```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv("water_potability.csv")


data.head()
```

```
         ph    Hardness        Solids  Chloramines     Sulfate
Conductivity  \
0       NaN  204.890455  20791.318981     7.300212  368.516441
564.308654
1  3.716080  129.422921  18630.057858     6.635246         NaN
592.885359
2  8.099124  224.236259  19909.541732     9.275884         NaN
418.606213
3  8.316766  214.373394  22018.417441     8.059332  356.886136
363.266516
4  9.092223  181.101509  17978.986339     6.546600  310.135738
398.410813

   Organic_carbon  Trihalomethanes  Turbidity  Potability
0       10.379783        86.990970   2.963135           0
1       15.180013        56.329076   4.500656           0
2       16.868637        66.420093   3.055934           0
3       18.436524       100.341674   4.628771           0
4       11.558279        31.997993   4.075075           0
```

```
#The rows and columns

data.shape
```

```
(3276, 10)
```

DATA CLEANING

```
data.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
```

```
data.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
```

```
data.fillna(data.mean(),inplace=True)
```

```
data.isnull().sum()
```

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

#EDP

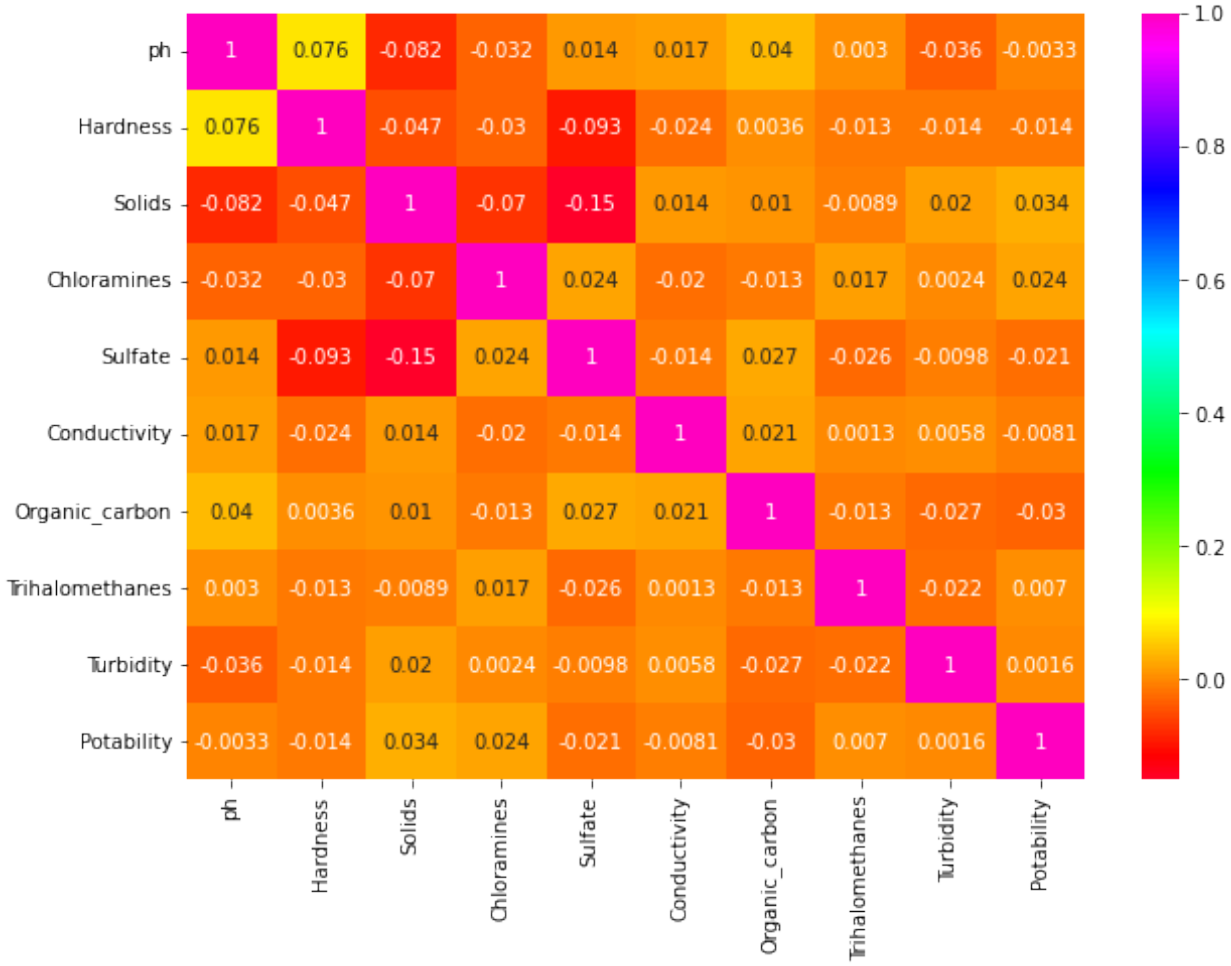
```
data.describe()
```

```
                ph     Hardness        Solids  Chloramines
Sulfate  \
count  2785.000000  3276.000000   3276.000000  3276.000000
2495.000000
mean      7.080795   196.369496  22014.092526     7.122277
333.775777
std       1.594320    32.879761   8768.570828     1.583085
41.416840
min       0.000000    47.432000    320.942611     0.352000
129.000000
```

