CAPSTONE PROJECT

Machine Learning – Python

Project Title :

Water Potability Classification using Python.

Abstract :

Our main objective of the this idea is to predict the given sample of water is Potable or not, in lamen terms whether the water is edible or not. The available dataset is divided into two parts i.e. the training and the test dataset. Models were developed based on the training dataset and applied to the test dataset to find out the accuracy of each model built based on the values predicted.

Submitted By:

TARUN M V

Table of Content:

Table for graphs:

1. Introduction on project Water potability classification

Drinking Not-Potable water is very dangerous. To prevent this we have been working on project to classify whether the water sample is potable or not. Here we are using dataset obtained from many samples as the training and testing data. This model can be further used to know the quality or the potability of water to avoid the risks and take preventive measures.

2. Importance of the project :

What is the need for checking whether water is potable or not-potable?

When you drink non-potable water, **you swallow organisms harmful to your body and expose yourself to a number of water-borne diseases**. Waterborne contaminants could be bacterial, viral, parasitic or chemical.

Non-Potable Water: A Grave Problem:

WHO studies suggest that deaths due to water-related diseases exceed a shocking 5 million annually throughout the world. Those who get medical help and survive often have to go through a prolonged recovery period.

The long-term effects of the illness and strong medication weaken the immune system, making them more prone to other diseases and illnesses.

The most common Waterborne Diseases are **Cholera, Typhoid** and **Dysentery**.

3. Dataset:

```
water.shape
```

```
(3276, 10)
```

The number of observations in data set are 3276.

The number of features in data set are 10, where 9 features are independent features and 1 feature is dichotomous i.e. categorical (0 or 1).

Dichotomous means there are only two possible classes.

1. All 10 independent features are numerical.

```
water.dtypes
```

```
ph                 float64
Hardness           float64
Solids             float64
Chloramines        float64
Sulfate            float64
Conductivity       float64
Organic_carbon     float64
Trihalomethanes    float64
Turbidity          float64
Potability           int64
dtype: object
```

Independent Feature description:

1. **Ph**: potential of hydrogen of water (0 to 14) is a scale used to specify the acidity or basicity of an aqueous solution.
2. **Hardness**: Capacity of water to precipitate soap in mg/L.
3. **Solids**: Total dissolved solids in ppm.
4. **Chloramines**: Amount of Chloramines in ppm.
5. **Sulfate**: Amount of Sulfates dissolved in mg/L.
6. **Conductivity**: Electrical conductivity of water in µS/cm.
7. **Organic carbon**: Amount of organic carbon in ppm.
8. **Trihalomethanes**: Amount of Trihalomethanes in µg/L.
9. **Turbidity**: Measure of light emitting property of water in NTU.

Units:

1. **mg/L** : milligram per litre
2. **ppm** : parts per million
3. **µS/cm** : Micro Siemens per cm
4. **µg/L** : Micrograms per Litre
5. **NTU** : nephelometric turbidity units

2. 1 target feature is dichotomous

Target Feature description:

1. Potability: Indicates if water is safe for human consumption. Potable -1 and Not potable -0

4. Summary of the data

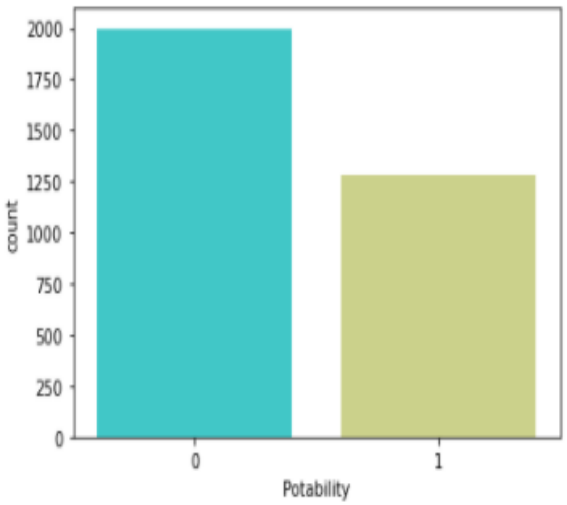
```
water.describe()
```

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
count	2785.000000	3276.000000	3276.000000	3276.000000	2495.000000	3276.000000	3276.000000	3114.000000	3276.000000	3276.000000
mean	7.080795	196.369496	22014.092526	7.122277	333.775777	426.205111	14.284970	66.396293	3.966786	0.390110
std	1.594320	32.879761	8768.570828	1.583085	41.416840	80.824064	3.308162	16.175008	0.780382	0.487849
min	0.000000	47.432000	320.942611	0.352000	129.000000	181.483754	2.200000	0.738000	1.450000	0.000000
25%	6.093092	176.850538	15666.690300	6.127421	307.699498	365.734414	12.065801	55.844536	3.439711	0.000000
50%	7.036752	196.967627	20927.833605	7.130299	333.073546	421.884968	14.218338	66.622485	3.955028	0.000000
75%	8.062066	216.667456	27332.762125	8.114887	359.950170	481.792305	16.557652	77.337473	4.500320	1.000000
max	14.000000	323.124000	61227.196010	13.127000	481.030642	753.342620	28.300000	124.000000	6.739000	1.000000

- As we can see the max count is 3276 which is the number of observation in the data set.
- We can also see some of the feature count is less than 3276 which means it has some missing data which will treated in future steps.
- We get to know the mean, standard deviation and quartile values using this command describe().
- We also get to know the minimum and miximum values in the particular features.

