```
import pandas as pd
import scipy.stats as stats
import numpy as np
from scipy.stats import f_oneway
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from scipy.stats import chi2_contingency
from scipy.stats import *
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from mlxtend.evaluate import bootstrap
from scipy.stats import norm
import statsmodels.stats.multicomp as mc
import statsmodels.api as sm
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.utils import resample
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.ensemble import RandomForestClassifier
import seaborn as sns
import io
from google.colab import files
from google.colab import drive
```

```
drive.mount('/content/drive')
df = pd.read_excel("drive/My Drive/Personal/water_potability.xlsx")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

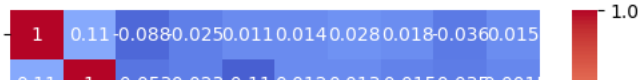
```
df
df=df.dropna()
```

```
df.ph.describe()
```

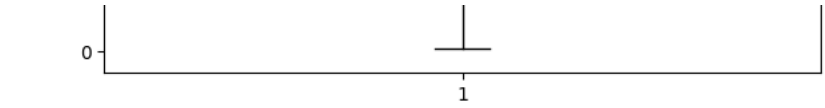
```
count    2011.000000
mean        7.085990
std         1.573337
min         0.227499
25%         6.089723
50%         7.027297
75%         8.052969
max        14.000000
Name: ph, dtype: float64
```

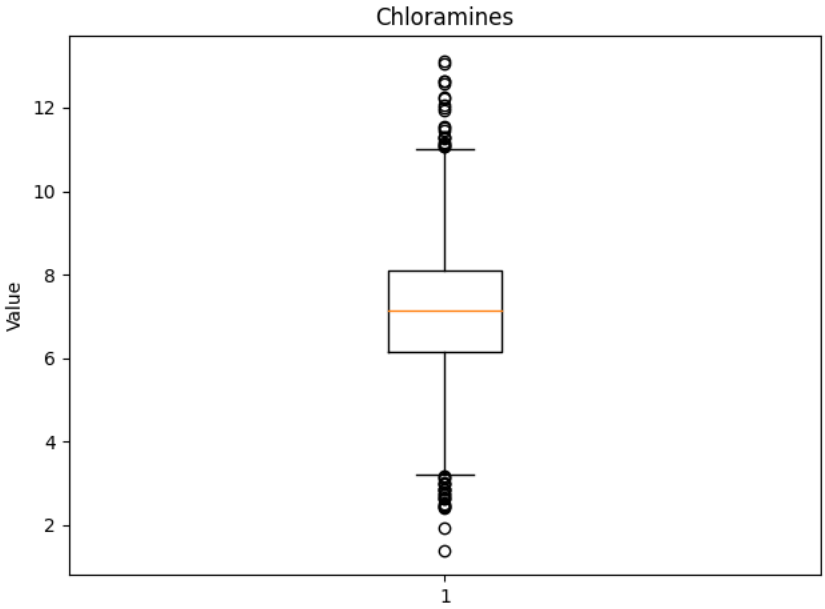
```
df.corr()
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
```

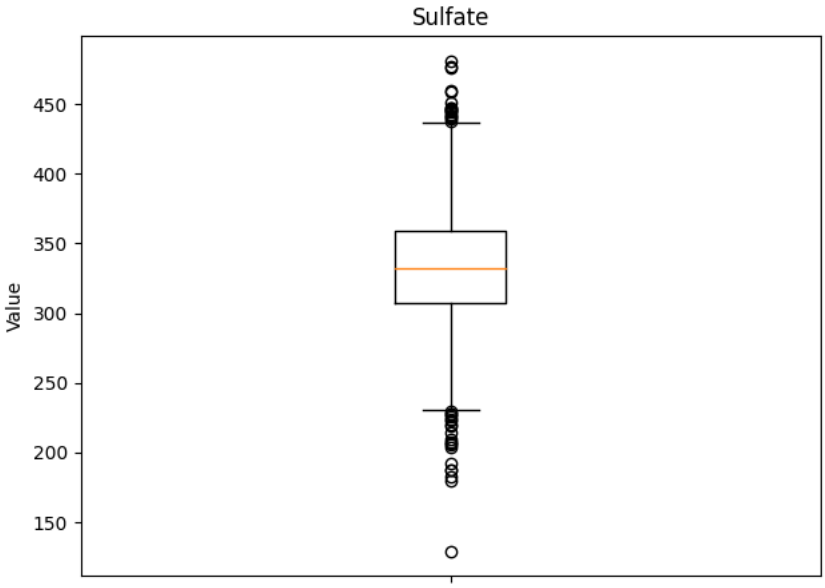
```
<Axes: >
```




```
cols_to_plot = [col for col in df.columns if col != 'Class']
for col in cols_to_plot:
  fig, ax = plt.subplots()
  ax.boxplot(df[col])
  ax.set_title(col)
  ax.set_ylabel('Value')
  plt.tight_layout()
  plt.show()
```


0
1


Chloramines
Value
12
10
8
6
4
2
1


Sulfate
Value
450
400
350
300
250
200
150

```
df
```

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Orgar
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	
5	5.584087	188.313324	28748.687739	7.544869	326.678363	280.467916	
6	10.223862	248.071735	28749.716544	7.513408	393.663396	283.651634	
7	8.635849	203.361523	13672.091764	4.563009	303.309771	474.607645	
...	...	...	...	...	...	...	
3267	8.989900	215.047358	15921.412018	6.297312	312.931022	390.410231	
3268	6.702547	207.321086	17246.920347	7.708117	304.510230	329.266002	
3269	11.491011	94.812545	37188.826022	9.263166	258.930600	439.893618	
3270	6.069616	186.659040	26138.780191	7.747547	345.700257	415.886955	
3271	4.668102	193.681735	47580.991603	7.166639	359.948574	526.424171	

2011 rows × 10 columns

```
df['Potability']
```

```
3       0
4       0
5       0
6       0
7       0
       ..
3267    1
3268    1
3269    1
3270    1
3271    1
Name: Potability, Length: 2011, dtype: int64
```

```
not_potable = df[df['Potability']==0]
potable= df[df['Potability']==1]


not_potable_mean = np.mean(not_potable['Hardness'])
potable_mean = np.mean(potable['Hardness'])
not_potable_std = np.std(not_potable['Hardness'])
potable_std = np.std(potable['Hardness'])
standard_error = np.sqrt(not_potable_std**2/len(not_potable) + potable_std**2/len(potable))
z_score = (not_potable_mean - potable_mean) / standard_error
p_value = 2 * norm.cdf(-np.abs(z_score))
significance_level = 0.05
print("Not Potable mean hardness: ", not_potable_mean)
print("Potable mean hardness: ", potable_mean)
print("Not Potable hardness standard deviation: ",not_potable_std)
print("Potable hardness standard deviation: ",potable_std)
print("P_value: ",p_value)
temp=p_value-significance_level
print(temp)
if p_value < significance_level:
  print("There is significant difference bettern mean of potable and not potable water")
else:
  print("There is no significant difference between mean of potable and not potable water")
```

```
Not Potable mean hardness:  196.00843971539516
Potable mean hardness:  195.90834062028307
Not Potable hardness standard deviation:  30.704840345986625
Potable hardness standard deviation:  35.27937569340146
```

```
P_value:  0.9476060340227065
0.8976060340227064
There is no significant difference between mean of potable and not potable water
```

```
not_potable_mean = np.mean(not_potable['ph'])
potable_mean = np.mean(potable['ph'])
not_potable_std = np.std(not_potable['ph'])
potable_std = np.std(potable['ph'])
standard_error = np.sqrt(not_potable_std**2/len(not_potable) + potable_std**2/len(potable))
z_score = (not_potable_mean - potable_mean) / standard_error
p_value = 2 * norm.cdf(-np.abs(z_score))
significance_level = 0.05
print("Not Potable mean ph: ", not_potable_mean)
print("Potable mean ph: ", potable_mean)
print("Not Potable ph standard deviation: ",not_potable_std)
print("Potable ph standard deviation: ",potable_std)
print("P_value: ",p_value)
if p_value < significance_level:
  print("There is significant difference bettern mean of potable and not potable water")
else:
  print("There is no significant difference between mean of potable and not potable water")
```

```
Not Potable mean ph:  7.067200989469905
Potable mean ph:  7.113790850108899
Not Potable ph standard deviation:  1.6584142049817716
Potable ph standard deviation:  1.4367361970848636
P_value:  0.5029374991503169
There is no significant difference between mean of potable and not potable water
```

```
not_potable_mean = np.mean(not_potable['Solids'])
potable_mean = np.mean(potable['Solids'])
not_potable_std = np.std(not_potable['Solids'])
potable_std = np.std(potable['Solids'])
standard_error = np.sqrt(not_potable_std**2/len(not_potable) + potable_std**2/len(potable))
z_score = (not_potable_mean - potable_mean) / standard_error
p_value = 2 * norm.cdf(-np.abs(z_score))
significance_level = 0.05
print("Not Potable mean Solids: ", not_potable_mean)
print("Potable mean Solids: ", potable_mean)
print("Not Potable Solids standard deviation: ",not_potable_std)
print("Potable Solids standard deviation: ",potable_std)
print("P_value: ",p_value)
temp=p_value-significance_level
print(temp)
if p_value < significance_level:
  print("There is significant difference bettern mean of potable and not potable water")
else:
  print("There is no significant difference between mean of potable and not potable water")
```

```
Not Potable mean Solids:  21628.535121541106
Potable mean Solids:  22344.922883169846
Not Potable Solids standard deviation:  8457.582495861785
Potable Solids standard deviation:  8886.064433266549
P_value:  0.07058222282826863
0.02058222282826863
There is no significant difference between mean of potable and not potable water
```

```
not_potable_mean = np.mean(not_potable['Chloramines'])
potable_mean = np.mean(potable['Chloramines'])
not_potable_std = np.std(not_potable['Chloramines'])
potable_std = np.std(potable['Chloramines'])
standard_error = np.sqrt(not_potable_std**2/len(not_potable) + potable_std**2/len(potable))
z_score = (not_potable_mean - potable_mean) / standard_error
p_value = 2 * norm.cdf(-np.abs(z_score))
significance_level = 0.05
print("Not Potable mean Chloramines: ", not_potable_mean)
print("Potable mean Chloramines: ", potable_mean)
print("Not Potable Chloramines standard deviation: ",not_potable_std)
print("Potable Chloramines standard deviation: ",potable_std)
print("P_value: ",p_value)
if p_value < significance_level:
  print("There is significant difference bettern mean of potable and not potable water")
else:
  print("There is no significant difference between mean of potable and not potable water")
```

```
Not Potable mean Chloramines:  7.107266894602623
Potable mean Chloramines:  7.174394917458128
Not Potable Chloramines standard deviation:  1.4759612371909365
Potable Chloramines standard deviation:  1.7317268825652565
P_value:  0.3659556745832798
There is no significant difference between mean of potable and not potable water
```

```
not_potable_mean = np.mean(not_potable['Sulfate'])
potable_mean = np.mean(potable['Sulfate'])
not_potable_std = np.std(not_potable['Sulfate'])
potable_std = np.std(potable['Sulfate'])
standard_error = np.sqrt(not_potable_std**2/len(not_potable) + potable_std**2/len(potable))
z_score = (not_potable_mean - potable_mean) / standard_error
p_value = 2 * norm.cdf(-np.abs(z_score))
significance_level = 0.05
print("Not Potable mean Sulfate: ", not_potable_mean)
print("Potable mean Sulfate: ", potable_mean)
print("Not Potable Sulfate standard deviation: ",not_potable_std)
print("Potable Sulfate standard deviation: ",potable_std)
print("P_value: ",p_value)
if p_value < significance_level:
  print("There is significant difference bettern mean of potable and not potable water")
else:
  print("There is no significant difference between mean of potable and not potable water")
```

```
Not Potable mean Sulfate:  333.742927889492
Potable mean Sulfate:  332.4578319377389
Not Potable Sulfate standard deviation:  36.38323430796971
Potable Sulfate standard deviation:  47.416929377399825
P_value:  0.5138908879837127
There is no significant difference between mean of potable and not potable water
```

```
not_potable_mean = np.mean(not_potable['Conductivity'])
potable_mean = np.mean(potable['Conductivity'])
not_potable_std = np.std(not_potable['Conductivity'])
potable_std = np.std(potable['Conductivity'])
standard_error = np.sqrt(not_potable_std**2/len(not_potable) + potable_std**2/len(potable))
z_score = (not_potable_mean - potable_mean) / standard_error
p_value = 2 * norm.cdf(-np.abs(z_score))
significance_level = 0.05
print("Not Potable mean Conductivity: ", not_potable_mean)
print("Potable mean Conductivity: ", potable_mean)
print("Not Potable Conductivity standard deviation: ",not_potable_std)
print("Potable Conductivity standard deviation: ",potable_std)
print("P_value: ",p_value)
if p_value < significance_level:
  print("There is significant difference bettern mean of potable and not potable water")
else:
  print("There is no significant difference between mean of potable and not potable water")
```

```
Not Potable mean Conductivity:  427.55434176504036
Potable mean Conductivity:  425.0054227392821
Not Potable Conductivity standard deviation:  79.84938596132312
Potable Conductivity standard deviation:  81.9004419192641
P_value:  0.4892037268953736
There is no significant difference between mean of potable and not potable water
```

