**K-means clustering**

**Code :** [**https://github.com/kvartak2/Data-Mining-Assignment**](https://github.com/kvartak2/Data-Mining-Assignment)

**Given** : ‘water-treatment.data’ (a randomized dataset)

Problem statement : apply k-means algorithm for clustering

In real world scenarios, we often encounter a randomized dataset (water-treatment.data)

Our goal is to apply k-means algorithm to find groups in a dataset

But first we need to make sure the data is clean.

**Data cleaning**

* Usually data scientists spend more time in data cleaning, organizing the data rather than actually perform analysis.
* Missing data is perhaps the most common trait of unclean data.
* What should we do about missing values:

Below are some options we have for handling missing values:

1. **drop missing values**

Data is a valuable asset and machine learning models almost always tend to perform better with more data. Therefore,depending on the situation, we may prefer replacing missing values instead of dropping

1. **fill missing values with test statistics**

Frequently, when dealing with missing data, the sequencing does not matter, and thus, the values used to replace missing values can be based on the entirety of available data. In such cases, you would typically replace the missing values with your best guess (i.e., mean or median of the available data)

* We could detect missing values using below commands:

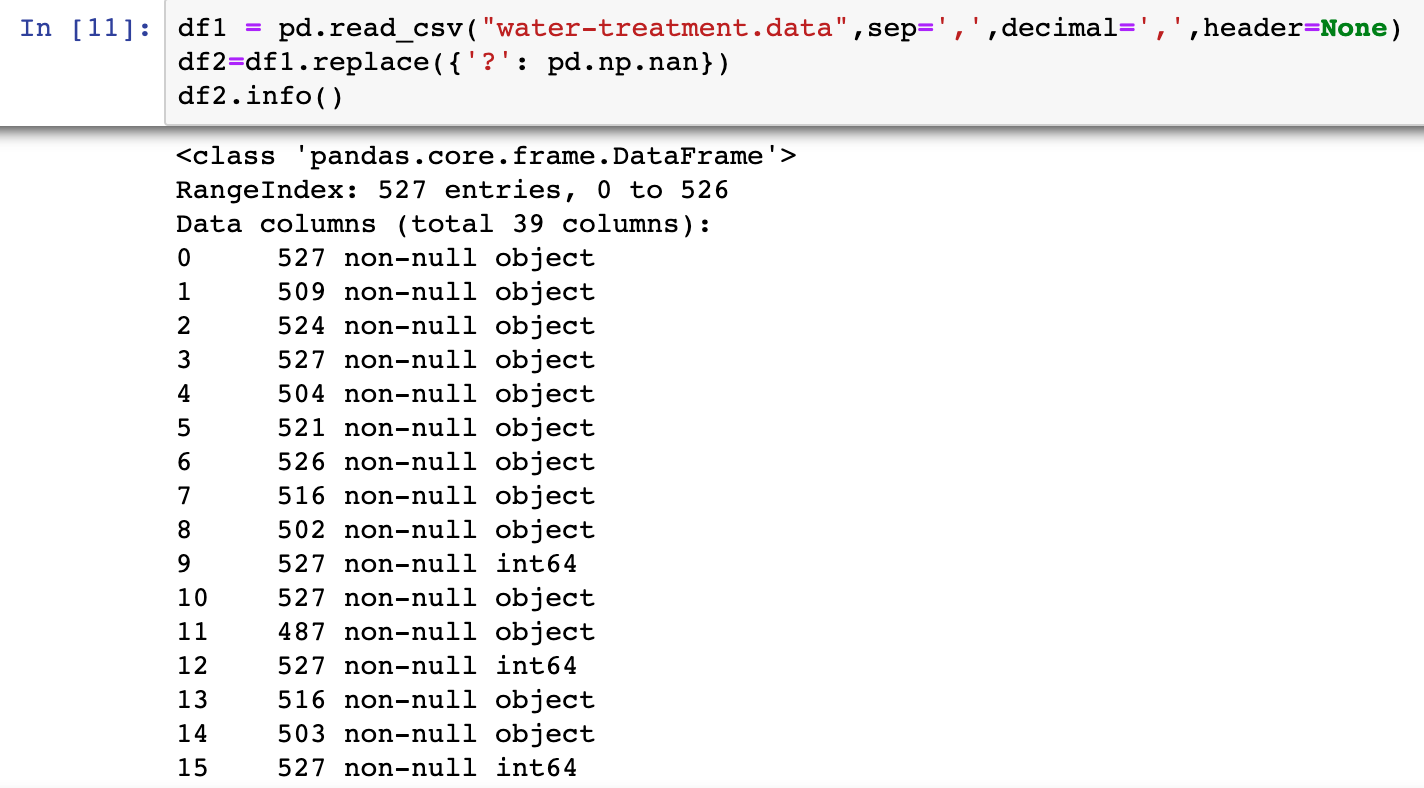
df.info() : to get a quick overview of dataset and also gives null values if any