5. Exploratory Data Analysis

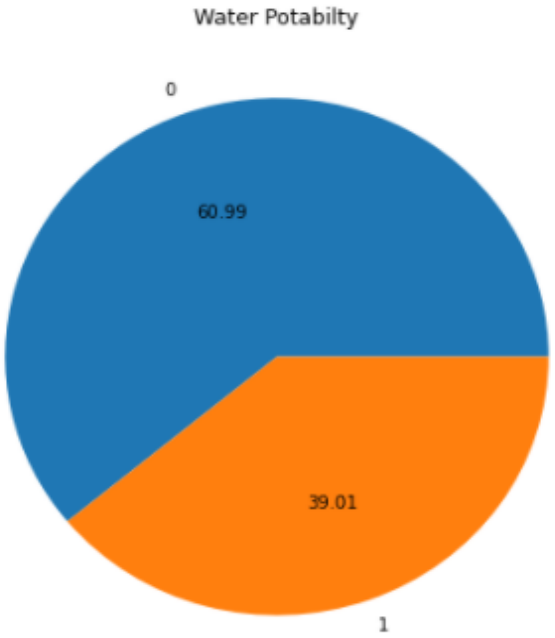
A. Count plot to understand target varible:



Potable -1 and Not potable -0

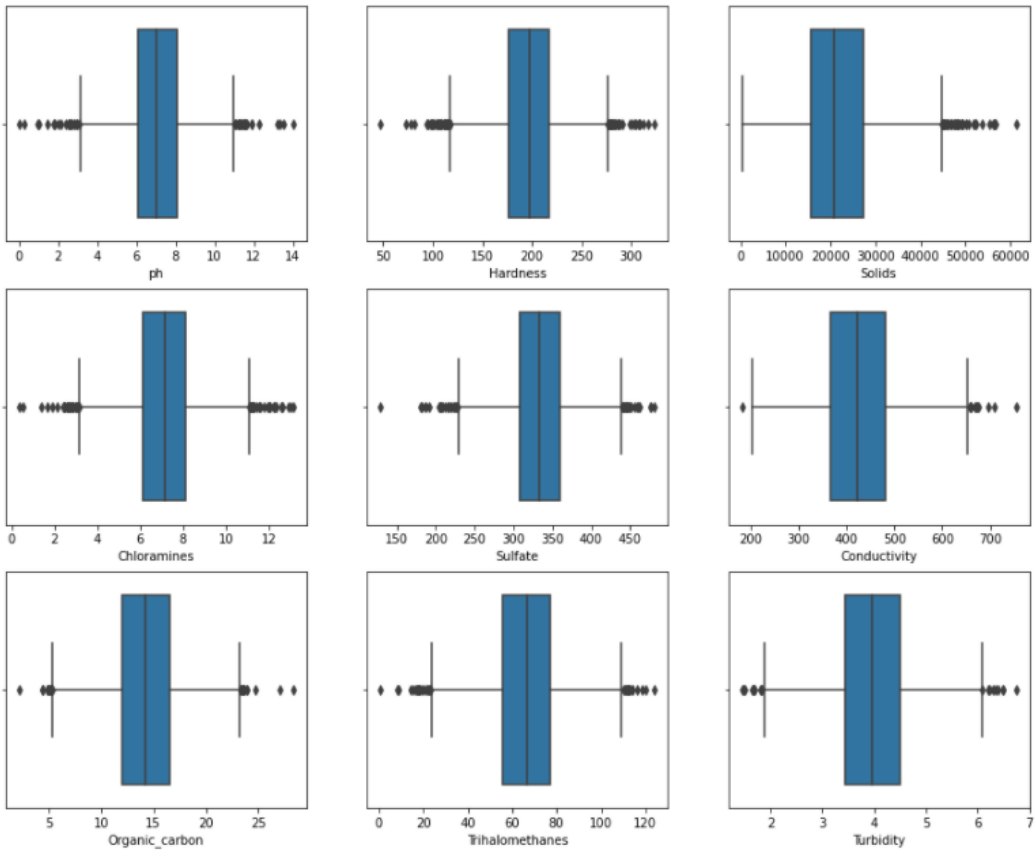
In count plot we can see Potability 0 and 1. Where 0 has more count than 1.

B. Pie plot to understand target variable:



In Pie plot we can see Potability 0 and 1. Where 0 has more area covered than 1. The outcome of both plot says 0 has more values but here in this plot it also says the percentage of 0's and 1's holding in the entire dataseta

C. Box Plot to find outliers:



Most used plot to find the outliers is Box Plot.

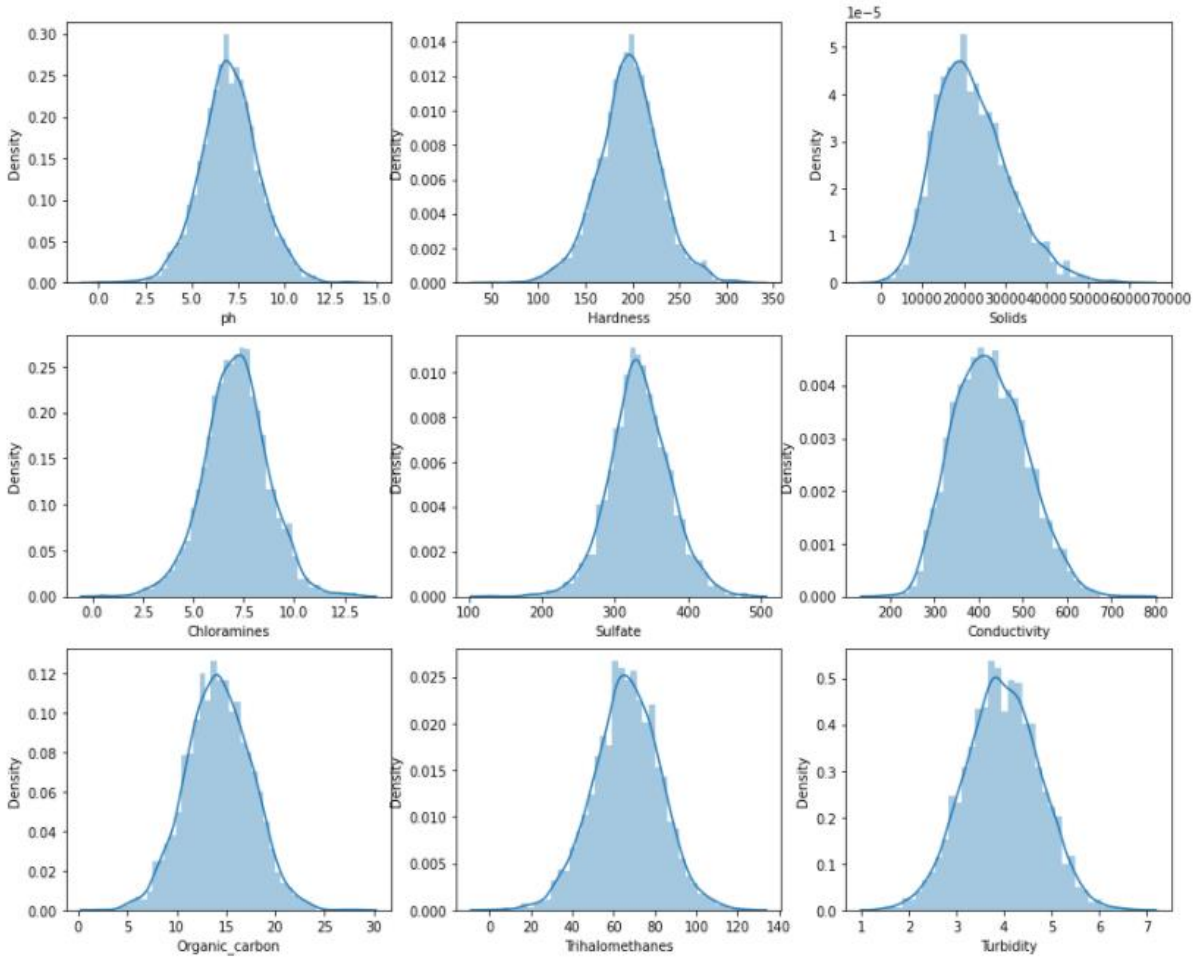
An **outlier** is an observation that lies in abnormal distance from other values in a random sample from a population.

As we can see above the black dots before/after the plot are the values which are considered as outliers

How to treat Outliers?

- Finding out the outliers using the plot
- Post analysis remove the ouliers if the count is negligible
- In this case since the count is not negligable we have replaced the values with the median of its particular feature.
- The values to be removed or replaced can be known by this formula
  Lower bound = (Q1–1.5*IQR)
  Upper bound = (Q3+1.5*IQR)

D. Distplot:



distplot() is used to visualize the parametric distribution of a dataset.

