Agglomerative clustering

Aggumulative clustering is a bottom-up approach to clustering. The algorithm starts with each datapoint as a cluster, and then merges the two closest clusters iteratively until the desired number of clusters is reached, according to some distance metric.

the algorithm runs acording to the following pseudo-code:

Input:

- data: A set of n data points, where data = {x1, x2, ..., xn}
- distance_function: A function that calculates the distance between two clusters, for example, d(ci, cj) denotes the distance between clusters ci and cj.

Output:

- clusters: A set of clusters obtained from the agglomerative clustering algorithm.

Initialization:

Initialize n clusters, each containing a single data point: $ci = xi$ for $i \in \{1 \ldots n\}$. Compute the distance matrix D with D[i, j] = distance_function(ci, cj) for all pairs of clusters.

Agglomerative Cluster(K, data x, distance_function f):

While the number of clusters is greater than 1, do the following:

1. Find the two closest clusters ca and cb based on the distance matrix D:

$$c_a, c_b = argmin_{c_i, c_j} D[i, j]$$

1. Merge the closest clusters c_a and c_b into a new cluster c_new:

$$c_{new} = c_a \cup c_b$$

2. Update the distance matrix D to reflect the new distances between the merged cluster c_new and the remaining clusters: $D[new, k] = f(c_{new}, c_k)$ for all clusters ck in the remaining clusters

Remove the rows and columns corresponding to c_a and c_b from D.

Output:

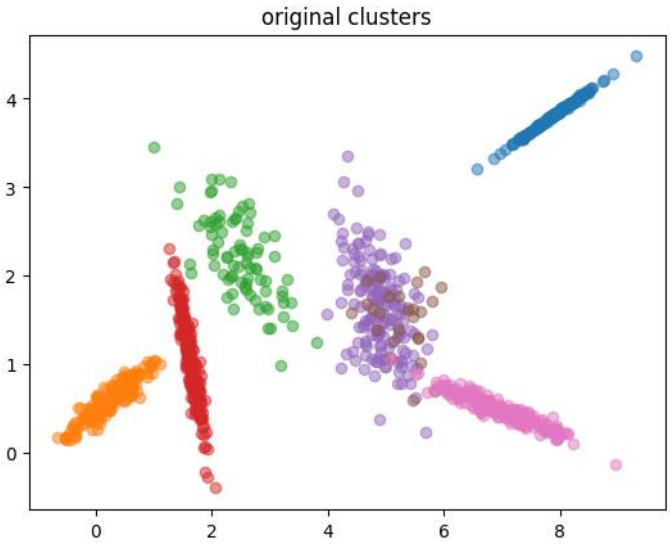
Return the final set of clusters after the algorithm terminates.

generate the data

parameters:

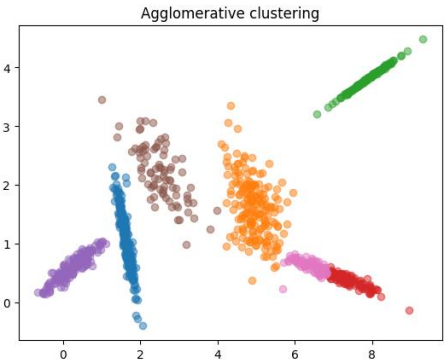
```
K = 7
d = 2
N = 1000
```

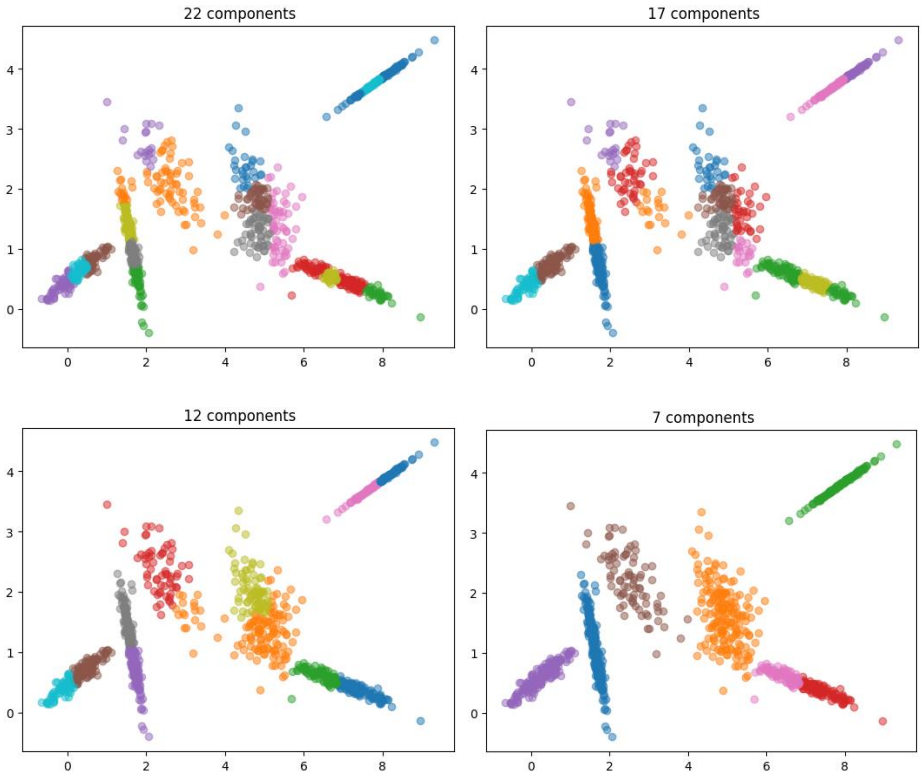
plot the data



try the algorithm on a dataset

here we will try the algorithm on the data we generated above, and see how it performs. it is a dataset consisting of 7 clusters, 1000 2D data points, and the clusters are separated by a mean distance of 6, which creates a noticeable distinction. we will try to cluster the data into 7 clusters, and see how the algorithm performs.





<Figure size 640x480 with 0 Axes>

different distance metrics

in the agglomerative clustering algorithm, we need to define a distance metric between two clusters for finding the two clusters to unify in step 1. note that the distance metric is not

the same as the distance metric between two points $d_{i,j}$, rather it is a distance metric

between two cluster of datapoints $d_{metric}(C_i, C_j)$, as we would like to find the two clusters to unify in this step. choosing linkage metric shuld reflect the nature of the data and the specific characteristics of clusters we expect to find. There is no one-size-fits-all answer, and one should try different linkage methods and compare their clustering outputs using his own validation methods.

There are several options for this metric, and we will discuss four of them, as presented in class, and try each on the same dataset

d_{min}

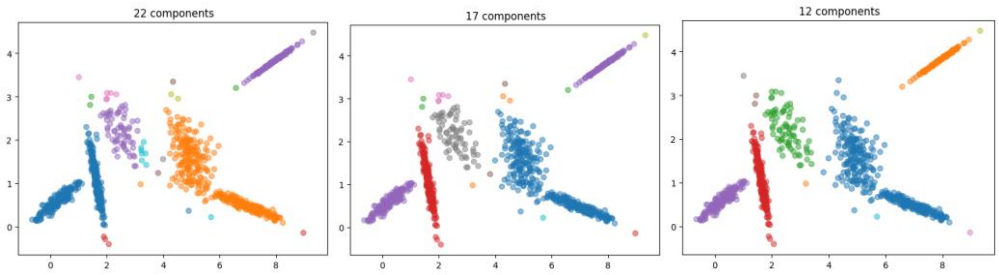
this distance metric is also known as the single linkage distance. It is defined as the distance between the two closest points in the two clusters. Formally, it is defined as:

$$d_{min}(c_i, c_j) = min_{x_i \in c_i, x_j \in c_j} | \quad |x_i, x_j| \quad |$$

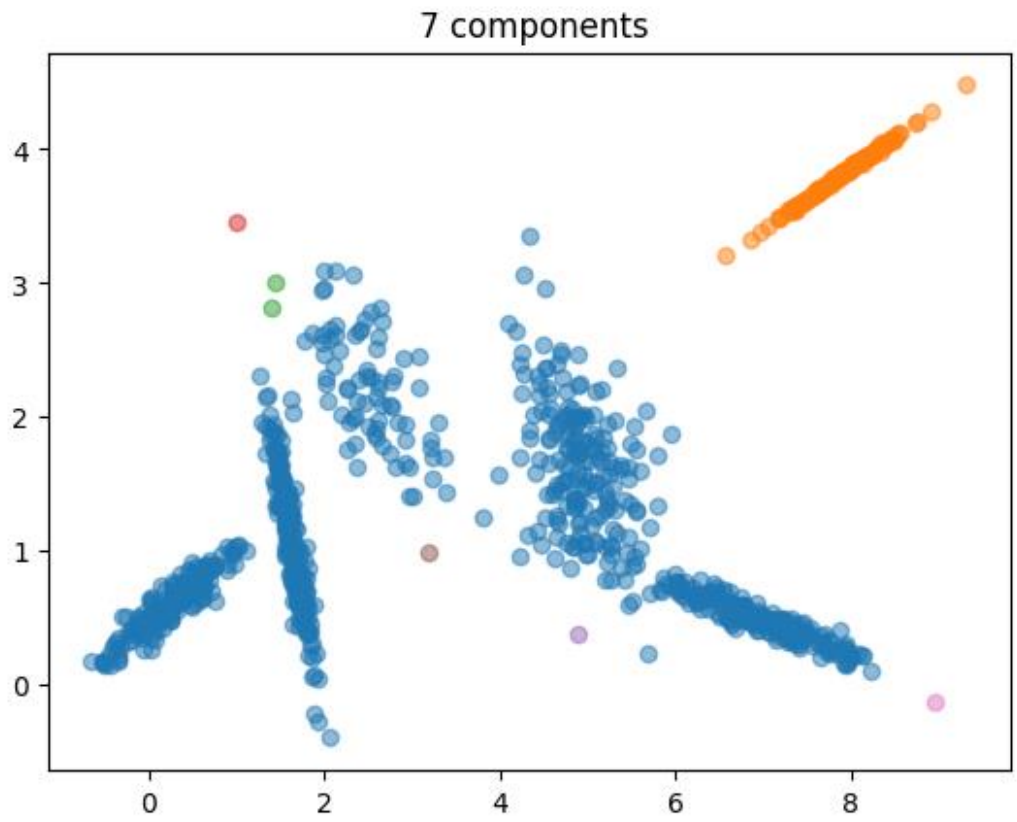
advantages : It can detect elongated clusters and is less sensitive to outliers. disadvantagae : It suffers from the "chaining" effect, as we will se in the next sections, where clusters can be connected as long chains, leading to long and straggly clusters.

try the distance measure

```
clusters = agglomerative_cluster_plot_steps(data,K, linkage='single')
```




```
<Figure size 640x480 with 0 Axes>
```



results

we applied the single linkage method to a 7-cluster dataset, resulting in the merging of four original clusters and the allocation of some outliers to their own clusters. Single linkage's sensitivity to elongated clusters and its preference for small, isolated clusters contributed to these clustering outcomes.

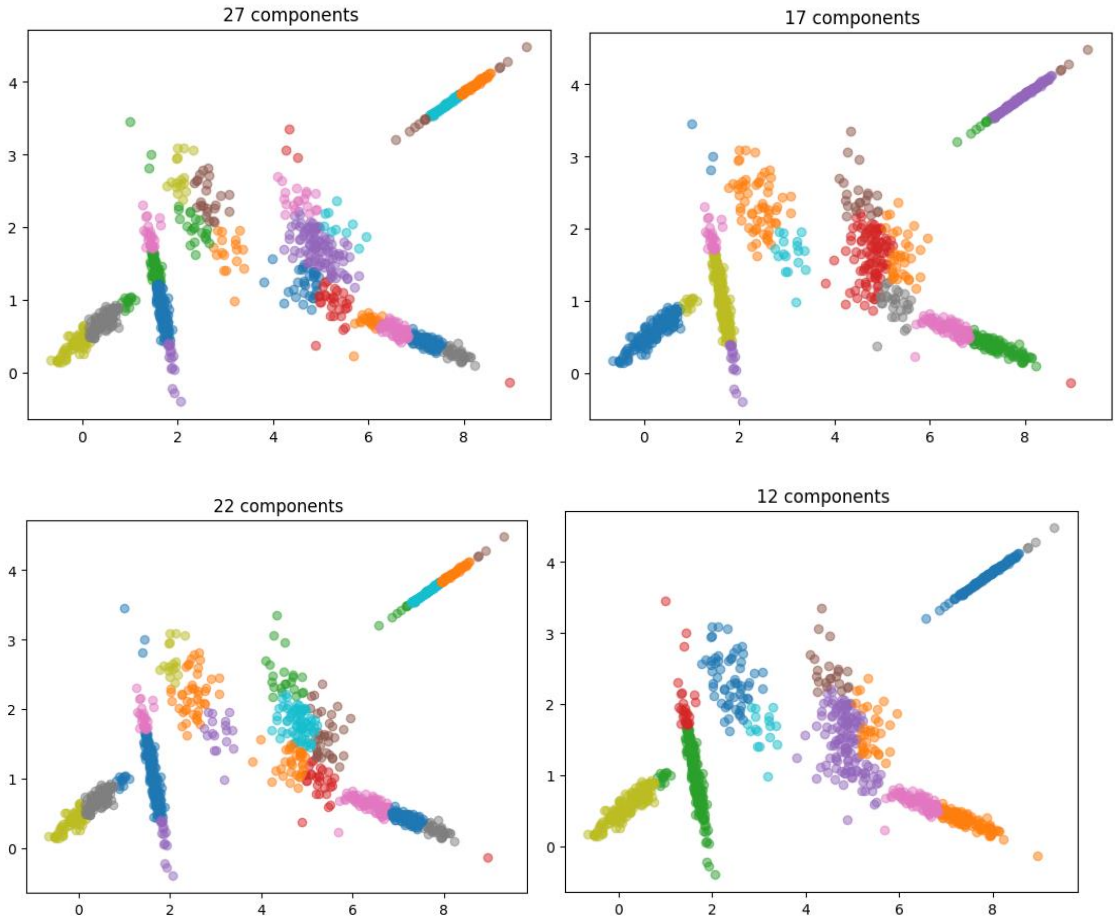
d_{max}

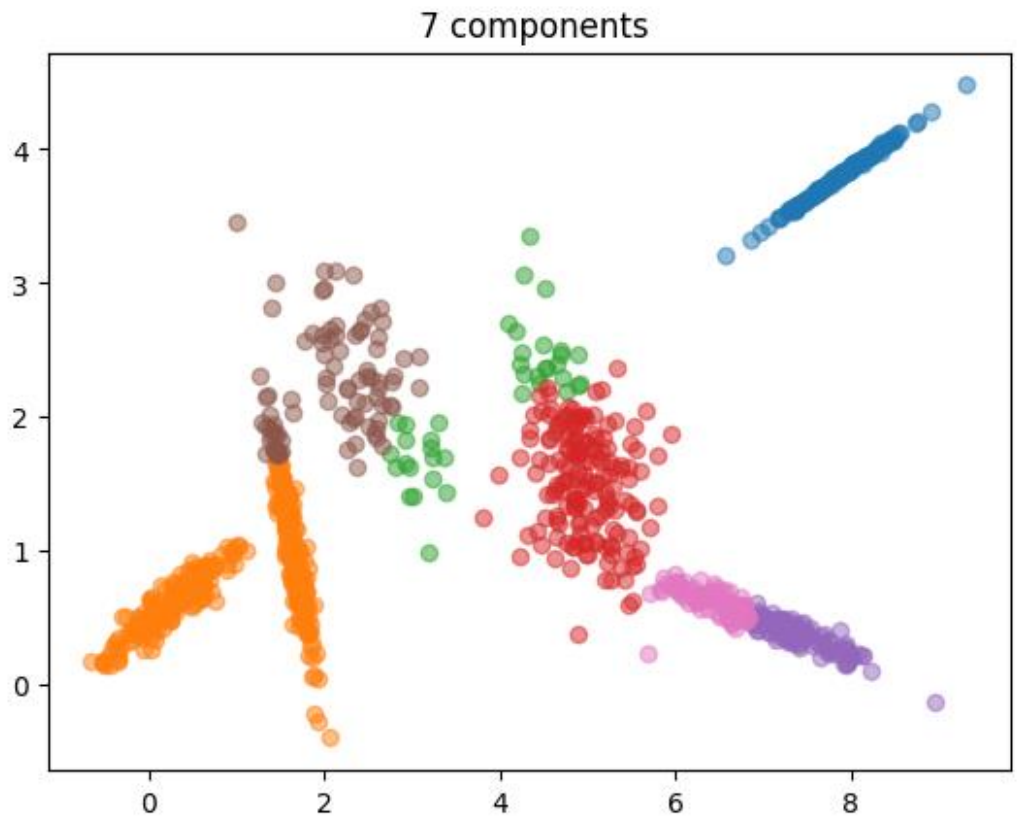
this linkage uses the maximum distances between all observations of the two sets. it seems counterintuitive to use the max distance, but we calculate it in order to merge clusters whose two farthest data points have the smallest distance.