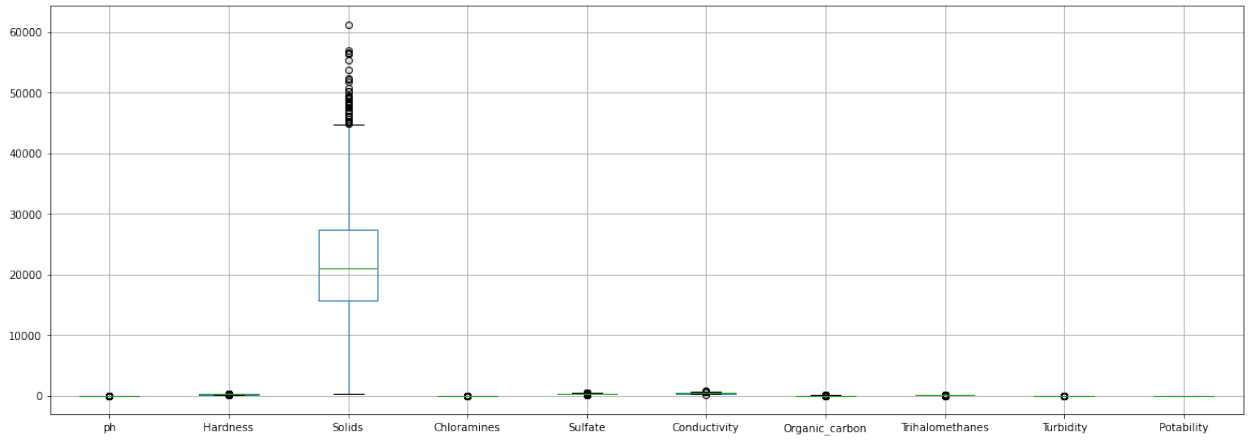
```
25%       6.093092   176.850538  15666.690297     6.127421
307.699498
50%       7.036752   196.967627  20927.833607     7.130299
333.073546
75%       8.062066   216.667456  27332.762127     8.114887
359.950170
max      14.000000   323.124000  61227.196008    13.127000
481.030642

       Conductivity  Organic_carbon  Trihalomethanes    Turbidity
Potability
count   3276.000000     3276.000000      3114.000000  3276.000000
3276.000000
mean     426.205111       14.284970        66.396293     3.966786
0.390110
std       80.824064        3.308162        16.175008     0.780382
0.487849
min      181.483754        2.200000         0.738000     1.450000
0.000000
25%      365.734414       12.065801        55.844536     3.439711
0.000000
50%      421.884968       14.218338        66.622485     3.955028
0.000000
75%      481.792304       16.557652        77.337473     4.500320
1.000000
max      753.342620       28.300000       124.000000     6.739000
1.000000
```

```python
# if reduce the dimensionality of a dataset is needed
# exploring the data
sns.heatmap(data.corr(),annot=True,cmap='gist_rainbow')
fig=plt.gcf()
fig.set_size_inches(10,7)
plt.show()
```




```
#Outlier using Box plot
#outlier is an observation that lies abnormally far away from other
values in a dataset
data.boxplot(figsize=(20,7))
plt.show()
```




```
data['Solids'].describe()
```

```
count     3276.000000
mean     22014.092526
std       8768.570828
min        320.942611
25%      15666.690297
50%      20927.833607
75%      27332.762127
max      61227.196008
Name: Solids, dtype: float64
```