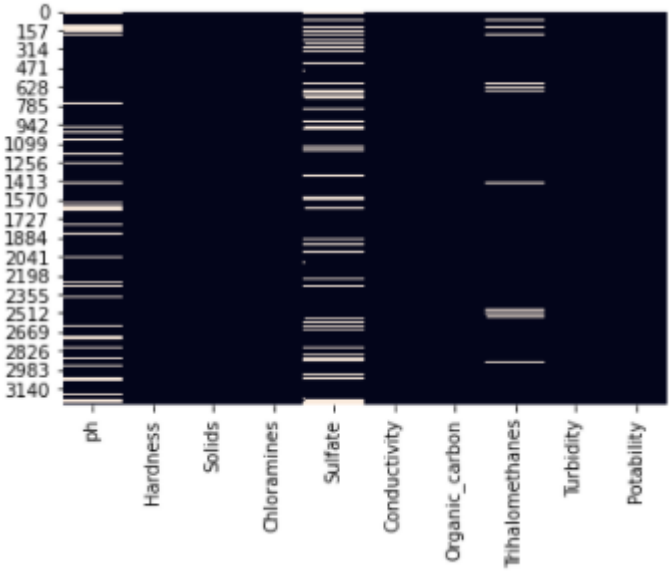
From the above plot we can say that all features are normally distributed

E. Heat map to check null values

```
water.isnull().sum()
```

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

The above command is used to check the number of null values present in the particular features in the dataset.

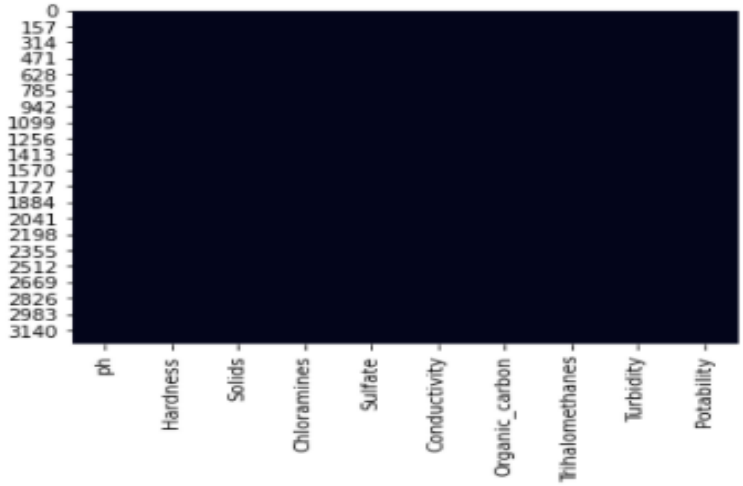


From the above heat map we can see the null values present in the features (ph, Sulfate and Trihalomethanes)

How to treat null values in the data?

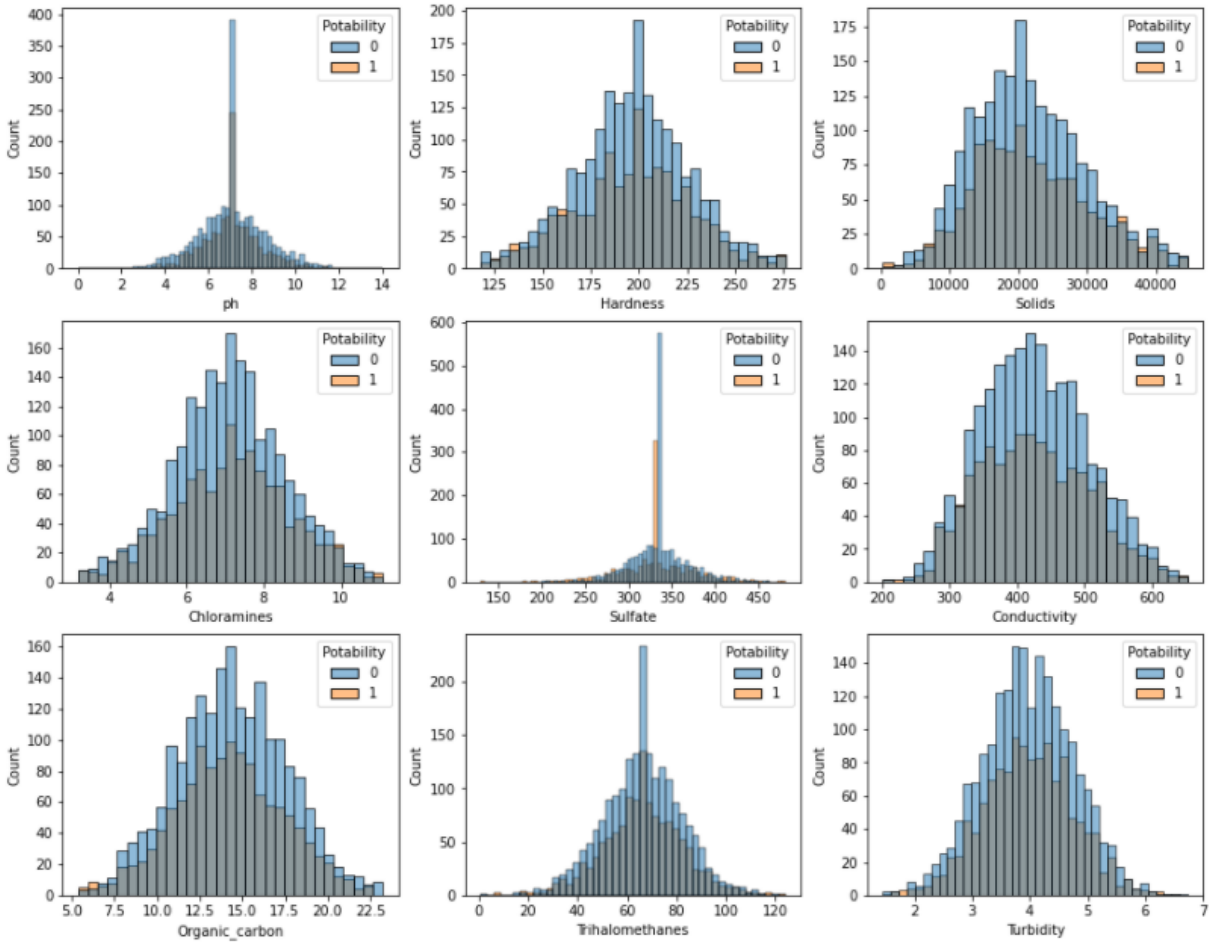
- Find th out the null values using code as well as using visualization
- Once the features are found with null values. Then we use imputation.
- **Imputation** is the process of replacing missing data with substituted values.
- Here in this case we are going to replace the null values of the above 3 features with the mean of the those particular features.
- Once this is completed there will be no missing/null values in the data.

