PROJECT REPORT ON

WATER QUALITY PREDICTION

FOR

BIG DATA AND DATA ANALYTICS

Submitted to:

Dr. Arti Jain

Submitted by:

Chandrika Rajvanshi

Enrolment No: 9920103096

Batch: F4

Department of CSE/IT

Jaypee Institute of Information Technology University, Noida

December, 2022

PROBLEM STATEMENT

The major goal of this project is to use machine learning techniques to measure water quality. A potability is a numerical phrase that is used to assess the quality of a body of water. The following water quality parameters were utilised to assess the overall water quality in terms of potability in this study. ph, Hardness, Solids, Chloramines, Sulphate, Conductivity, Organic Carbon, Trihalomethanes, Turbidity were the parameters. To depict the water quality, these parameters are used as a feature vector. To estimate the water quality class, we used one type of classification algorithms: Decision Tree (DT). Experiments were carried out utilising a synthetic dataset generated at random using parameters. According to the findings, machine learning approaches are capable of accurately predicting the potability. Potability, Water Quality Parameters, Data Mining, and Classification are all index terms.

INTRODUCTION

Water quality analysis is a complex topic due to the different factors that influence it. This concept is inextricably linked to the various purposes for which water is used. Different needs necessitate different standards. There is a lot of study being done on water quality prediction. Water quality is normally determined by a set of physical and chemical parameters that are closely related to the water's intended usage. The acceptable and unacceptable values for each variable must then be established. Water that meets the predetermined parameters for a specific application is considered appropriate for that application. If the water does not fulfil these requirements, it must be treated before it may be used. Water quality can be assessed using a variety of physical and chemical properties. As a result, studying the behaviour of each individual variable independently is not possible in practise to accurately describe water quality on a spatial or temporal basis. The more challenging method is to combine the values of a group of physical and chemical variables into a single value. A quality value function (usually linear) represented the equivalence between the variable and its quality level was included in the index for each variable. These functions were created using direct measurements of a substance's concentration or the value of a physical variable derived from water sample studies. The major goal of this project is to examine how machine learning algorithms may be used to predict water quality.

DETAILED DESCRIPTION OF THE PROJECT

The proposed system is intended to determine potability. It is divided into two phases, one for training and the other for testing. The following procedures are carried out in both sections. Data on training pH and hardness testing data Solids, chloramines, sulphate, conductivity, organic carbon, trihalomethanes, turbidity, and potability are all terms that can be used to describe something. The data set was chosen as follows: The collection of essential parameters that affect water quality, identification of the number of data samples, and definition of the class labels for each data sample present in the data are all factors that go into selecting the water quality data set, which is a prerequisite to model construction. Ten indicator parameters make up the data sets used in this study. pH value and hardness are examples of these factors. Solids, chloramines, sulfate, conductivity, organic carbon, trihalomethanes, turbidity, and potability are all terms that can be used to describe the properties of a substance. The proposed approach, however, is not constrained by the number of parameters or the selection of parameters. A k-fold cross validation technique is employed to set the learning and testing framework in this study, corresponding to each data sample in the data set. The dataset is separated into k-disjointed sets of equal size, each with roughly the same class distribution, using this technique. This division's subsets are utilised as the test set in turn, with the remaining subsets serving as the training set. In terms of the underlying relational structure between the indicator parameters and the class label, each of these strategies takes a different approach. As a result, each technique's performance for the same data set is likely to differ. Validating the performance of different classifiers on an unknown data set: Data mining provides several metrics for validating the performance of different classifiers on an unknown data set. The following procedure was used to apply the classification algorithm:

1. The data set was split into two parts: training (80%) and testing (20%).
2. The training set was subjected to repeated cross-validation, with the number of iterations fix ed to Classifiers were trained in this manner.
3. The model's optimal parameter configuration was selected, resulting in the maximum accuracy.
4. The model was scrutinized.

IMPLEMENTATION

Classification To estimate river water quality class, data mining method used was: Decision Tree(DT). This method is both parametric and nonparametric classifiers, and its goal is to develop a function that maps input variables to output variables from a training dataset. Because the function's form is unknown, different algorithms make different assumptions about the function's form and how training data is learned to produce the output. The parametric learning classifier makes more confident assumptions about the data. If the assumptions for any data set are true, these classifiers will make rectification judgments. However, if the assumptions are incorrect, the same classifier performs poorly. In order to learn classification tasks, this classifier does not rely on the quantity of the sample data set; rather, its working principles are its assumptions. This classifier is susceptible to prediction mistakes such as bias, in addition to its parametric character.

When the model makes multiple assumptions, the Decision Tree yields substantial bias. Nonparametric classifiers, unlike parametric learning classifiers, do not make any assumptions about the form of the mapping function, and by not making any assumptions, they are having more accuracy. These classifiers can create any function from the training data set. The DT classifier is included in this category. Learning techniques are used in DT, whereas the similarity principle is used in KNN. To put it another way, DT Small data sets with complete domain expertise, on the other hand, are equally advantageous for these classifiers. Unlike other classifiers, DT does not rely on domain expertise. To make classification decisions, it simply calculates the distance between two characteristics. Because each algorithm's mode of operation differs, a comparison of all of them is necessary to determine which one is better at approximating the underlying function for the same training and testing water quality datasets.

MODELING AND ANALYSIS

Data mining techniques require domain knowledge in order to generate predictions. For water quality applications, it is vital to understand how various water quality parameters influence water quality. This information can come from a domain expert or historical data collections. For the forecasting task, one type of data sets was used: a carefully created huge synthetic data set. The fact that data set is examined on an equal number of indicator parameters is the key similarity between them. The amount of samples considered in data set differs amongst the data sets. The developed synthetic data set, captures identical relational structures and water quality parameters have the same distribution as in the real-world scenario. Ten water quality parameters were utilised to evaluate the overall water quality in terms of potability for each data set. These variables are pH and Hardness. Solids, chloramines, sulphate, conductivity, organic carbon, trihalomethanes, turbidity, and potability are all terms that can be used to describe something. The choice of parameters was influenced by the fact that they are all commonly monitored critical parameters with well-defined water quality standards. The predictive modeling described in this paper, on the other hand, is flexible enough to function with any number of parameters.

A target data set is necessary for the use of data mining methods. If data mining is to be used to find patterns in data, the data collection should be large enough to contain these patterns as a general rule. A synthetic data collection was created to provide a realistic technique to obtaining this enormous data set. This synthetic data set was carefully produced by taking into account possible water quality parameter ranges. The benefit of using these concentration ranges was that they were developed after careful consideration of water quality standards assigned by various national and international organization's such as the European Union (EU), the World Health Organization (WHO), the Central Pollution Control Board (CPCB), and others Scientific data was reported. Each sample reflected the occurrence of one instance of the 10 parameter concentration values under investigation. The data set that will be utilised to develop a predictive model using the classification technique must be supervised. The following step was to establish a supervised environment for the numerical data set, which was generated by assigning a label to each instance in order to forecast the water contamination level. To do this, the potability was determined for each instance of concentration values for the 10 parameters chosen.

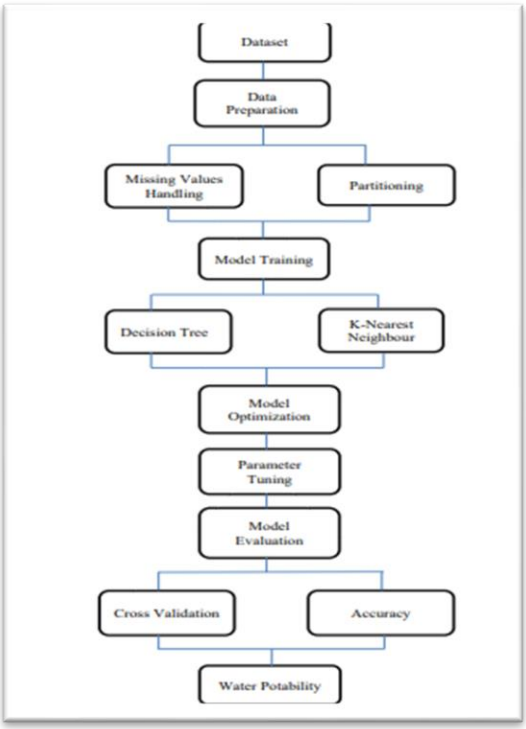
	A	B	C	D	E	F	G	H	I	J
1	ph	Hardness	Solids	Chloramin	Sulfate	Conductiv	Organic_c	Trihalome	Turbidity	Potability
2		204.8905	20791.32	7.300212	368.5164	564.3087	10.37978	86.99097	2.963135	0
3	3.71608	129.4229	18630.06	6.635246		592.8854	15.18001	56.32908	4.500656	0
4	8.099124	224.2363	19909.54	9.275884		418.6062	16.86864	66.42009	3.055934	0
5	8.316766	214.3734	22018.42	8.059332	356.8861	363.2665	18.43652	100.3417	4.628771	0
6	9.092223	181.1015	17978.99	6.5466	310.1357	398.4108	11.55828	31.99799	4.075075	0
7	5.584087	188.3133	28748.69	7.544869	326.6784	280.4679	8.399735	54.91786	2.559708	0
8	10.22386	248.0717	28749.72	7.513408	393.6634	283.6516	13.7897	84.60356	2.672989	0
9	8.635849	203.3615	13672.09	4.563009	303.3098	474.6076	12.36382	62.79831	4.401425	0
10		118.9886	14285.58	7.804174	268.6469	389.3756	12.70605	53.92885	3.595017	0
11	11.18028	227.2315	25484.51	9.0772	404.0416	563.8855	17.92781	71.9766	4.370562	0
12	7.36064	165.5208	32452.61	7.550701	326.6244	425.3834	15.58681	78.74002	3.662292	0
13	7.974522	218.6933	18767.66	8.110385		364.0982	14.52575	76.48591	4.011718	0
14	7.119824	156.705	18730.81	3.606036	282.3441	347.715	15.92954	79.50078	3.445756	0
15		150.1749	27331.36	6.838223	299.4158	379.7618	19.37081	76.51	4.413974	0
16	7.496232	205.345	28388	5.072558		444.6454	13.22831	70.30021	4.777382	0
17	6.347272	186.7329	41065.23	9.629596	364.4877	516.7433	11.53978	75.07162	4.376348	0
18	7.051786	211.0494	30980.6	10.0948		315.1413	20.39702	56.6516	4.268429	0
19	9.18156	273.8138	24041.33	6.90499	398.3505	477.9746	13.38734	71.45736	4.503661	0
20	8.975464	279.3572	19460.4	6.204321		431.444	12.88876	63.82124	2.436086	0
21	7.37105	214.4966	25630.32	4.432669	335.7544	469.9146	12.50916	62.79728	2.560299	0
22		227.435	22305.57	10.33392		554.8201	16.33169	45.38282	4.133423	0
23	6.660212	168.2837	30944.36	5.858769	310.9309	523.6713	17.88424	77.04232	3.749701	0
24		215.9779	17107.22	5.60706	326.944	436.2562	14.18906	59.85548	5.459251	0
25	3.902476	196.9032	21167.5	6.996312		444.4789	16.60903	90.18168	4.528523	0
26	5.400302	140.7391	17266.59	10.05685	328.3582	472.8741	11.25638	56.93191	4.824786	0

