Case Study- Water Potability Analysis

Source: [https://www.kaggle.com/adityakadiwal/water-potability (https://www.kaggle.com/adityakadiwal/water-potability)](https://www.kaggle.com/adityakadiwal/water-potability)

The water_potability.csv file contains water quality metrics for 3276 different water bodies and 10 variables.

1. PH value: PH is an important parameter in evaluating the acid–base balance of water. WHO has recommended maximum permissible limit of pH from 6.5 to 8.5.
2. Hardness: Hardness defined as the capacity of water to precipitate soap caused by Calcium and Magnesium. 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.
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.

Import Libraries

In [74]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Read the CSV file

In [75]:
```
df=pd.read_csv(r'C:\Users\lxm\Downloads\water_potability.csv')
df
```

Out[75]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.180013
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.868637
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279
...	...	...	...	...	...	...	...
3271	4.668102	193.681735	47580.991603	7.166639	359.948574	526.424171	13.894419
3272	7.808856	193.553212	17329.802160	8.061362	NaN	392.449580	19.903225
3273	9.419510	175.762646	33155.578218	7.350233	NaN	432.044783	11.039070
3274	5.126763	230.603758	11983.869376	6.303357	NaN	402.883113	11.168946
3275	7.874671	195.102299	17404.177061	7.509306	NaN	327.459760	16.140368

3276 rows × 10 columns

In [76]:
```
df.columns
```

Out[76]:
```
Index(['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'Conductivity',
       'Organic_carbon', 'Trihalomethanes', 'Turbidity', 'Potability'],
      dtype='object')
```

Shapes of the Dataset

In [77]:
```
df.shape
```

Out[77]:
```
(3276, 10)
```

Checking the null values in the dataset

In [78]:
```
df.isna().sum()
```

Out[78]:
```
ph                 491
Hardness             0
Solids               0
Chloramines          0
Sulfate            781
Conductivity         0
Organic_carbon       0
Trihalomethanes    162
Turbidity            0
Potability           0
dtype: int64
```

Statistical details about dataset

In [79]:
```
df.describe().T
```

Out[79]:

	count	mean	std	min	25%	50%	
ph	2785.0	7.080795	1.594320	0.000000	6.093092	7.036752	8
Hardness	3276.0	196.369496	32.879761	47.432000	176.850538	196.967627	216
Solids	3276.0	22014.092526	8768.570828	320.942611	15666.690297	20927.833607	27332
Chloramines	3276.0	7.122277	1.583085	0.352000	6.127421	7.130299	8
Sulfate	2495.0	333.775777	41.416840	129.000000	307.699498	333.073546	359
Conductivity	3276.0	426.205111	80.824064	181.483754	365.734414	421.884968	481
Organic_carbon	3276.0	14.284970	3.308162	2.200000	12.065801	14.218338	16
Trihalomethanes	3114.0	66.396293	16.175008	0.738000	55.844536	66.622485	77
Turbidity	3276.0	3.966786	0.780382	1.450000	3.439711	3.955028	4
Potability	3276.0	0.390110	0.487849	0.000000	0.000000	0.000000	1

Fill the null values by the median value in the dataset

In [80]:
```
df['ph'].fillna(7.5,inplace=True)
```

In [81]:
```
df['Sulfate'].fillna(333.073546,inplace=True)
```

In [82]:
```
df['Trihalomethanes'].fillna(66.622485,inplace=True)
```

In [83]:
```
df.isna().sum()
```

Out[83]:
```
ph                 0
Hardness           0
Solids             0
Chloramines        0
Sulfate            0
Conductivity       0
Organic_carbon     0
Trihalomethanes    0
Turbidity          0
Potability         0
dtype: int64
```

Checked the correlation in datset

In [84]:
```
df.corr()
```

Out[84]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_ca
ph	1.000000	0.076091	-0.080277	-0.032296	0.016710	0.014498	0.03
Hardness	0.076091	1.000000	-0.046899	-0.030054	-0.092833	-0.023915	0.00
Solids	-0.080277	-0.046899	1.000000	-0.070148	-0.149747	0.013831	0.01
Chloramines	-0.032296	-0.030054	-0.070148	1.000000	0.023762	-0.020486	-0.01
Sulfate	0.016710	-0.092833	-0.149747	0.023762	1.000000	-0.014182	0.02
Conductivity	0.014498	-0.023915	0.013831	-0.020486	-0.014182	1.000000	0.02
Organic_carbon	0.038138	0.003610	0.010242	-0.012653	0.027102	0.020966	1.00
Trihalomethanes	0.001600	-0.012707	-0.008799	0.016614	-0.025657	0.001184	-0.01
Turbidity	-0.037100	-0.014449	0.019546	0.002363	-0.009767	0.005798	-0.02
Potability	-0.005853	-0.013837	0.033743	0.023779	-0.020476	-0.008128	-0.03