```
not_potable_mean = np.mean(not_potable['Organic_carbon'])
potable_mean = np.mean(potable['Organic_carbon'])
not_potable_std = np.std(not_potable['Organic_carbon'])
potable_std = np.std(potable['Organic_carbon'])
standard_error = np.sqrt(not_potable_std**2/len(not_potable) + potable_std**2/len(potable))
z_score = (not_potable_mean - potable_mean) / standard_error
p_value = 2 * norm.cdf(-np.abs(z_score))
significance_level = 0.05
print("Not Potable mean Organic_carbon: ", not_potable_mean)
print("Potable mean Organic_carbon: ", potable_mean)
print("Not Potable Organic_carbon standard deviation: ",not_potable_std)
print("Potable Organic_carbon standard deviation: ",potable_std)
print("P_value: ",p_value)
if p_value < significance_level:
  print("There is significant difference bettern mean of potable and not potable water")
else:
  print("There is no significant difference between mean of potable and not potable water")
```

```
Not Potable mean Organic_carbon:  14.400250029135737
Potable mean Organic_carbon:  14.294763978633684
Not Potable Organic_carbon standard deviation:  3.368791561047154
Potable Organic_carbon standard deviation:  3.255908191346215
P_value:  0.4821855899430869
There is no significant difference between mean of potable and not potable water
```

```
not_potable_mean = np.mean(not_potable['Trihalomethanes'])
potable_mean = np.mean(potable['Trihalomethanes'])
not_potable_std = np.std(not_potable['Trihalomethanes'])
potable_std = np.std(potable['Trihalomethanes'])
standard_error = np.sqrt(not_potable_std**2/len(not_potable) + potable_std**2/len(potable))
z_score = (not_potable_mean - potable_mean) / standard_error
p_value = 2 * norm.cdf(-np.abs(z_score))
significance_level = 0.05
print("Not Potable mean Trihalomethanes: ", not_potable_mean)
print("Potable mean Trihalomethanes: ", potable_mean)
print("Not Potable Trihalomethanes standard deviation: ",not_potable_std)
print("Potable Trihalomethanes standard deviation: ",potable_std)
print("P_value: ",p_value)
if p_value < significance_level:
  print("There is significant difference bettern mean of potable and not potable water")
else:
  print("There is no significant difference between mean of potable and not potable water")
```

```
Not Potable mean Trihalomethanes:  66.27871178932854
Potable mean Trihalomethanes:  66.58159561081533
Not Potable Trihalomethanes standard deviation:  15.925313389393967
Potable Trihalomethanes standard deviation:  16.287661826403898
P_value:  0.6797808091128006
There is no significant difference between mean of potable and not potable water
```

```
not_potable_mean = np.mean(not_potable['Turbidity'])
potable_mean = np.mean(potable['Turbidity'])
not_potable_std = np.std(not_potable['Turbidity'])
potable_std = np.std(potable['Turbidity'])
standard_error = np.sqrt(not_potable_std**2/len(not_potable) + potable_std**2/len(potable))
z_score = (not_potable_mean - potable_mean) / standard_error
p_value = 2 * norm.cdf(-np.abs(z_score))
significance_level = 0.05
print("Not Potable mean Turbidity: ", not_potable_mean)
print("Potable mean Turbidity: ", potable_mean)
print("Not Potable Turbidity standard deviation: ",not_potable_std)
print("Potable Turbidity standard deviation: ",potable_std)
print("P_value: ",p_value)
if p_value < significance_level:
  print("There is significant difference bettern mean of potable and not potable water")
else:
  print("There is no significant difference between mean of potable and not potable water")
```

```
Not Potable mean Turbidity:  3.9551813104832747
Potable mean Turbidity:  3.9912540600698114
Not Potable Turbidity standard deviation:  0.78265817302107
Potable Turbidity standard deviation:  0.7759289851238195
P_value:  0.3081384336085471
There is no significant difference between mean of potable and not potable water
```

```
import numpy as np
import scipy
from scipy.stats import f_oneway
ph_potable=potable['ph']
ph_not_potable=not_potable['ph']
# Perform one-way ANOVA
f_stat_ph, p_value_ph = scipy.stats.f_oneway(ph_potable,ph_not_potable)
#Print the results
print("F-statistic:", f_stat)
print("P-value:", p_value)
# Set significance level (e.g., 0.05)
alpha = 0.05
# Compare the p-value to the significance level to make a decision
if p_value_ph < alpha:
    print("Reject the null hypothesis: There is a statistically significant difference among the means of the ph based on the ANOVA test
else:
    print("Fail to reject the null hypothesis: There is no statistically significant difference among the means of the ph based on the A
```

```
F-statistic: 12815.173534180929
P-value: 0.554714033807364
Fail to reject the null hypothesis: There is no statistically significant difference among the means of the ph based on the ANOVA te
```

