First Name - Sai Prathyusha

Last Name - Devarapalli

NJIT UCID - sd894

Email address – sd894@njit.edu

Language Worked - Python

Water Quality Detection Using Machine Learning

Introduction :

Access to clean drinking water is critical to health, a basic human right, and a component of any health-protection strategy. This is significant as a national, regional, and local health and development concern. It has been demonstrated in some locations that investments in water supply and sanitation can result in a net economic gain since the reductions in adverse health impacts and health care expenditures surpass the price of carrying out the interventions.

Water consumption and hydration are linked to a lower risk of urinary tract infections (UTIs), lower blood pressure, and heart disease. As a result, drinking water is critical for healthy heart health.

Data Dictionary:

1. pH value: PH is an important parameter in evaluating the acid–base balance of water. It is also the indicator of acidic or alkaline condition of water status. WHO has recommended maximum permissible limit of pH from 6.5 to 8.5. The current investigation ranges were 6.52–6.83 which are in the range of WHO standards.
2. Hardness: Hardness is mainly caused by calcium and magnesium salts. These salts are dissolved from geologic deposits through which water travels. The length of time water is in contact with hardness producing material helps determine how much hardness there is in raw water. Hardness was originally defined as the capacity of water to precipitate soap caused by Calcium and Magnesium.
3. Solids (Total dissolved solids - TDS): Water has the ability to dissolve a wide range of inorganic and some organic minerals or salts such as potassium, calcium, sodium, bicarbonates, chlorides, magnesium, sulfates etc. These minerals produced un-wanted taste and diluted color in appearance of water. This is the important parameter for the use of water. The water with high

TDS value indicates that water is highly mineralized. Desirable limit for TDS is 500 mg/l and maximum limit is 1000 mg/l which prescribed for drinking purpose.

4. **Chloramines:** Chlorine and chloramine are the major disinfectants used in public water systems. Chloramines are most commonly formed when ammonia is added to chlorine to treat drinking water. Chlorine levels up to 4 milligrams per liter (mg/L or 4 parts per million (ppm)) are considered safe in drinking water.
5. **Sulfate:** Sulfates are naturally occurring substances that are found in minerals, soil, and rocks. They are present in ambient air, groundwater, plants, and food. The principal commercial use of sulfate is in the chemical industry. Sulfate concentration in seawater is about 2,700 milligrams per liter (mg/L). It ranges from 3 to 30 mg/L in most freshwater supplies, although much higher concentrations (1000 mg/L) are found in some geographic locations.
6. **Conductivity:** Pure water is not a good conductor of electric current rather's a good insulator. Increase in ions concentration enhances the electrical conductivity of water. Generally, the amount of dissolved solids in water determines the electrical conductivity. Electrical conductivity (EC) actually measures the ionic process of a solution that enables it to transmit current. According to WHO standards, EC value should not exceeded 400 µS/cm.
7. **Organic_carbon:** Total Organic Carbon (TOC) in source waters comes from decaying natural organic matter (NOM) as well as synthetic sources. TOC is a measure of the total amount of carbon in organic compounds in pure water. According to US EPA < 2 mg/L as TOC in treated / drinking water, and < 4 mg/Lit in source water which is use for treatment.
8. **Trihalomethanes:** THMs are chemicals which may be found in water treated with chlorine. The concentration of THMs in drinking water varies according to the level of organic material in the water, the amount of chlorine required to treat the water, and the temperature of the water that is being treated. THM levels up to 80 ppm is considered safe in drinking water.
9. **Turbidity:** The turbidity of water depends on the quantity of solid matter present in the suspended state. It is a measure of light emitting properties of water and the test is used to indicate the quality of waste discharge with respect to colloidal matter. The mean turbidity value obtained for Wondo Genet Campus (0.98 NTU) is lower than the WHO recommended value of 5.00 NTU.
10. **Potability:** Indicates if water is safe for human consumption where 1 means Potable and 0 means Not potable.

Data Description:

The DataFrame consists of 3276 rows with10 columns as part of data cleaning I have

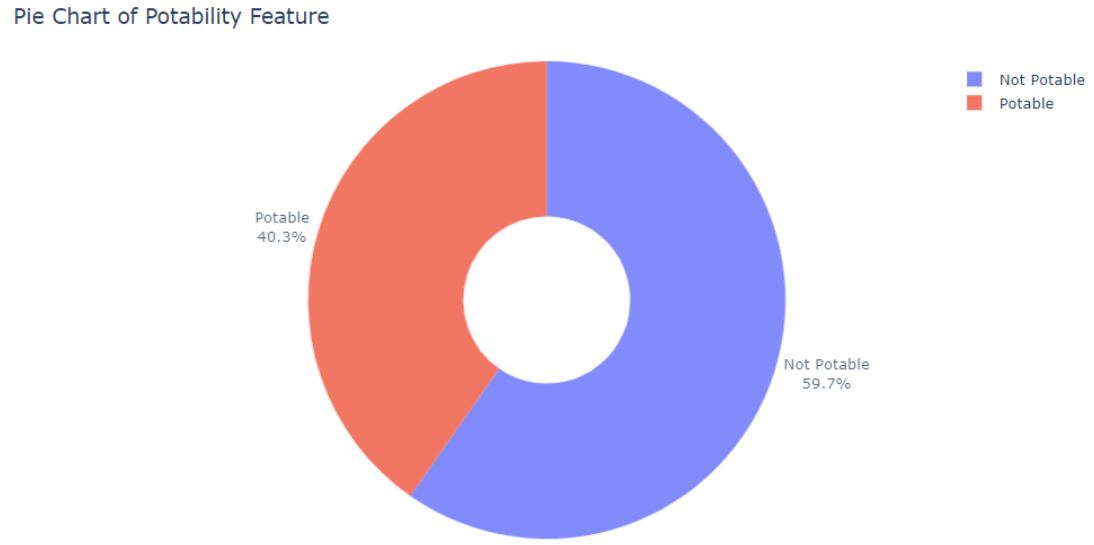
started with identifying Missing values in the data.

```
#Check for missing values
df.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
```