I considerd not to remove it because there are lots of outliers which can mean that excess solid shows bad quality

continuing with EDA

```
data.head()
```

```
         ph    Hardness        Solids  Chloramines     Sulfate
Conductivity  \
0  7.080795  204.890455  20791.318981     7.300212  368.516441
564.308654
1  3.716080  129.422921  18630.057858     6.635246  333.775777
592.885359
2  8.099124  224.236259  19909.541732     9.275884  333.775777
418.606213
3  8.316766  214.373394  22018.417441     8.059332  356.886136
363.266516
4  9.092223  181.101509  17978.986339     6.546600  310.135738
398.410813

   Organic_carbon  Trihalomethanes  Turbidity  Potability
```

```
0       10.379783        86.990970   2.963135           0
1       15.180013        56.329076   4.500656           0
2       16.868637        66.420093   3.055934           0
3       18.436524       100.341674   4.628771           0
4       11.558279        31.997993   4.075075           0

data.shape

(3276, 10)

data.info

<bound method DataFrame.info of             ph    Hardness
Solids  Chloramines     Sulfate  \
0     7.080795  204.890455  20791.318981     7.300212  368.516441
1     3.716080  129.422921  18630.057858     6.635246  333.775777
2     8.099124  224.236259  19909.541732     9.275884  333.775777
3     8.316766  214.373394  22018.417441     8.059332  356.886136
4     9.092223  181.101509  17978.986339     6.546600  310.135738
...        ...         ...           ...          ...         ...
3271  4.668102  193.681735  47580.991603     7.166639  359.948574
3272  7.808856  193.553212  17329.802160     8.061362  333.775777
3273  9.419510  175.762646  33155.578218     7.350233  333.775777
3274  5.126763  230.603758  11983.869376     6.303357  333.775777
3275  7.874671  195.102299  17404.177061     7.509306  333.775777

      Conductivity  Organic_carbon  Trihalomethanes  Turbidity
Potability
0       564.308654       10.379783        86.990970   2.963135
0
1       592.885359       15.180013        56.329076   4.500656
0
2       418.606213       16.868637        66.420093   3.055934
0
3       363.266516       18.436524       100.341674   4.628771
0
4       398.410813       11.558279        31.997993   4.075075
0
...            ...             ...              ...        ...
...
3271    526.424171       13.894419        66.687695   4.435821
1
3272    392.449580       19.903225        66.396293   2.798243
1
3273    432.044783       11.039070        69.845400   3.298875
1
3274    402.883113       11.168946        77.488213   4.708658
1
3275    327.459760       16.140368        78.698446   2.309149
1
```

```
[3276 rows x 10 columns]>
```

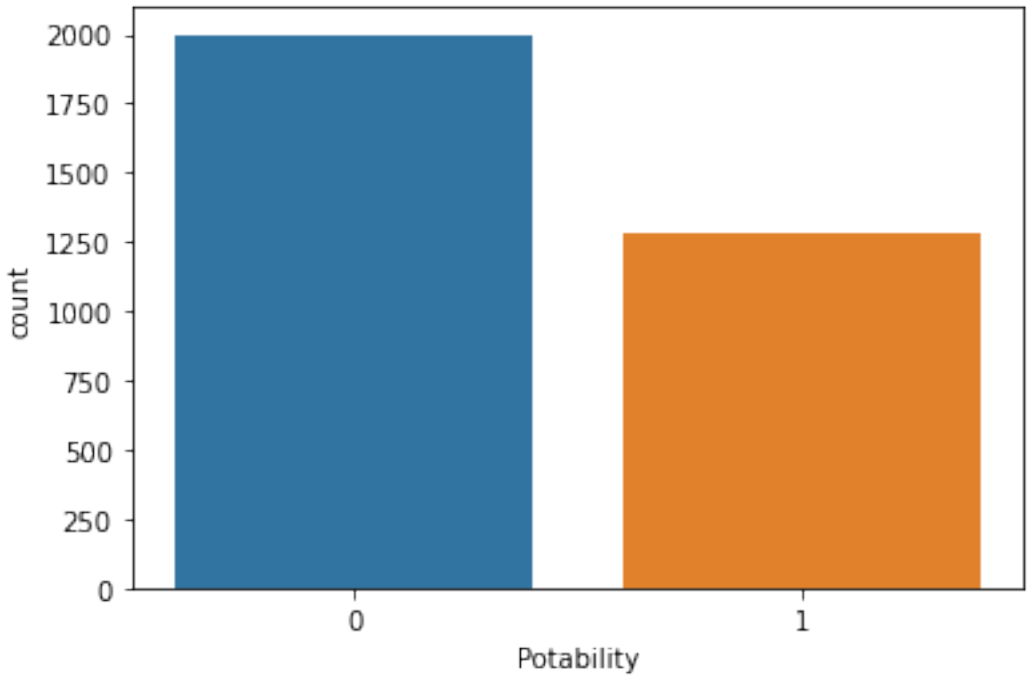
```
# 1 means Potable and 0 means Not potable.

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

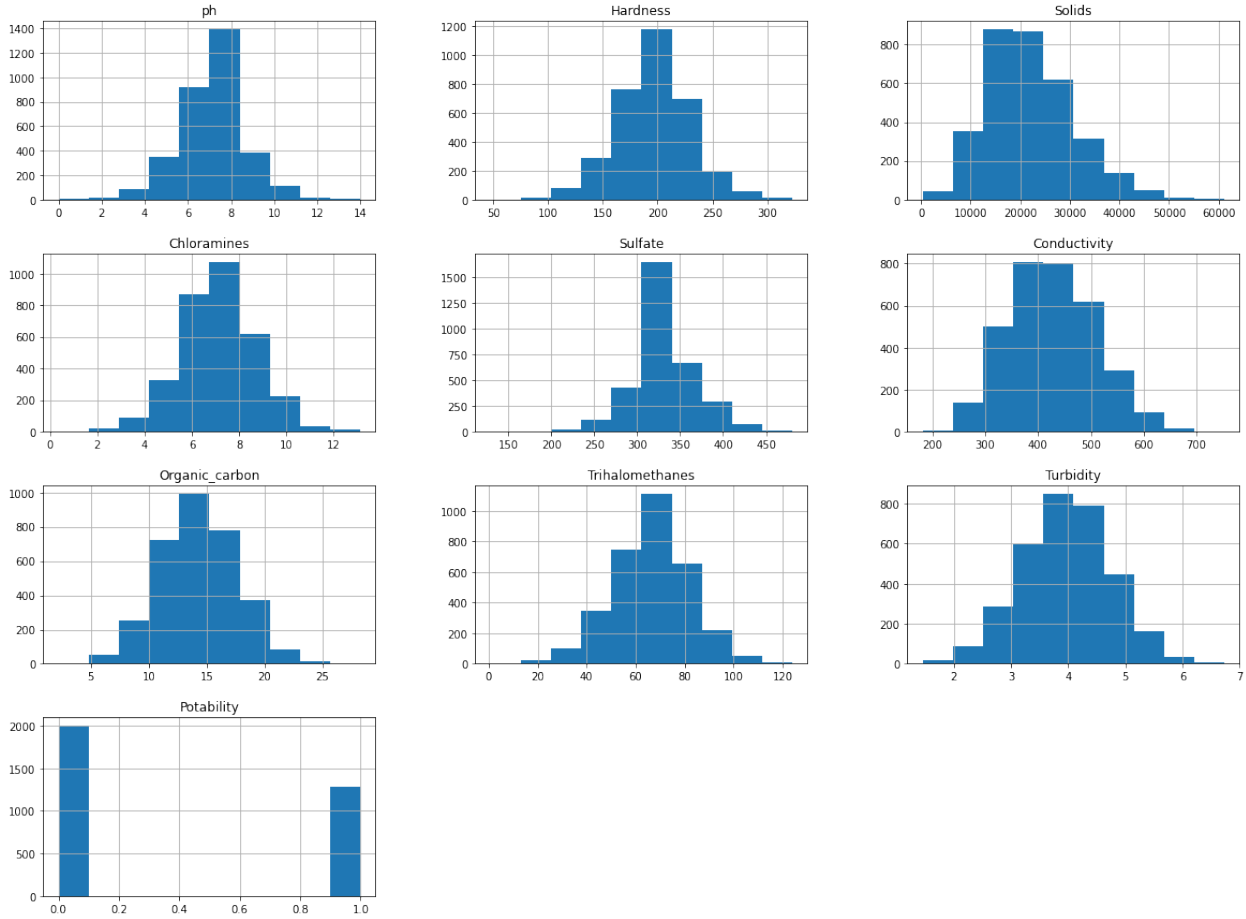
```
0    1998
1    1278
Name: Potability, dtype: int64
```