Grouping the Potability by mean of others variable

In [85]:
```
grouped_data=df.groupby(['Potability']).agg({'ph':np.mean,'Turbidity':np.mean,'Or
grouped_data
```

Out[85]:

	Potability	ph	Turbidity	Organic_carbon
0	0	7.150539	3.965800	14.364335
1	1	7.132813	3.968328	14.160893

Maximum and minimum sulfate are in the water and then Is the water drinkable or not?

In [86]:
```
df[df['Sulfate']==df['Sulfate'].max()]
```

Out[86]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon
2318	5.057736	137.689344	11229.137777	6.41141	481.030642	580.095225	15.390304

In [107]:
```
df[df['Sulfate']==df['Sulfate'].min()]
```

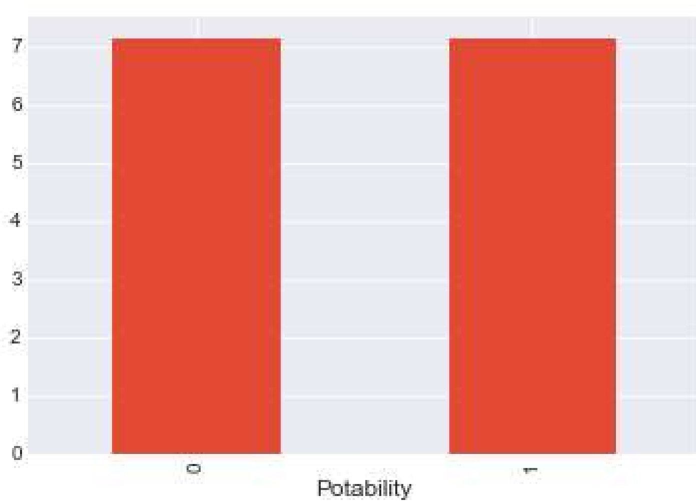
Out[107]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trih
1554	8.942046	215.673786	56488.672413	3.231438	129.0	541.915468	9.313771	

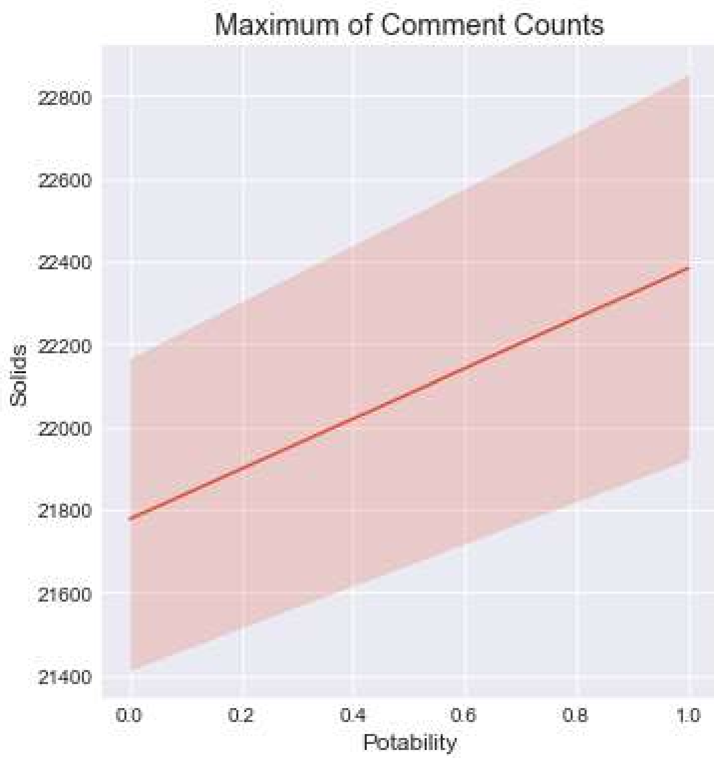
Data Visualizations

In [104]:
```
g=df.groupby('Potability')['ph'].mean().sort_values(ascending=False).plot.bar()
g
```