F. Heap map after treating missing/null values:



G. Histplot

Visualizing the features with target variable (Potability) as hue.



As we saw in count plot and pie plot not only Potability feature, in hisplot the data in other features also have more count for Potability-0.

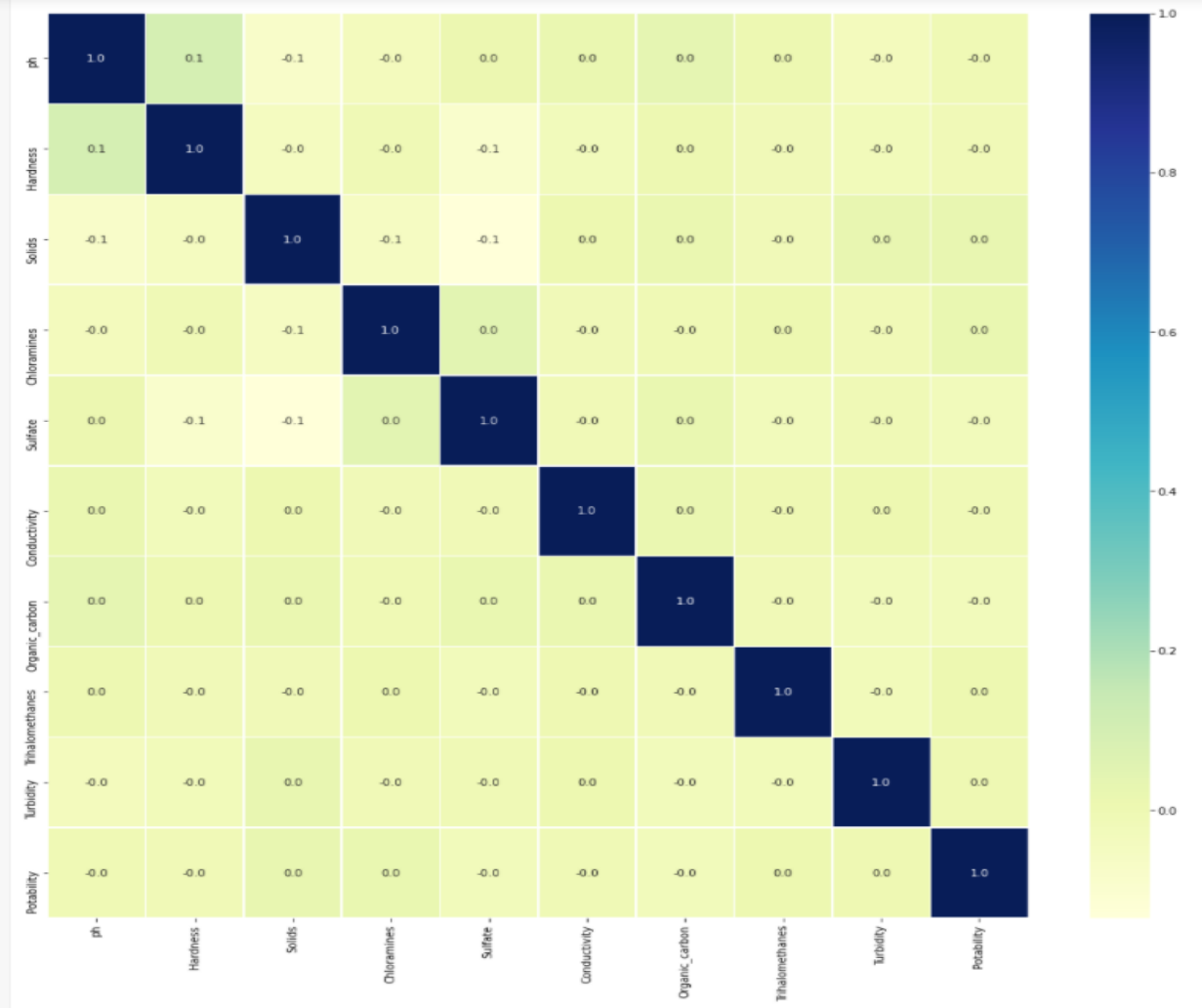
Multivariate analysis:

Correlation matrix is a square matrix giving the covariance between each pair of elements of a given random vector.

