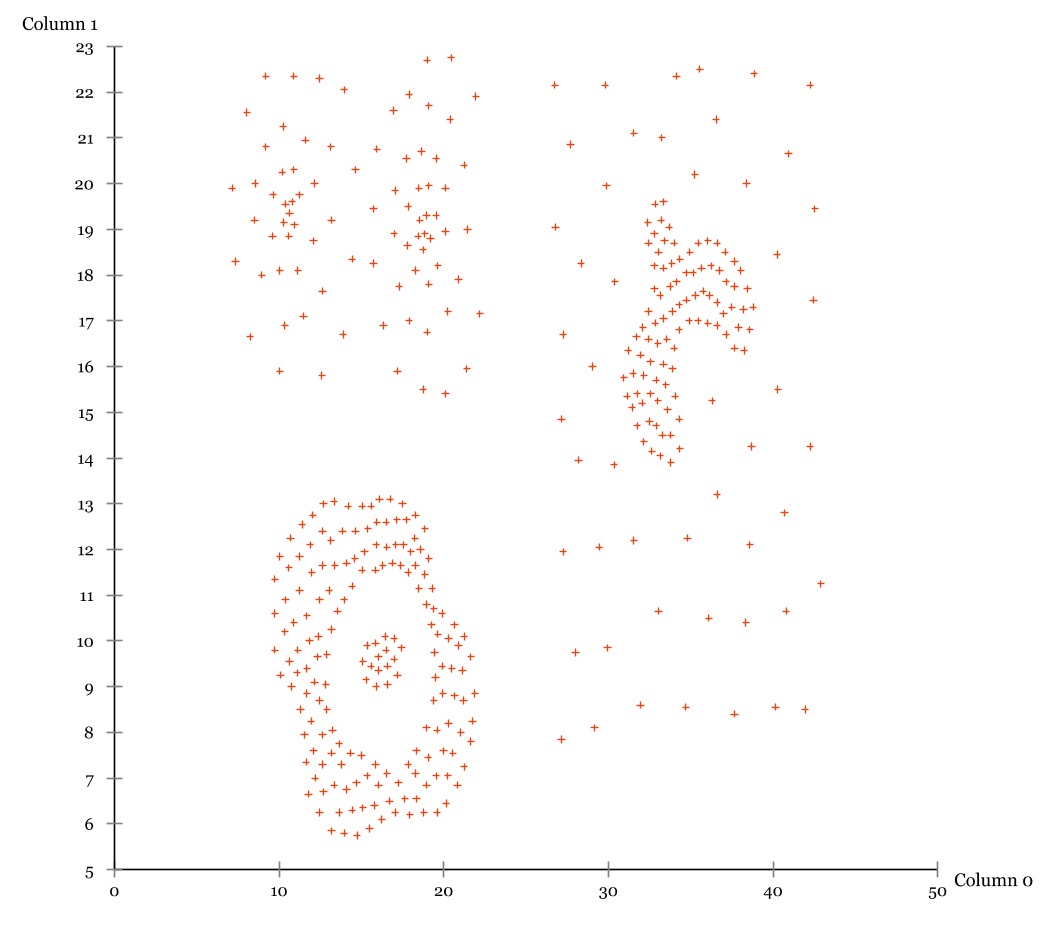
Data Mining – Project 1/Exercise 11

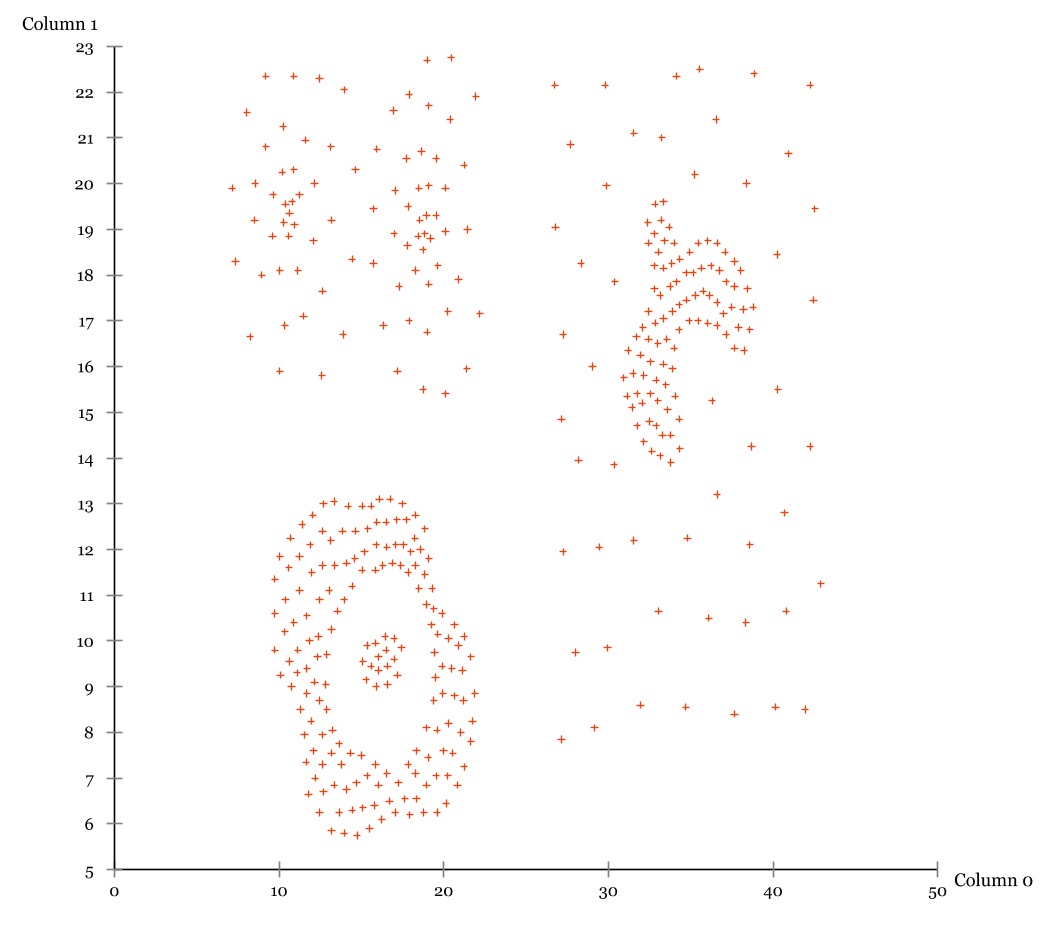
**Zahn’s Compound**

**Trivial (all in one):**



**Choosing the right algorithms**

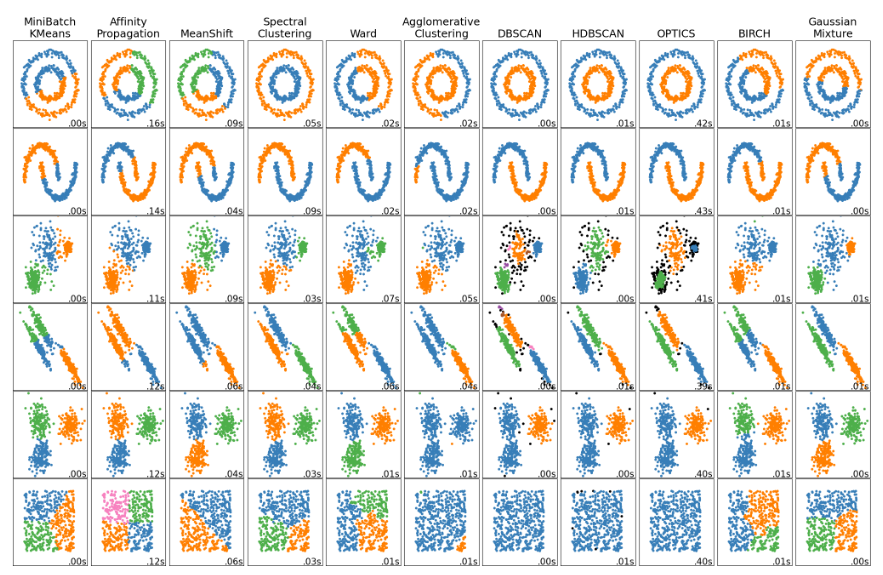
By eye, looking at the unclustered dataset, we may roughly see 5 clusters – we do this by identifying the different clumps that have around the same density.



We also see that the clusters are nested – for example this spiral here [point out] and here [point out].

This nesting indicates that a relatively standard clustering method, for example k-means clustering, would not be the best suited for our dataset, as this type of clustering has difficulty differentiating when working with nested clusters.

We therefore instead look towards algorithms that are more appropriate for this kind of data. To visualize and exemplify this we may take a look at the following overview:



SciKit: <https://scikit-learn.org/stable/modules/clustering.html>

Comparing our dataset to the ones shown in the overview, Zahn’s compound would perhaps be deemed most similar to dataset three from the top. Looking at how the different algorithms perform on this dataset, we may pick out DBSCAN and OPTICS as better suited for the complexity of our dataset. On the contrary, we do not think that k-means will do particularly well on our dataset – but to exemplify the differences and comparisons of these algorithms we will be showing how all three perform on our dataset.

**K-Means**

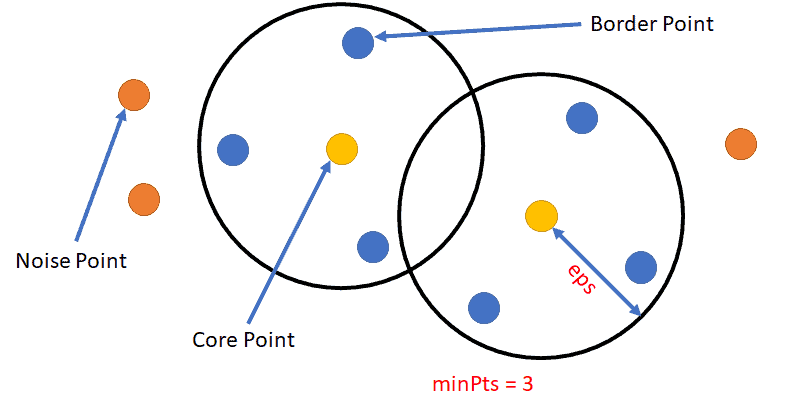
We start with the k-means clustering method…

TEKST OG ANALYSE (INKL. BILLEDER)

**DBSCAN**

Now going to DBSCAN. The DBSCAN algorithm views clusters as areas of high density separated by areas of low density. This means that clusters found be DBSCAN can be any shape – whereas k-means clustering assumes that clusters are convex shaped (alt: spherical or circular).

When performing DBSCAN, there are two parameters to the algorithm; minimum samples and epsiolon. These define the density necessary to form a closter. This may be explained as such: we define core samples in the data set such that there exist the number of minimum samples within a distance of epsilon. These points are defined as neighbors of the core sample.



A cluster is a set of cores samples that can be built recursively by taking a core sample, finding all of its neighbors that are core samples, finding all of their neighbors that are core samples and so on. A cluster also has a set of non-care samples, which are neighbors of a core sample in the cluster but not themselves core samples. These are on the outskirts of a cluster.

**Performing DBSCAN in Elki**

For using DBSCAN, it is recommended to always use an index. We use SimpleCoverTree index, which works for most data sets. This requires only one further parameter, a distance function. Here we use Minkowski Euclidean distance function.

***Selecting MinPts:***

Minimum number of points required to form a dense region.

As general rule of thumb MinPts=2\*number of dimensions. In our case 2\*2=4, so we set MinPts to 4 as a starting point.