```
import numpy as np
import scipy
from scipy.stats import f_oneway
Solids_potable=potable['Solids']
Solids_not_potable=not_potable['Solids']
# Perform one-way ANOVA
f_stat_Solids, p_value_Solids = scipy.stats.f_oneway(Solids_potable,Solids_not_potable)
#Print the results
print("F-statistic:", f_stat_Solids)
print("P-value:", p_value_Solids)
# Set significance level (e.g., 0.05)
alpha = 0.05
# Compare the p-value to the significance level to make a decision
if p_value_Solids < alpha:
    print("Reject the null hypothesis: There is a statistically significant difference among the means of the Solids based on the ANOVA
else:
    print("Fail to reject the null hypothesis: There is no statistically significant difference among the means of the Solids based on t
```

```
F-statistic: 3.3291754630439674
P-value: 0.06820943709924715
Fail to reject the null hypothesis: There is no statistically significant difference among the means of the Solids based on the ANOV
```

```
import numpy as np
import scipy
from scipy.stats import f_oneway
Hardness_potable=potable['Hardness']
Hardness_not_potable=not_potable['Hardness']
# Perform one-way ANOVA
f_stat_Hardness, p_value_Hardness = scipy.stats.f_oneway(Hardness_potable,Hardness_not_potable)
#Print the results
print("F-statistic:", f_stat_Hardness)
print("P-value:", p_value_Hardness)
# Set significance level (e.g., 0.05)
alpha = 0.05
# Compare the p-value to the significance level to make a decision
if p_value_Hardness < alpha:
    print("Reject the null hypothesis: There is a statistically significant difference among the means of the Hardness based on the ANOV
else:
    print("Fail to reject the null hypothesis: There is no statistically significant difference among the means of the Hardness based or
```

```
F-statistic: 0.004550569085124058
P-value: 0.9462238843630211
Fail to reject the null hypothesis: There is no statistically significant difference among the means of the Hardness based on the AN
```

```
import numpy as np
import scipy
from scipy.stats import f_oneway
Chloramines_potable=potable['Chloramines']
Chloramines_not_potable=not_potable['Chloramines']
# Perform one-way ANOVA
f_stat_Chloramines, p_value_Chloramines = scipy.stats.f_oneway(Chloramines_potable,Chloramines_not_potable)
#Print the results
print("F-statistic:", f_stat_Chloramines)
print("P-value:", p_value_Chloramines)
# Set significance level (e.g., 0.05)
alpha = 0.05
# Compare the p-value to the significance level to make a decision
if p_value_Chloramines < alpha:
    print("Reject the null hypothesis: There is a statistically significant difference among the means of the Chloramines based on the A
else:
    print("Fail to reject the null hypothesis: There is no statistically significant difference among the means of the Chloramines based
```

```
F-statistic: 0.8681793208518837
P-value: 0.3515717996483123
Fail to reject the null hypothesis: There is no statistically significant difference among the means of the Chloramines based on the
```

```
import numpy as np
import scipy
from scipy.stats import f_oneway
Sulfate_potable=potable['Sulfate']
Sulfate_not_potable=not_potable['Sulfate']
# Perform one-way ANOVA
f_stat_Sulfate, p_value_Sulfate = scipy.stats.f_oneway(Sulfate_potable,Sulfate_not_potable)
#Print the results
print("F-statistic:", f_stat_Sulfate)
print("P-value:", p_value_Sulfate)
# Set significance level (e.g., 0.05)
alpha = 0.05
# Compare the p-value to the significance level to make a decision
if p_value_Sulfate < alpha:
    print("Reject the null hypothesis: There is a statistically significant difference among the means of the Sulfate based on the ANOVA
else:
    print("Fail to reject the null hypothesis: There is no statistically significant difference among the means of the Sulfate based on
```

```
F-statistic: 0.47059062857385214
P-value: 0.4927947335655132
Fail to reject the null hypothesis: There is no statistically significant difference among the means of the Sulfate based on the ANO
```

```
import numpy as np
import scipy
from scipy.stats import f_oneway
Conductivity_potable=potable['Conductivity']
Conductivity_not_potable=not_potable['Conductivity']
# Perform one-way ANOVA
f_stat_Conductivity, p_value_Conductivity = scipy.stats.f_oneway(Conductivity_potable,Conductivity_not_potable)
#Print the results
print("F-statistic:", f_stat_Conductivity)
print("P-value:", p_value_Conductivity)
# Set significance level (e.g., 0.05)
alpha = 0.05
# Compare the p-value to the significance level to make a decision
if p_value_Conductivity < alpha:
    print("Reject the null hypothesis: There is a statistically significant difference among the means of the Conductivity based on the
else:
    print("Fail to reject the null hypothesis: There is no statistically significant difference among the means of the Conductivity base
```

```
F-statistic: 0.482511760520502
P-value: 0.48736711195012805
Fail to reject the null hypothesis: There is no statistically significant difference among the means of the Conductivity based on th
```

```
import numpy as np
import scipy
from scipy.stats import f_oneway
Organic_carbon_potable=potable['Organic_carbon']
Organic_carbon_not_potable=not_potable['Organic_carbon']
# Perform one-way ANOVA
f_stat_Organic_carbon, p_value_Organic_carbon = scipy.stats.f_oneway(Organic_carbon_potable,Organic_carbon_not_potable)
#Print the results
print("F-statistic:", f_stat_Organic_carbon)
print("P-value:", p_value_Organic_carbon)
# Set significance level (e.g., 0.05)
alpha = 0.05
# Compare the p-value to the significance level to make a decision
if p_value_Organic_carbon < alpha:
    print("Reject the null hypothesis: There is a statistically significant difference among the means of the Organic_carbon based on th
else:
    print("Fail to reject the null hypothesis: There is no statistically significant difference among the means of the Organic_carbon ba
```

```
F-statistic: 0.4869638587459594
P-value: 0.4853655467233867
Fail to reject the null hypothesis: There is no statistically significant difference among the means of the Organic_carbon based on
```

```
import numpy as np
import scipy
from scipy.stats import f_oneway
Trihalomethanes_potable=potable['Trihalomethanes']
Trihalomethanes_not_potable=not_potable['Trihalomethanes']
# Perform one-way ANOVA
f_stat_Trihalomethanes, p_value_Trihalomethanes = scipy.stats.f_oneway(Trihalomethanes_potable,Trihalomethanes_not_potable)
#Print the results
print("F-statistic:", f_stat_Trihalomethanes)
print("P-value:", p_value_Trihalomethanes)
# Set significance level (e.g., 0.05)
alpha = 0.05
# Compare the p-value to the significance level to make a decision
if p_value_Trihalomethanes < alpha:
    print("Reject the null hypothesis: There is a statistically significant difference among the means of the Trihalomethanes based on t
else:
    print("Fail to reject the null hypothesis: There is no statistically significant difference among the means of the Trihalomethanes b
```

```
F-statistic: 0.17169088386987577
P-value: 0.6786576528829604
Fail to reject the null hypothesis: There is no statistically significant difference among the means of the Trihalomethanes based or
```

```
import numpy as np
import scipy
from scipy.stats import f_oneway
Turbidity_potable=potable['Turbidity']
Turbidity_not_potable=not_potable['Turbidity']
# Perform one-way ANOVA
f_stat_Turbidity, p_value_Turbidity = scipy.stats.f_oneway(Turbidity_potable,Turbidity_not_potable)
#Print the results
print("F-statistic:", f_stat_Turbidity)
print("P-value:", p_value_Turbidity)
# Set significance level (e.g., 0.05)
alpha = 0.05
# Compare the p-value to the significance level to make a decision
if p_value_Turbidity < alpha:
    print("Reject the null hypothesis: There is a statistically significant difference among the means of the Turbidity based on the ANO
else:
    print("Fail to reject the null hypothesis: There is no statistically significant difference among the means of the Turbidity based o
```

```
F-statistic: 1.0341446923242343
P-value: 0.3093096320238314
Fail to reject the null hypothesis: There is no statistically significant difference among the means of the Turbidity based on the A
```

```
tukey_ph = pairwise_tukeyhsd(df['ph'],df['Potability'],
alpha=0.05)

print(tukey_ph)
```

```
Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
group1 group2 meandiff p-adj   lower  upper  reject
-----------------------------------------------------
     0      1   0.0466 0.5149 -0.0937 0.1869  False
-----------------------------------------------------
```

```
tukey_Hardness = pairwise_tukeyhsd(endog=df['Hardness'], groups=df['Potability'],
alpha=0.05)

print(tukey_Hardness)
```

```
Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
group1 group2 meandiff p-adj   lower  upper reject
-----------------------------------------------------
     0      1  -0.1001 0.9462 -3.0102  2.81  False
-----------------------------------------------------
```

```
tukey_Solids = pairwise_tukeyhsd(endog=df['Solids'], groups=df['Potability'],
alpha=0.05)

print(tukey_Solids)
```

```
  Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
group1 group2 meandiff p-adj   lower     upper   reject
-----------------------------------------------------
     0      1 716.3878 0.0682 -53.6103 1486.3858  False
-----------------------------------------------------
```