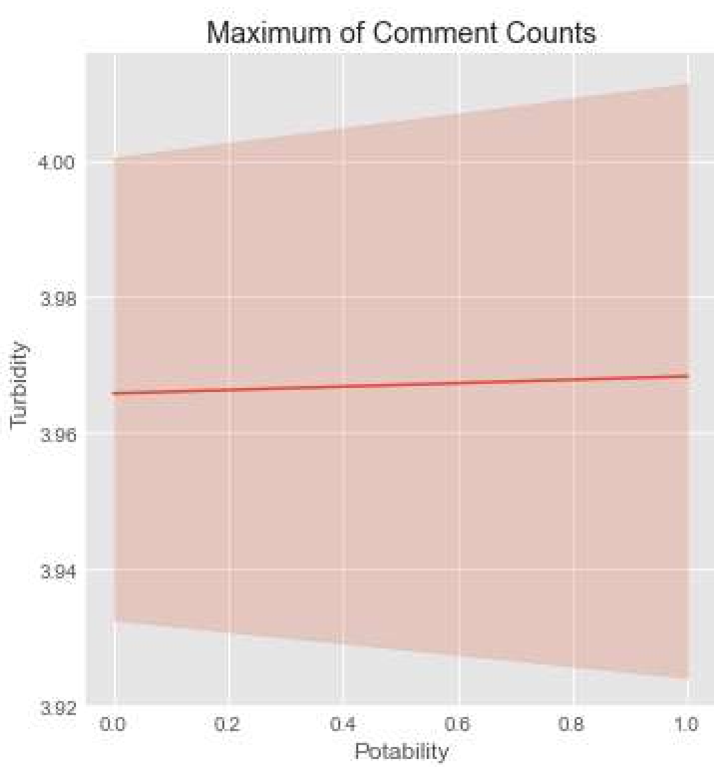
Out[104]: <AxesSubplot:xlabel='Potability'>



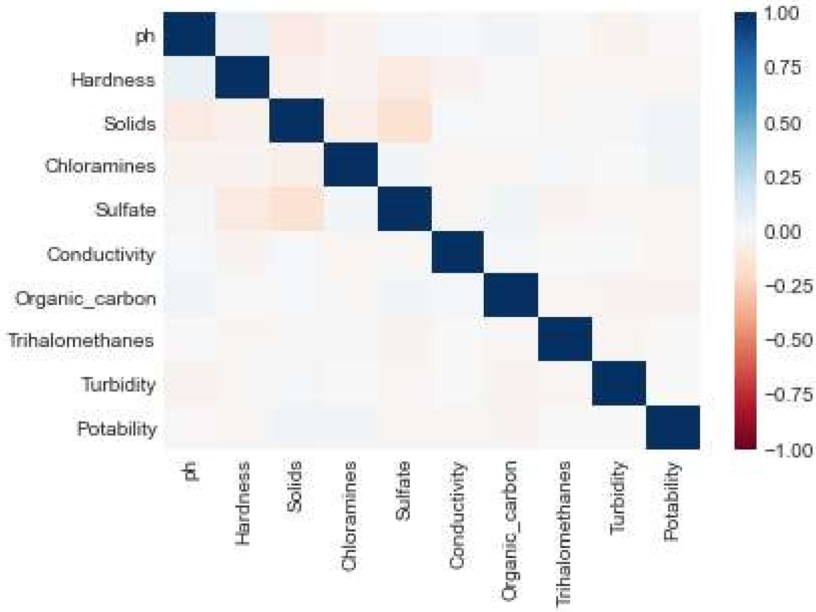
In [93]:
```
sns.relplot(x='Potability',y='Solids',kind='line',data=df)
plt.show()
```



In [56]:
```
sns.relplot(x='Potability',y='Turbidity',kind='line',data=df).set(title='Maximum
plt.show()
```

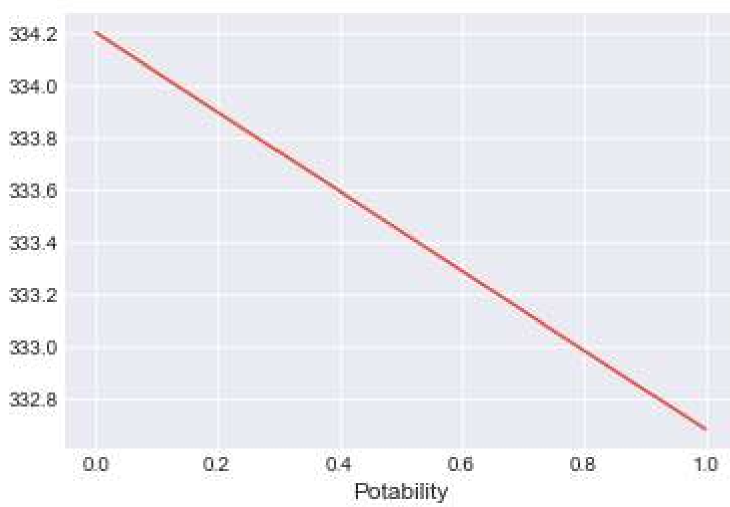


In [61]:
```
sns.heatmap(df.corr(),cmap='RdBu',vmin=-1,vmax=1)
plt.show()
```



In [63]:
```
df.groupby('Potability')['Sulfate'].mean().nlargest(20).plot.line()
```