```
#Checking the balance of the data
sns.countplot(data['Potability'])
plt.show()
```

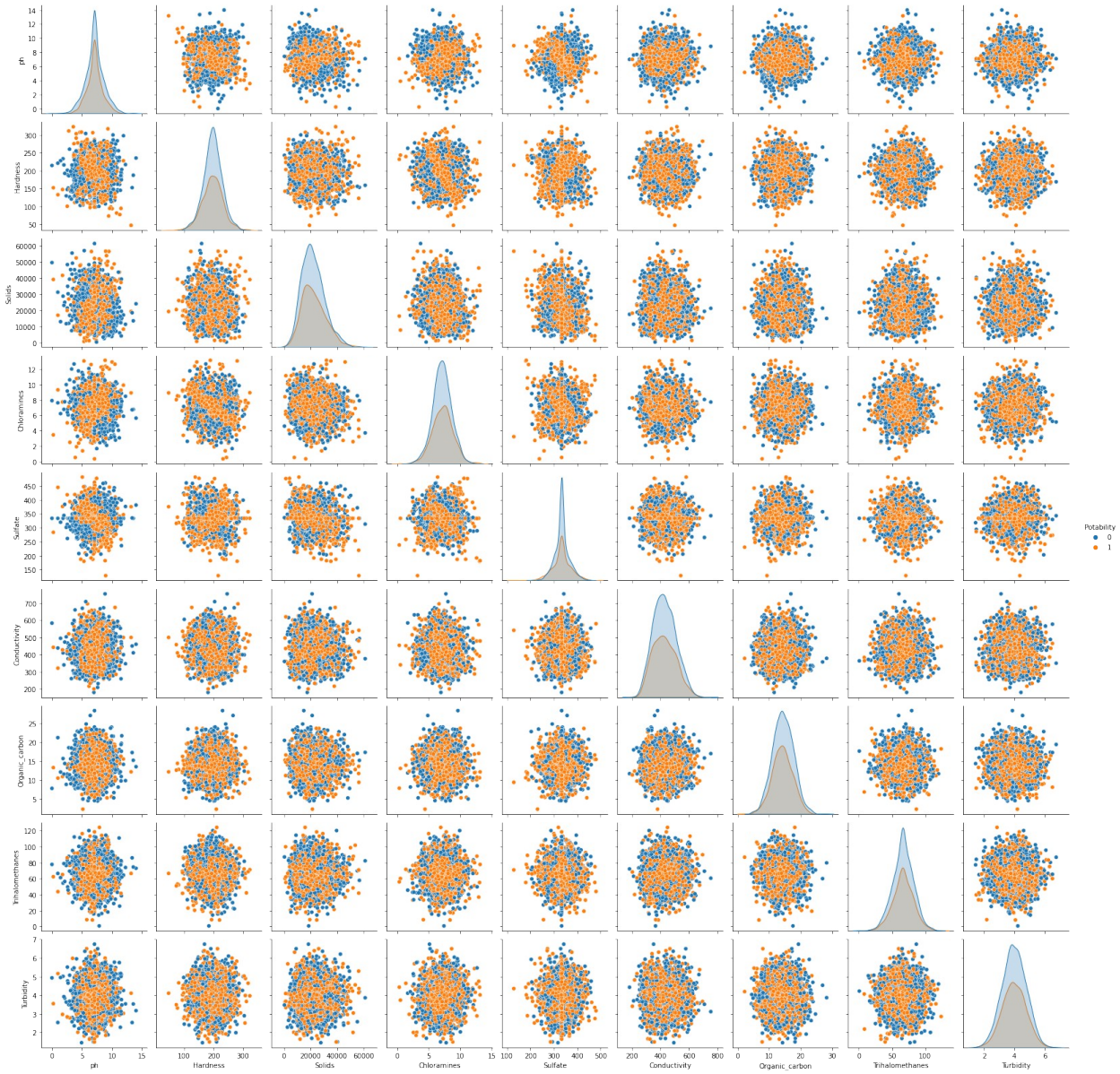
```
C:\Users\malsh\AppData\Local\Programs\Python\Python310\lib\site-
packages\seaborn\_decorators.py:36: FutureWarning: Pass the following
variable as a keyword arg: x. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```




```
data.hist(figsize=(20,15))
plt.show()
```

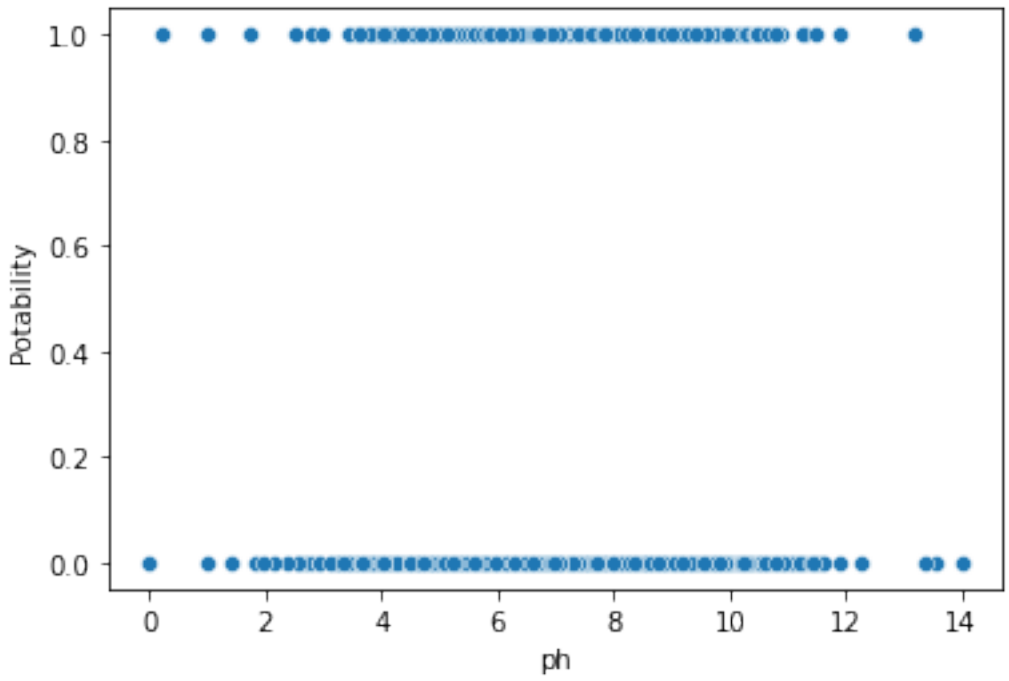



```
#Assigning a hue variable adds a semantic mapping and changes the
default marginal plot to a layered kernel density estimate (KDE):
sns.pairplot(data,hue='Potability')
plt.show()
```