$$d_{max}(C_i, C_j) = max_{u \in C_i, v \in C_j} dist(u, v)$$

Pros: It is less affected by outliers and is capable of detecting compact and spherical clusters. Cons: It can create clusters of relatively equal diameter, and it may not perform well for elongated or irregularly shaped clusters.

run the algorithm






```
<Figure size 640x480 with 0 Axes>
```

results

here we employed the complete linkage method on the same 7-cluster dataset, which yielded the formation of elongated clusters separated into distinct "stripes." Complete linkage's preference for compact and spherical clusters contributed to this clustering pattern, emphasizing the cohesive nature of the resulting stripes.

d_{avg}

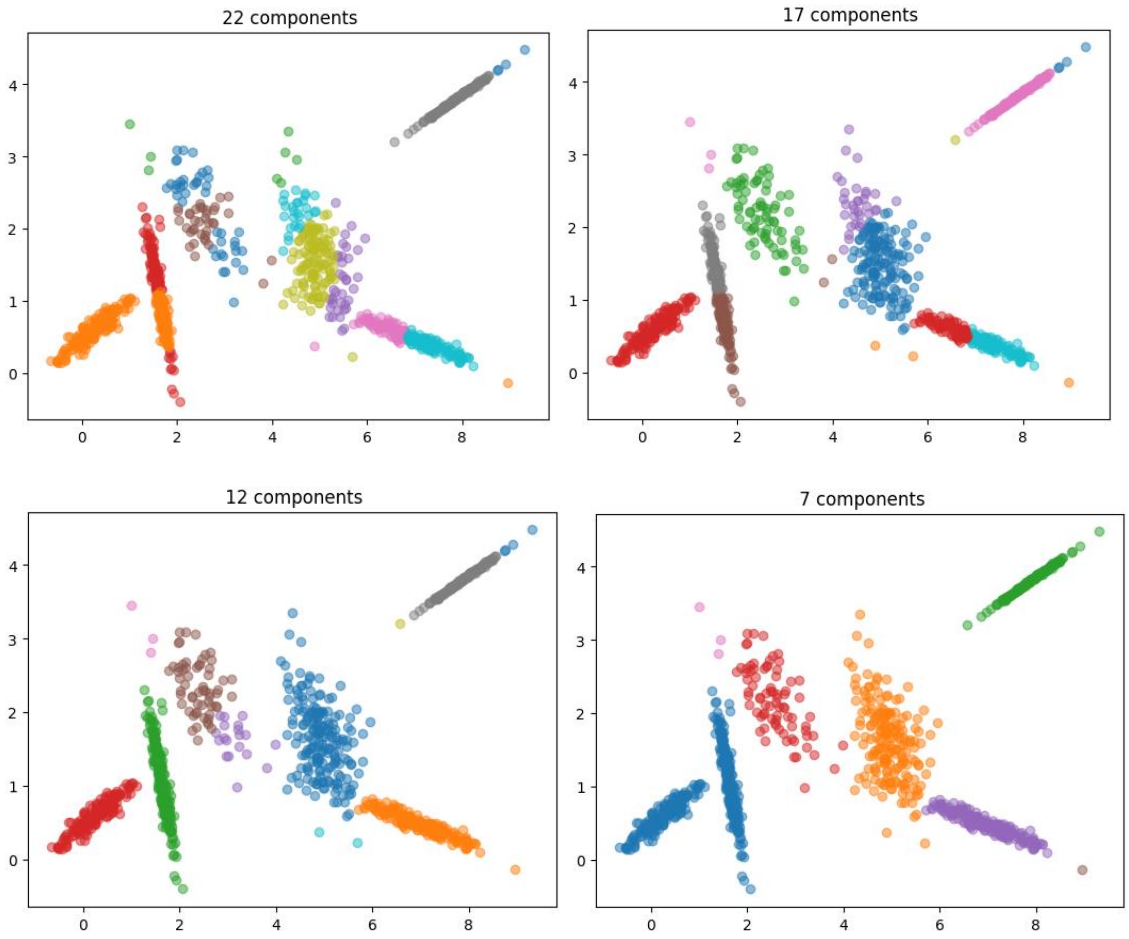
in this metric, we merge clusters based on the average proximity of all pairs of data points from the two clusters.

