WATER QUALITY ANALYSIS

PROJECT FINAL REPORT

An Introduction to Water Quality Analysis

Abstract :

Water is perhaps the most precious natural resource after air. Though the surface of the earth is mostly consists of water, only a small part of it is usable, which makes this resource very limited. This precious and limited resource, therefore, must be used with prudence. As water is required for different purposes, the suitability of it must be checked before use. Also, sources of water must be monitored regularly to determine whether they are in sound health or not. Poor condition of water bodies are not only the indictor of environmental degradation, it is also a threat to the ecosystem. In industries, improper quality of water may cause hazards and severe economic loss. Thus, the quality of water is very important in both environmental and economic aspects. Thus, water quality analysis is essential for using it in any purpose. After years of research, water quality analysis is now consists of some standard protocols. There are guidelines for sampling, preservation and analysis of the samples. Here the standard chain of action is discussed briefly so that it may be useful to the analysts and researchers.

INTRODUCTION :

What is Water Quality?

Water Quality can be defined as the chemical, physical and biological characteristics of water, usually in respect to its suitability for a designated use. Water can be used for recreation, drinking, fisheries, agriculture or industry. Each of these designated uses has different defined chemical, physical and biological standards necessary to fulfil the respective purpose. For example, there are stringent standards for water to be used for drinking or swimming compared to that used in agriculture or industry

What is Water Quality Analysis?

After many years of research, water quality standards are put in place to ensure the suitability of efficient use of water for a designated purpose. Water quality analysis is to measure the required parameters of water, following standard methods, to check whether they are in accordance with the standard.

Why Water Quality Analysis is required?

Water quality analysis is required mainly for monitoring purpose. Some importance of such assessment includes:

(i) To check whether the water quality is in compliance with the standards, and hence, suitable or not for the designated use. (ii) To monitor the efficiency of a system, working for

(ii) To monitor the efficiency of a system, working for water quality maintenance

(iii) To check whether upgradation / change of an existing system is required and to decide what changes should take place

(iv) To monitor whether water quality is in compliance with rules and regulations.

PROCEDURES OF WATER QUALITY ANALYSIS :

The steps for water quality analysis in general is mentioned in Figure-1.

Selection of Parameters:

The parameters of water quality are selected entirely according to the need for a specific use of that water. Some examples are

Drinking : As per WHO/CPCB Standards

Irrigation : pH

Conductivity

Sodium & Potassium

Nutrients

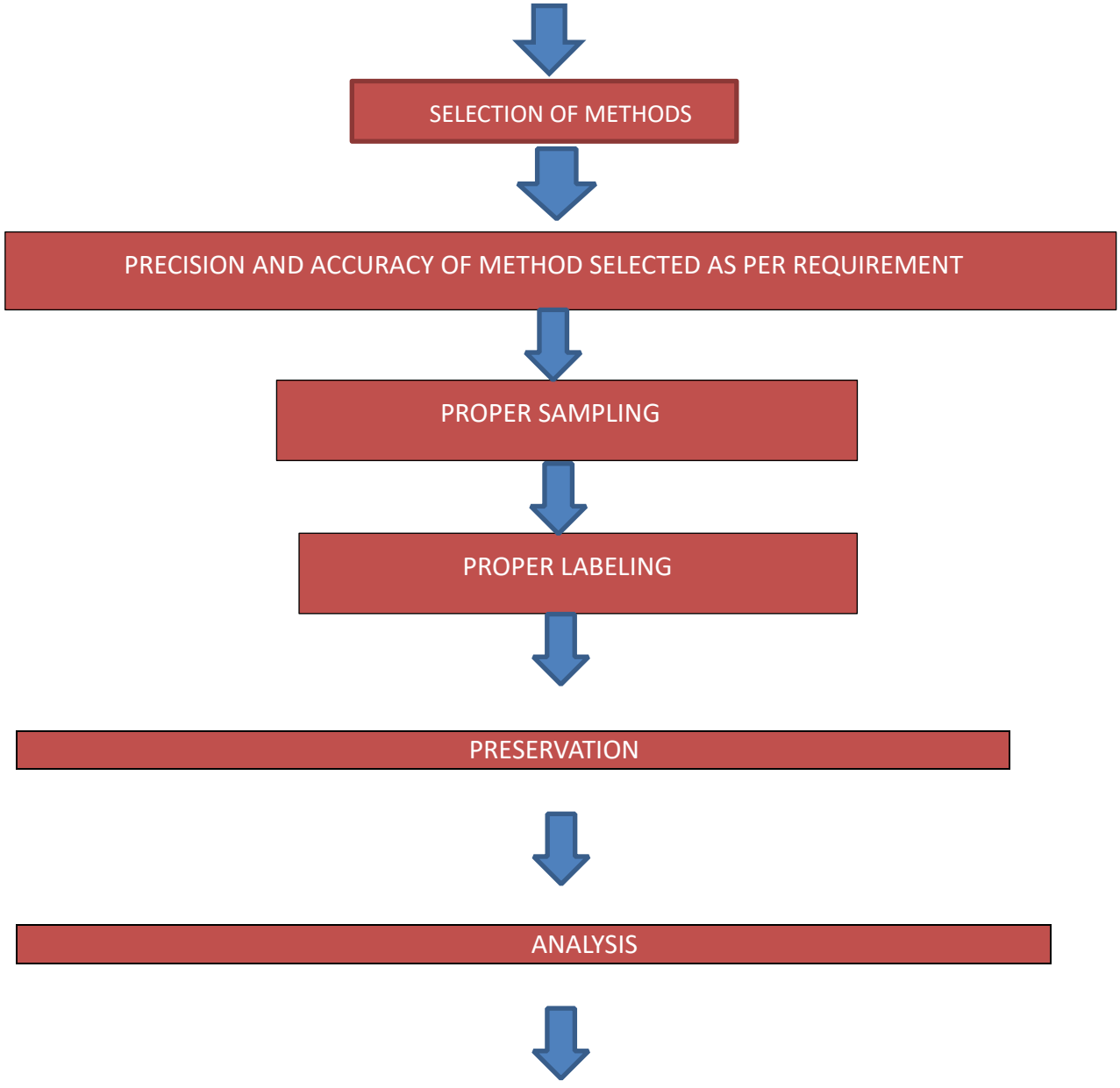
Specific compounds

Industries: As per specific requirement

Domestic Consumption: As per BIS Standards

Water Bodies: As per CPCB guidelines

Figure -1: Steps for Water Quality Analysis



Selection of Methods :