Correlation matrix

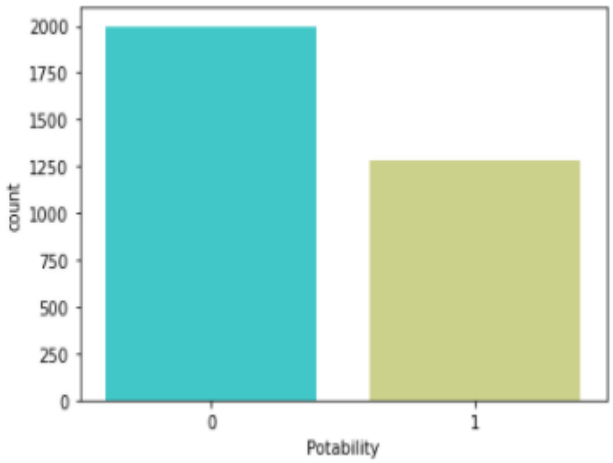
	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
ph	1.000000	0.097792	-0.076046	-0.036547	0.014785	0.017619	0.040496	0.003009	-0.036211	-0.003848
Hardness	0.097792	1.000000	-0.044991	-0.008571	-0.090677	-0.028899	0.001916	-0.016446	-0.021104	-0.012704
Solids	-0.076046	-0.044991	1.000000	-0.055797	-0.135929	0.006705	0.016834	-0.017745	0.026166	0.025051
Chloramines	-0.036547	-0.008571	-0.055797	1.000000	0.046348	-0.018787	-0.006775	0.010410	-0.013485	0.021052
Sulfate	0.014785	-0.090677	-0.135929	0.046348	1.000000	-0.014027	0.023661	-0.025797	-0.009523	-0.026957
Conductivity	0.017619	-0.028899	0.006705	-0.018787	-0.014027	1.000000	0.015019	-0.000583	0.007564	-0.008881
Organic_carbon	0.040496	0.001916	0.016834	-0.006775	0.023661	0.015019	1.000000	-0.013525	-0.025371	-0.027250
Trihalomethanes	0.003009	-0.016446	-0.017745	0.010410	-0.025797	-0.000583	-0.013525	1.000000	-0.021540	0.007305
Turbidity	-0.036211	-0.021104	0.026166	-0.013485	-0.009523	0.007564	-0.025371	-0.021540	1.000000	0.001581
Potability	-0.003848	-0.012704	0.025051	0.021052	-0.026957	-0.008881	-0.027250	0.007305	0.001581	1.000000

Heat Map for the correlation matrix:



From the above map we can say there is no correlation between features.

Balancing data

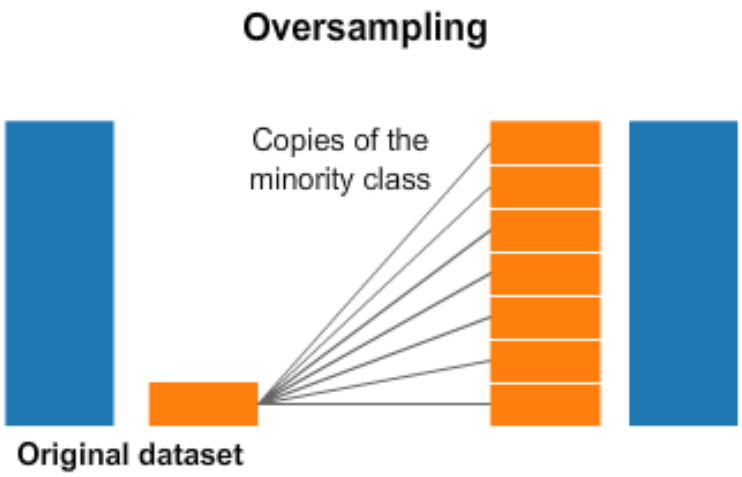



```
water['Potability'].value_counts()
```

```
0    1998
1    1278
Name: Potability, dtype: int64
```

As we can see the values of feature Potability (0 & 1) are imbalanced. Hence we have to balance the data using SMOTE() method before splitting the data into training dataset and testing dataset.

SMOTE (Synthetic Minority Over-sampling Technique) is an oversampling technique and creates new minority class synthetic samples.