```
tukey_Chloramines = pairwise_tukeyhsd(endog=df['Chloramines'], groups=df['Potability'],
alpha=0.05)

print(tukey_Chloramines)
```

```
     Multiple Comparison of Means - Tukey HSD, FWER=0.05
     ===================================================
     group1 group2 meandiff p-adj   lower  upper  reject
     ---------------------------------------------------
          0      1   0.0671 0.3516 -0.0742 0.2084  False
     ---------------------------------------------------
```

```
tukey_Sulfate = pairwise_tukeyhsd(endog=df['Sulfate'], groups=df['Potability'],
alpha=0.05)

print(tukey_Sulfate)
```

```
     Multiple Comparison of Means - Tukey HSD, FWER=0.05
     ===================================================
     group1 group2 meandiff p-adj  lower  upper  reject
     ---------------------------------------------------
          0      1  -1.2851 0.4928 -4.959 2.3888  False
     ---------------------------------------------------
```

```
tukey_Conductivity = pairwise_tukeyhsd(endog=df['Conductivity'], groups=df['Potability'],
alpha=0.05)

print(tukey_Conductivity)
```

```
     Multiple Comparison of Means - Tukey HSD, FWER=0.05
     ===================================================
     group1 group2 meandiff p-adj   lower  upper  reject
     ---------------------------------------------------
          0      1  -2.5489 0.4874 -9.7453 4.6474  False
     ---------------------------------------------------
```

```
tukey_Organic_carbon = pairwise_tukeyhsd(endog=df['Organic_carbon'], groups=df['Potability'],
alpha=0.05)

print(tukey_Organic_carbon)
```

```
     Multiple Comparison of Means - Tukey HSD, FWER=0.05
     ===================================================
     group1 group2 meandiff p-adj   lower  upper reject
     ---------------------------------------------------
          0      1  -0.1055 0.4854 -0.4019 0.191  False
     ---------------------------------------------------
```

```
tukey_Trihalomethanes = pairwise_tukeyhsd(endog=df['Trihalomethanes'], groups=df['Potability'],
alpha=0.05)

print(tukey_Trihalomethanes)
```

```
     Multiple Comparison of Means - Tukey HSD, FWER=0.05
     ===================================================
     group1 group2 meandiff p-adj   lower  upper  reject
     ---------------------------------------------------
          0      1   0.3029 0.6787 -1.1307 1.7364  False
     ---------------------------------------------------
```

```
tukey_Turbidity = pairwise_tukeyhsd(endog=df['Turbidity'], groups=df['Potability'],
alpha=0.05)

print(tukey_Turbidity)
```

```
     Multiple Comparison of Means - Tukey HSD, FWER=0.05
     ===================================================
     group1 group2 meandiff p-adj   lower  upper  reject
     ---------------------------------------------------
          0      1   0.0361 0.3093 -0.0335 0.1056  False
     ---------------------------------------------------
```

Double-click (or enter) to edit

```
median_maj_axis_len = df["ph"].median()
df["Group"] = pd.cut(df["ph"], bins=[0, median_maj_axis_len, float("inf")], labels=["<= Median", "> Median"])

le = LabelEncoder()

# Fit and transform the Class column with the LabelEncoder
df['Group10'] = le.fit_transform(df['Group'])

# Print the updated dataframe
df.head()

# Create a contingency table of the "Potability" and "Group" attributes
contingency_table = pd.crosstab(df["Potability"], df["Group"])

# Perform Fisher's exact test
odds_ratio, p_value = fisher_exact(contingency_table)

print(contingency_table)

# Print the results
print("Odds ratio:", odds_ratio)
print("P-value:", p_value)
```

```
Group       <= Median  > Median
Potability
0                 607       593
1                 399       412
Odds ratio: 1.0569594306170147
P-value: 0.554714033807364
```

```
df
```

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Orgar
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	
5	5.584087	188.313324	28748.687739	7.544869	326.678363	280.467916	
6	10.223862	248.071735	28749.716544	7.513408	393.663396	283.651634	
7	8.635849	203.361523	13672.091764	4.563009	303.309771	474.607645	
...	...	...	...	...	...	...	
3267	8.989900	215.047358	15921.412018	6.297312	312.931022	390.410231	
3268	6.702547	207.321086	17246.920347	7.708117	304.510230	329.266002	

```
# Define the response variable and the predictor variables
df=df.dropna()
y = df['Potability']
X = df[['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'Conductivity', 'Organic_carbon','Trihalomethanes','Turbidity']]
print(X.shape)
X=X.dropna()
print(X.shape)