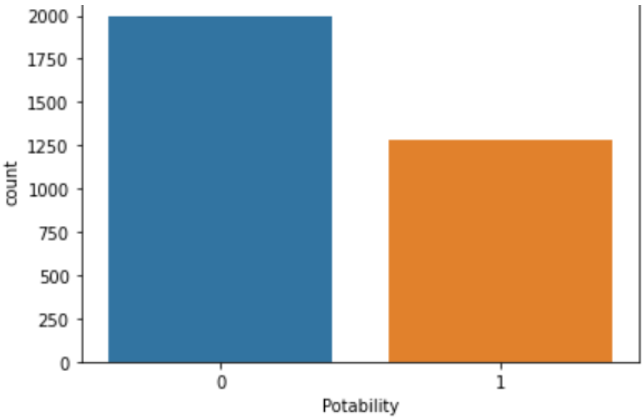
water_potability

Data set created artificially

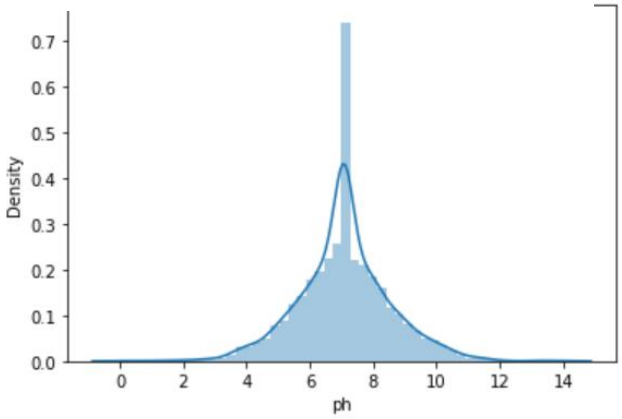
The water_potability.csv file contains water quality metrics for 3276 different water bodies.