Out[63]: <AxesSubplot:xlabel='Potability'>



Conclusion

There was ph: 491, Sulfate: 781, and Trihalomethanes: 162 null values in the dateset which has been filled by 7.5, 333.073546, and 66.622485

When the water is not drinkable the level of ph is 7.150539 and when it is drinkable the level of ph is 7.132813

As level of Solids increase then the water is potable

As Sulfate area in the water reduces it is more safe to drink

Sklearn.naive_bayes

In [49]:
```
x=df[df.columns[:-1]]
y=df.Potability
```

In [50]:
```
x
```

Out[50]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon
0	7.500000	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783
1	3.716080	129.422921	18630.057858	6.635246	333.073546	592.885359	15.180013
2	8.099124	224.236259	19909.541732	9.275884	333.073546	418.606213	16.868637
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279
...	...	...	...	...	...	...	...
3271	4.668102	193.681735	47580.991603	7.166639	359.948574	526.424171	13.894419
3272	7.808856	193.553212	17329.802160	8.061362	333.073546	392.449580	19.903225
3273	9.419510	175.762646	33155.578218	7.350233	333.073546	432.044783	11.039070
3274	5.126763	230.603758	11983.869376	6.303357	333.073546	402.883113	11.168946
3275	7.874671	195.102299	17404.177061	7.509306	333.073546	327.459760	16.140368

3276 rows × 9 columns

In [51]:
```
y
```

Out[51]:
```
0       0
1       0
2       0
3       0
4       0
       ..
3271    1
3272    1
3273    1
3274    1
3275    1
Name: Potability, Length: 3276, dtype: int64
```

Import Library Train_Test_Split

In [52]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=49)
```

In [53]:
```
x_train.shape,y_train.shape
```
Out[53]:
```
((2293, 9), (2293,))
```

In [54]:
```
x_test.shape,y_test.shape
```
Out[54]:
```
((983, 9), (983,))
```

In [55]:
```
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
```

BernoulliNB Test

In [56]:
```
bnb=BernoulliNB()
bnb.fit(x_train,y_train)
print("BernoulliNB Training Accuracy:", bnb.score(x_train,y_train))
print("BernoulliNB Testing Accuracy:", bnb.score(x_test,y_test))
```
```
BernoulliNB Training Accuracy: 0.6061927605756651
BernoulliNB Testing Accuracy: 0.6185147507629705
```

MultinomialNB Test

In [57]:
```
mnb= MultinomialNB()
mnb.fit(x_train,y_train)
print("MultinomialNB Training Accuracy:", mnb.score(x_train,y_train))
print("MultinomialNB Testing Accuracy:", mnb.score(x_test,y_test))
```
```
BernoulliNB Training Accuracy: 0.5211513301351941
BernoulliNB Testing Accuracy: 0.5483214649033571
```

GaussianNB Test

In [58]:
```
gnb= GaussianNB()
gnb.fit(x_train,y_train)
print("GaussianNB Training Accuracy:", gnb.score(x_train,y_train))
print("GaussianNB Testing Accuracy:", gnb.score(x_test,y_test))
```
```
BernoulliNB Training Accuracy: 0.6275621456607064
BernoulliNB Testing Accuracy: 0.6185147507629705
```

In [59]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y)
```

In [72]:
```
svc=SVC(kernel='rbf',gamma=2,C=1)
svc.fit(x_train,y_train)
print(svc.score(x_train,y_train))
print(svc.score(x_test,y_test))
```

```
1.0
0.5921855921855922
```

Conclusion

After dividing the data into train and test we fit the naive_bayes into the train test data

GaussianNB gives GaussianNB Training Accuracy: 0.6275621456607064 and GaussianNB Testing Accuracy: 0.6185147507629705

MultinomialNB gives MultinomialNB Training Accuracy: 0.5211513301351941 and MultinomialNB Testing Accuracy: 0.5483214649033571

BernoulliNB gives BernoulliNB Training Accuracy: 0.6061927605756651 and BernoulliNB Testing Accuracy: 0.61851475076297

In [ ]: