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 K-Means with K as 4
kmeans = KMeans(n_clusters=4, init='k-means++', max_iter=300, n_init=10, random_state=0)
pred_y = kmeans.fit_predict(df)


# Adjusting the Clustering output from 0-3 to 1-4
for i in range(len(pred_y)):
    if pred_y[i]==0:
        pred_y[i]=1
    elif pred_y[i]==1:
        pred_y[i]=2
    elif pred_y[i]==2:
        pred_y[i]=3
    else:
        pred_y[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_y:
    if not k in l1:
        l1.append(k)
        cnt=cnt+1
        l2.append(cnt)
for k in range(len(pred_y)):
    for k1 in range(len(l1)):
        if (pred_y[k]==l1[k1]):
            pred_y[k]=l2[k1]
            break

print('Clustering Output After Ordering In Specified Order')
print(pred_y)

# MyFile=open('question3_output.txt','w')
# i=1
# for element in pred_y:
#     MyFile.write(str(i))
#     i+=1
#     MyFile.write(' ')
#     MyFile.write(str(element))
#     MyFile.write('\n')

# MyFile.close()
```

```
Clustering Output After Ordering In Specified Order
[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 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
 1 2 2 1 2 1 1 1 1 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 2
 2 2 2 2 2 1 1 1 1 2 2 1 1 1 1 1 2 2 2 2 1 1 2 2 2 2 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 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 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 [ ]:
```