The methods of water quality analysis are selected according to the requirement. The factors playing key role for the selection of methods are

(i) Volume and number of sample to be analyzed
(ii) Cost of analysis
(iii) Precision required
(iv) Promptness of the analysis as required

Precision and Accuracy of Method Selected as per Requirement :

What precision and accuracy to be maintained against a particular method is decided according to the objective of the monitoring. The factors influencing this decision includes:

Budget of Monitoring System

Parameters to be Monitored

Use of the Water

Proper Sampling :

Proper sampling is a vital condition for correct measurement of water quality parameters. Even if advanced techniques and sophisticated tools are used, the parameters can give an incorrect image of the actual scenario due to improper sampling. Proper Labeling

Proper labeling :

prevents sample misidentification and ensures the responsibility and accountability of the collector. The sample container should be labeled properly, preferably by attaching an appropriately inscribed tag or label. Alternatively, the bottle can be labeled directly with a waterproof marker. Barcode labels are also available nowadays. Information on the sample container or the tag should include at least:

(i) Sample code number (identifying location)
(ii) Date and time of sampling
(iii) Source and type of sample
(iv) Pre-treatment or preservation carried out on the sample

(v) Any special notes for the analyst

(vi) Sampler's name

Preservation :

Usually a delay occurs between the collection and analysis of a sample. The characteristics of the sample can be changed during this period. Therefore proper preservation is required in the way to laboratory after collection, and in the laboratory upto when analysis starts.

Complete and unequivocal preservation of samples, whether domestic wastewater, industrial wastes, or natural waters, is a practical impossibility because complete stability for every constituent never can be achieved. At best, preservation techniques only retard chemical (especially, hydrolysis of constituents) and biological changes that inevitably continue after sample collection.

No single method of preservation is entirely satisfactory; the preservative is chosen with due regard to the determinations to be made. Preservation methods are limited to pH control, chemical addition, the use of amber and opaque bottles, refrigeration, filtration, and freezing.

Analysis :

The samples, after reaching laboratory, are analyzed, according to the requisite parameters, following standard methods and protocols.

Reporting :

The ultimate procedure of water analysis is to prepare a proper report against the submitted requisition. The report must be authenticated before handing over the authority. All data should be kept in the laboratory log and preferably in laboratory database.

An alternative way to present the overall quality of water is to express it in the form of Water Quality Index (WQI). WQI is a concise numerical representation of overall water quality of a water body, which is convenient to interpret and used widely. WQI expresses the overall quality of water with a single digit, instead of many digits for all the WQP. Thus, it is readily conceivable for common people.

PROBLEM STATEMENT :

Harmful germs and chemical can get in the water from may source ,including: fertilizers, pesticides or other chemicals that have been applied to land near the water . concentrated feeding operations manufacturing operations

Every year, more people will die from unsafe water that from all forms of violence, including war (Decade, water for life). The main reason behind water pollution is due to plastics , ship water and how those can affect human civilization.

Innovation part in water quality analysis for projects

Innovations in water quality analysis are crucial for ensuring safe and sustainable water resources for various projects. Advanced technologies and approaches can enhance the accuracy, efficiency, and cost-effectiveness of water quality analysis. Here are some innovative aspects to consider for water quality analysis in projects

Sensor Technology

Utilize advanced sensor technologies that can continuously monitor water quality parameters in real-time. These sensors can measure parameters such as pH, dissolved oxygen, turbidity, conductivity, and specific ions.

Remote Sensing and GIS:

Integrate remote sensing and Geographic Information System (GIS) technologies to monitor and analyze water quality over a large geographic area. These technologies can provide spatial and temporal data for better decision-making.

Machine Learning and AI:

• Apply machine learning and artificial intelligence algorithms to analyze vast amounts of water quality data and identify patterns, trends, and potential anomalies. This can help in early detection of pollution events and proactive management of water quality.

Blockchain for Data Integrity:

• Implement blockchain technology to ensure the integrity and traceability of water quality data. This can be particularly useful in projects where data authenticity and tamper-proof records are crucial.

Automated Sampling and Analysis:

• Develop automated systems for sampling and analysis that can collect water samples at specified intervals and perform analyses on-site. This reduces the need for manual sampling and transportation of samples to laboratories.

Mobile Applications for Data Collection:

• Develop mobile applications that field personnel can use to input and transmit water quality data directly from the field to a centralized database. This enhances data collection efficiency and reduces the risk of errors.

Miniaturized Lab-on-a-Chip Technology:
• Utilize miniaturized lab-on-a-chip devices that can conduct multiple water quality tests using very small amounts of sample, reducing the time and resources required for analysis.

Nanotechnology for Detection:
• Implement nanotechnology-based sensors for detecting trace levels of contaminants in water. Nanomaterials can provide high sensitivity and selectivity for various pollutants.

Citizen Science and Crowdsourcing:
• Involve the community through citizen science initiatives and crowdsourcing to gather water quality data. This not only expands data collection capabilities but also promotes public awareness and engagement in water quality issues.

Augmented Reality (AR) for Field Analysis:
• Incorporate augmented reality into field equipment to provide real-time overlays of data, graphs, and analysis results during field measurements. This can enhance on-the-spot decision-making and interpretation of results.

Integration of Environmental DNA (eDNA) Analysis:
• Utilize eDNA analysis to detect and identify aquatic organisms, pathogens, and invasive species, providing additional insights into the ecosystem's health and potential sources of pollution

Development Part

Developing a water quality analysis project typically involves several key steps and components. Here is a high-level overview of the development process for such a project:

1. Project Definition
Define the project's objectives: Determine the specific goals of your water quality analysis project. What are you trying to achieve? For example, you might want to monitor and improve the water quality in a specific area or analyze the impact of certain pollutants on aquatic ecosystems.
2. Data Collection:
• Select data sources: Identify the sources of water quality data, such as sensors, monitoring stations, or historical datasets.
• Set up data collection methods: Implement data collection methods, which may include field sampling, sensor installations, or data acquisition from external sources.
3. Sensor Deployment (if applicable):
• Choose appropriate sensors: Select the right sensors for measuring parameters like pH, turbidity, temperature, dissolved oxygen, nutrients, and contaminants. • Install and calibrate

sensors: Properly install the sensors at designated locations and calibrate them to ensure accurate data collection.