We have found that 3 columns naming ph,sulfates,thihalmethanes are having missing values So strait up we have dropped the missing values using isnull() function in python and procced with next task to check the balance of the y value in the data.

```
d = pd.DataFrame(df["Potability"].value_counts())
fig = px.pie(d, values = "Potability", names = ["Not Potable", "Potable"], hole = 0.35, opacity = 0.8,
             labels = {"label" :"Potability","Potability":"Number of Samples"})
fig.update_layout(title = dict(text = "Pie Chart of Potability Feature"))
fig.update_traces(textposition = "outside", textinfo = "percent+label")
fig.show()
```



As seen with above plot we can say that the data is imbalance and we need to
balance it
using resampling package in python I have resampled it

```
zero  = df[df['Potability']==0]   #zero values in Potability column
one = df[df['Potability']==1]  # one values in Potability column
```

```
from sklearn.utils import resample
```

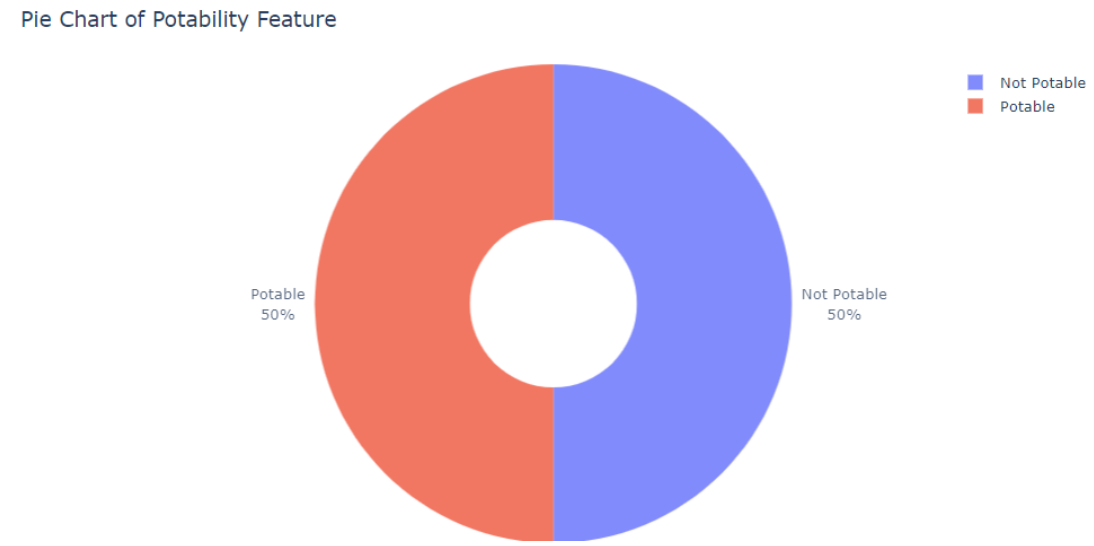
```
df_minority_upsampled = resample(one, replace = True, n_samples = 1200)
```

```
df = pd.concat([zero, df_minority_upsampled])

from sklearn.utils import shuffle
df = shuffle(df)
```

