# Phase-1 Machine Learning Project – Water Quality Assessment – Rushikesh Reddy(Group-7)

## Data Loading

In [20]:

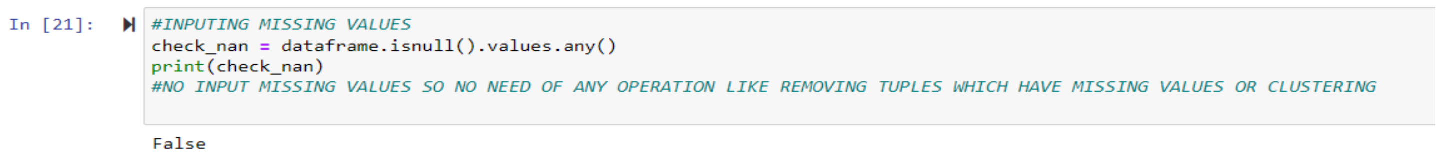
```python
#Importing data
import pandas as p
import numpy as n
from pandas import read_csv as csv
url="D:\\semister 5\\machine learning\\PROJECT\\4.csv"
data=csv(url)
arr=n.array(data)
dataframe=p.DataFrame(arr)
print(dataframe)
list(data.columns)
```

```
         0      1     2     3      4     5     6     7     8     9  ...  \
0     1.65   9.08  0.04  2.85  0.007  0.35  0.83  0.17  0.05  0.20  ...
1     2.32  21.16  0.01  3.31  0.002  5.28  0.68  0.66  0.90  0.65  ...
2     1.01  14.02  0.04  0.58  0.008  4.24  0.53  0.02  0.99  0.05  ...
3     1.36  11.33  0.04  2.96  0.001  7.23  0.03  1.66  1.08  0.71  ...
4     0.92  24.33  0.03  0.20  0.006  2.67  0.69  0.57  0.61  0.13  ...
...    ...    ...   ...   ...    ...   ...   ...   ...   ...   ...  ...
7994  0.05   7.78  0.00  1.95  0.040  0.10  0.03  0.03  1.37  0.00  ...
7995  0.05  24.22  0.02  0.59  0.010  0.45  0.02  0.02  1.48  0.00  ...
7996  0.09   6.85  0.00  0.61  0.030  0.05  0.05  0.02  0.91  0.00  ...
7997  0.01  10.00  0.01  2.00  0.000  2.00  0.00  0.09  0.00  0.00  ...
7998  0.04   6.85  0.01  0.70  0.030  0.05  0.01  0.03  1.00  0.00  ...

         11     12    13     14     15    16    17    18    19   20
0     0.054  16.08  1.13  0.007  37.75  6.78  0.08  0.34  0.02  1.0
1     0.100   2.01  1.93  0.003  32.26  3.21  0.08  0.27  0.05  1.0
2     0.078  14.16  1.11  0.006  50.28  7.07  0.07  0.44  0.01  0.0
3     0.016   1.41  1.29  0.004   9.12  1.72  0.02  0.45  0.05  0.0
4     0.117   6.74  1.11  0.003  16.90  2.41  0.02  0.06  0.02  1.0
...     ...    ...   ...    ...    ...   ...   ...   ...   ...  ...
7994  0.197  14.29  1.00  0.005   3.57  2.13  0.09  0.06  0.03  1.0
7995  0.031  10.27  1.00  0.001   1.48  1.11  0.09  0.10  0.08  1.0
7996  0.182  15.92  1.00  0.000   1.35  4.84  0.00  0.04  0.05  1.0
7997  0.000   0.00  0.00  0.000   0.00  0.00  0.00  0.00  0.00  1.0
7998  0.182  15.92  1.00  0.000   1.35  4.84  0.00  0.04  0.05  1.0

[7999 rows x 21 columns]
```

Out[20]:

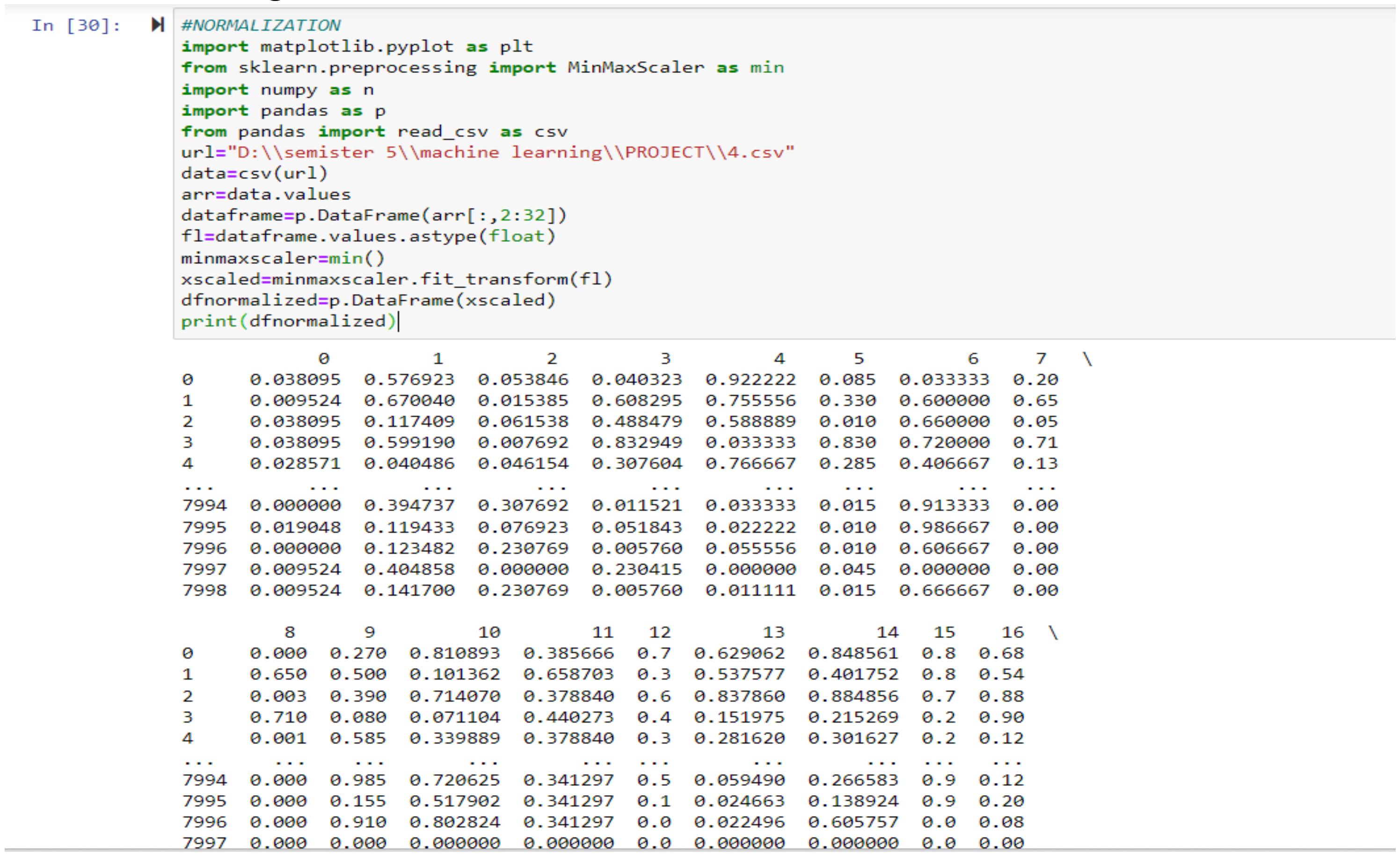
```
['aluminium',
 'ammonia',
 'arsenic',
 'barium',
 'cadmium',
 'chloramine',
 'chromium',
 'copper',
 'flouride',
 'bacteria',
 'viruses',
 'lead',
 'nitrates',
 'nitrites',
 'mercury',
 'perchlorate',
 'radium',
 'selenium',
 'silver',
 'uranium',
 'is_safe']
```

## Data Transformation

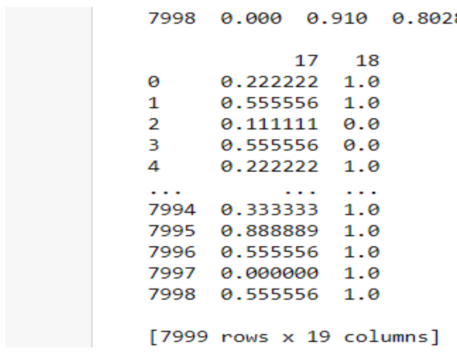


```
In [21]:  #INPUTING MISSING VALUES
          check_nan = dataframe.isnull().values.any()
          print(check_nan)
          #NO INPUT MISSING VALUES SO NO NEED OF ANY OPERATION LIKE REMOVING TUPLES WHICH HAVE MISSING VALUES OR CLUSTERING

          False
```