4. Data Storage and Management:

• Establish a database: Set up a database system to store and manage the collected data. This can be a local database or cloud-based solution.

• Data quality control: Implement data quality control measures to filter out erroneous or inconsistent data.

5. Data Analysis:

• Develop analysis methods: Choose appropriate analytical techniques to assess water quality. This may involve statistical analysis, machine learning models, or GIS (Geographic Information System) tools.

• Visualizations: Create informative visualizations (graphs, maps, charts) to communicate the results effectively.

6. Reporting and Interpretation:

• Generate reports: Produce reports or dashboards summarizing the analysis findings.

• Interpret results: Analyze the data to draw conclusions about water quality trends, potential issues, and areas of improvement.

7. Feedback and Decision-Making

• Share results: Communicate the findings with relevant stakeholders, including government agencies, environmental organizations, and the public

• Inform decision-making: Use the analysis to inform policy decisions, water management strategies, and environmental initiatives.

8. Continuous Monitoring and Improvement:

• Establish ongoing monitoring: Continue to collect and analyze water quality data over time to track changes and ensure that water quality goals are met.

• Adapt and improve: Use feedback and new data to adapt and improve the project, including sensor maintenance, data collection methods, and analysis techniques.

9. Regulatory Compliance:

• Ensure compliance: If your project is subject to environmental regulations or standards, ensure that your monitoring and analysis processes adhere to these requirements.

10. Public Engagement and Education:

• Engage the community: Involve the local community and stakeholders in the project to raise awareness about water quality issues and encourage responsible water use.

11. Environmental Mitigation:

• If the analysis reveals water quality issues, develop and implement mitigation strategies to address the problems, such as pollution control measures, restoration projects, or policy changes.

12. Long-Term Sustainability:

• Consider the long-term sustainability of the project, including funding, personnel, and ongoing support.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
plt.style.use('dark_background')
import seaborn as sns
color = sns.color_palette()
import plotly.express          as ex
import plotly.graph_objs       as go
import plotly.offline          as pyo
import scipy.stats             as stats
import pymc3                   as pm
import theano.tensor           as tt
from matplotlib.colors import ListedColormap
from scipy.stats import norm, boxcox
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score
from collections import Counter
from scipy import stats
from tqdm import tqdm_notebook

from sklearn import metrics
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error,
confusion_matrix, r2_score, accuracy_score
from sklearn.model_selection import (GridSearchCV, KFold, train_test_split,
cross_val_score)

from imblearn.over_sampling import SMOTE
from collections import Counter

from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from sklearn import svm
from xgboost.sklearn import XGBClassifier
from sklearn.tree import DecisionTreeClassifier
```

Importing The Dataset

```
path = "water_potability.csv"
df = pd.read_csv(path)
```

Initial Analysis

```
df.shape
```

```
(3276, 10)
```

```
df.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
```

```
df.nunique()
```

```
ph                2785
Hardness          3276
Solids            3276
Chloramines       3276
Sulfate           2495
Conductivity      3276
Organic_carbon    3276
Trihalomethanes   3114
Turbidity         3276
Potability           2
dtype: int
```

Statistical Analysis

```
df.describe().T.style.background_gradient(subset=['mean','std','50%','count'],
cmap='PuBu')
```

```
#Portability is 1 - means good for Human
df[df['Potability']==1].describe().T.style.background_gradient(subset=['mean',
'std','50%','count'], cmap='PuBu')
```

```
# Portability is 0 - means not good for Human
df[df['Potability']==0].describe().T.style.background_gradient(subset=['mean',
'std','50%','count'], cmap='RdBu')
```

Check for missing values

```
plt.title('Missing Values Per Feature')
nans = df.isna().sum().sort_values(ascending=False).to_frame()
sns.heatmap(nans,annot=True,fmt='d',cmap='vlag')
```

```
################################### Imputing 'ph' value
#####################################

phMean_0 = df[df['Potability'] == 0]['ph'].mean(skipna=True)
df.loc[(df['Potability'] == 0) & (df['ph'].isna()), 'ph'] = phMean_0
phMean_1 = df[df['Potability'] == 1]['ph'].mean(skipna=True)
df.loc[(df['Potability'] == 1) & (df['ph'].isna()), 'ph'] = phMean_1

################################### Imputing 'Sulfate' value
#####################################

SulfateMean_0 = df[df['Potability'] == 0]['Sulfate'].mean(skipna=True)
df.loc[(df['Potability'] == 0) & (df['Sulfate'].isna()), 'Sulfate'] =
SulfateMean_0
SulfateMean_1 = df[df['Potability'] == 1]['Sulfate'].mean(skipna=True)
df.loc[(df['Potability'] == 1) & (df['Sulfate'].isna()), 'Sulfate'] =
SulfateMean_1

############################## Imputing 'Trihalomethanes' value
#####################################

TrihalomethanesMean_0 = df[df['Potability'] ==
0]['Trihalomethanes'].mean(skipna=True)
```

```
df.loc[(df['Potability'] == 0) & (df['Trihalomethanes'].isna()),
'Trihalomethanes'] = TrihalomethanesMean_0
TrihalomethanesMean_1 = df[df['Potability'] ==
1]['Trihalomethanes'].mean(skipna=True)
df.loc[(df['Potability'] == 1) & (df['Trihalomethanes'].isna()),
'Trihalomethanes'] = TrihalomethanesMean_1
```

```
print('Checking to see any more missing data \n')
df.isna().sum()
```

Checking to see any more missing data

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

Exploratory Data Analysis

```
Corrmat = df.corr()
plt.subplots(figsize=(7,7))
sns.heatmap(Corrmat, cmap="YlGnBu", square = True, annot=True, fmt='.2f')
plt.show()
```

```
fig =  ex.pie (df, names = "Potability", hole = 0.4, template = "plotly_dark")
fig.show ()
```

```
sns.violinplot(x='Potability', y='ph', data=df, palette='rocket')
```

```
print('Boxplot and density distribution of different features by
Potability\n')

fig, ax = plt.subplots(ncols=2, nrows=9, figsize=(14, 28))
```

```
features = list(df.columns.drop('Potability'))
i=0
for cols in features:
    sns.kdeplot(df[cols], fill=True, alpha=0.4, hue = df.Potability,
                palette=('indianred', 'steelblue'), multiple='stack',
ax=ax[i,0])

    sns.boxplot(data= df, y=cols, x='Potability', ax=ax[i, 1],
               palette=('indianred', 'steelblue'))
    ax[i,0].set_xlabel(' ')
    ax[i,1].set_xlabel(' ')
    ax[i,1].set_ylabel(' ')
    ax[i,1].xaxis.set_tick_params(labelsize=14)
    ax[i,0].tick_params(left=False, labelleft=False)
    ax[i,0].set_ylabel(cols, fontsize=16)
    i=i+1

plt.show()
```

SMOTE

```
##################### Preparing the Data for Modelling #####################

X = df.drop('Potability', axis = 1).copy()
y = df['Potability'].copy()

############################## Train-Test split ############################
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25)

########################## Synthetic OverSampling ##########################
print('Balancing the data by SMOTE - Oversampling of Minority level\n')
smt = SMOTE()
counter = Counter(y_train)
print('Before SMOTE', counter)
X_train, y_train = smt.fit_resample(X_train, y_train)
counter = Counter(y_train)
print('\nAfter SMOTE', counter)

################################## Scaling #################################
ssc = StandardScaler()

X_train = ssc.fit_transform(X_train)
X_test = ssc.transform(X_test)

modelAccuracy = list()
```

Modelling and Prediction

```
model = [LogisticRegression(), DecisionTreeClassifier(), GaussianNB(),
RandomForestClassifier(),
        svm.LinearSVC(), XGBClassifier()]
trainAccuracy = list()
testAccuracy = list()
kfold = KFold(n_splits=10, random_state=7, shuffle=True)

for mdl in model:
    trainResult = cross_val_score(mdl, X_train, y_train, scoring='accuracy',
cv=kfold)
    trainAccuracy.append(trainResult.mean())
    mdl.fit(X_train, y_train)
    y_pred = mdl.predict(X_test)
    testResult = metrics.accuracy_score(y_test, y_pred)
    testAccuracy.append(testResult)
```

```
print('Random Forest Classifier\n')
Rfc = RandomForestClassifier()
Rfc.fit(X_train, y_train)

y_Rfc = Rfc.predict(X_test)
print(metrics.classification_report(y_test, y_Rfc))
print(modelAccuracy.append(metrics.accuracy_score(y_test, y_Rfc)))

sns.heatmap(confusion_matrix(y_test, y_Rfc), annot=True, fmt='d')
plt.show()
```

```
Random Forest Classifier

              precision    recall  f1-score   support

           0       0.83      0.79      0.81       506
           1       0.69      0.74      0.72       313

    accuracy                           0.77       819
   macro avg       0.76      0.77      0.76       819
weighted avg       0.78      0.77      0.78       819

None
```

```
################################### XGB Classifier() ######################
print('XGB Classifier\n')
xgb = XGBClassifier()
xgb.fit(X_train, y_train)
```

OUTPUT :

	count	mean	std	min	25%	50%	75%	
ph	2785.000000	7.080795	1.594320	0.000000	6.093092	7.036752	8.062066	14.0
Hardness	3276.000000	196.369496	32.879761	47.432000	176.850538	196.967627	216.667456	323.1
Solids	3276.000000	22014.092526	8768.570828	320.942611	15666.690297	20927.833607	27332.762127	61227.1
Chloramines	3276.000000	7.122277	1.583085	0.352000	6.127421	7.130299	8.114887	13.1
Sulfate	2495.000000	333.775777	41.416840	129.000000	307.699498	333.073546	359.950170	481.0
Conductivity	3276.000000	426.205111	80.824064	181.483754	365.734414	421.884968	481.792304	753.3
Organic_carbon	3276.000000	14.284970	3.308162	2.200000	12.065801	14.218338	16.557652	28.3
Trihalomethanes	3114.000000	66.396293	16.175008	0.738000	55.844536	66.622485	77.337473	124.0
Turbidity	3276.000000	3.966786	0.780382	1.450000	3.439711	3.955028	4.500320	6.7

From the above table, we can see that the count of each feature are not same. so there must me some null values. Feature Solids has the high mean and standard deviation comparted to other feature. so the distribution must be high. However, the above description is for overall population. lets try the same for 2 samples based on Portability feature

```
#Portability is 1 - means good for Human
df[df['Potability']==1].describe().T.style.background_gradient(subset=['mean','std','50%','count'], cma
```

	count	mean	std	min	25%	50%	75%	
ph	1101.000000	7.073783	1.448048	0.227499	6.179312	7.036752	7.933068	13.1
Hardness	1278.000000	195.800744	35.547041	47.432000	174.330531	196.632907	218.003420	323.1
Solids	1278.000000	22383.991018	9101.010208	728.750830	15668.985035	21199.386614	27973.236446	56488.6
Chloramines	1278.000000	7.169338	1.702988	0.352000	6.094134	7.215163	8.199261	13.1
Sulfate	985.000000	332.566990	47.692818	129.000000	300.763772	331.838167	365.941346	481.0
Conductivity	1278.000000	425.383800	82.048446	201.619737	360.939023	420.712729	484.155911	695.3
Organic_carbon	1278.000000	14.160893	3.263907	2.200000	12.033897	14.162809	16.356245	23.6
Trihalomethanes	1223.000000	66.539684	16.327419	8.175876	56.014249	66.678214	77.380975	124.0
Turbidity	1278.000000	3.968328	0.780842	1.492207	3.430909	3.958576	4.509569	6.4