After resampling the data

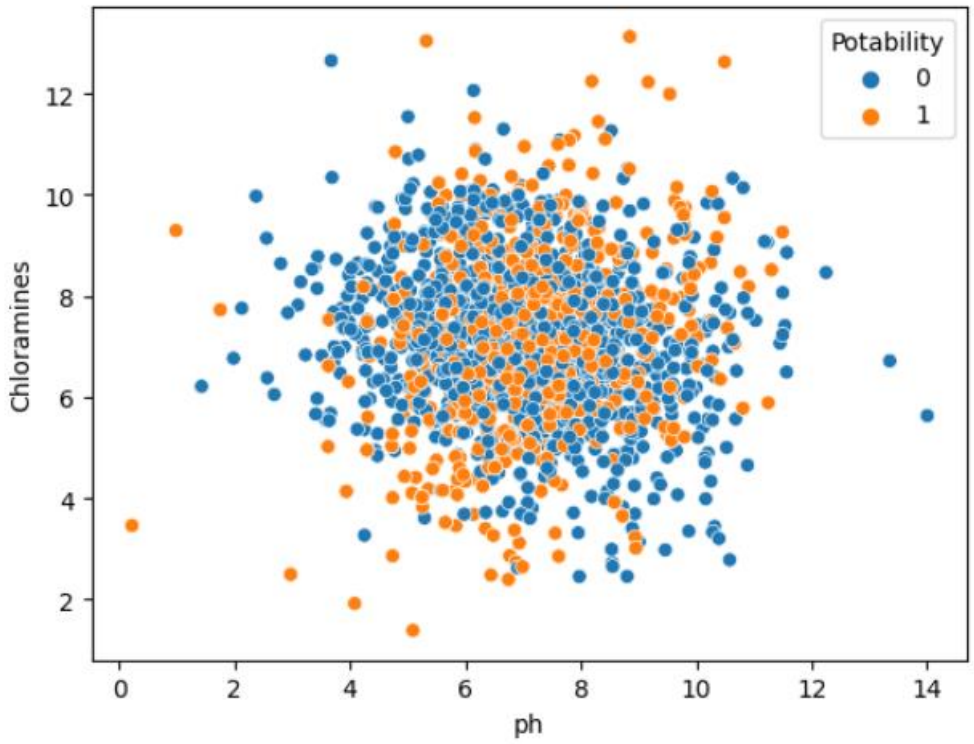
```
d = pd.DataFrame(df["Potability"].value_counts())
fig = px.pie(d, values = "Potability", names = ["Not Potable", "Potable"], hole = 0.35, opacity = 0.8,
             labels = {"label" :"Potability","Potability":"Number of Samples"})
fig.update_layout(title = dict(text = "Pie Chart of Potability Feature"))
fig.update_traces(textposition = "outside", textinfo = "percent+label")
fig.show()
```



As part of identifying the pattern I have plotted the scatter plot

```
1 sns.scatterplot(x=df["ph"], y=df["Chloramines"], hue=df.Potability,data=df)
```

```
<AxesSubplot:xlabel='ph', ylabel='Chloramines'>
```



But there is no particular pattern to be identified

Model Building

As part of model building we have choose decision tree model

The internal nodes of this decision tree represent the various qualities, while the branches between the nodes indicate the potential values that these attributes can have in the observed samples, and the terminal nodes indicate the dependent variable's ultimate value (classification). The predicted attribute is the dependent variable in the dataset, whereas the other characteristics in the tree are the independent variables.

No coming to the hyper parameters to the model

```
#parameters for decision tree
para_dt = {'criterion':['gini','entropy'],'max_depth':np.arange(1, 50), 'min_samples_leaf':[1,2,4,5,10,20,30,40,80,100]}
grid_dt = GridSearchCV(dt, param_grid=para_dt, cv=10) #grid search decision tree for 10 fold cv
```

With cv 10 we have build the model

And the best parameterd for the model are :`Best parameters for Decision Tree: {'criterion': 'entropy', 'max_depth': 22, 'min_samples_leaf': 1}`

```
from sklearn.metrics import accuracy_score

for classifier_name, classifier in classifiers:

    # Fit clf to the training set
    classifier.fit(X_train, y_train)

    # Predict y_pred
    y_pred = classifier.predict(X_test)
    accuracy = accuracy_score(y_test,y_pred)



    # Evaluate clf's accuracy on the test set
    print('{:s} : {:.2f}'.format(classifier_name, accuracy))
```

```
Decision Tree : 0.81
Random Forest : 0.88
```

Following with the confusion matrix:

```
y_pred_dt= dt.predict(X_test)
print(classification_report(y_test, y_pred_dt))
```

```
              precision    recall  f1-score   support

           0       0.84      0.76      0.79       115
           1       0.79      0.86      0.83       125

    accuracy                           0.81       240
   macro avg       0.82      0.81      0.81       240
weighted avg       0.81      0.81      0.81       240
```

Second model

Random Forest :

Multiple random decision trees make up a random forest. The trees incorporate two different kinds of randomization. Each tree is initially constructed using a random sampling of the original data. Second, to provide the best split, a subset of features are chosen at random for each tree node.

```
params_rf = {'n_estimators':[100,200, 350, 500], 'min_samples_leaf':[2, 10, 30]}
grid_rf = GridSearchCV(rf, param_grid=params_rf, cv=10)
```

And the best fit for the data is

```
Best parameters for Random Forest: {'min_samples_leaf': 2,
 'n_estimators': 200}
```

Accuracy achieved by the the random forest is :`Random Forest : 0.88`

```
from sklearn.metrics import classification_report

y_pred_rf= rf.predict(X_test)
print(classification_report(y_test, y_pred_rf))
```

```
              precision    recall  f1-score   support

           0       0.84      0.91      0.87       115
           1       0.91      0.84      0.87       125

    accuracy                           0.88       240
   macro avg       0.88      0.88      0.87       240
weighted avg       0.88      0.88      0.87       240
```

The precision of class 0 is 84% and that of class 1 is 91% It means the model predicts 84% of class 0 and 91% of class 1 correctly.