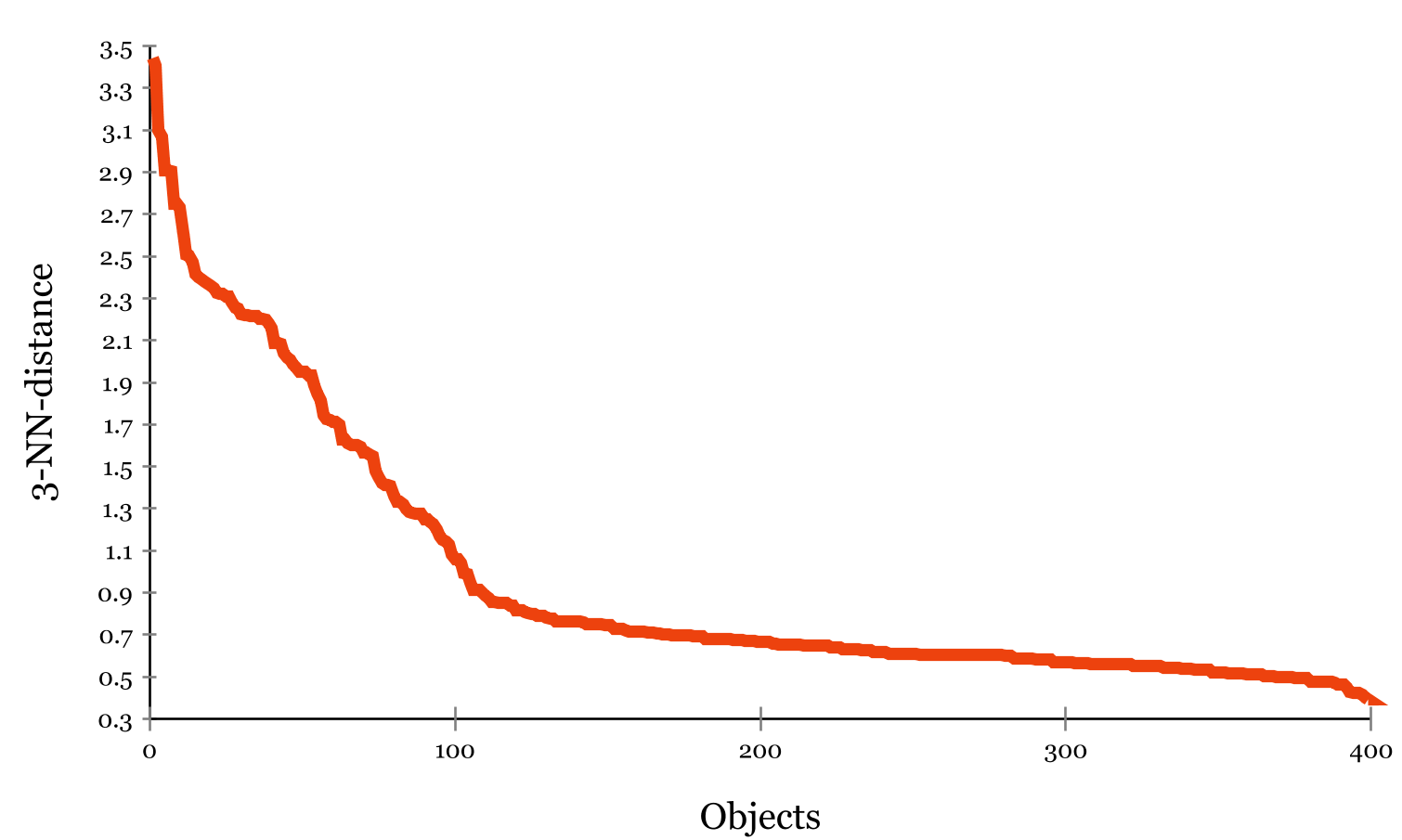
We may want to increase this to avoid creating too many small clusters.

***Selecting Epsilon:***

Determines the maximum distance between two points for them to be considered neighbors.

For selecting epsilon, using domain knowledge regarding one’s dataset could be a starting point for estimating what distance is meaningful in the specific case. As we do not have this knowledge, we use the elbow method.