```
# Portability is 0 - means not good for Human
df[df['Potability']==0].describe().T.style.background_gradient(subset=['mean','std','50%','count'], cma
```

	count	mean	std	min	25%	50%	75%	
ph	1684.000000	7.085378	1.683499	0.000000	6.037723	7.035456	8.155510	14.0
Hardness	1998.000000	196.733292	31.057540	98.452931	177.823265	197.123423	216.120687	304.2
Solids	1998.000000	21777.490788	8543.068788	320.942611	15663.057382	20809.618280	27006.249009	61227.1
Chloramines	1998.000000	7.092175	1.501045	1.683993	6.155640	7.090334	8.066462	12.6
Sulfate	1510.000000	334.564290	36.745549	203.444521	311.264006	333.389426	356.853897	460.1
Conductivity	1998.000000	426.730454	80.047317	181.483754	368.498530	422.229331	480.677198	753.3
Organic_carbon	1998.000000	14.364335	3.334554	4.371899	12.101057	14.293508	16.649485	28.3
Trihalomethanes	1891.000000	66.303555	16.079320	0.738000	55.706530	66.542198	77.277704	120.0
Turbidity	1998.000000	3.965800	0.780282	1.450000	3.444062	3.948076	4.496106	6.7

Mean and std of almost all features are similar for both samples. there are few differnces in Solids feature. Further analysis using hypothetical testing could help us to identify the significance.

Out[11]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	P
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.180013	56.329076	4.500656	
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.868637	66.420093	3.055934	
11	7.974522	218.693300	18767.656682	8.110385	NaN	364.098230	14.525746	76.485911	4.011718	
14	7.496232	205.344982	28388.004887	5.072558	NaN	444.645352	13.228311	70.300213	4.777382	
16	7.051786	211.049406	30980.600787	10.094796	NaN	315.141267	20.397022	56.651604	4.268429	
...	...	...	...	...	...	...	...	...	...	
3266	8.372910	169.087052	14622.745494	7.547984	NaN	464.525552	11.083027	38.435151	4.906358	
