In [4]:

```
import csv
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split

#I've converted the .data file to .csv which is read and replaced ? to NaN in the files
COLUMNS_COUNT = 2

with open('water-treatment.data', 'r') as f:
    columns = [next(f).strip() for line in range(COLUMNS_COUNT)]
temp_df = pd.read_csv('water-treatment.data', skiprows=COLUMNS_COUNT, header=None, delimit
er=';', skip_blank_lines=True)
even_df = temp_df.iloc[::2].reset_index(drop=True)
odd_df = temp_df.iloc[1::2].reset_index(drop=True)
df = pd.concat([even_df, odd_df], axis=1)
df.columns = columns
df.to_csv('out.csv', index=False)
text = open("out.csv", "r")
text = ''.join([i for i in text]) \
    .replace("?", "NaN")
x = open("out.csv","w")
x.writelines(text)
x.close()

reader=pd.read_csv('water-treatment.csv',header=None,delimiter=',');
df=pd.DataFrame(reader)
# print('Before Cleaning Up the DataSet\n')
# print(df)

#Calculating the Median of each Column and Replacing "NaN" with the Corresponding Median v
alues
for i in range(1,39):
    mean = df.loc[:,i].mean()
#     print('The mean of column :'+str(i))
#     print(mean)
    df.loc[:,i].fillna(mean, inplace=True)

for i in range(1,39):
    for j in range(0,527):
        mean=df.loc[:,i].mean();
        stdevi=df.loc[:,i].std();
        df.loc[j,i]=(df.loc[j,i]-mean)/stdevi;

print('\n')
# print('After Cleaning Up the DataSet and performing Normalization\n')
# print(df)

#Dropping the Date Column
print('\n')
# print('After Dropping\n')
df.drop(df.columns[0], axis=1, inplace=True)
```

```
# print(df)

# Implementing PCA with 2 components On the Normalized DataFrame
df1 = StandardScaler().fit_transform(df)
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(df1)
principalDf = pd.DataFrame(data = principalComponents
                           , columns = ['principal component 1', 'principal component 2'])
print('PRINCIPAL COMPONENT ANALYSIS\n')
print('Principal Components\n')
print(principalDf)

# K-Means Implementation on the PCA applied Dataset
kmeans1 = KMeans(n_clusters=4, init='k-means++', max_iter=300, n_init=10, random_state=0)
pred_y1 = kmeans1.fit_predict(principalDf)


# Adjusting the Clustering output from 0-3 to 1-4
for i in range(len(pred_y1)):
    if pred_y1[i]==0:
        pred_y1[i]=1
    elif pred_y1[i]==1:
        pred_y1[i]=2
    elif pred_y1[i]==2:
        pred_y1[i]=3
    else:
        pred_y1[i]=4


# Adjusting the Output to the desired form so that the Clusters get renamed and appear in
 order
l1=[]
l2=[]
cnt=0
for k in pred_y1:
    if not k in l1:
        l1.append(k)
        cnt=cnt+1
        l2.append(cnt)
for k in range(len(pred_y1)):
    for k1 in range(len(l1)):
        if (pred_y1[k]==l1[k1]):
            pred_y1[k]=l2[k1]
            break

print('Clustering Output of K-Means With PCA')
print(pred_y1)


# Writing the Clustering Output to the File
# MyFile=open('output.txt','w')
# i=1
# for element in pred_y1:
#     MyFile.write(str(i))
#     i+=1
#     MyFile.write(' ')
#     MyFile.write(str(element))
```

PRINCIPAL COMPONENT ANALYSIS

Principal Components

	principal component 1	principal component 2
0	-4.259950	-1.224329
1	-3.006076	-0.699610
2	-4.539135	-1.172044
3	-3.170685	-0.569993
4	-3.907692	-1.208180
5	-3.833486	-0.560928
6	-3.525585	-0.374845
7	-4.256650	0.270398
8	-4.767118	-0.969559
9	-3.695791	-0.467272
10	-5.423437	2.964510
11	-3.452051	1.515667
12	-5.858433	-0.553929
13	-3.522004	-1.148490
14	-5.140316	-1.573328
15	-4.284590	-0.467223
16	-4.055189	-0.380601
17	-3.551764	-0.034783
18	-3.669021	-0.354988
19	-4.193923	-0.148645
20	-5.018161	-0.769919
21	-4.038778	-1.453991
22	-3.744958	-0.049668
23	-4.134236	-1.066055
24	-3.572877	-0.746917
25	-3.735657	-0.660862
26	-4.301407	-0.597759
27	-4.905522	-1.431215
28	-4.510180	-1.114859
29	-3.455764	-0.844948
..	...	...
497	9.248392	1.465688
498	9.040609	0.172655
499	9.891089	0.566244
500	9.553056	-0.907416
501	11.706029	3.369268
502	10.727060	-0.279378
503	10.005346	-0.713758
504	9.635081	-1.359008
505	10.327988	-0.098921
506	10.863301	-0.522474
507	10.495026	-0.659999
508	12.319223	4.160751
509	12.055994	0.636669
510	9.774266	-1.165305
511	10.939580	-1.343885
512	11.563551	-1.741280
513	11.452881	-2.050057
514	13.904230	-0.264248

```
515          14.486136          -1.619392
516          15.179874          -1.981372
517          15.567796          -2.115335
518          16.448255          -1.905527
519          18.267418          -2.774714
520          18.488493          -3.520711
521          20.395226           0.082644
522          21.325591          -2.950235
523          26.869162          -2.241796
524          30.881646          -3.471193
525          40.283087          -4.075141
526          51.800433          -6.473373

[527 rows x 2 columns]
Clustering Output of K-Means With PCA
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1
 2 2 2 1 2 1 2 2 1 1 2 2 2 2 2 1 1 1 2 1 1 1 2 1 1 1 1 1 2 1 2 1 2 2 2 2 2
 2 2 2 2 2 1 1 1 1 2 2 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 4 4 4 4]
```

In [ ]: