## 1. Overview

Clustering is basically a type of [unsupervised learning method](https://www.geeksforgeeks.org/supervised-unsupervised-learning/) .

An unsupervised learning method is a method in which we draw references from datasets consisting of input data without labeled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent in a set of examples.  
Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups. In simple words, the aim is to segregate groups with similar traits and assign them into clusters.

Let’s understand this with an example. Suppose, you are the head of a rental store and wish to understand preferences of your costumers to scale up your business. Is it possible for you to look at details of each costumer and devise a unique business strategy for each one of them? Definitely not. But, what you can do is to cluster all of your costumers into say 10 groups based on their purchasing habits and use a separate strategy for costumers in each of these 10 groups. And this is what we call clustering.

These data points are clustered by using the basic concept that the data point lies within the given constraint from the cluster center. Various distance methods and techniques are used for calculation of the outliers.

**Why Clustering ?**

Clustering is very much important as it determines the intrinsic grouping among the unlabeled data present. There are no criteria for a good clustering. It depends on the user, what is the criteria they may use which satisfy their need. For instance, we could be interested in finding representatives for homogeneous groups (data reduction), in finding “natural clusters” and describe their unknown properties (“natural” data types), in finding useful and suitable groupings (“useful” data classes) or in finding unusual data objects (outlier detection). This algorithm must make some assumptions which constitute the similarity of points and each assumption make different and equally valid clusters.

**2. Types of Clustering**

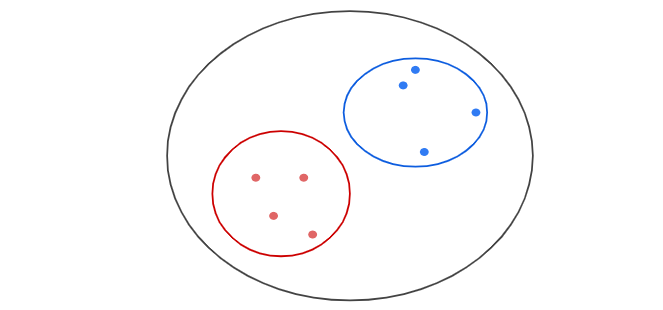
Broadly speaking, clustering can be divided into two subgroups :

* **Hard Clustering:** In hard clustering, each data point either belongs to a cluster completely or not. For example, in the above example each customer is put into one group out of the 10 groups.
* **Soft Clustering:** In soft clustering, instead of putting each data point into a separate cluster, a probability or likelihood of that data point to be in those clusters is assigned. For example, from the above scenario each costumer is assigned a probability to be in either of 10 clusters of the retail store.

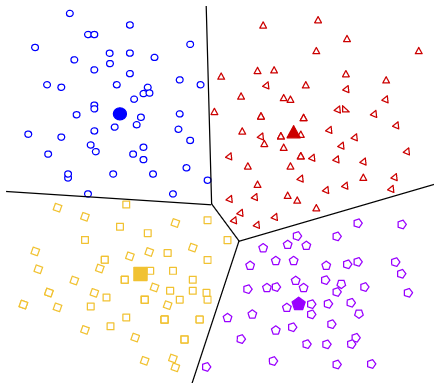
**3. Types of clustering algorithms**

Since the task of clustering is subjective, the means that can be used for achieving this goal are plenty. Every methodology follows a different set of rules for defining the ‘*similarity’* among data points. In fact, there are more than 100 clustering algorithms known. But few of the algorithms are used popularly, let’s look at them in detail:

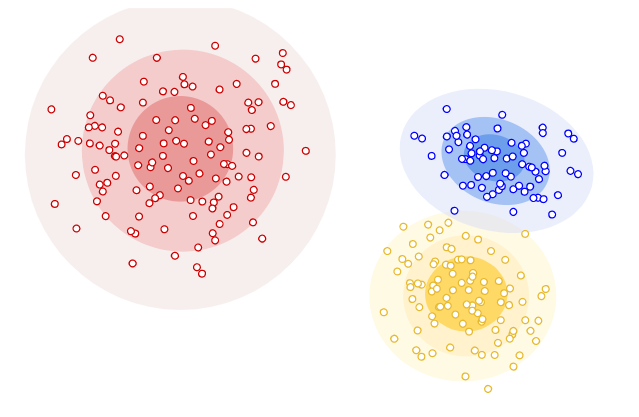
* **Connectivity models:** As the name suggests, these models are based on the notion that the data points closer in data space exhibit more similarity to each other than the data points lying farther away. These models can follow two approaches. In the first approach, they start with classifying all data points into separate clusters & then aggregating them as the distance decreases. In the second approach, all data points are classified as a single cluster and then partitioned as the distance increases. Also, the choice of distance function is subjective. These models are very easy to interpret but lacks scalability for handling big datasets. Examples of these models are hierarchical clustering algorithm and its variants.



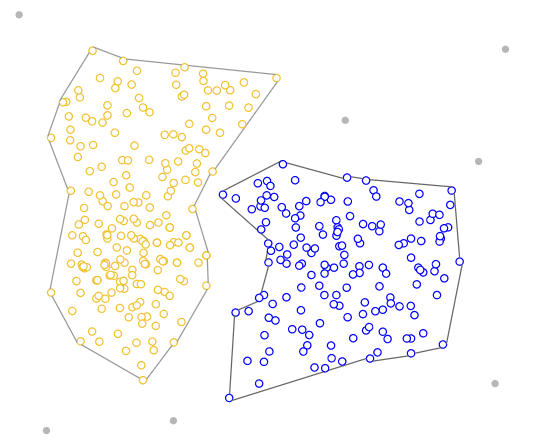
* **Centroid models:** These are iterative clustering algorithms in which the notion of similarity is derived by the closeness of a data point to the centroid of the clusters. K-Means clustering algorithm is a popular algorithm that falls into this category. In these models, the no. of clusters required at the end have to be mentioned beforehand, which makes it important to have prior knowledge of the dataset. These models run iteratively to find the local optima.



* **Distribution models:** These clustering models are based on the notion of how probable is it that all data points in the cluster belong to the same distribution (For example: Normal, Gaussian). These models often suffer from overfitting. A popular example of these models is Expectation-maximization algorithm which uses multivariate normal distributions.



* **Density Models:**These models search the data space for areas of varied density of data points in the data space. It isolates various different density regions and assign the data points within these regions in the same cluster. Popular examples of density models are DBSCAN and OPTICS.

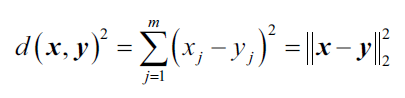


**4. K-Means Clustering**

**Key Concepts**

* **Squared Euclidean Distance**

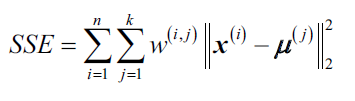
The most commonly used distance in K-Means is the squared Euclidean distance. An example of this distance between two points *x and y*in *m-dimensional*space is:



Here,*j*is the *jth* dimension (or feature column) of the sample points *x and y.*

* **Cluster Inertia**

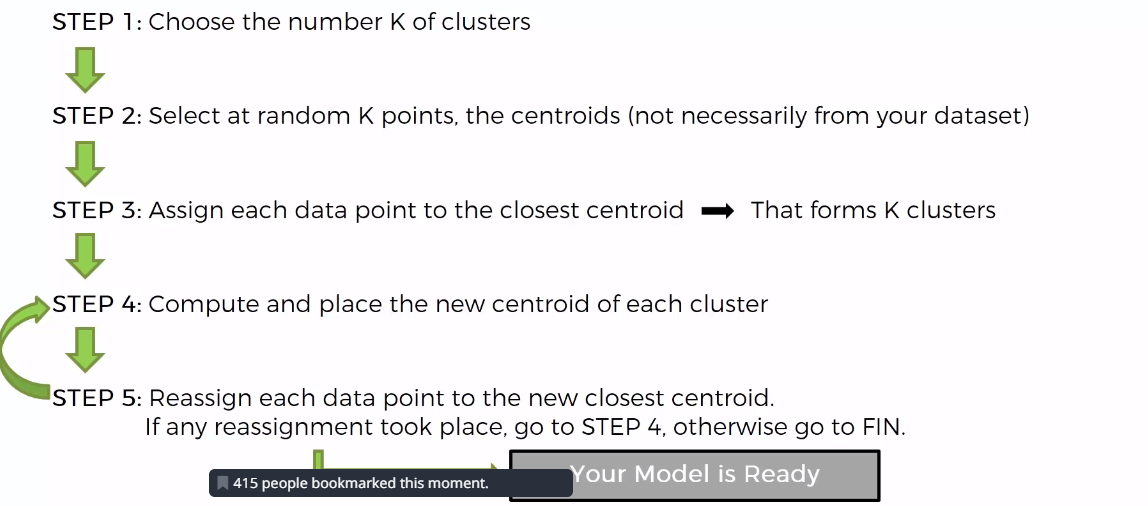
Cluster inertia is the name given to the Sum of Squared Errors (Also called as Within Cluster Sum of Squares - WCSS) within the clustering context, and is represented as follows:



Where *μ(j)*is the centroid for cluster *j,* and *w(i,j)* is 1 if the sample *x(i)* is in cluster *j*and 0 otherwise.

K-Means can be understood as an algorithm that will try to minimize the cluster inertia factor.

**Algorithm Steps**



1. First, we need to choose k, the number of clusters that we want to find.
2. Then, the algorithm will select randomly the centroids of each cluster.
3. It will be assigned each datapoint to the closest centroid (using euclidean distance).
4. It will be computed the cluster inertia.
5. The new centroids will be calculated as the mean of the points that belong to the centroid of the previous step. In other words, by calculating the minimum quadratic error of the datapoints to the center of each cluster, moving the center towards that point
6. Back to step 3.

**K-Means Hyperparameters**

* Number of clusters: The number of clusters and centroids to generate.
* Maximum iterations: Of the algorithm for a single run.
* Number initial: The number of times the algorithm will be run with different centroid seeds. The final result will be the best output of the number defined of consecutives runs, in terms of inertia.

**Challenges of K-Means**

* The output for any fixed training set won’t be always the same, because the initial centroids are set randomly and that will influence the whole algorithm process. This is called Random Initialization Trap.
* As stated before, due to the nature of Euclidean distance, it is not a suitable algorithm when dealing with clusters that adopt non-spherical shapes.

**Points to be Considered When Applying K-Means**

* Features must be measured on the same scale, so it may be necessary to perform z-score standardization or max-min scaling.
* When dealing with categorical data, we will use the get dummies function.
* Exploratory Data Analysis (EDA) is very helpful to have an overview of the data and determine if K-Means is the most appropiate algorithm.
* The minibatch method is very useful when there is a large number of columns, however, it is less accurate.

**Random Initialization Trap – Solution**

The final clustering result can depend on the selection of initial centroids, so a lot of thought has been given to this problem. One simple solution is just to run K-Means a couple of times with random initial assignments. We can then select the best result by taking the one with the minimal sum of distances from each point to its cluster – the error value that we are trying to minimize in the first place.

Other approaches to selecting initial points can rely on selecting distant points. This can lead to better results, but we may have a problem with outliers, those rare alone points that are just “off” that may just be some errors. Since they are far from any meaningful cluster, each such point may end up being its own ‘cluster’. A good balance is [K-Means++](https://en.wikipedia.org/wiki/K-means%2B%2B) variant, whose initialization will still pick random points, but with probability proportional to square distance from the previously assigned centroids. Points that are further away will have higher probability to be selected as starting centroids. Consequently, if there’s a group of points, the probability that a point from the group will be selected also gets higher as their probabilities add up, resolving the outlier problem we mentioned.

K-Means++ is also the default initialization for Python’s [Scikit-learn](http://scikit-learn.org/) K-Means implementation. If you’re using Python, this may be your library of choice.

**How to Choose the Right K Number**

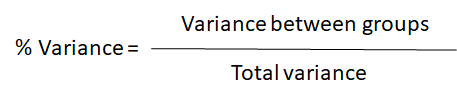
Choosing the right number of clusters is one of the key points of the K-Means algorithm. To find this number there are some methods:

* Field knowledge
* Business decision
* Elbow Method
* Silhouette Method

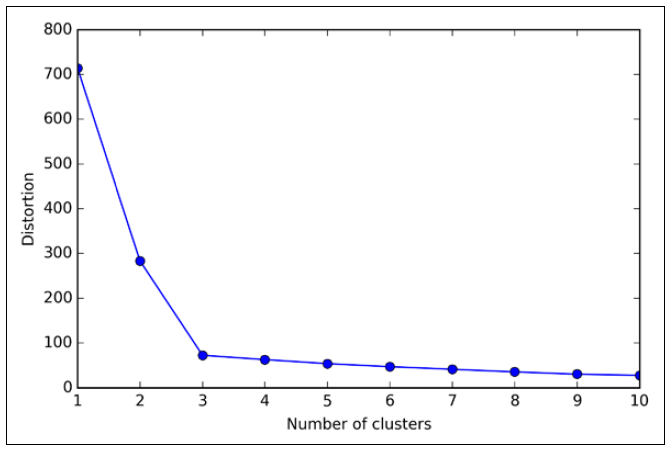
As being aligned with the motivation and nature of Data Science, the elbow mehtod is the prefered option as it relies on an analytical method backed with data, to make a decision.

**Elbow Method**

The elbow method is used for determining the correct number of clusters in a dataset. It works by plotting the ascending values of K versus the total error obtained when using that K.



The goal is to find the k that for each cluster will not rise significantly the variance

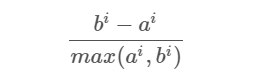


In this case, we will choose the k=3, where the elbow is located.

**Silhouette Method**

**Silhouette analysis** can be used to determine the degree of separation between clusters. For each sample:

* Compute the average distance from all data points in the same cluster (ai).
* Compute the average distance from all data points in the closest cluster (bi).
* Compute the coefficient:



The coefficient can take values in the interval [-1, 1].

* If it is 0 –> the sample is very close to the neighboring clusters.
* If it is 1 –> the sample is far away from the neighboring clusters.
* If it is -1 –> the sample is assigned to the wrong clusters.

Therefore, we want the coefficients to be as big as possible and close to 1 to have a good clusters.