df.describe(): gives basic statistical details like percentile, mean, std etc.

df.head(x) : this will give first ‘n’ rows and allows us to take a quick look at dataset

* Not all missing values come in nice and clean np.nan or None format. For example, “?” and “- -“ characters in column\_c of our dataframe do not give us any valuable information or insight so essentially they are missing values. However, these characters cannot be detected as missing value by Pandas.
* we have to first convert or replace such non NaN entry with a NaN.

﻿df2=df1.replace({'?': pd.np.nan})

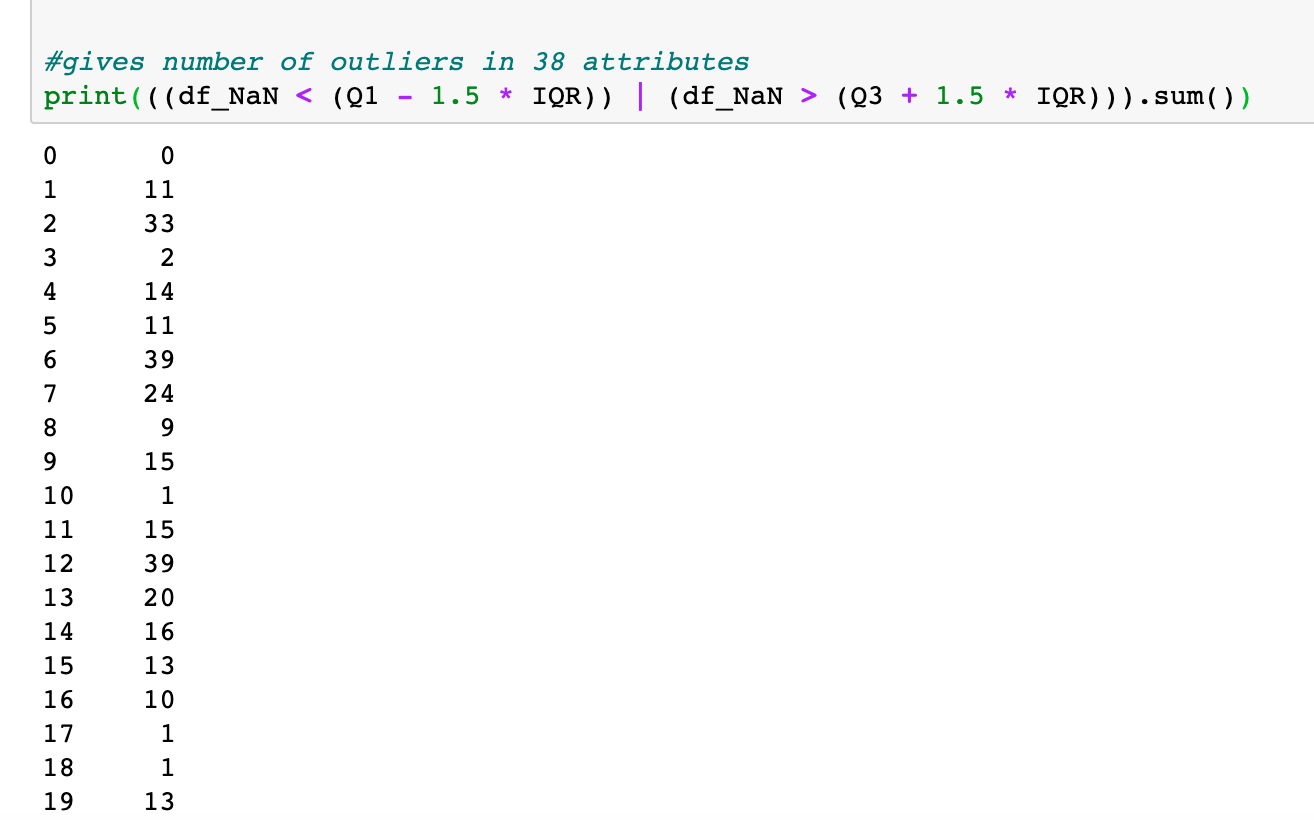


* Now we need to decide how we should fill these missing values. Usually, it is a good practice to fill the missing values with mean or median.
* One way to decide is by checking the outliers in the attributes. As outlier is a data point that differs significantly from other observations it will help us decide.

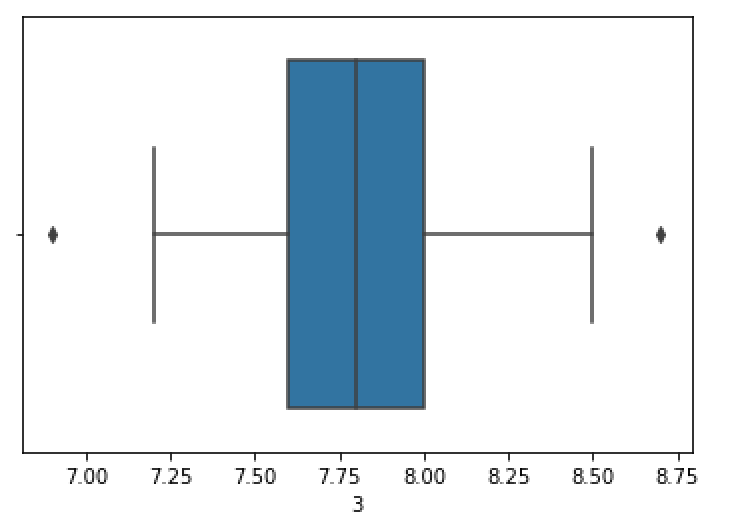
## Interquartile Rule to Find Outliers

1. Calculate the interquartile range for the data.
2. Multiply the interquartile range (IQR) by 1.5 (a constant used to discern outliers).
3. Add 1.5 x (IQR) to the third quartile. Any number greater than this is a suspected outlier.
4. Subtract 1.5 x (IQR) from the first quartile. Any number less than this is a suspected outlier.

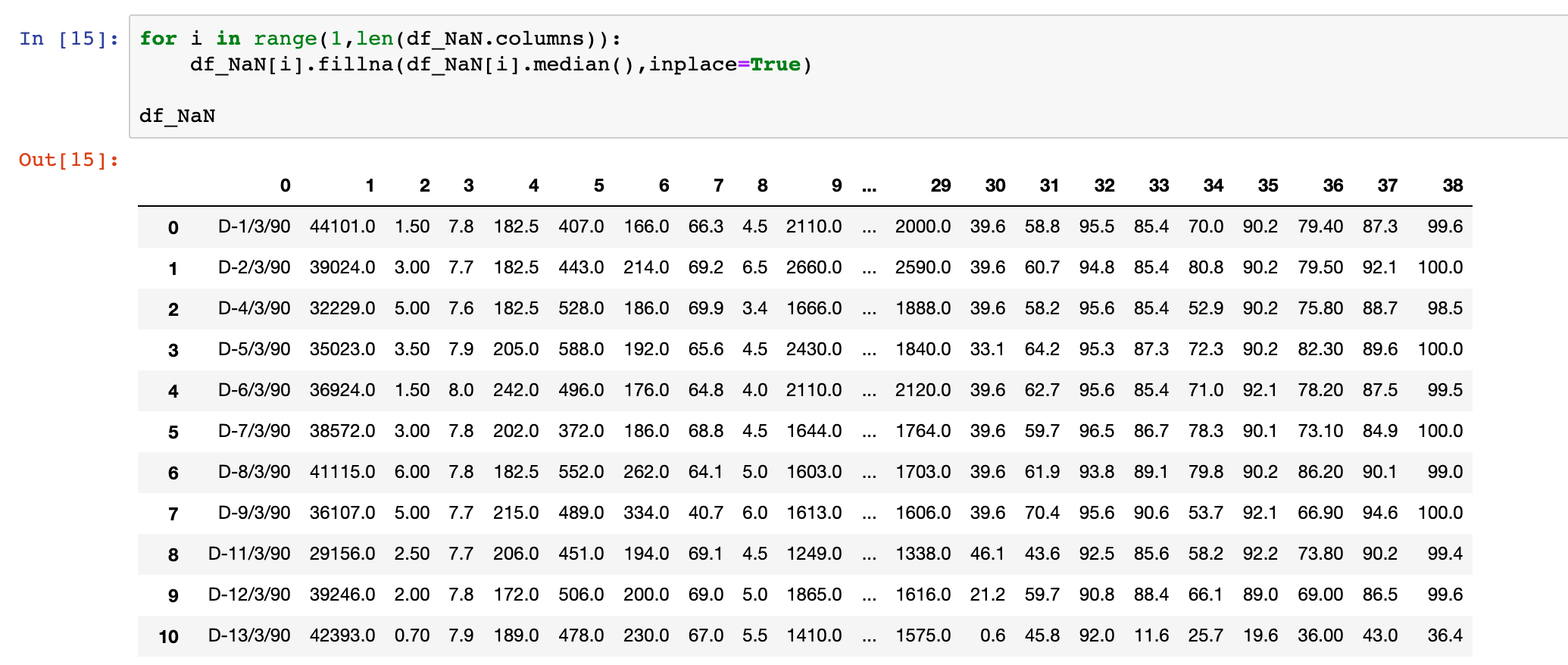
* We should remember that the interquartile rule is only a rule of thumb that generally holds but does not apply to every case.



* To confirm, below is a box plot of the 3rd attribute.



* Now, as the data is continuous and with outliers, we replaced the missing value by the median of each attribute. If we fill the values with mean, the data will be skewed.
* Lets replace NaN with median using fillna()



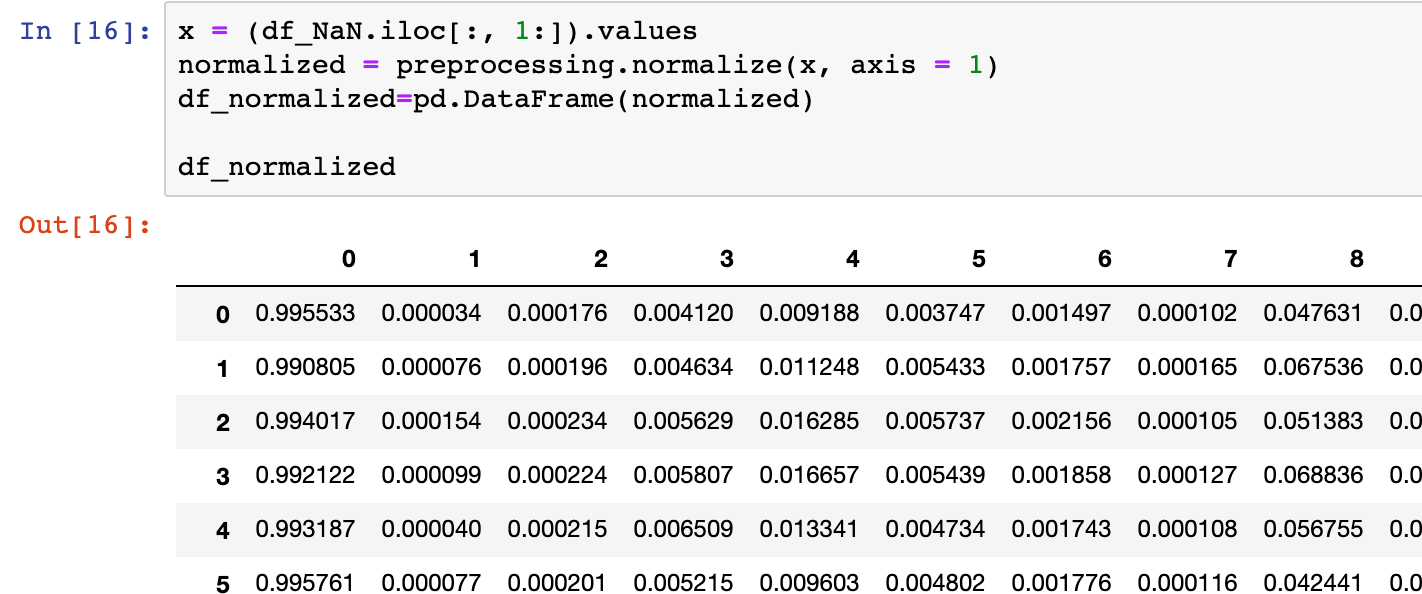
**To normalize or standardize**

First let's look at some definitions:

**Standardization** : Standardization is useful when your data has varying scales and the algorithm you are using does make assumptions about your data having a Gaussian distribution, such as linear regression, logistic regression, and linear discriminant analysis.

**Normalization** : Normalization is useful when your data has varying scales and the algorithm you are using does not make assumptions about the distribution of your data, such as k-nearest neighbors and artificial neural networks.

For normalizing the values, used ﻿sklearn.preprocessing ﻿ normalize which normalizes the dataset and translates each feature individually such that it is in the given range on the training set that is between 0 and 1.



**k-means**

The basic idea behind k-means consists of defining k clusters such that total within-cluster **variation (or error) is minimum**.

The aim of k-means clustering is to find these k clusters and their centers while reducing the total error.

I will explain in detail **The Elbow Method** that can be useful to find k in k-Means.

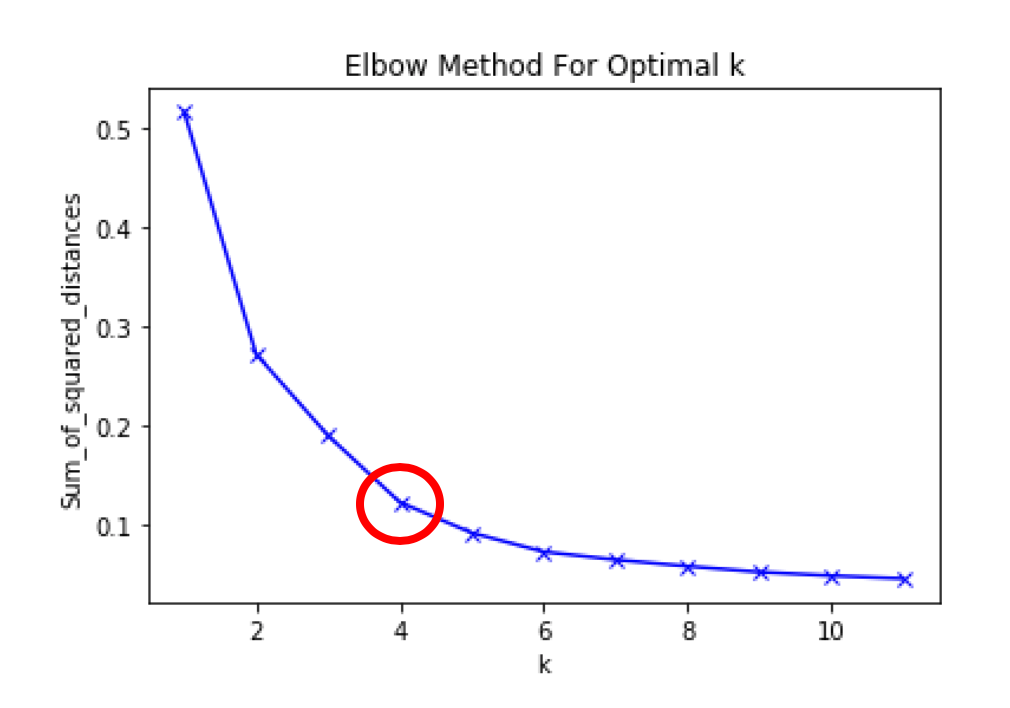
For the k-means clustering method, the most common approach for answering this question is the so-called elbow method. It involves running the algorithm multiple times over a loop, with an increasing number of cluster choices and then plotting a clustering score as a function of the number of clusters.

We now define the following:-

1. **Distortion:** It is calculated as the average of the squared distances from the cluster centers of the respective clusters. Typically, the Euclidean distance metric is used.
2. **Inertia:** It is the sum of squared distances of samples to their closest cluster center.

We iterate the values of k over K and calculate the values of distortions for each value of k and inertia for each value of k in the given range.

We obtain the following plot for our dataset.



To determine the optimal number of clusters, we have to select the value of k at the “elbow” i.e. the point after which the distortion/inertia start decreasing in a linear fashion. Thus for the given data, we conclude that the optimal number of clusters for the data is **4**.

We can verify that there are number of clusters which are 4 has same data which means our prediction of k=4 is True

﻿{2: 1, 3: 53, 0: 225, 1: 248}