# Intelligent K-Means Clustering Algorithm


## Yusif Ibrahimov[1] and Fahmin Guliyev[2]

[1] *French-Azerbaijani University, Computer Science department.*



## Abstract

Unsupervised learning is a method of machine learning in which the model does not need to be overseen by the users. Instead, it allows the model to operate on its own to discover trends and previously undetected knowledge. It deals primarily with unlabelled data. K-means clustering is a simple and elegant approach to partitioning data set into independent, non-overlapping clusters of K. Performing K-means clustering, we must first determine the optimal number of clusters K; then the the K-means algorithm assigns each observation to precisely one of the K Clusters.We claim that K-Means is the unsupervised clustering algorithm , however we assign the optimal number of clusters. Then we supervise it? Is it really unsupervised? How can we make it really unsupervised? This is the scope of the mini-research project. The optimal number of clusters in the data will be calculated using the following methods:

**1) Measuring the diversity of the clusters** and

**2)Measuring the quality of the clusters**


## 1 Introduction

The K-Means algorithm is a straightforward algorithm capable of clustering datasets very rapidly and effectively, sometimes in only a few iterations. It was suggested by Stuart Lloyd at Bell Labs in 1957 as a pulse code modulation technique, but was only released outside the company in 1982. Edward W. Forgy published nearly the same algorithm in 1965, so Lloyd-Forgyy is often referred to as K Means.

Let's have a look the dataset that we will work on it. The Figure 1 is the visualization of our data set.

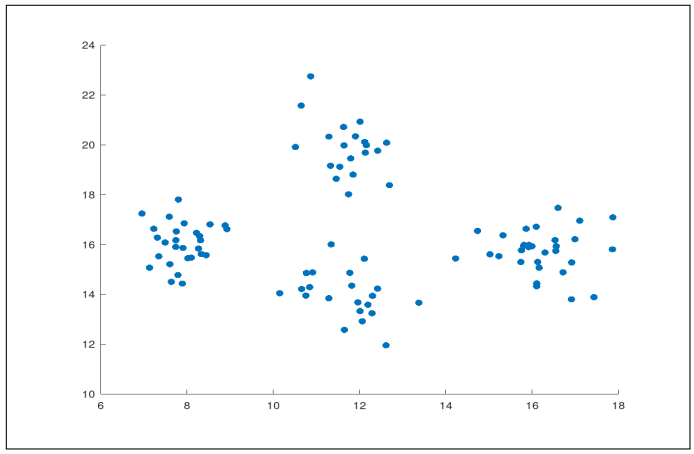



**Figure 1.** The dataset that we will work on it..

With our eyes, we can clearly see that there are 4 clusters. But the computers don't have eyes, so for using the K-Means we have to initialize the number of clusters K and we perform the Algorithm 1. Let's see the results of the algorithm using the coloring technique.

If we re-cluster the same dataset we will get the following clustering model

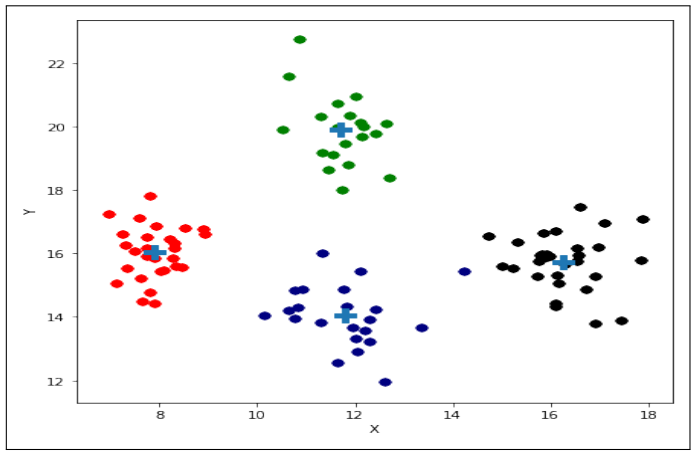



**Figure 2.** Clustered Data set using Ordinary k-means algorithm. .

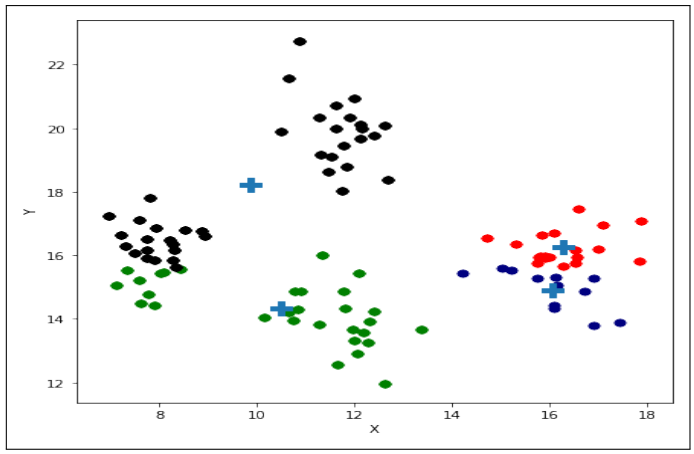



**Figure 3.** Clustered Data set using Ordinary k-means algorithm.

Did we perform the perfect clustering? Even we gave the correct number of clusters, we have not achieved the perfect clustering.What could be the reason.Of course the initialization of the cluster centroids. We initialize all of them randomly,therefore, we get bad clusters. Let's see the different cluster locations that are caused by the randomly initialized clusters.

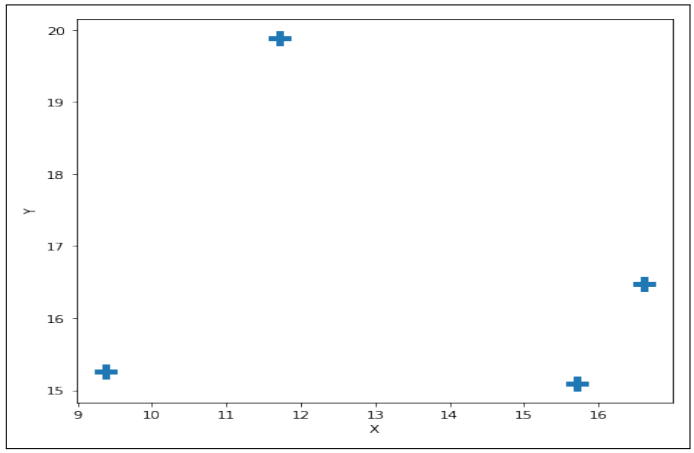



**Figure 4.** Cluster locations .

So we can see from the cluster locations, we can get the unperfect clusters. But what to do?
We know that the reason is the initializing. We must choose the separate points in order to perform the perfect clustering.
The answer is **K-Means++** algorithm.

## 1.1 K-Means ++ algorithm

K-Means ++ is almost the same with ordinary K-Means algorithm. The only difference is the centroid initialization. We do not initialize all the centroids randomly. We take only the first one randomly, and the others are chosen related to the data , and all the centers are maximally far from one another. The initialization algorithm is as the following:

Now, let's observe the results after applying the K-Means++ initialization. First, let's have a look the locations of the centroids:

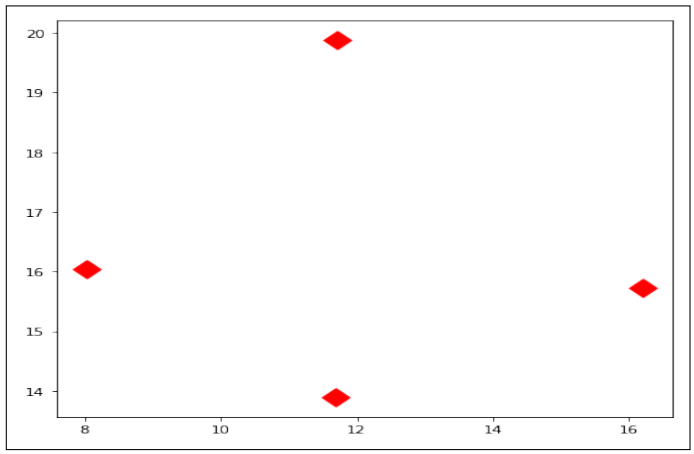



**Figure 5.** Cluster locations .

Now, let's have a look the following figure. We can easily say that it's perfectly clustered. K-

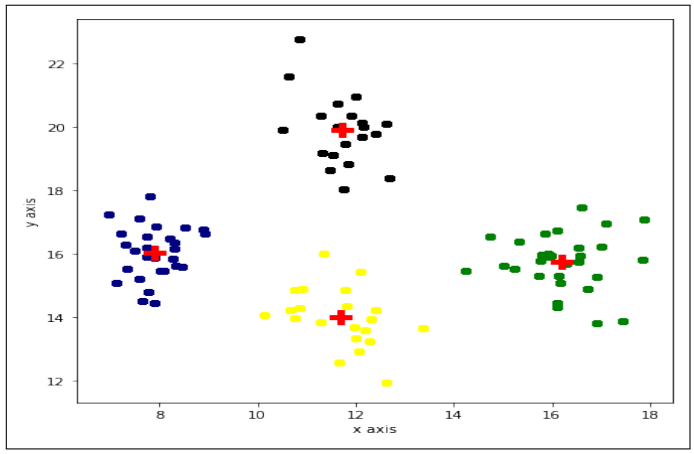



**Figure 6.** Cluster locations .

means++ is also stochastic since we have to randomly select the first center of the points, and the outcome will be somewhat different based on where we choose the first points, even though we give the correct number of centers. Fortunately, k-means++ is a way of choosing several varied starting points, but a first center must still be selected at random, meaning that the results of k-means++ may vary based on where this center is located, but the results of n k-means++ with n different initial centers are likely to be more reliable than the results of n k-means with k different initial centers.

### 1.2 Correct number of clusters

Now, let's return to the scope again. We demonstrated that, K-Means++ is better on the centroid initialization. A better initialization of the centers speeds up the general algorithms (less iterations are needed) and reduces the uncertainty of the outcomes (k-means finds a local optima).But , the thing is, We have to declare the number of clusters in both cases. It means that, K-Means++ is not the solution yet. So it isn't always unsupervised unless k can be determined automatically.
There are 2 perfect clusters:
a) data clustering in a single cluster
b) N number of cluster, each contains 1 sample.
Will the n-1 clusters be a satisfying cluster of n elements? It's definitely not. What's about 1+1 = 2 clusters? Yeah, this is going to be rewarding if the clusters are in good quality.
From the logic above, we can deduce that having fewer clusters is more beneficial, so let's say that we want to minimize the number of clusters, but we have to mention the definition of the "cluster in good quality"

## 2 Measuring the diversity of n K-Means ++ clusterings

As we said, our goal is to determine the correct number of clusters, and different methods can be proposed for this. The first method is to find out how different clusters differ from each other. As we can see in the picture, we have 4 different clusters. And it turns out that these cannot be present in many combinations. They can only vary slightly. So, what if we tell our program that it contains 3 clusters or another number of clusters in a situation where we see that there are 4 clusters with our eyes? Let's review the situations that may arise.

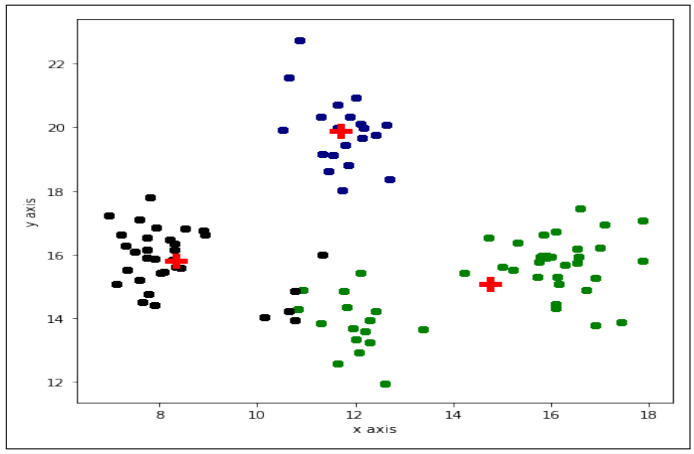



**Figure 7.** One of the Possible Situation K = 3

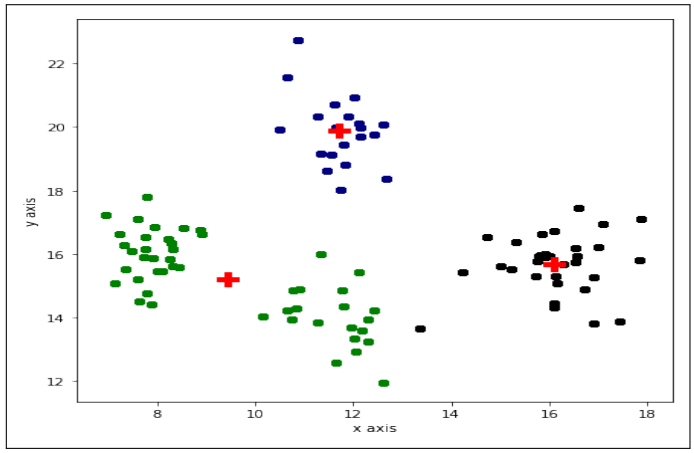



**Figure 8.** One of the Possible Situation K = 3

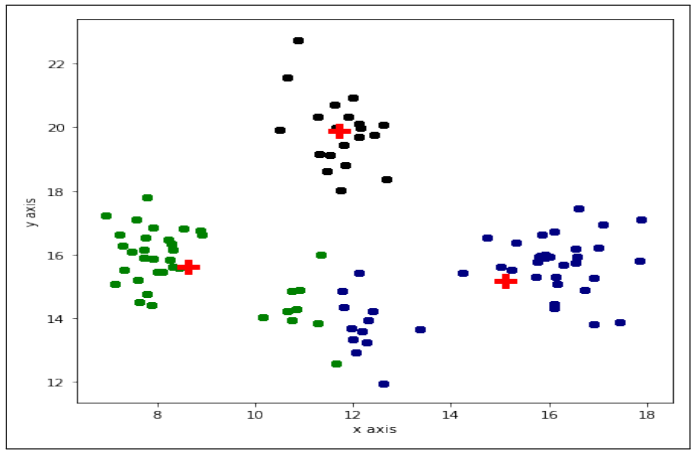



**Figure 9.** One of the Possible Situation K = 3

Regarding the results, what comes to mind first? Yes, we are getting very different results. More precisely, every time we restart the clustering process, we get different results. Let's keep that in mind, this is just a small set of experimental data. Larger data sets and p. and then the differences can become more enlightened. Well let's ask the same question to ourselves again. We know these differences with our own eyes. But computers are spared of such a capability, and there are no eyes, they cannot see differences as we do. Let's try to look at the issue from a different lens. We can present these differences with another expression. "Randomness". Clausius used this scale for the first time in physical science and called it "Entropy".This s the second law of thermodynamics and states that, "Entropy of an isolated system always increases, or in other word, "Entropy can be created but can not be destroyed.Entropy in physics is calculated using the formula below.

$$\delta S = S_f - S_i = \int \frac{dq_{rev}}{T} \tag{1}$$

So how will this formula help computer science and, to be more precise, us now that we are facing it? Answer: Unfortunately, this formula cannot help us at the moment. But there were a number of Entropy formulas that were inspired by this formula, which are used directly to measure the confusion in computer science, such as Shannon's Entropy(1948), Schneider's Entropy, (1991), Shenkin's Entropy (1991), Gerstein's Entropy (1995) and Gap Normalized Entropy (2004) ). All of those actually come from Shannon's Entropy, so we'll use the following equation to reflect this in our calculations.

$$H(f) = \sum_{v=1}^{n} p(v) log_2(\frac{1}{p(v)}) = -\sum_{v=1}^{n} p(v) log_2(p(v)) \tag{2}$$

If we approach the issue algorithmically, the algorithm below will help solve our problem. Now back to our point, how will Shannon Entropy help us? As we just said, when we try to divide a data set that actually contains 4 clusters into 3 clusters, we risk getting a different result each time. This is the same for all other situations. So let's change our perspective on the matter a little bit. And for the current situation, let's try the cluster number for 3, 4, and 5 for both, and test each 32 times. Let's put the results we got in 32 iterations for both 3, 4 and 5 into a list. And let's calculate the entropy of this List. We get the following results. When I look at the above table, we can now be

**Table 1. Relations of K and Entropy**

| Num. of clusters | Entropy |
|---|---|
| 3 | 3.0314 |
| 4 | 2.1972 |
| 5 | 3.8073 |

aware of the situation. The least Entropy side is taken when our randomness cluster count is equal to 4. As it can be seen from here, too much entropy actually does not work for us. The smallest number is also taken in 4. So what does this mean? Of course, in this range, among the numbers 3, 4 and 5, the most successful is 4.

Now let's change our perspective. We have no idea about values above 5. Let's review more values and review more values. But let's be careful that we are still in the local area. We clearly see how the Radnomness changes in the values in the 3-10 cluster count range. The lowest entropy value still remained at 4. This is also a very good development because even if we expand our score range, our claims are still confirmed.

Now we're raising the bar even higher and we're going to consider changing points at a global alignment. Let's see if the number of clusters is maximally high, will our previous assumptions be correct?

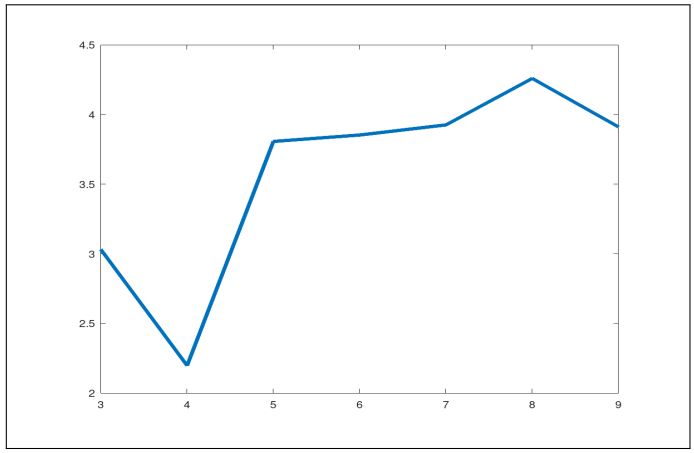



**Figure 10.** Range of K is : 3-10 and the entropies

We got a very good result. We have seen that when we give K the maximum values possible, we get

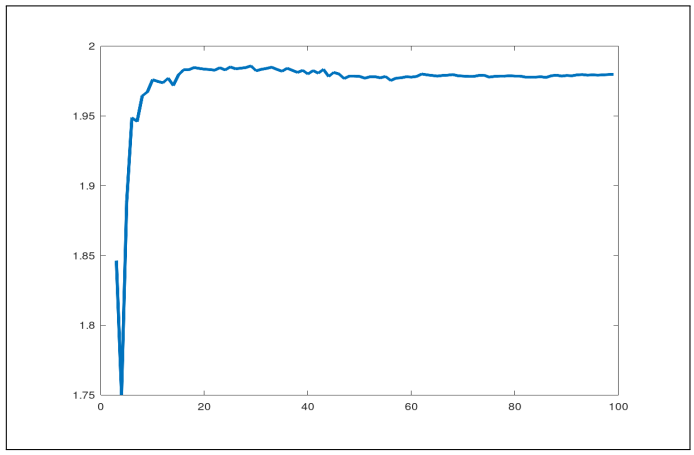



**Figure 11.** Global change of the entropies

the lowest score of 4. This was already a known value for the available data. This officially proves and confirms all of our assumptions.

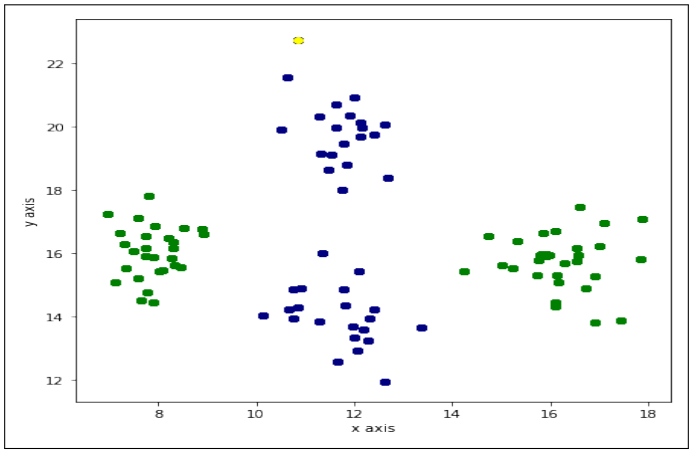
Here we tested the value of K with 10 times 30 times. First of all, we need to know that this takes a lot of time and can tire our computer. So what do we have to do this time? At this time, we prefer to use Gini imputiy instead of Entropy because it makes our job a little easier. Gini impurity is calculated using the formula below.