## Normalzing The Data Set



```
In [30]:  #NORMALIZATION
          import matplotlib.pyplot as plt
          from sklearn.preprocessing import MinMaxScaler as min
          import numpy as n
          import pandas as p
          from pandas import read_csv as csv
          url="D:\\semister 5\\machine learning\\PROJECT\\4.csv"
          data=csv(url)
          arr=data.values
          dataframe=p.DataFrame(arr[:,2:32])
          fl=dataframe.values.astype(float)
          minmaxscaler=min()
          xscaled=minmaxscaler.fit_transform(fl)
          dfnormalized=p.DataFrame(xscaled)
          print(dfnormalized)

                 0         1         2         3         4      5         6     7  \
0     0.038095  0.576923  0.053846  0.040323  0.922222  0.085  0.033333  0.20
1     0.009524  0.670040  0.015385  0.608295  0.755556  0.330  0.600000  0.65
2     0.038095  0.117409  0.061538  0.488479  0.588889  0.010  0.660000  0.05
3     0.038095  0.599190  0.007692  0.832949  0.033333  0.830  0.720000  0.71
4     0.028571  0.040486  0.046154  0.307604  0.766667  0.285  0.406667  0.13
...        ...       ...       ...       ...       ...    ...       ...   ...
7994  0.000000  0.394737  0.307692  0.011521  0.033333  0.015  0.913333  0.00
7995  0.019048  0.119433  0.076923  0.051843  0.022222  0.010  0.986667  0.00
7996  0.000000  0.123482  0.230769  0.005760  0.055556  0.010  0.606667  0.00
7997  0.009524  0.404858  0.000000  0.230415  0.000000  0.045  0.000000  0.00
7998  0.009524  0.141700  0.230769  0.005760  0.011111  0.015  0.666667  0.00

          8      9        10        11   12        13        14   15    16  \
0     0.000  0.270  0.810893  0.385666  0.7  0.629062  0.848561  0.8  0.68
1     0.650  0.500  0.101362  0.658703  0.3  0.537577  0.401752  0.8  0.54
2     0.003  0.390  0.714070  0.378840  0.6  0.837860  0.884856  0.7  0.88
3     0.710  0.080  0.071104  0.440273  0.4  0.151975  0.215269  0.2  0.90
4     0.001  0.585  0.339889  0.378840  0.3  0.281620  0.301627  0.2  0.12
...     ...    ...       ...       ...  ...       ...       ...  ...   ...
7994  0.000  0.985  0.720625  0.341297  0.5  0.059490  0.266583  0.9  0.12
7995  0.000  0.155  0.517902  0.341297  0.1  0.024663  0.138924  0.9  0.20
7996  0.000  0.910  0.802824  0.341297  0.0  0.022496  0.605757  0.0  0.08
7997  0.000  0.000  0.000000  0.000000  0.0  0.000000  0.000000  0.0  0.00
```



```
7998  0.000  0.910  0.802824  0.341297  0.0  0.022496  0.605757  0.0  0.08

            17   18
0     0.222222  1.0
1     0.555556  1.0
2     0.111111  0.0
3     0.555556  0.0
4     0.222222  1.0
...        ...  ...
7994  0.333333  1.0
7995  0.888889  1.0
7996  0.555556  1.0
7997  0.000000  1.0
7998  0.555556  1.0

[7999 rows x 19 columns]
```

## Standardizing The DataSet

```
In [29]:
#STANDARDIZATION
from sklearn.preprocessing import StandardScaler
import numpy as n
import pandas as p
from pandas import read_csv as csv
url="D:\\semister 5\\machine learning\\PROJECT\\4.csv"
data=csv(url)
arr=data.values
X=arr[:,2:32]
Y=arr[:,8]
scaler=StandardScaler().fit(X)
rescaledX=scaler.transform(X)
print(rescaledX[0:2,:])
print(X[0:2,:])
```

```
[[-0.48082883  1.05449775 -0.99331288 -0.71169675  2.1534498  -0.97300495
  -1.65745451 -0.36321006 -0.86910037 -0.78134877  1.12997554 -0.34886073
   0.6088263   1.203735    1.66150641  1.05377233  1.33911238 -0.91713868
   2.78934778]
 [-0.59960559  1.4327825  -1.13202009  1.20893261  1.59917419 -0.22319416
   0.29501817  1.00264114  0.85014726  0.00946081 -1.40928413  1.04685264
  -0.73936432  0.89332646  0.12461018  1.05377233  0.85145064  0.19801013
   2.78934778]]
[[4.000e-02 2.850e+00 7.000e-03 3.500e-01 8.300e-01 1.700e-01 5.000e-02
  2.000e-01 0.000e+00 5.400e-02 1.608e+01 1.130e+00 7.000e-03 3.775e+01
  6.780e+00 8.000e-02 3.400e-01 2.000e-02 1.000e+00]
 [1.000e-02 3.310e+00 2.000e-03 5.280e+00 6.800e-01 6.600e-01 9.000e-01
  6.500e-01 6.500e-01 1.000e-01 2.010e+00 1.930e+00 3.000e-03 3.226e+01
  3.210e+00 8.000e-02 2.700e-01 5.000e-02 1.000e+00]]
```