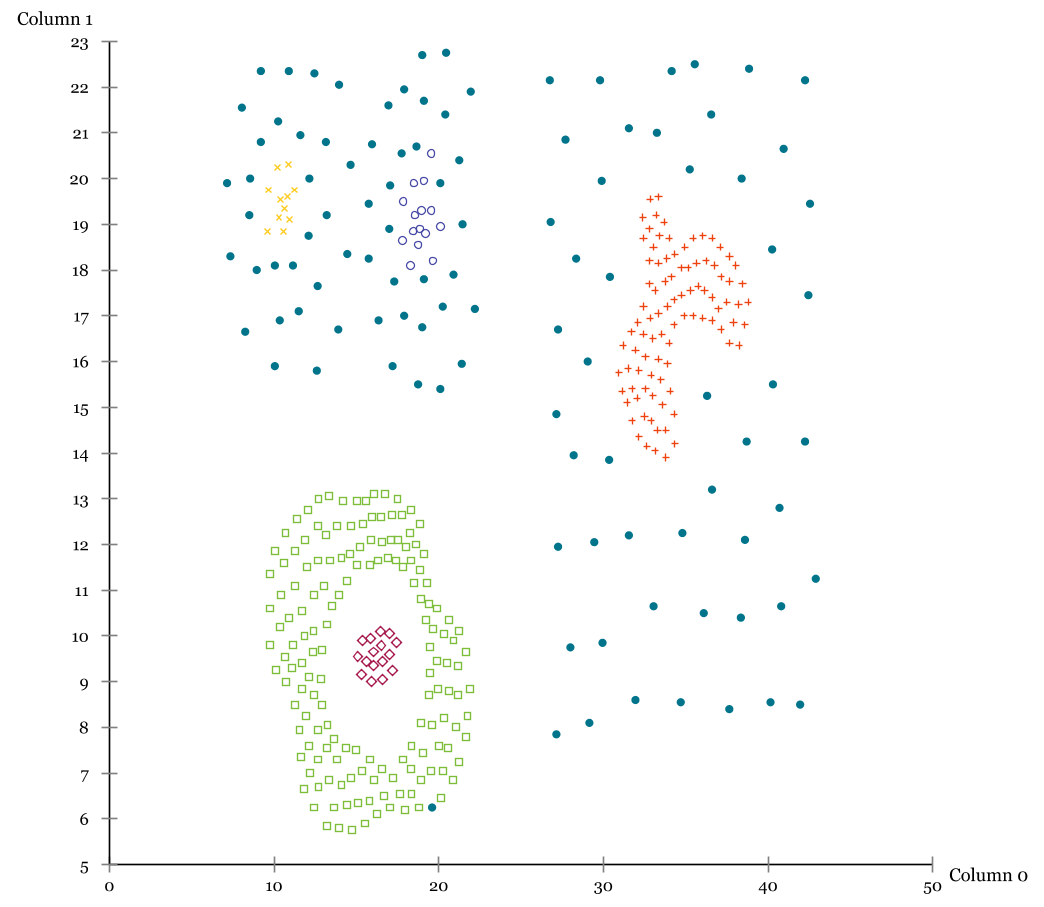
We plot a K-distance graph and look at the “elbow” or “knee”. In Elki we use KNNDistancesSampler with k=MinPts-1.

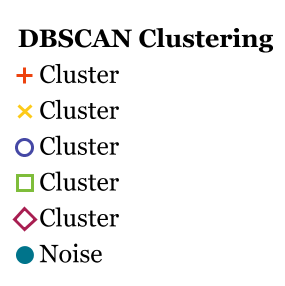


We see that the elbow lies around 0.8, so we use this as our epsilon.

Putting all this together:

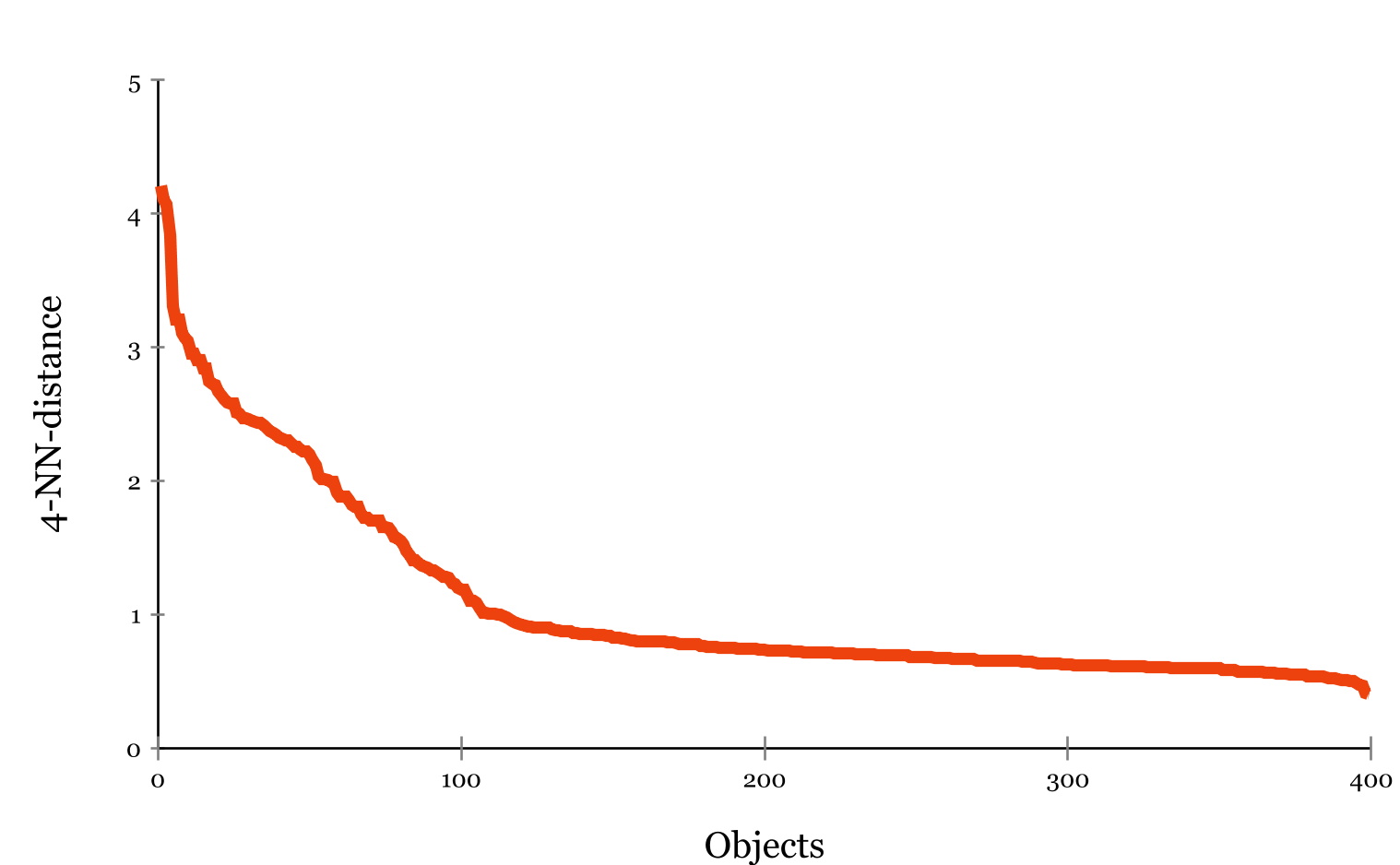
Eps=0.8, MinPts=4





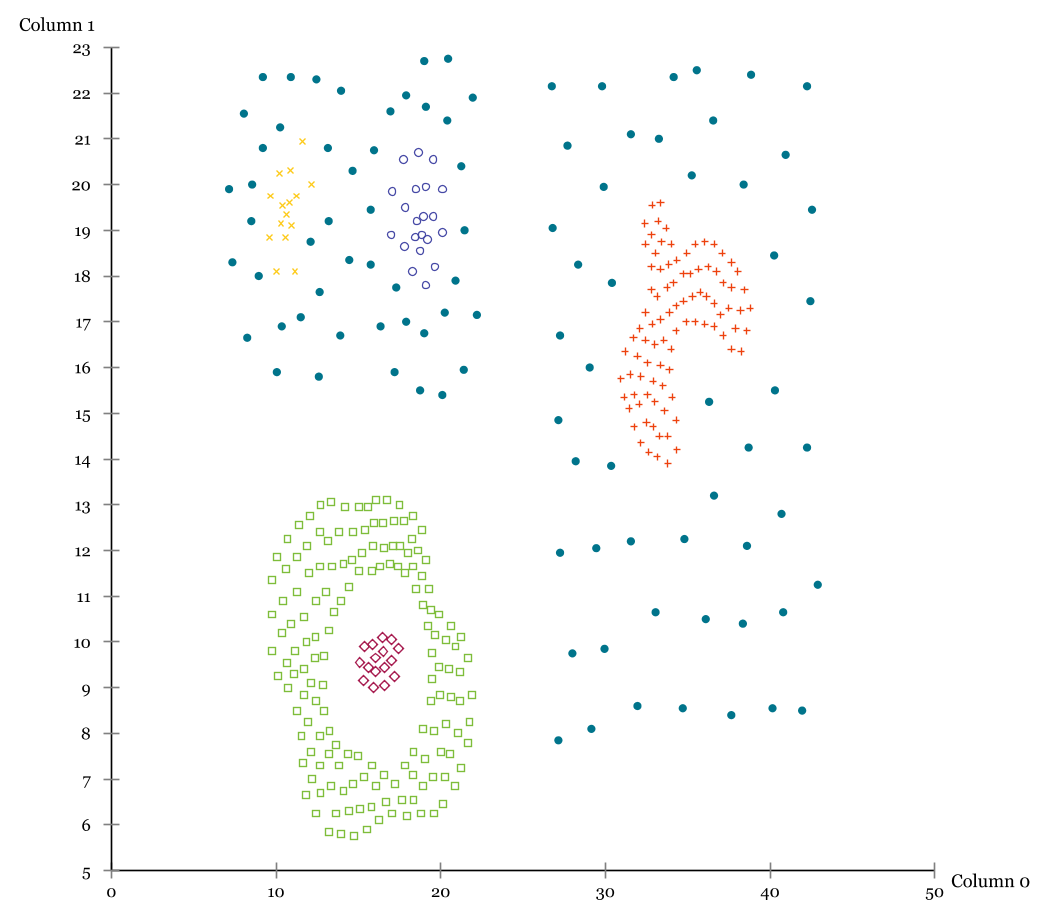
We try to increase MinPts to 5.

Adjusting epsilon for MinPts=5



Eps=1

Now performing DBSCAN with eps=1 and MinPts=5



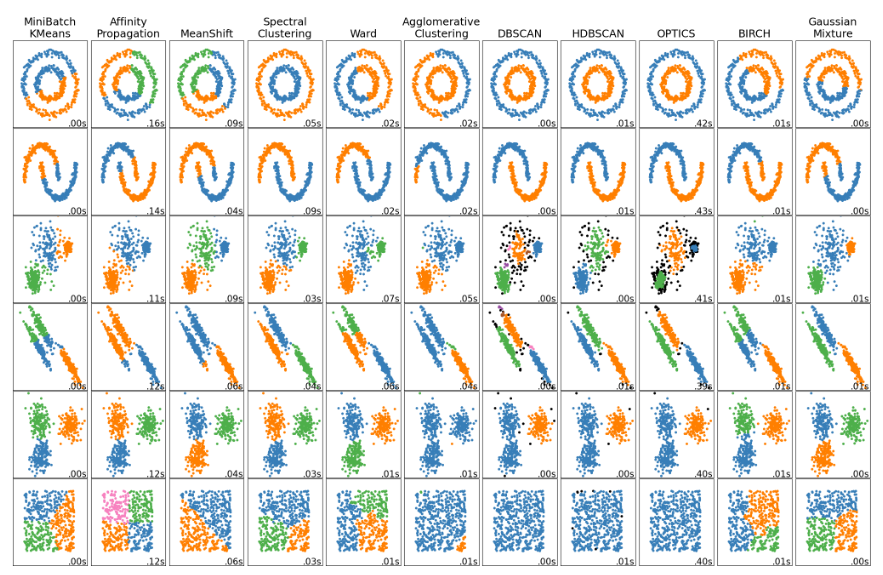
Her kan man evt. have et slide med sammenligning af minpts 4 og 5 og deres respektive epsilon. Hvilken passer bedst og hvorfor? Forskelle i hvilke punkt der kommer med/ud? Osv osv.

OPTICS

Our last algorithm is OPTICS. This…

**Comparison of our chosen algorithms**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithm | Parameters | Scalability | Uses | Geometry (metric used) |
| K-Means | Number of clusters | Very large n\_samples, medium n\_clusters with MiniBatch Code | General-purpose, even cluster size, flat geometry, not too many clusters, inductive | Distances between points |
| DBSCAN | Neighborhood size | Very large n\_samples, medium n\_clusters | Non-flat geometry, uneven cluster size, outlier removal, transductive | Distances between nearest points |
| OPTICS | Minimum cluster membership | Very large n\_samples, large n\_clusters | Non-flat geometry, uneven cluster sizes, variavle cluster density, outlier removal, transductive | Distances between points |



SciKit: <https://scikit-learn.org/stable/modules/clustering.html>