WORKFLOW DIAGRAM

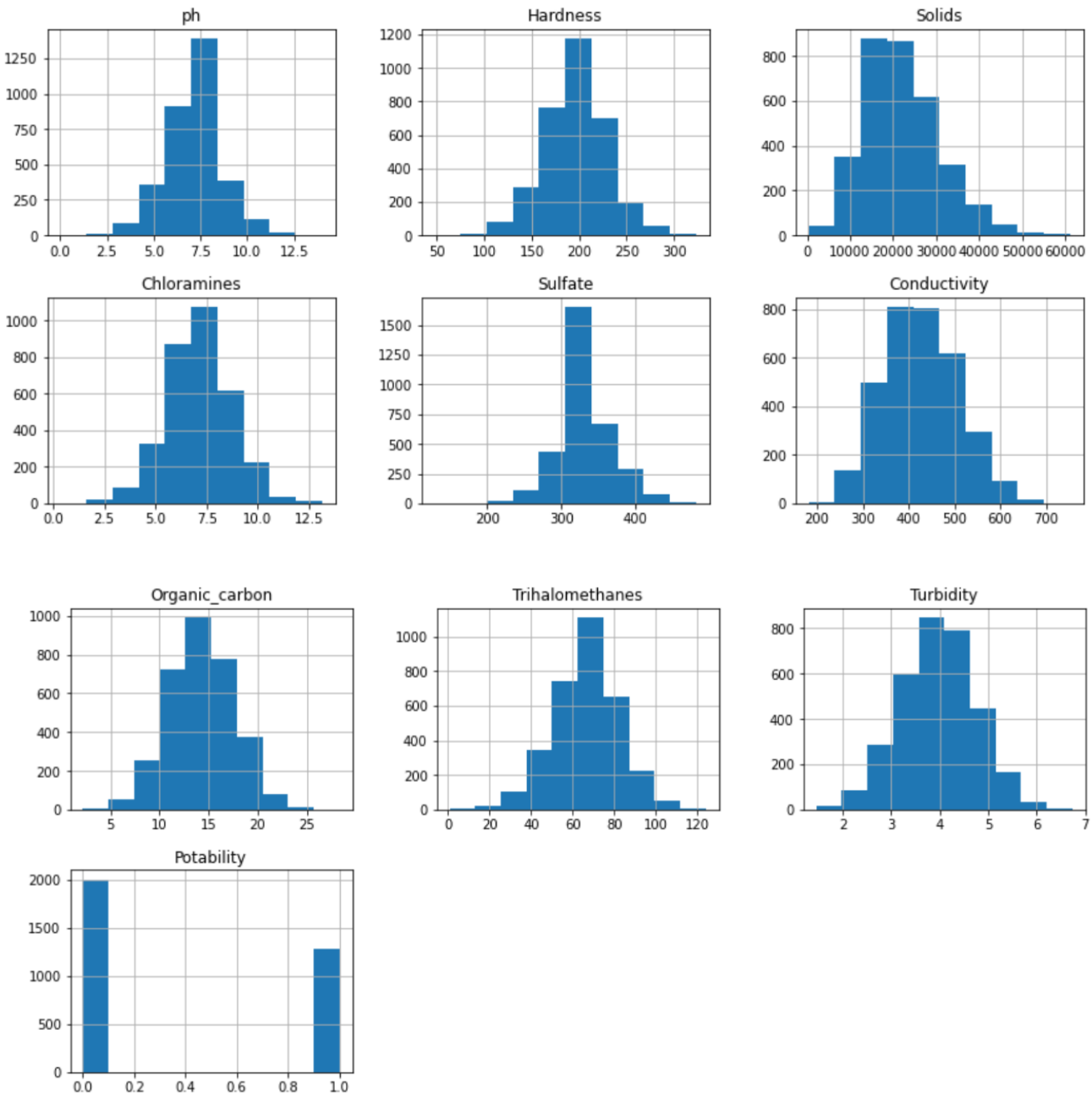




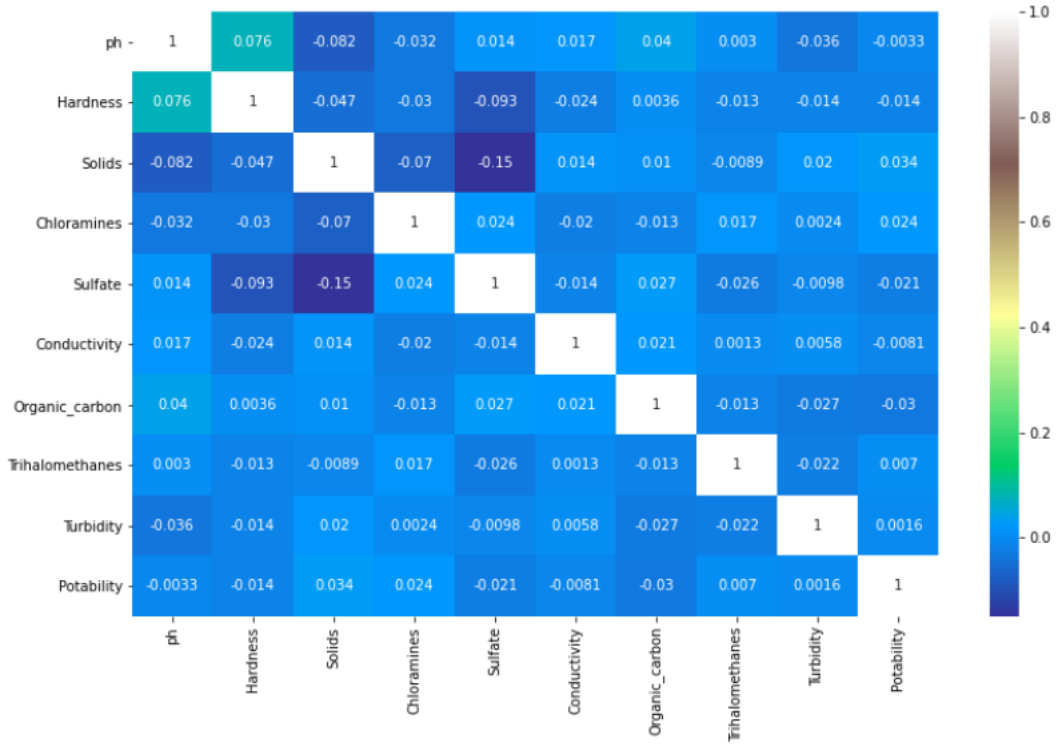
Visualization of the potability using a count plot function of seaborn



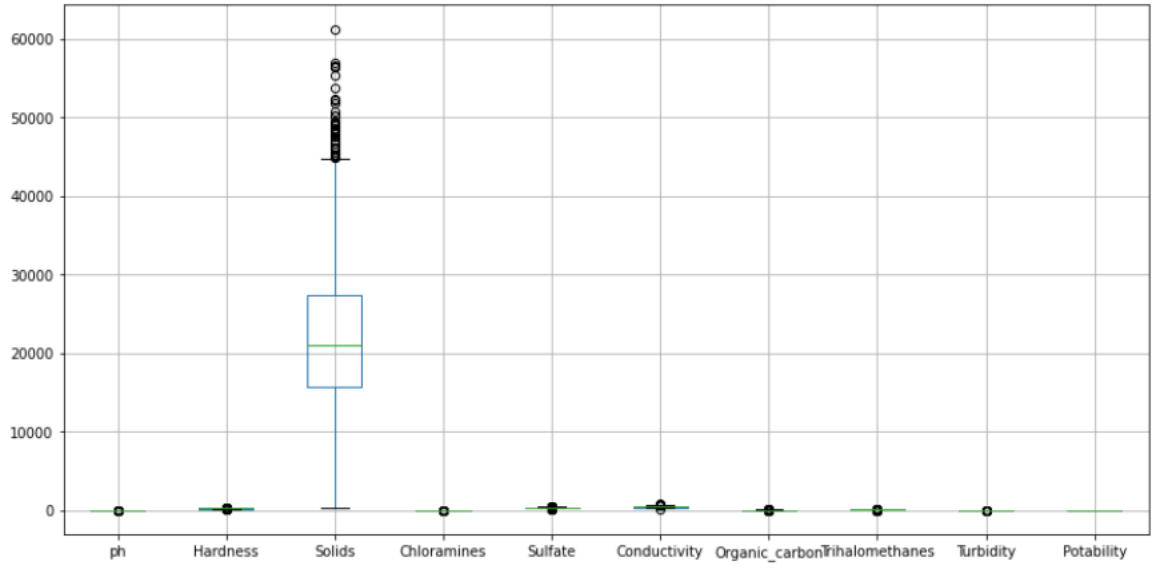
Visualization of the pH value using a distplot function



Visualization of all the features of the data sets



Visualization of the correlation of all the features using a heat map function of seaborn



Seeing the outlier using a boxplot function

PROGRAM CODE

Now it's time to prepare the data set. Divide the data into the independent and dependent features. All are independent features except Potability because Potability is our dependent feature. Split the data set into the training and testing using the train_test_split function which returns four data sets.

```
X = df.drop('Potability',axis=1)
Y= df['Potability']
```

```
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=101,shuffle=True)
```

Now define the decision tree classifier model and train the model using the data set (X_train, Y_train). Then test the model using the test data set (X_test). Now it's time to evaluate the model using the accuracy score, confusion matrix and classification report. Evaluation techniques take two parameters; one is the actual data and the other one is a predicted data. And You can see that overall accuracy is 59%.

Train Decision Tree Classifier and check accuracy

In [17]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
dt=DecisionTreeClassifier(criterion= 'gini', min_samples_split= 10, splitter= 'best')
dt.fit(X_train,Y_train)
```

Out[17]:
```
DecisionTreeClassifier(min_samples_split=10)
```

In [18]:
```
prediction=dt.predict(X_test)
print(f"Accuracy Score = {accuracy_score(Y_test,prediction)*100}")
print(f"Confusion Matrix =\n {confusion_matrix(Y_test,prediction)}")
print(f"Classification Report =\n {classification_report(Y_test,prediction)}")
```

```
Accuracy Score = 58.536585365853654
Confusion Matrix =
 [[271 131]
 [141 113]]
Classification Report =
               precision    recall  f1-score   support

           0       0.66      0.67      0.67       402
           1       0.46      0.44      0.45       254

    accuracy                           0.59       656
   macro avg       0.56      0.56      0.56       656
weighted avg       0.58      0.59      0.58       656
```

Then test the model on a custom data set and you can also see the result in the below image.

In [19]:
```
res = dt.predict([[5.735724, 158.318741,25363.016594,7.728601,377.543291,568.304671,13.626624,75.952337,4.732954]])[0]
res
```

```
C:\Python 3.10\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

Out[19]: 1

Now apply a hyper parameter tuning on the decision tree classifier using a GridSearchCV. Only three parameters to tune the model as you can see in the image below.