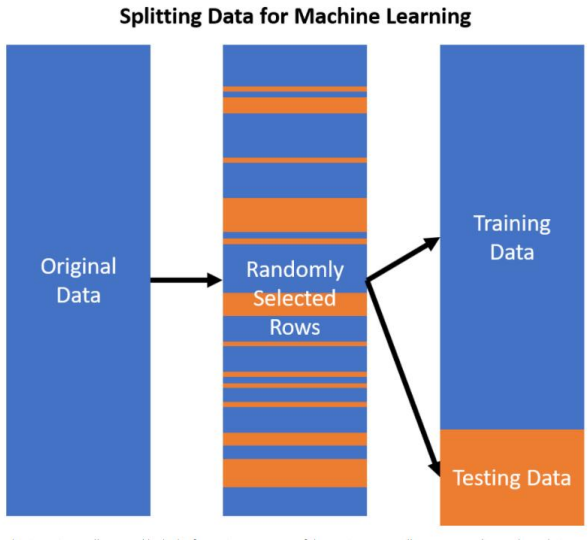
Creating Training and Test dataset

Creating a Training dataset with 80% of the original dataset and testing dataset with remaining 20% of original dataset.

```
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,random_state=1)
```

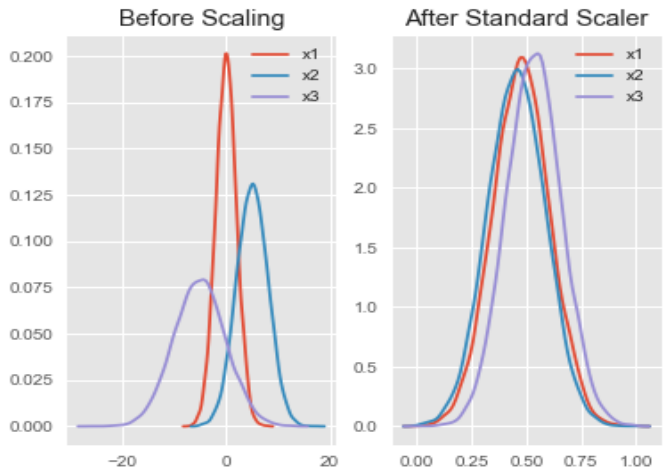
The size of training and testing dataset are:

```
x_train (3250, 9)
x_test (656, 9)
y_train (3250,)
y_test (656,)
```



Feature scaling

Feature Scaling is a technique to standardize the independent features present in the data in a fixed range.



Model building:

A. Classification Models

1. Logistic Regression:

   Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. Logistic regression describes and estimates the relationship between one dependent binary variable and independent variables.

2. Naïve Bayes:

   Naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features (see Bayes classifier). They are among the simplest Bayesian network models

3. K-Nearest Neighbour (KNN):

   K-Nearest Neighbours algorithm (k-NN) is a non-parametric supervised learning. KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label

4. Support Vector Machine (SVM):

   Support Vector Machine or SVM is a supervised and linear Machine Learning algorithm most commonly used for solving classification problems and is also referred to as Support Vector Classification.

5. Decision tree:

   The goal of this algorithm is to create a model that predicts the value of a target variable, for which the decision tree uses the tree representation to solve the problem in which the leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

6. Random Forest:

   Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

B. Boosting algorithms

1. Adaptive Boosting (ADA):

AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method. The most common algorithm used with AdaBoost is decision trees with one level that means with Decision trees with only 1 split.

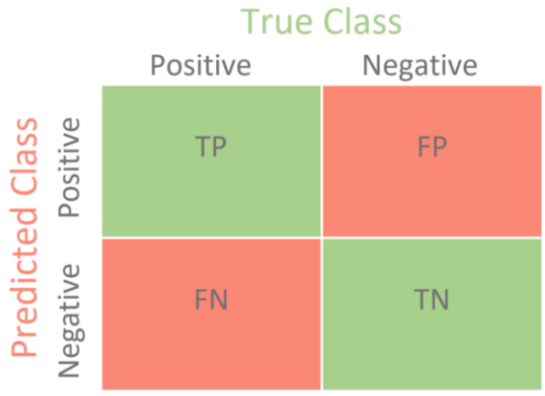
2. Extreme Gradient Boosting Model (XGBM):

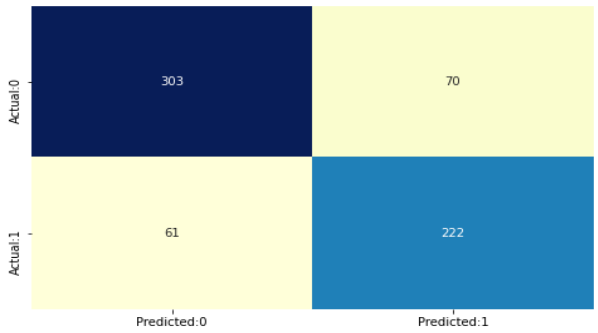
Extreme Gradient Boosting (XGBoost) is an open-source library that provides an efficient and effective implementation of the gradient boosting algorithm.