$$G(f) = 1 - \sum_{v=1}^{n} p(v)^2 \tag{3}$$

As a result, we saw that the lowest randomness value was verified when our cluster count equals 4. So, the solution we propose can help us find the most suitable cluster number.

## 3 Measuring the quality of the clusters

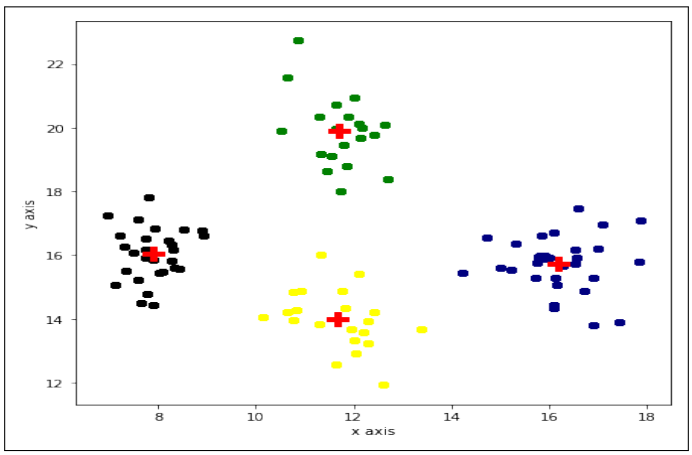
Another method of giving an idea about clusters is to calculate the quality of the clusters. It is very important that the cluster quality is high, because otherwise we would not be considered correct. Suppose we get the following type of clustering as a result. Let's think about it.
When we look carefully, we realize that the result we get is not good. We have 4 clusters. The blue




cluster and green cluster contain too many elements, while the yellow cluster contains only one element. Green and Blue colored clusters show very large scatter. This is not a good situation either. As can be seen from the picture above, we need to give an opinion on two important details in order to give an idea about **density**. The first detail is the cluster frequency. How much space does the cluster take and how many elements does it contain? The second detail is how **scattered** the elements show.

## 3.1 Density of clusters

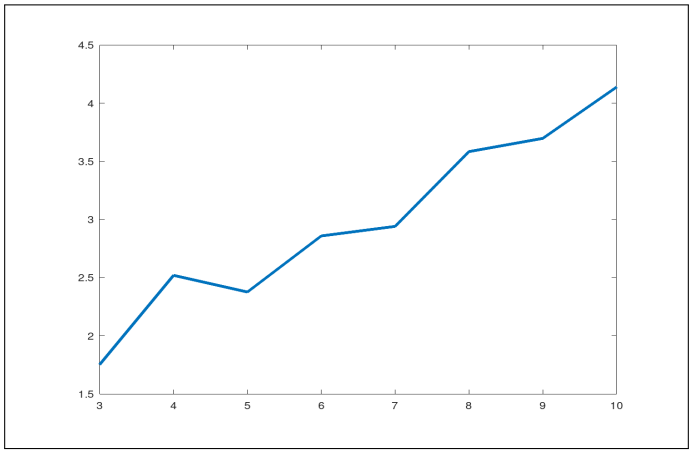
As mentioned above, one of the two important details is density. So how do we calculate the frequency of a cluster? Usually we take the frequency by dividing it by the field. In this case, the frequency will not be a different technique. We will calculate the number of elements containing the cluster by dividing by the area of the cluster.
This time, let's consider a clustering situation like the following.