This is normal distribution

```
sns.scatterplot(x=data['ph'], y=data['Potability'])
plt.show()
```



for more ph values, the water quality is good and once it goes over 12 its bad quality water

partitioning

```
x=data.drop('Potability',axis=1) #input data

y=data['Potability'] #target variable

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, shuffle=True, random_state=404)

y_train
```

```
536     0
2709    1
1367    0
995     0
3106    0
       ..
1935    1
1012    0
1206    1
1898    1
1788    0
Name: Potability, Length: 2620, dtype: int64
```

```
x_train
```

```
            ph    Hardness        Solids  Chloramines     Sulfate  \
536   9.606859  200.842143  18907.642841     7.515087  333.775777
2709  7.080795  189.823418  20278.338272     6.799940  333.775777
1367  6.906575  199.638124  15201.339954     5.136599  333.775777
995   8.312380  203.744548   8727.247349     7.456302  333.775777
3106  7.080795  156.773181  23084.066585     7.269795  334.956100
...        ...         ...           ...          ...         ...
1935  7.080795  169.775475  37273.429223     8.027830  240.897629
1012  7.581688  180.749140  11989.246243     4.977307  328.176978
1206  7.080795  134.679257  30211.832991     4.792361  234.285621
1898  6.203573  139.129083   6698.239095     3.876813  333.775777
1788  7.436537  167.328466  31935.690705     7.896365  333.775777

      Conductivity  Organic_carbon  Trihalomethanes  Turbidity
536     370.903807       11.358735        64.371183   4.405352
2709    314.834175       10.092087        46.629299   4.940015
1367    306.023975       15.212798        90.579020   3.282750
995     543.392988       15.470400        81.508682   2.988093
3106    378.253869       19.247141        81.571554   5.564902
...            ...             ...              ...        ...
1935    454.543765       18.621698        63.519516   3.573741
1012    617.883513       13.561253        39.215917   4.457282
1206    391.820964       18.999154        66.396293   3.840889
1898    601.526167       13.368165        68.298689   4.305549
1788    398.574215       14.824433        69.252783   4.497629

[2620 rows x 9 columns]
```

Model trainning

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

AB = DecisionTreeClassifier(criterion = 'entropy',min_samples_split =
9, splitter='best')

AB.fit(x_train,y_train)

DecisionTreeClassifier(criterion='entropy', min_samples_split=9)

y_test
```

```
65      0
190     0
841     0
1066    0
2831    1
```

```
      ..
1644    0
805     1
2848    1
875     0
2834    1
Name: Potability, Length: 656, dtype: int64
```