C. Bagging classifier:

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction.

Confusion matrix:





Real time confusion matrix of Ranfom forest

A confusion matrix is a tabular way of visualizing the performance of your prediction model. Each entry in a confusion matrix denotes the number of predictions made by the model where it classified the classes correctly or incorrectly.

True Positive (TP): No. of predictions where the classifier correctly predicts the positive class as positive.

True Negative (TN): No. of predictions where the classifier correctly predicts the negative class as negative.

False Positive (FP): No. of predictions where the classifier incorrectly predicts the negative class as positive.

False Negative (FN): No. of predictions where the classifier incorrectly predicts the positive class as negative.

Accuracy: It gives you the overall accuracy of the model, means the fraction of the total samples that were correctly classified by the classifier. Formula***: (TP+TN)/(TP+TN+FP+FN)***.

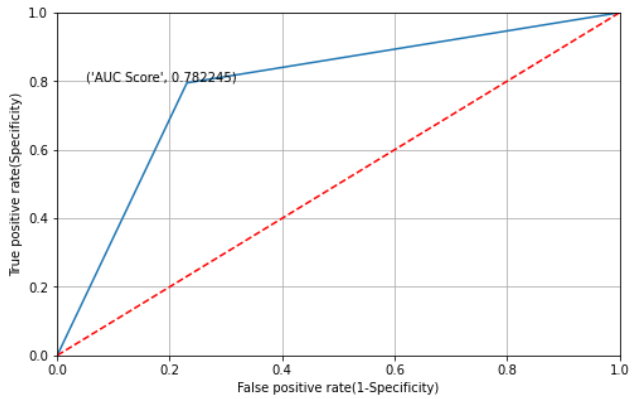
Precision: Fraction of predictions as a positive class were actually positive. Formula: ***TP/(TP+FP)***.

Recall: Fraction of all positive samples were correctly predicted as positive by the classifier. Formula: ***TP/(TP+FN)***

F1-score: It combines precision and recall into a single measure. Mathematically it's the harmonic mean of precision and recall. It can be calculated as follows:

$$F_1\text{−score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$$

AUC or AUROC is Area Under ROC curve:



The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.

The **Receiver Operator Characteristic (ROC)** curve is an evaluation metric for binary classification problems. It is a probability curve that plots the **TPR** against **FPR** at various threshold values and essentially **separates the 'signal' from the 'noise'**. The **Area under the Curve (AUC)** is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

Model Evaluation :

Evaulationg all models on their obtained values.

	Model	AUC Score	Precision Score	Recall Score	Accuracy Score	f1-score
0	logistic_regression	0.502591	0.434084	0.477032	0.506098	0.454545
1	Naive Bayes	0.534763	0.470383	0.477032	0.542683	0.473684
2	KNN	0.574300	0.502994	0.593640	0.571646	0.544571
3	SVM	0.609579	0.545161	0.597173	0.611280	0.569983
4	Decision tree	0.609579	0.545161	0.597173	0.611280	0.569983
5	Random Forest	0.804181	0.768966	0.787986	0.806402	0.778360
6	ADA BOOST	0.737583	0.654971	0.791519	0.730183	0.716800
7	XGBM	0.770489	0.736842	0.742049	0.774390	0.739437

Conclusion:

- Comparing all the models the best fit is **RANDOM FOREST** as it has consistent in precision, recall and accuracy.
- Even when comes to specificity and sensitivity random forest has the good results
    - Specificity: the ability of a test to correctly identify samples of potable water. – [TN/(FP+TN)] => [222/(222+70)] = **76%**

    - Sensitivity: the ability of a test to correctly identify sample that is not potable. – [TP/(TP+FN)] => [303/(303+61)] = **83%**

	Model	AUC Score	Precision Score	Recall Score	Accuracy Score	f1-score
0	logistic_regression	0.502591	0.434084	0.477032	0.506098	0.454545
1	Naive Bayes	0.534763	0.470383	0.477032	0.542683	0.473684
2	KNN	0.574300	0.502994	0.593640	0.571646	0.544571
3	SVM	0.609579	0.545161	0.597173	0.611280	0.569983
4	Decision tree	0.609579	0.545161	0.597173	0.611280	0.569983
5	Random Forest	0.804181	0.768966	0.787986	0.806402	0.778360
6	ADA BOOST	0.737583	0.654971	0.791519	0.730183	0.716800
7	XGBM	0.770489	0.736842	0.742049	0.774390	0.739437

References:

1. https://en.wikipedia.org/ [Theory information]
2. https://www.geeksforgeeks.org/ [Theory & Python packages]
3. https://palmeramia.com/blogs [Water related issues]
4. https://towardsdatascience.com/ [Theory & Understanding]
5. https://seaborn.pydata.org/ [Visualization information]
6. www.i2tutorials.com [Theory & Understanding]
7. https://stackoverflow.com/ [Theory & Understanding]
8. https://images.google.com/ [Concepts related images]