3272	7.808856	193.553212	17329.802160	8.061362	NaN	392.449580	19.903225	NaN	2.798243	
3273	9.419510	175.762646	33155.578218	7.350233	NaN	432.044783	11.039070	69.845400	3.298875	
3274	5.126763	230.603758	11983.869376	6.303357	NaN	402.883113	11.168946	77.488213	4.708658	
3275	7.874671	195.102299	17404.177061	7.509306	NaN	327.459760	16.140368	78.698446	2.309149	

781 rows × 10 columns

In [12]:
```
df[df['ph'].isnull()]
```

Out[12]:

ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135	0
NaN	118.988579	14285.583854	7.804174	268.646941	389.375566	12.706049	53.928846	3.595017	0
NaN	150.174923	27331.361962	6.838223	299.415781	379.761835	19.370807	76.509996	4.413974	0
NaN	227.435048	22305.567414	10.333918	NaN	554.820086	16.331693	45.382815	4.133423	0
NaN	215.977859	17107.224226	5.607060	326.943978	436.256194	14.189062	59.855476	5.459251	0
...	...	...	...	...	...	...	...	...	...
NaN	198.218700	31081.735264	7.419106	NaN	517.925946	11.711419	85.428785	3.345543	1
NaN	203.204659	10643.186771	6.828936	NaN	384.597711	16.011328	72.911573	3.065910	1
NaN	225.754109	28194.452646	5.892830	366.201583	418.272901	17.306832	103.912548	3.855895	1
NaN	188.536608	24711.414927	7.129520	NaN	555.548534	16.959269	56.038702	4.331691	1
NaN	134.736856	9000.025591	9.026293	NaN	428.213987	8.668672	74.773392	3.699558	1

's × 10 columns

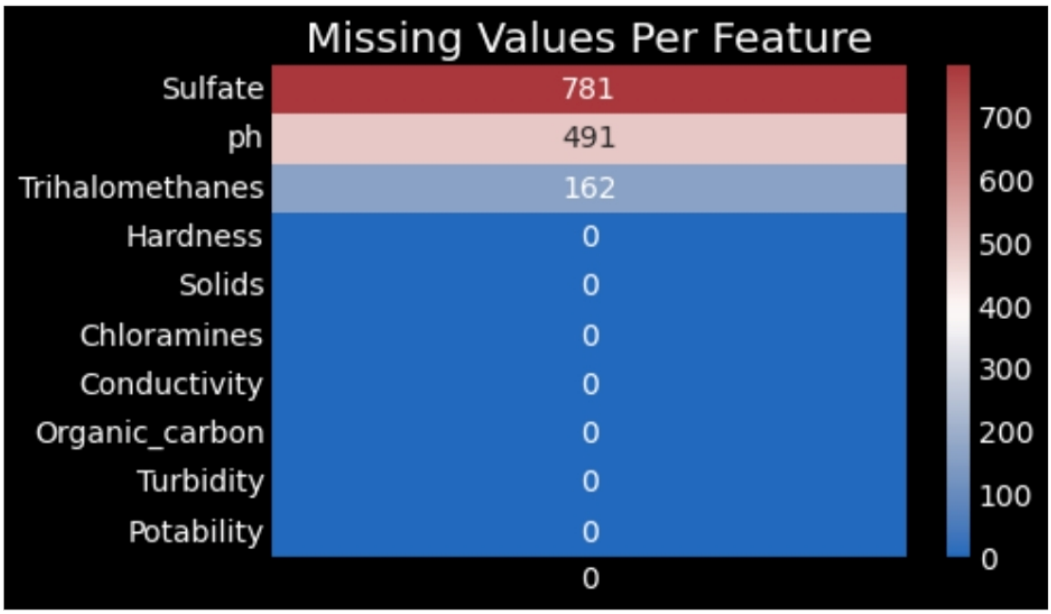
In [13]:
```
df[df['Trihalomethanes'].isnull()]
```

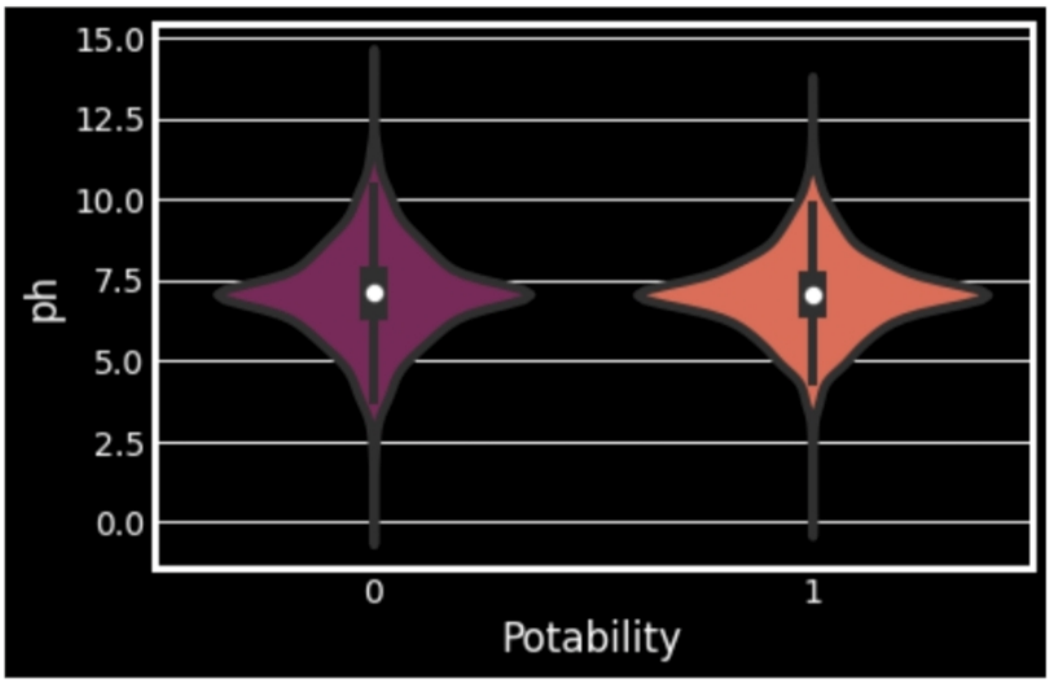
Out[13]:

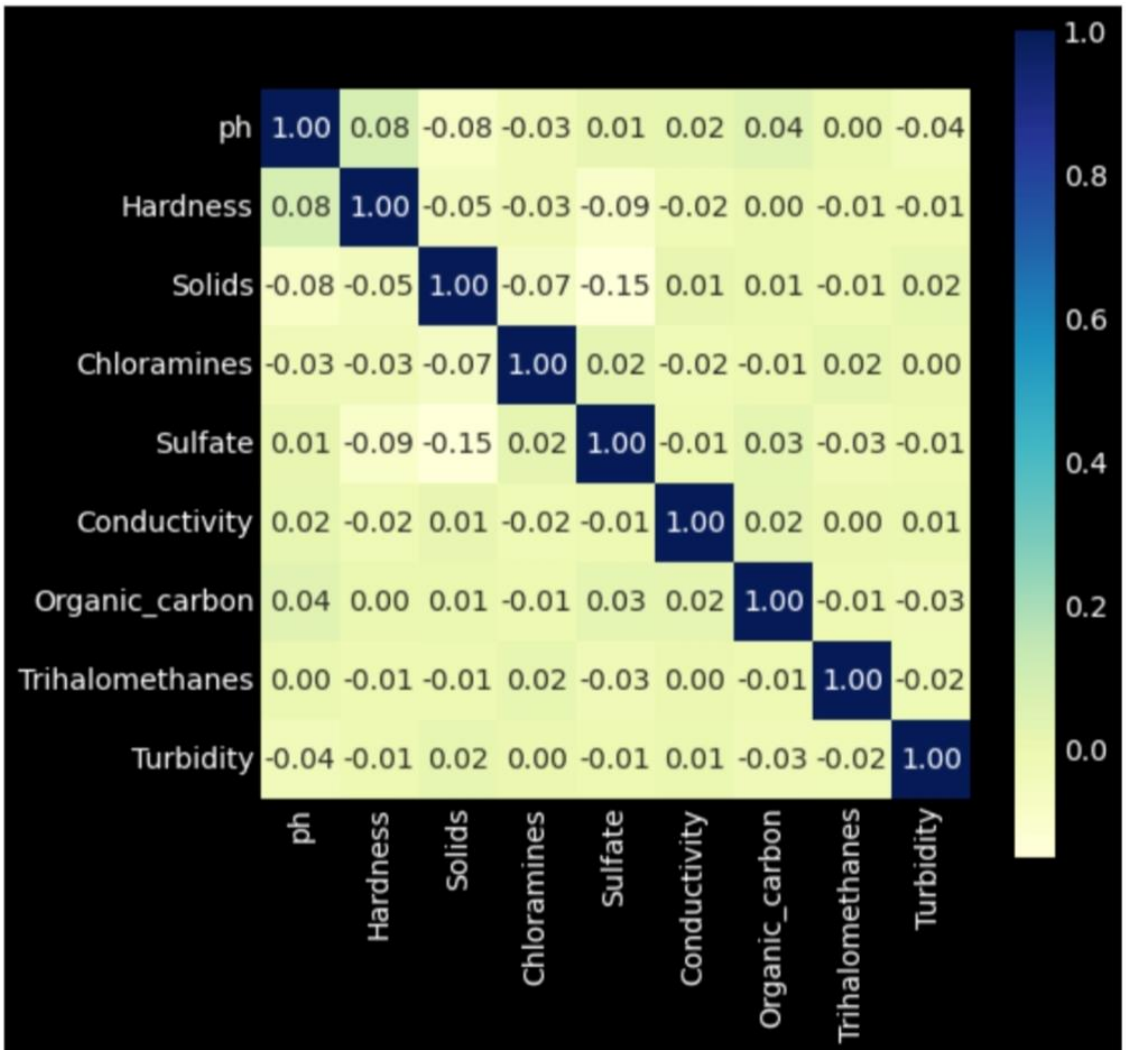
	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidit
62	NaN	229.485694	35729.692709	8.810843	384.943779	296.397547	16.927092	NaN	3.85560
81	5.519126	168.728583	12531.601921	7.730723	NaN	443.570372	18.099078	NaN	3.75899