# Define the logistic regression model
lr = LogisticRegression(max_iter=1000)

# Perform forward selection
sfs_forward = SequentialFeatureSelector(lr, direction='forward', n_features_to_select=5)
sfs_forward.fit(X, y)
selected_features_forward = X.columns[sfs_forward.get_support()]

# Perform backward selection
sfs_backward = SequentialFeatureSelector(lr, direction='backward', n_features_to_select=4)
sfs_backward.fit(X, y)
selected_features_backward = X.columns[sfs_backward.get_support()]

# Print the selected features for both methods
print("Selected features (forward):", selected_features_forward)
print("Selected features (backward):", selected_features_backward)
```

```
(2011, 9)
(2011, 9)
Selected features (forward): Index(['ph', 'Chloramines', 'Sulfate', 'Conductivity', 'Organic_carbon'], dtype='object')
Selected features (backward): Index(['ph', 'Chloramines', 'Organic_carbon', 'Turbidity'], dtype='object')
```

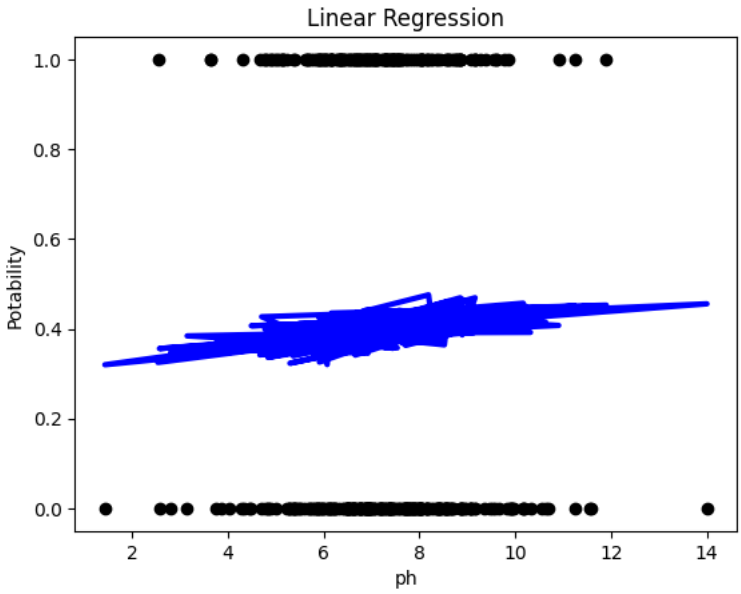
```
X = df[['ph','Chloramines','Sulfate','Conductivity','Organic_carbon']]
y = df['Potability']

LR = LogisticRegression()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
LR.fit(X_train,y_train)

y_pred = LR.predict(X_test)

accuracy = accuracy_score(y_test,y_pred)
print(accuracy)
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
0.5910596026490066
Confusion Matrix:
 [[355   0]
 [247   2]]
```

```
X = df[['ph','Chloramines','Sulfate','Conductivity','Organic_carbon']]
y = df['Potability']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
lr = LinearRegression()

# Fit the model to the training data
lr.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = lr.predict(X_test)

# Evaluate the performance of the model using mean squared error and R-squared
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)


print('Mean Squared Error:', mse)
print('R-squared:', r2)
```

```
Mean Squared Error: 0.2478945456844096
R-squared: -0.013296719773967958
```

```
plt.scatter(X_test['ph'], y_test, color='black')
plt.plot(X_test['ph'], y_pred, color='blue', linewidth=3)
plt.title('Linear Regression')
plt.xlabel('ph')
plt.ylabel('Potability')
plt.show()
```



```
X = df[['ph','Chloramines','Sulfate','Conductivity','Organic_carbon']]
y = df['Potability']

# Define the number of folds
K = 7
kf = KFold(n_splits=K, shuffle=True, random_state=42)

# Initialize the lists to store the scores and confusion matrices
scores = []


# Loop over the folds
for train_index, test_index in kf.split(X):
    # Split the data into training and testing sets for this fold
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    # Fit a logistic regression model on the training data for this fold
    model = LogisticRegression()
    model.fit(X_train, y_train)

    # Predict the labels of the test data using the trained model
    y_pred = model.predict(X_test)

    # Calculate the accuracy score and confusion matrix for this fold
    score = accuracy_score(y_test, y_pred)
    print(score)

    # Store the score and confusion matrix for this fold
    scores.append(score)


# Print the average score and confusion matrix across all folds
print("Average score:", np.mean(scores))
```

```
0.6006944444444444
0.5625
0.6202090592334495
0.6480836236933798
0.5365853658536586
0.5958188153310104
0.6132404181184669
Average score: 0.5967331038106299
```