## Data Summarization

```
In [31]:
#DATA SUMMARIZATION
description=data.describe()
print(description)
print(data.shape)
```

```
         aluminium      ammonia      arsenic       barium      cadmium  \
count  7999.000000  7999.000000  7999.000000  7999.000000  7999.000000
mean      0.666158    14.274201     0.161445     1.567715     0.042806
std       1.265145     8.879813     0.252590     1.216091     0.036049
min       0.000000    -0.080000     0.000000     0.000000     0.000000
25%       0.040000     6.560000     0.030000     0.560000     0.008000
50%       0.070000    14.130000     0.050000     1.190000     0.040000
75%       0.280000    22.130000     0.100000     2.480000     0.070000
max       5.050000    29.840000     1.050000     4.940000     0.130000

        chloramine     chromium       copper     flouride     bacteria  ...  \
count  7999.000000  7999.000000  7999.000000  7999.000000  7999.000000  ...
mean      2.176831     0.247226     0.805857     0.771565     0.319665  ...
std       2.567027     0.270640     0.653539     0.435373     0.329485  ...
min       0.000000     0.000000     0.000000     0.000000     0.000000  ...
25%       0.100000     0.050000     0.090000     0.405000     0.000000  ...
50%       0.530000     0.090000     0.750000     0.770000     0.220000  ...
75%       4.240000     0.440000     1.390000     1.160000     0.610000  ...
max       8.680000     0.900000     2.000000     1.500000     1.000000  ...

              lead     nitrates     nitrites      mercury  perchlorate  \
count  7999.000000  7999.000000  7999.000000  7999.000000  7999.000000
mean      0.099450     9.818822     1.329961     0.005194    16.460299
std       0.058172     5.541331     0.573219     0.002967    17.687474
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.048000     5.000000     1.000000     0.003000     2.170000
50%       0.102000     9.930000     1.420000     0.005000     7.740000
75%       0.151000    14.610000     1.760000     0.008000    29.480000
max       0.200000    19.830000     2.930000     0.010000    60.010000

            radium     selenium       silver      uranium      is_safe
count  7999.000000  7999.000000  7999.000000  7999.000000  7999.000000
mean      2.920548     0.049685     0.147781     0.044673     0.113889
std       2.323009     0.028770     0.143551     0.026904     0.317697
[illegible]
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.820000     0.020000     0.040000     0.020000     0.000000
50%       2.410000     0.050000     0.080000     0.050000     0.000000
75%       4.670000     0.070000     0.240000     0.070000     0.000000
max       7.990000     0.100000     0.500000     0.090000     1.000000

[8 rows x 21 columns]
(7999, 21)
```

## Correlation

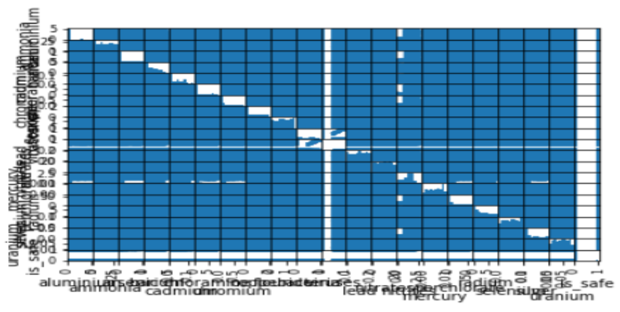
```
In [32]:  #data correlation
          data.corr()
```

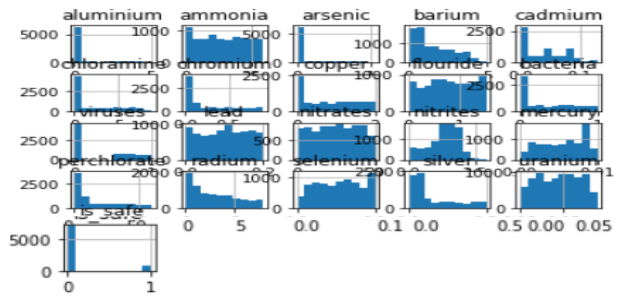
Out[32]:

| | aluminium | ammonia | arsenic | barium | cadmium | chloramine | chromium | copper | flouride | bacteria | ... | lead | nitrates | nitrit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **aluminium** | 1.000000 | 0.067572 | 0.225773 | 0.294145 | -0.099911 | 0.369309 | 0.353218 | 0.168612 | -0.009784 | -0.078238 | ... | 0.020792 | -0.003810 | 0.2373 |
| **ammonia** | 0.067572 | 1.000000 | 0.046920 | 0.070279 | -0.006586 | 0.105089 | 0.125068 | 0.016073 | -0.027949 | 0.063823 | ... | -0.037501 | 0.006619 | -0.0635 |
| **arsenic** | 0.225773 | 0.046920 | 1.000000 | 0.362945 | 0.334682 | 0.356559 | 0.312475 | -0.036444 | 0.003792 | 0.035688 | ... | -0.087756 | 0.027554 | 0.3050 |
| **barium** | 0.294145 | 0.070279 | 0.362945 | 1.000000 | -0.037803 | 0.446928 | 0.415972 | 0.065426 | -0.018548 | 0.101259 | ... | -0.042888 | -0.011331 | 0.3127 |
| **cadmium** | -0.099911 | -0.006586 | 0.334682 | -0.037803 | 1.000000 | -0.144370 | -0.157766 | -0.109024 | 0.004880 | -0.092431 | ... | -0.034959 | 0.020194 | -0.0156 |
| **chloramine** | 0.369309 | 0.105089 | 0.356559 | 0.446928 | -0.144370 | 1.000000 | 0.555938 | 0.119059 | 0.004400 | 0.154510 | ... | -0.030479 | -0.001551 | 0.3796 |
| **chromium** | 0.353218 | 0.125068 | 0.312475 | 0.415972 | -0.157766 | 0.555938 | 1.000000 | 0.113043 | -0.002284 | 0.142041 | ... | -0.050501 | -0.012793 | 0.3357 |
| **copper** | 0.168612 | 0.016073 | -0.036444 | 0.065426 | -0.109024 | 0.119059 | 0.113043 | 1.000000 | 0.011683 | 0.149110 | ... | 0.121765 | 0.002332 | 0.1620 |
| **flouride** | -0.009784 | -0.027949 | 0.003792 | -0.018548 | 0.004880 | 0.004400 | -0.002284 | 0.011683 | 1.000000 | 0.014134 | ... | 0.011905 | -0.008140 | -0.0166 |
| **bacteria** | -0.078238 | 0.063823 | 0.035688 | 0.101259 | -0.092431 | 0.154510 | 0.142041 | 0.149110 | 0.014134 | 1.000000 | ... | -0.027525 | -0.033920 | 0.2462 |
| **viruses** | -0.070863 | 0.106203 | 0.011703 | -0.002276 | 0.021183 | 0.003687 | 0.002430 | 0.006292 | 0.018418 | 0.618480 | ... | 0.017598 | -0.044544 | -0.0915 |
| **lead** | 0.020792 | -0.037501 | -0.087756 | -0.042888 | -0.034959 | -0.030479 | -0.050501 | 0.121765 | 0.011905 | -0.027525 | ... | 1.000000 | 0.034978 | -0.0524 |
| **nitrates** | -0.003810 | 0.006619 | 0.027554 | -0.011331 | 0.020194 | -0.001551 | -0.012793 | 0.002332 | -0.008140 | -0.033920 | ... | 0.034978 | 1.000000 | 0.0169 |
| **nitrites** | 0.237307 | -0.063519 | 0.305005 | 0.312711 | -0.015682 | 0.379685 | 0.335708 | 0.162093 | -0.016669 | 0.246252 | ... | -0.052405 | 0.016936 | 1.0000 |
| **mercury** | -0.003306 | 0.020476 | -0.015404 | 0.005987 | -0.016174 | -0.021472 | -0.022787 | 0.017626 | -0.004400 | -0.004471 | ... | -0.007832 | -0.020458 | -0.0167 |
| **perchlorate** | 0.363069 | 0.091246 | 0.332279 | 0.462234 | -0.149344 | 0.588769 | 0.524532 | 0.104564 | -0.016191 | 0.147652 | ... | -0.027709 | -0.014020 | 0.3461 |
| **radium** | 0.243217 | 0.050233 | 0.218204 | 0.286569 | -0.099259 | 0.388806 | 0.315271 | 0.026215 | 0.007688 | 0.099298 | ... | -0.048741 | -0.021410 | 0.2728 |
| **selenium** | -0.003672 | 0.029771 | -0.007009 | 0.035242 | 0.010145 | 0.011399 | 0.030539 | -0.003267 | 0.022629 | -0.006971 | ... | 0.031888 | 0.043109 | 0.0121 |
| **silver** | 0.334993 | 0.075777 | 0.307837 | 0.431606 | -0.155408 | 0.522447 | 0.510768 | 0.089333 | 0.014554 | 0.148225 | ... | -0.057351 | 0.005218 | 0.3327 |
| **uranium** | 0.014711 | 0.014554 | 0.001455 | -0.002440 | -0.005633 | -0.007658 | -0.005526 | 0.006978 | 0.016792 | 0.044839 | ... | -0.009151 | 0.000824 | -0.0099 |
| **is_safe** | 0.333961 | -0.022630 | -0.123181 | 0.090505 | -0.255672 | 0.186099 | 0.182784 | 0.029040 | 0.006340 | -0.022497 | ... | -0.009523 | -0.071503 | 0.0469 |

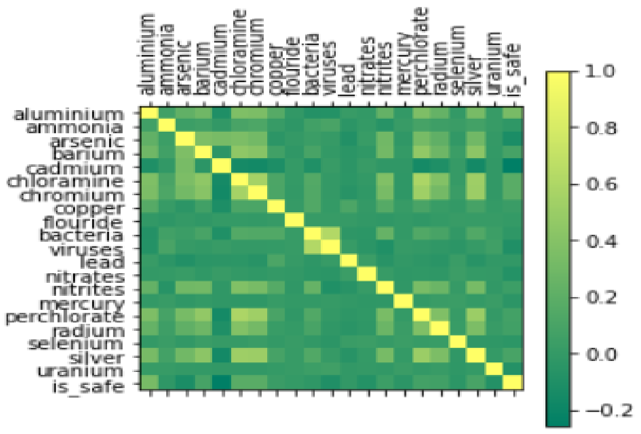
21 rows × 21 columns

```
#DATA SUMMARIZATION USING GRAPHS
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
scatter_matrix(data[['aluminium','ammonia','arsenic','barium','cadmium','chloramine','chromium','copper','flouride',
                     'bacteria','viruses','lead','nitrates','nitrites','mercury','perchlorate','radium','selenium',
                     'silver','uranium','is_safe']])
plt.show()
data.hist()
plt.show()
```

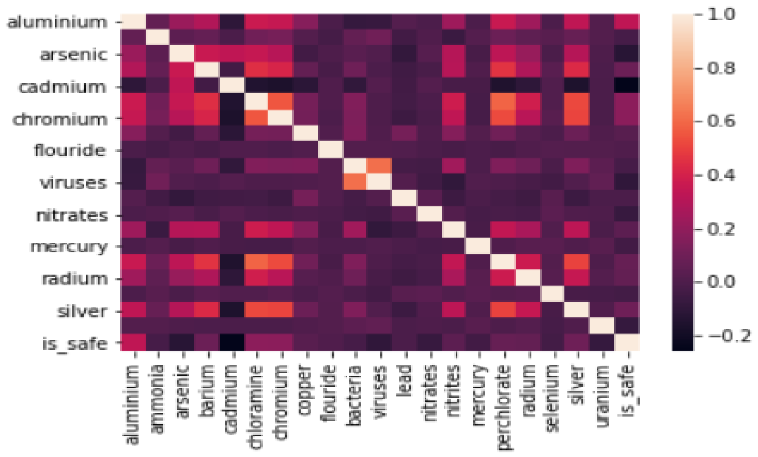




```python
#DATA SUMMARIZATION USING GRAPHS
import matplotlib.pylab as plt
%matplotlib inline
plt.matshow(data.corr(), cmap='summer')
plt.colorbar()
plt.xticks(list(range(len(data.columns))), data.columns, rotation ='vertical')
plt.yticks(list(range(len(data.columns))), data.columns, rotation ='horizontal')
plt.show()
```

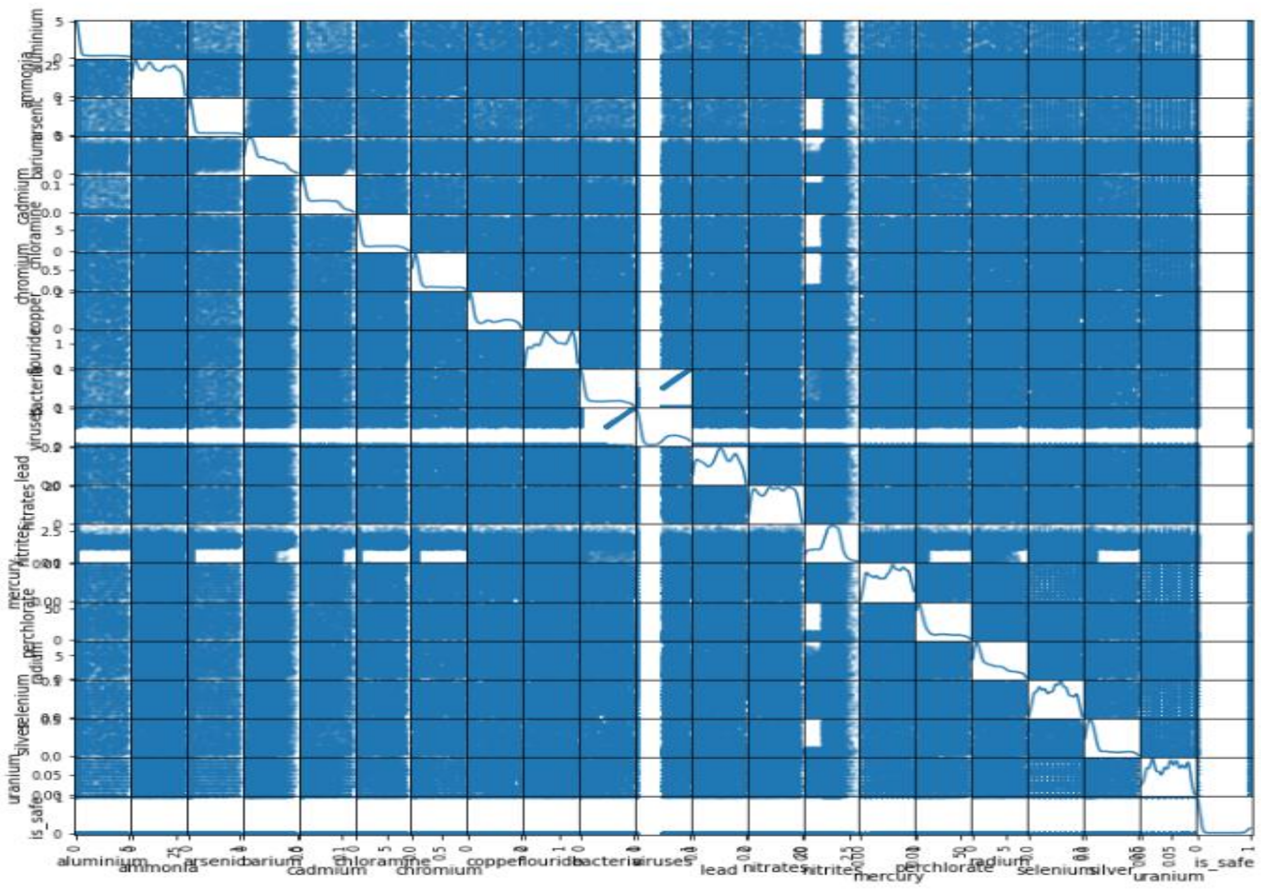


```python
#DATA SUMMARIZATION USING GRAPHS
correlations =data.corr()
import seaborn as sns
sns.heatmap(correlations)
plt.show()
```



## Covariance

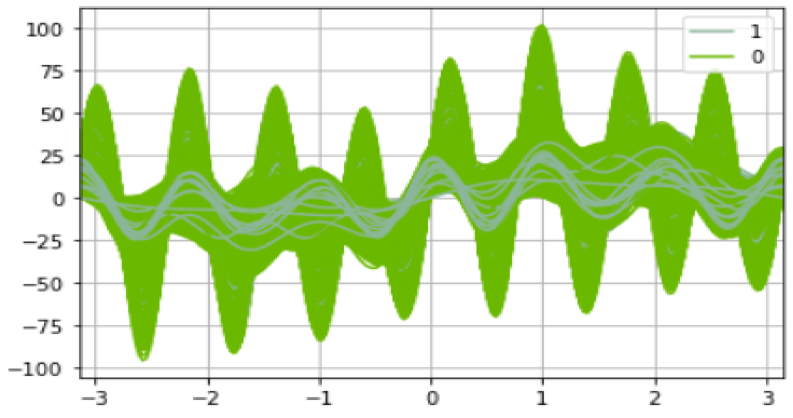
```
#DATA SUMMARIZATION(COVARIANCE) USING GRAPHS
p.plotting.scatter_matrix(data, alpha =0.2, figsize=(12,12), diagonal ='kde')
plt.show()
```



**Data Visualization**
**Andrews Curve For Target Value "is_safe"**

```
#DATA VISUALIZAION USING GRAPHS(Andrews Curve For Target Value "is_safe")
from pandas.plotting import andrews_curves
andrews_curves(data,'is_safe')
plt.show
```

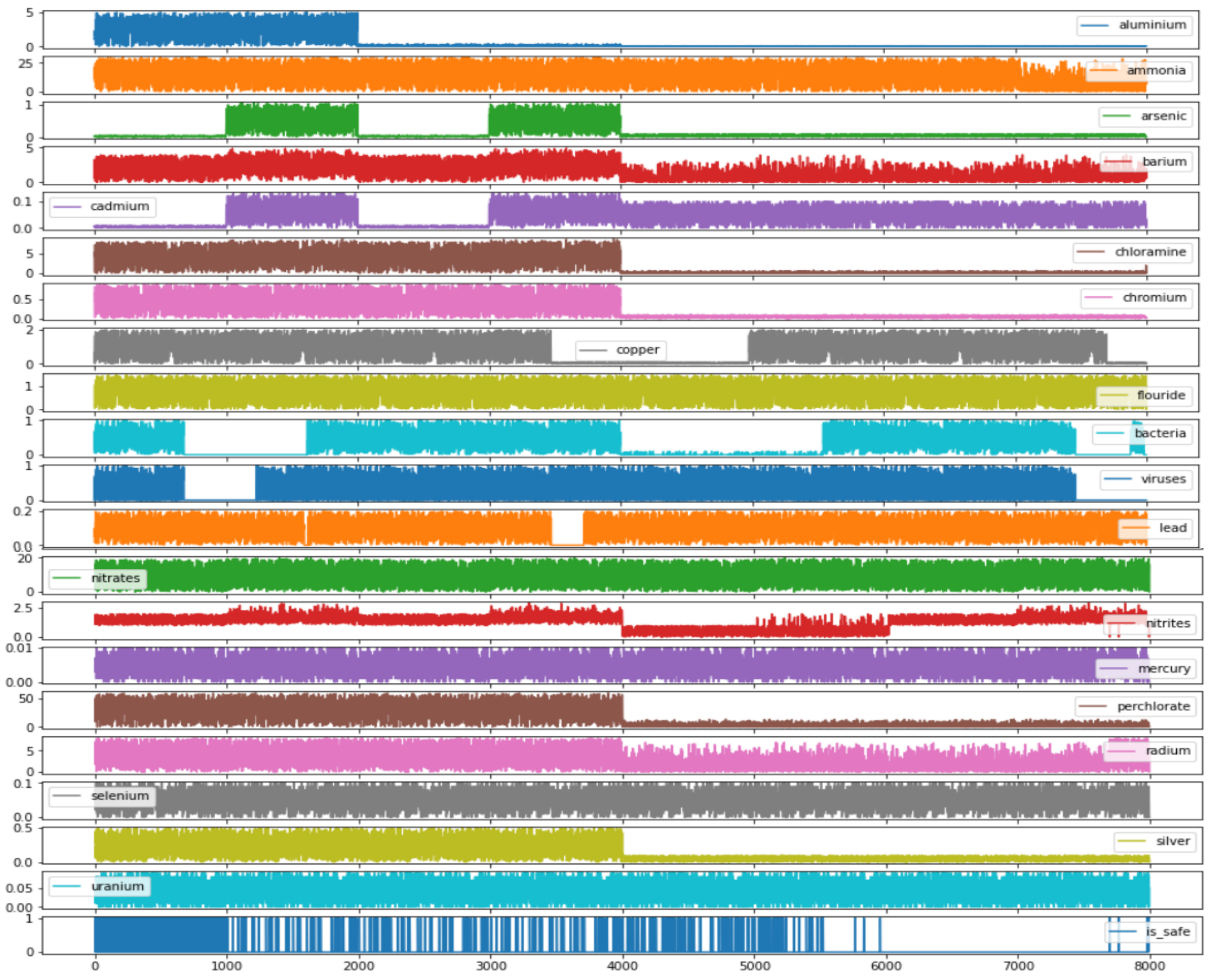
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



## Subplots for Each Attribute

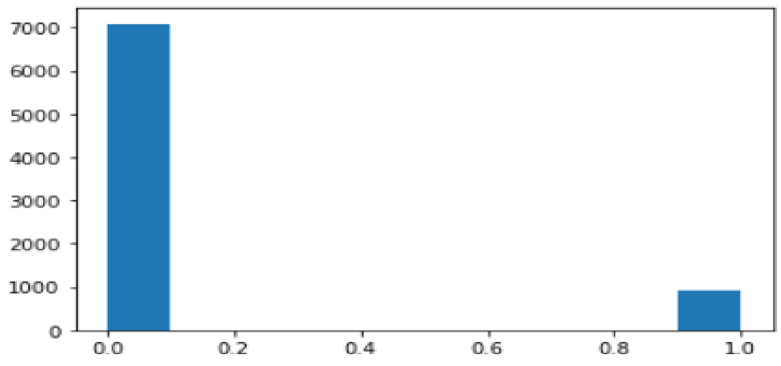
```
#DATA VISUALIZATION USING GRAPHS(Subplots for Each Attribute)
data.plot(subplots = True,figsize =(16,16))
```

```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>,
       <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>,
       <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>,
       <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>,
       <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>,
       <AxesSubplot:>], dtype=object)
```



## Histogram For Target Value "is_safe"

```
#DATA VISUALIZATION USING GRAPHS(Histogram For Target Value "is_safe")
plt.hist(data.is_safe.values)
plt.show()
```


7000
6000
5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0