# K-Means\_Clustering

Clustering is similar to classification, but the basis is different.

In Clustering you don`t know what you are looking for, and you are trying to identify some segments or clusters in your data. When you use clustering algorithms on your dataset, unexpected things can suddenly pop up like structures, clusters and groupings you would have never thought of otherwise.

In this part, you will understand and learn how to implement the following Machine Learning Clustering models:

K-Means Clustering

Enjoy Machine Learning!

**Solve Business problem in Python and R**

There is a big Mall in a specific city that contains information of its clients, the clients that subscribed to the membership card. When the clients subscribed to the membership card, they provided info like their gender, age, annual income, spending score ( values between 1 and 100 so that the closest spending score less the client spend, and close to 100 spending score more the client spend, score based to some criteria: income, the amount of dollars spent, number of times per week shown in mall etc. ).

My job is to segment the clients into groups based to annual income and spending score ( 2 variables for simplicity ).

The mall doesn’t know – which are the segments and how many segments, this is typically clustering problem because we don’t know the answers.

Let’s dive into it !

K-Means Clustering Intuition

In this section will talk about K-Means Clustering Algorithm. It allows you to cluster data, it’s very convenient tool for discovering categories groups of data set and in this section will learn how to understand K-Means in intuitive levels. Let’s dive into it:

Let’s decide we have 2 variables in our data set and we decide to plot those variables on X and Y axes. The question is:

*Can we identify certain groups among variables and how would we go about doing it ?!*

Yes, let’s see how! Are you excited ?! Because I really am !

What K-Means does for you !? It takes out the complexity from this decision making process and allows you to very easily identify those clusters actually called clusters of data points in your dataset. In this example we have 2 dimensions ( 2 variables ) but K-Means can work with multi-dimensions.

STEPS:

1. *Choose the numbers K of clusters*
2. *Select a random K points, the centroids (and not necessarily from your data set, they can be actual points in your dataset or they can be random points in scatterplot)*
3. *Assign each data point to the closest centroid -> that forms K clusters (for the purpose of this project we’ll use Euclidian distance. Basically, for every data point in dataset we’ll identify which centroid is closest. We’re going to use a quick hack , something that we learned from geometry. So, we’re going to connect the centroids with a line and then we’ll find the centrum of the line and we’ll put a perpendicular line exactly throw the central, so from the geometry that we know, it’s a very straight concept that every point of the perpendicular line is equity distant to the both centroids )*
4. *Compute and place the new centroid of each cluster ( in the center of mass, of gravity )*
5. *Reassign each data point to the new closest centroid. If any reassignment took place, go to step 4, otherwise go to FIN (Finnish) -> Your Model is Ready ( so, at the end, you can see this time the equit distant line does not make any points reassign, so, every point are already in the correct cluster and that mean no-reassignment to place during this step so we can proceed to complete our algorithm that mean the algorithm has converged. Now we can remove our centroids and distant line-> Model Ready )*

Random Initialization Trap

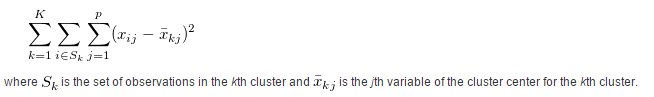
What if we select the centroid in different location, are we able to change the results ?! We don’t want the selection of centroids to effect how the clustering is going to happen. So, what would happen if we had a bad random initialization ?! There is a additional or a modification to K-means algorithm that allows you to correctly select the centroids and the solution is K-Means++ algorithm. At the same time I want to mention that we’ll not actually going into k-means++ algorithm, it is quietly involves approach in how the selection occurs, but the good news is that all this happens in background ( you don’t need to actually implement ) so, its good idea to be aware this issue. Keep in mind!

Choosing the right number of clusters

We talked about the random initialization trap (K-Means++), we worked with predetermined number of clusters. Let’s talk about the algorithm behind finding out the right number of clusters, so we’ll learn how to decide what number of clusters to input into K-Means algorithm.

So, let’s get straight into it! We got a challenge, a data science problem, again, we got only 2 variables ( X and Y coordinates, just for simplicity, in reality can be any number of columns, variables ). If we run K-Means clustering algorithm, we got the tree clusters, we need a certain metric, a way to evaluate how a certain a number of clusters performs compared to a different number of clusters and preferably that the metrics shouldn’t be a quantifiable . So, what kind of metrics can we use upon our clustering algorithm that will tell us something about final result ?! The answer is:

***Within Clusters Sum of Squares ( WCSS )***



What we are summing?! The distance between each point inside each cluster and the centroid of cluster and then we squaring the distances and we take the sum of all the squares of all these distances for each cluster. And we get the total sum and that is going to be our metric. This is quietly a good metric in terms of understanding or comparing the goodness of fit between 2 different of K-Means clustering. How do we know that ?! Let’s say (hypothetically) we have just one cluster, the WCSS become really big ( big distances between points and centroid) . Adding one more cluster, the total WCSS becoming less than when we had one centroid. Adding another cluster, the distance decrease and so on. What is the limit of that? How far its going decrease ? How many clusters of maximum can we have ?! We can have as many clusters as points of observation. In that case, WCSS will be 0 because every single point has its own centroid and the distance between point and centroid is 0.

But how do we find the optimal goodness of fit?! Is there a sacrifice that comes with improvement ?!

**THE ELBOW METHOD**

Its is actually very visual. You can see how WCSS changes in terms of adding more number of clusters. WCSS starts at a quite large number, what matters is how change is ( big gaps, massive change of units ). Firsts improvements created huge jumps, going forward, WCSS going not so substantial.

The Elbow method looks for that change ( elbow ) where the drop goes from being quite substantial to being not as substantial and there is the optimal number of clusters. This is quietly arbitrary. You decide how many of clusters its optimal for your certain of problem you’re trying to solve.