110	9.286155	222.661551	12311.268366	7.289866	332.239359	353.740100	14.171763	NaN	5.23998
118	7.397413	122.541040	8855.114121	6.888689	241.607532	489.851600	13.365906	NaN	3.14915
119	7.812804	196.583886	42550.841816	7.334648	NaN	442.545775	14.666917	NaN	6.20484
...	...	...	...	...	...	...	...	...	
3174	6.698154	198.286268	34675.862845	6.263602	360.232834	430.935009	12.176678	NaN	3.75818
3185	6.110022	234.800957	16663.539074	5.984536	348.055211	437.892115	10.059523	NaN	2.81778
3219	6.417716	209.702425	31974.481631	7.263425	321.382124	289.450118	11.369071	NaN	4.21032
3259	9.271355	181.259617	16540.979048	7.022499	309.238865	487.692788	13.228441	NaN	4.33395
3272	7.808856	193.553212	17329.802160	8.061362	NaN	392.449580	19.903225	NaN	2.79824

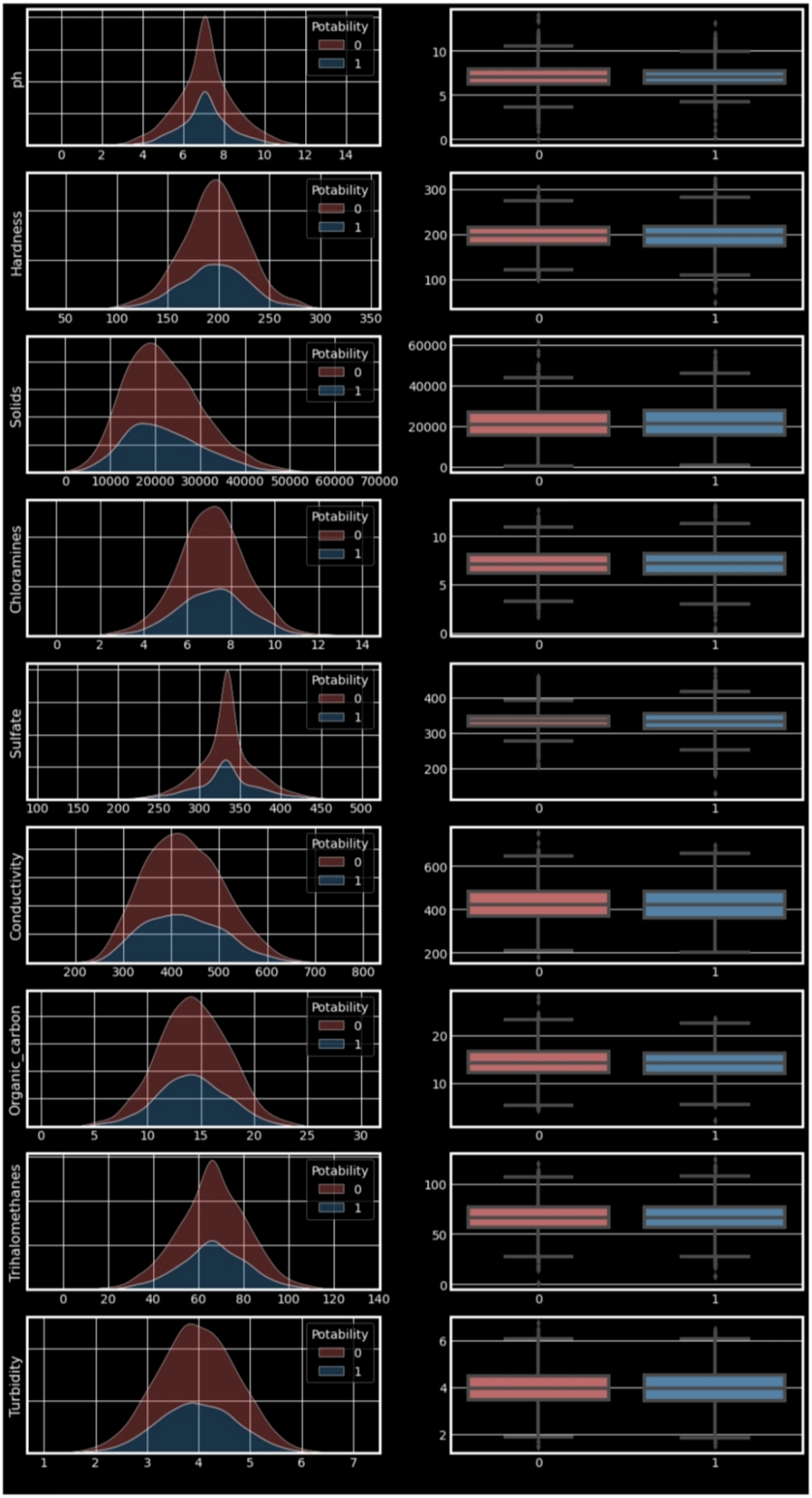
162 rows × 10 columns

Since the missing values are on both classess (Potability 1 & 0), we can replace it with population mean. so, we will replace the Nan values bases on sample mean from both classes.

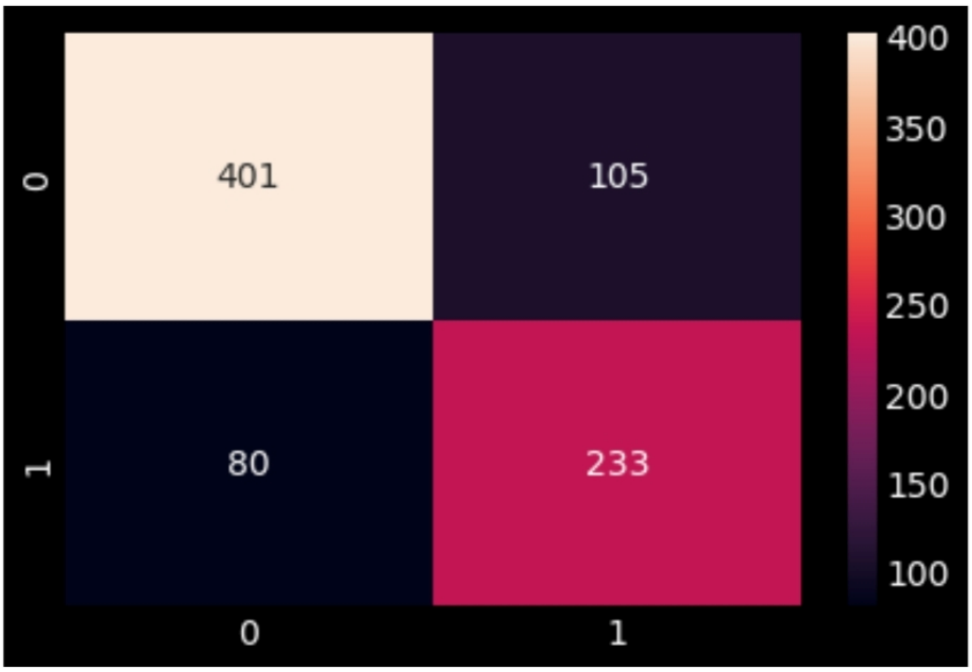

Missing Values Per Feature
Sulfate
ph
Trihalomethanes
Hardness
Solids
Chloramines
Conductivity
Organic_carbon
Turbidity
Potability
781
491
162
0
0
0
0
0
0
0
700
600
500
400
300
200
100
0


15.0
12.5
10.0
7.5
5.0
2.5
0.0
ph
0
1
Potability

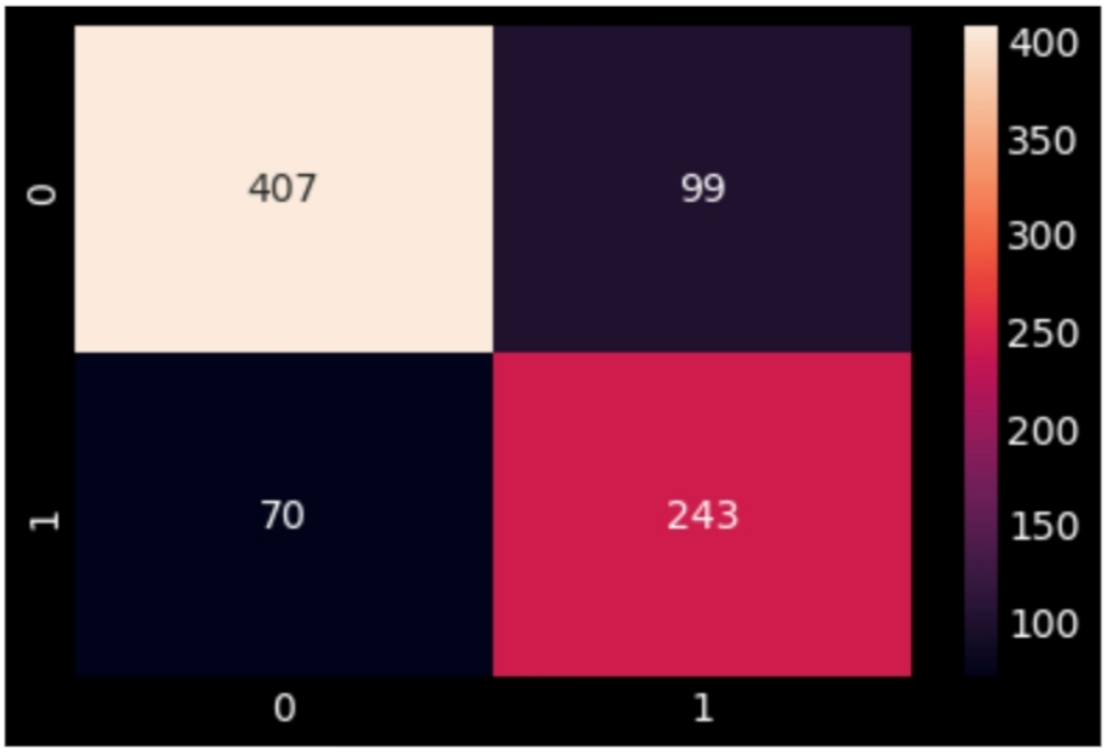

ph 1.00 0.08 -0.08 -0.03 0.01 0.02 0.04 0.00 -0.04
Hardness 0.08 1.00 -0.05 -0.03 -0.09 -0.02 0.00 -0.01 -0.01
Solids -0.08 -0.05 1.00 -0.07 -0.15 0.01 0.01 -0.01 0.02
Chloramines -0.03 -0.03 -0.07 1.00 0.02 -0.02 -0.01 0.02 0.00
Sulfate 0.01 -0.09 -0.15 0.02 1.00 -0.01 0.03 -0.03 -0.01
Conductivity 0.02 -0.02 0.01 -0.02 -0.01 1.00 0.02 0.00 0.01
Organic_carbon 0.04 0.00 0.01 -0.01 0.03 0.02 1.00 -0.01 -0.03
Trihalomethanes 0.00 -0.01 -0.01 0.02 -0.03 0.00 -0.01 1.00 -0.02
Turbidity -0.04 -0.01 0.02 0.00 -0.01 0.01 -0.03 -0.02 1.00
ph
Hardness
Solids
Chloramines
Sulfate
Conductivity
Organic_carbon
Trihalomethanes
Turbidity
1.0
0.8
0.6
0.4
0.2
0.0


ph
Hardness
Solids
Chloramines
Sulfate
Conductivity
Organic_carbon
Trihalomethanes
Turbidity
Potability
0
1

None


401
105
80
233
0
1
0
1
400
350
300
250
200
150
100

None


407
99
70
243
0
1
0
1
400
350
300
250
200
150
100

```
y_xgb = xgb.predict(X_test)
print(metrics.classification_report(y_test, y_xgb))
print(modelAccuracy.append(metrics.accuracy_score(y_test, y_xgb)))

sns.heatmap(confusion_matrix(y_test, y_xgb), annot=True, fmt='d')
plt.show()
```

```
The use of label encoder in XGBClassifier is deprecated and will be
removed in a future release. To remove this warning, do the
following: 1) Pass option use_label_encoder=False when constructing
XGBClassifier object; and 2) Encode your labels (y) as integers
starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].

              precision    recall  f1-score   support

           0       0.85      0.80      0.83       506
           1       0.71      0.78      0.74       313

    accuracy                           0.79       819
   macro avg       0.78      0.79      0.79       819
weighted avg       0.80      0.79      0.80       819

None
```

Conclusion

The Solid levels seem to contain some descripency since its values are on an average 40 folds more than the upper limit for safe drinking water.(Desirable limit for TDS is 500 mg/l and maximum limit is 1000 mg/l which prescribed for drinking purpose.)

The data contains almost equal number of acidic and basic pH level water samples.

The correlation coefficients between the features were very low.

Random Forest and XGBoost worked the best to train the model, both gives us f1 score (Balanced with precision & recall) as around 76%.