing and methods that can be used to improve the accuracy of a machine learning model.

3. We experienced about how different supervised classification algorithms works for the same dataset.

4. For the first time, we learned how to add AI layer to traditional web application from deploying ML/DL models.

5. Importance of ensemble learning approaches than traditional ML approaches.

6. Time management skills.

7. Group management skills like leadership, decision making, conflicts resolving, emotional support and shared resources.

06.03. Acknowledgement

We would like to thank all the lecturers and laboratory instructors including Dr. Prasanna Sumathipala, and Dr. Amitha Lal Caldera who gave us knowledge theoretically and practically for the successful completion of this project and our parents who supported us from various resources.

-The End-