Apply Hyper Parameter Tuning

In [20]:
```
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import GridSearchCV

# define models and parameters
model = DecisionTreeClassifier()
criterion = ["gini", "entropy"]
splitter = ["best", "random"]
min_samples_split = [2,4,6,8,10,12,14]

# define grid search
grid = dict(splitter=splitter, criterion=criterion, min_samples_split=min_samples_split)
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
grid_search_dt = GridSearchCV(estimator=model, param_grid=grid, n_jobs=-1, cv=cv,
                              scoring='accuracy',error_score=0)
grid_search_dt.fit(X_train, Y_train)
```

Out[20]:

GridSearchCV

```
GridSearchCV(cv=RepeatedStratifiedKFold(n_repeats=3, n_splits=10, random_state=1),
             error_score=0, estimator=DecisionTreeClassifier(), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'min_samples_split': [2, 4, 6, 8, 10, 12, 14],
                         'splitter': ['best', 'random']},
             scoring='accuracy')
```

estimator: DecisionTreeClassifier

DecisionTreeClassifier

DecisionTreeClassifier()

Now see how much time the model has trained with different parameters. Also check the training and testing accuracy as you can see that the training accuracy is 90% and testing accuracy is 60% which is okay.

```
print(f"Best: {grid_search_dt.best_score_:.3f} using {grid_search_dt.best_params_}")
means = grid_search_dt.cv_results_['mean_test_score']
stds = grid_search_dt.cv_results_['std_test_score']
params = grid_search_dt.cv_results_['params']

for mean, stdev, param in zip(means, stds, params):
    print(f"{mean:.3f} ({stdev:.3f}) with: {param}")

print("Training Score:",grid_search_dt.score(X_train, Y_train)*100)
print("Testing Score:", grid_search_dt.score(X_test, Y_test)*100)
```

```
Best: 0.603 using {'criterion': 'gini', 'min_samples_split': 14, 'splitter': 'random'}
0.580 (0.028) with: {'criterion': 'gini', 'min_samples_split': 2, 'splitter': 'best'}
0.567 (0.027) with: {'criterion': 'gini', 'min_samples_split': 2, 'splitter': 'random'}
0.583 (0.033) with: {'criterion': 'gini', 'min_samples_split': 4, 'splitter': 'best'}
0.573 (0.030) with: {'criterion': 'gini', 'min_samples_split': 4, 'splitter': 'random'}
0.584 (0.031) with: {'criterion': 'gini', 'min_samples_split': 6, 'splitter': 'best'}
0.583 (0.031) with: {'criterion': 'gini', 'min_samples_split': 6, 'splitter': 'random'}
0.584 (0.029) with: {'criterion': 'gini', 'min_samples_split': 8, 'splitter': 'best'}
0.585 (0.038) with: {'criterion': 'gini', 'min_samples_split': 8, 'splitter': 'random'}
0.592 (0.029) with: {'criterion': 'gini', 'min_samples_split': 10, 'splitter': 'best'}
0.584 (0.025) with: {'criterion': 'gini', 'min_samples_split': 10, 'splitter': 'random'}
0.588 (0.026) with: {'criterion': 'gini', 'min_samples_split': 12, 'splitter': 'best'}
0.584 (0.036) with: {'criterion': 'gini', 'min_samples_split': 12, 'splitter': 'random'}
0.588 (0.026) with: {'criterion': 'gini', 'min_samples_split': 14, 'splitter': 'best'}
0.603 (0.022) with: {'criterion': 'gini', 'min_samples_split': 14, 'splitter': 'random'}
0.583 (0.028) with: {'criterion': 'entropy', 'min_samples_split': 2, 'splitter': 'best'}
0.584 (0.028) with: {'criterion': 'entropy', 'min_samples_split': 2, 'splitter': 'random'}
0.584 (0.028) with: {'criterion': 'entropy', 'min_samples_split': 4, 'splitter': 'best'}
0.584 (0.030) with: {'criterion': 'entropy', 'min_samples_split': 4, 'splitter': 'random'}
0.583 (0.029) with: {'criterion': 'entropy', 'min_samples_split': 6, 'splitter': 'best'}
0.587 (0.036) with: {'criterion': 'entropy', 'min_samples_split': 6, 'splitter': 'random'}
0.588 (0.028) with: {'criterion': 'entropy', 'min_samples_split': 8, 'splitter': 'best'}
0.587 (0.027) with: {'criterion': 'entropy', 'min_samples_split': 8, 'splitter': 'random'}
0.586 (0.032) with: {'criterion': 'entropy', 'min_samples_split': 10, 'splitter': 'best'}
0.583 (0.029) with: {'criterion': 'entropy', 'min_samples_split': 10, 'splitter': 'random'}
0.589 (0.029) with: {'criterion': 'entropy', 'min_samples_split': 12, 'splitter': 'best'}
0.591 (0.028) with: {'criterion': 'entropy', 'min_samples_split': 12, 'splitter': 'random'}
0.589 (0.031) with: {'criterion': 'entropy', 'min_samples_split': 14, 'splitter': 'best'}
0.587 (0.030) with: {'criterion': 'entropy', 'min_samples_split': 14, 'splitter': 'random'}
Training Score: 81.56488549618321
Testing Score: 61.58536585365854
```