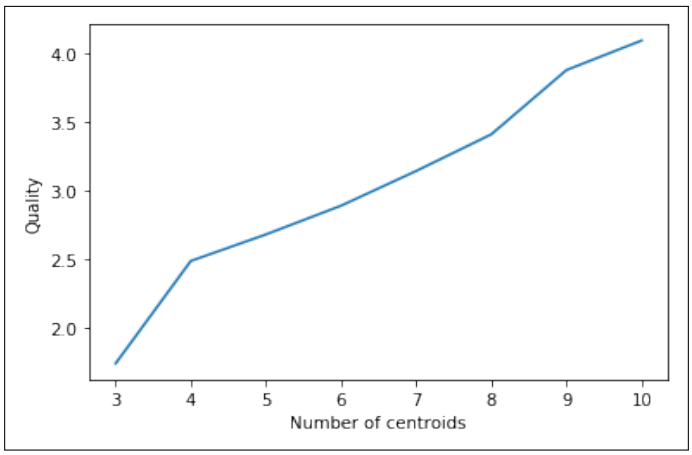

Now let's calculate the density of clusters colored in black. The number of elements can be calculated, but the point is that calculating the area might make us think. Since we are working in a 2 dimensional coordinate system, we know that each element has both x and y values. Based on this, we can calculate the area with the formula below.

$$A_{cluster} = (max\{X\} - min\{X\}) * (max\{Y\} - min\{Y\}) \tag{4}$$

And thanks to the source codes we wrote, the result we got is that the density of the cluster colored in black is 2.3189.
Now let's first try to analyze how the frequencies change for different cluster numbers.
It is not difficult to observe that densities tend to increase on average as the number of clusters




increases. The reason for this is actually good. As the number of clusters increases, we get more frequent clusters, so this value increases. In fact, the dense cluster is a good result for us, but as we see, we have difficulty getting the right result.

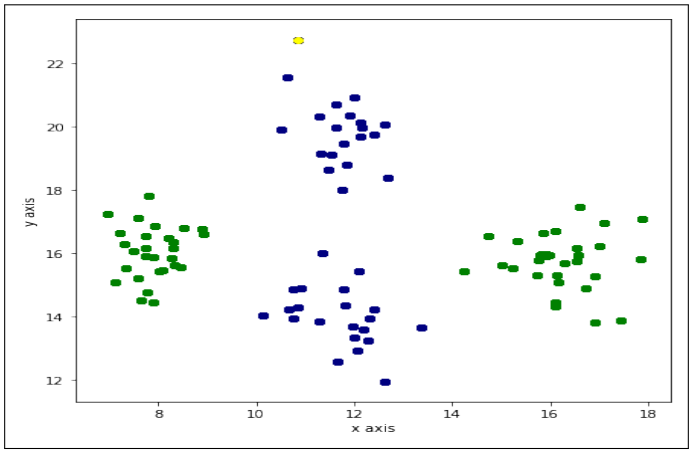



As we can see above, even if we make 30 evaluations for each k value, we observe an increase in this value on average.
As a result, we can say with certainty that density is never enough to convey an opinion about quality.

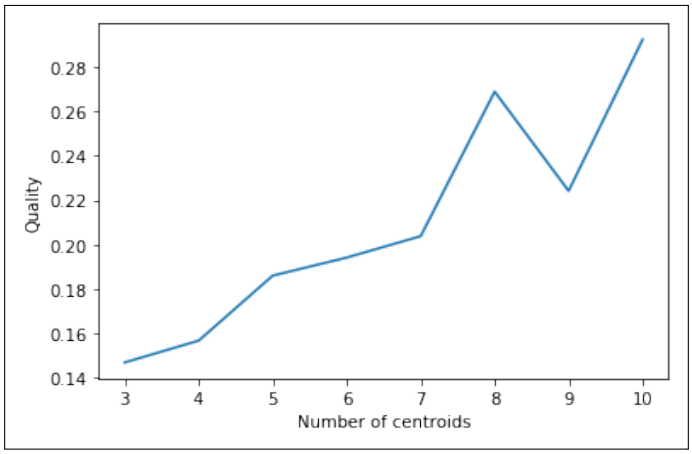
## 3.2 Scattering of clusters

Suppose we have a cluster, and there are too many elements in it, and the density value is too high. But we proved that this is not enough at all. Let's pay attention to the graphic below.

If we calculate, we will see that green and blue clusters have a high density. And at the same time, they show very high scattering. Indeed, is high scatter good quality? No, as can be seen from the picture, high scattering gives us wrong results.




Now the question is, how will scattering be calculated? In a very easy way, if we want to calculate scattering, we have to look at how the x and y values are scattered, that is, what sort of randomness they show. And if this value, the entropy value, is high, then we have shown too much scatter. Entropy is cumulative. So we add the entropy of the x and y values. It is necessary to divide the result by one, because low value of scattering is important for us. For blue and black clusters, the entropy value on the X value is 4.0347, on the Y value the entropy value is 4.0929, the total entropy value is 8.1276. Scattering is 0.1230.
Now let's see how the scattering value changes as the value of k changes.

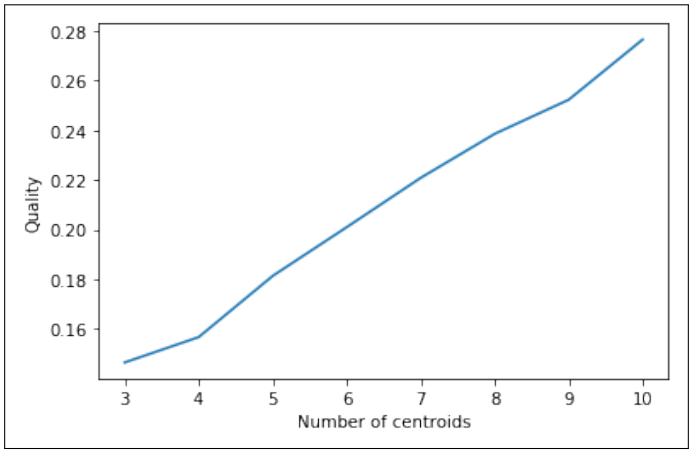



As we can see from the graphs above, the higher the k value, the higher the scattering. Now let's calculate 30 times for each k value again.

We can now clearly observe that the scattering value has increased. So the scattering value will not be correct for us. So what should we do? Maybe we should use both at the same time.

## 4 Combination

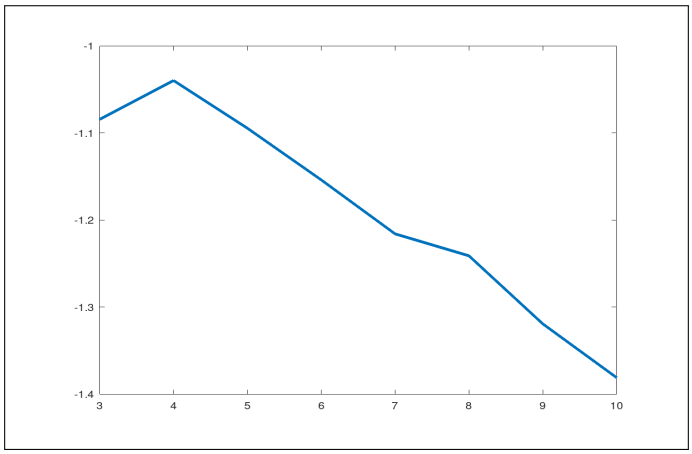
It seems that both values, density and scattering are important to us at the same time. That's why we have to use both at the same time. So how?
The formula below can answer our questions. By multiplying both values by a certain fixed value, we can put them together.




$$Quality_k = \sum_{i=1}^{k} \omega(i)(\alpha d(i) + \beta s(i)) \tag{5}$$

i one of the k clusters, $\omega()$ the weight of the cluster, $\alpha$ the weight of the density d() and the $\beta$ weight of the scattering s(). Omega is equal to the number of elements in the current cluster. Alpha and Beta are constant variables and are the same for each iteration. We take these into account some fixed situations, on the density value too high and s. It was determined as 0.002 and -0.1 considering the situations.

As we can see from the last graph, the highest quality has been achieved again at a value of 4. This




indicates that we have chosen successful fixed values.

## 5 Conclusion

If we are to express the results of our research briefly, we can see that there is a chance that both choices will lead us to success. Both ways are correct. but the first method may cause us to get results faster, but both ways are logically correct