```
y_prediction=AB.predict(x_test)
```

```
accuracy_score(y_prediction,y_test) *100
```

```
62.5
```

```
confusion_matrix(y_prediction,y_test)
```

```
array([[264, 128],
       [118, 146]], dtype=int64)
```

```
#The lenght
y_test.shape
```

```
(656,)
```

model optimization

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RepeatedStratifiedKFold

AB = DecisionTreeClassifier()


criterion = ["gini", "entropy"]
splitter = ["best", "random"]
min_samples_split=range(1,10)


parameters = dict(criterion = criterion, splitter =
splitter,min_samples_split = min_samples_split)
cv = RepeatedStratifiedKFold(n_splits = 5, random_state = 250)


grid_search_cv_AB = GridSearchCV(estimator=AB, param_grid=parameters,
scoring='accuracy',  cv=cv)

grid_search_cv_AB.fit(x_train, y_train)
```

```
C:\Users\malsh\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\model_selection\_validation.py:378: FitFailedWarning:

200 fits failed out of a total of 1800.
The score on these train-test partitions for these parameters will be
set to nan.
If these failures are not expected, you can try to debug them by
setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------
----------
200 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\malsh\AppData\Local\Programs\Python\Python310\lib\
site-packages\sklearn\model_selection\_validation.py", line 686, in
_fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\malsh\AppData\Local\Programs\Python\Python310\lib\
site-packages\sklearn\tree\_classes.py", line 969, in fit
    super().fit(
  File "C:\Users\malsh\AppData\Local\Programs\Python\Python310\lib\
site-packages\sklearn\tree\_classes.py", line 265, in fit
    check_scalar(
  File "C:\Users\malsh\AppData\Local\Programs\Python\Python310\lib\
site-packages\sklearn\utils\validation.py", line 1480, in check_scalar
    raise ValueError(
ValueError: min_samples_split == 1, must be >= 2.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\malsh\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\model_selection\_search.py:953: UserWarning: One or
more of the test scores are non-finite: [       nan        nan
0.57847328 0.57061069 0.58022901 0.57652672
 0.58274809 0.57770992 0.58087786 0.58091603 0.58446565 0.57744275
 0.58473282 0.59003817 0.58427481 0.58801527 0.58412214 0.58885496
        nan        nan 0.58160305 0.57416031 0.58656489 0.58171756
 0.5859542  0.57946565 0.58458015 0.58465649 0.58725191 0.58610687
 0.58931298 0.58889313 0.58641221 0.58438931 0.5870229  0.58198473]
  warnings.warn(

GridSearchCV(cv=RepeatedStratifiedKFold(n_repeats=10, n_splits=5,
random_state=250),
             estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'min_samples_split': range(1, 10),
                         'splitter': ['best', 'random']},
             scoring='accuracy')

print(grid_search_cv_AB.best_params_)
```

```
{'criterion': 'gini', 'min_samples_split': 7, 'splitter': 'random'}
```

```
prediction_grid=grid_search_cv_AB.predict(x_test)
```

```
accuracy_score(prediction_grid,y_test) *100
```

```
58.6890243902439
```

```
confusion_matrix(y_test,prediction_grid)
```

```
array([[267, 115],
       [156, 118]], dtype=int64)
```

KNN Algorithm

```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn=KNeighborsClassifier(metric='manhattan', n_neighbors=22)
knn.fit(x_train,y_train)
```

```
KNeighborsClassifier(metric='manhattan', n_neighbors=22)
```

```
prediction_knn=knn.predict(x_test)
accuracy_knn=accuracy_score(y_test,prediction_knn)*100
print('accuracy_score
score:',accuracy_score(y_test,prediction_knn)*100,'%')
```

```
accuracy_score score: 58.536585365853654 %
```

```
confusion_matrix(prediction_grid,y_test)
```

```
array([[267, 156],
       [115, 118]], dtype=int64)
```