RESULTS

Performance Measures Results True Positives (TP) are when the model predicts the positive class properly. True Negatives (TN) is one of the components of a confusion matrix designed to demonstrate how classification algorithms work. Positive outcomes that the model predicted incorrectly are known as False Positives (FP). False Negatives (FN) are negative outcomes that the model predicts negative class. Accuracy is the most basic and intuitive performance metric, consisting of the ratio of successfully predicted observations to total observations. Accuracy = TP+TN/(TP+FP+FN+TN)

SN.	Algorithm Type	Accuracy score	Precision	Recall	f1-Score
1	Decision Tree	58.5	0.42	0.38	0.40

CONCLUSION

Potability determines the quality of water, which is one of the most important resources for existence. Traditionally, testing water quality required an expensive and time-consuming lab analysis. This study looked into an alternative machine learning method for predicting water quality using only a few simple water quality criteria. To estimate, a set of representative supervised machine learning algorithms was used. It would detect water of bad quality before it was released for consumption and notify the appropriate authorities It will hopefully reduce the number of individuals who drink low-quality water, lowering the risk of diseases like typhoid and diarrhea. In this case, using a prescriptive analysis based on projected values would result in future capabilities to assist decision and policy makers.

REFERENCES

[1] F. A. Algalil, “Applied Bionics and Biomechanics,” [Online]. Available: https://www.hindawi.com/journals/abb/2020/6659314/. [Accessed Thursday December 2022].

[2] S. D. B. S. K. Rajiv Das Kangabam, “Development of a water quality index (WQI) for the Loktak Lake,” 26 June 2017. [Online]. Available: https://link.springer.com/content/pdf/10.1007/s13201-017-0579-4.pdf. [Accessed Thursday December 2022].

[3] R. B. R. K. Ashwani Kumar Thukral, “Water Quality Indices,” January 2005. [Online]. Available: https://www.researchgate.net/publication/257650627_Water_Quality_Indices. [Accessed Wednesday December 2022].

[4] P. K. Garima Srivastava, “WATER QUALITY INDEX WITH MISSING PARAMETERS,” April 2003. [Online]. Available: https://www.semanticscholar.org/paper/WATER-QUALITY-INDEX-WITH-MISSING-PARAMETERS-Srivastava-Kumar/d9ba99bbef3f8d95930ad97780e1812683d28945. [Accessed Thursday December 2022].

[5] D. Arsenault, “Water Quality Standards Handbook,” [Online]. Available: https://www.epa.gov/wqs-tech/water-quality-standards-handbook. [Accessed Thursday December 2022].