$$d_{avg}(C_i, C_j) = \frac{1}{n_i \cdot n_j} \sum_{u \in C_i} \sum_{v \in C_j} d\,ist(u, v)$$

advantage: It strikes a balance between single and complete linkage and is less sensitive to outliers. disadvantage: It can still suffer from the "chaining" effect, but to a lesser extent compared to single linkage.

```
clusters = agglomerative_cluster_plot_steps(data,K,'average')
```

```
<Figure size 640x480 with 0 Axes>
```




```
<Figure size 640x480 with 0 Axes>
```

d_{mean}

in this metric, we use the norm of the distance of means of each two cluster as our linkage metric, when the idea is to ultimately try to minimize the variance within each cluster when merging two clusters. It calculates the increase in the total sum of squares after

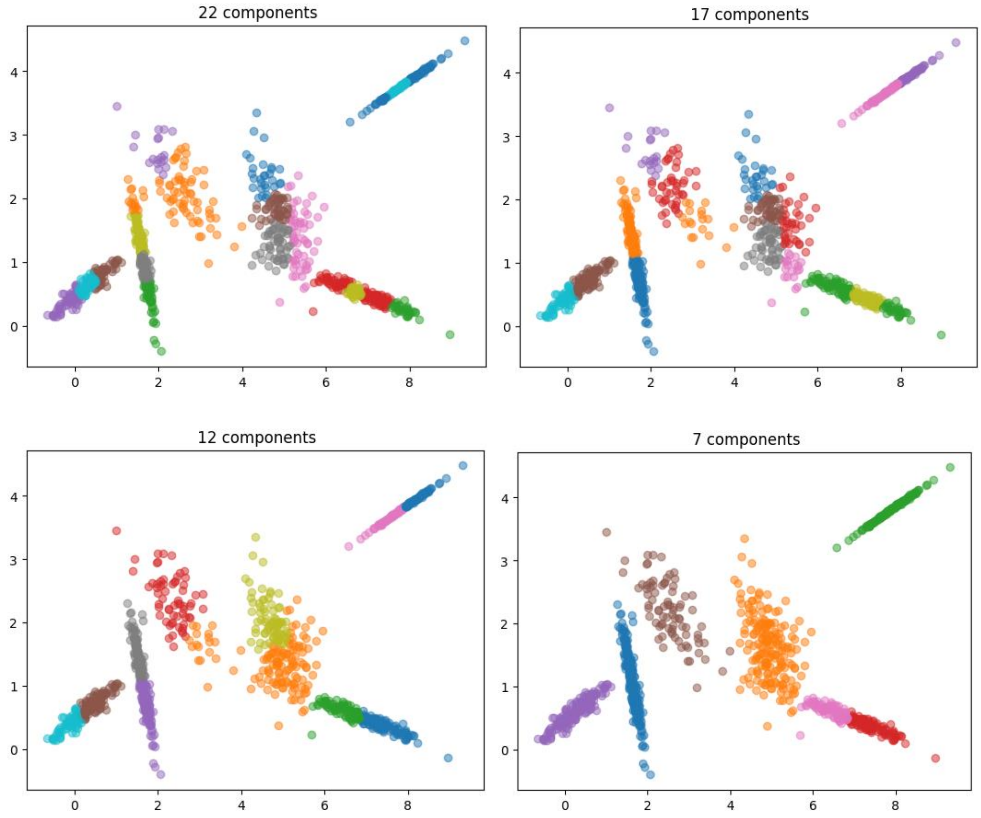
merging two clusters. there are two common ways to calculate d_mean, normalized, and un normalized.

$$d_{mean}(C_i, C_j) = ||c_i - c_j||$$

$$d_{mean}'(C_i, C_j) = \alpha \cdot ||c_i - c_j||$$

where - $ \alpha = \sqrt{\frac{n_i \cdot n_j}{n_i + n_j}}$ Pros: It tends to produce more balanced and compact clusters. Cons: It is sensitive to outliers, and the resulting clusters may not always be visually apparent.

run the algorithm with mean distance metric




```
<Figure size 640x480 with 0 Axes>
```

results

with the mean metric, the clustering results on the 7-cluster dataset exhibited remarkable performance, nearly reproducing the original clusters. However, it encountered challenges when dealing with two blended together clusters, resulting in their merging, it also split one of the clusters into two halves due to its sensitivity to the variance minimization criterion. mean metric perfoemed best on this dataset

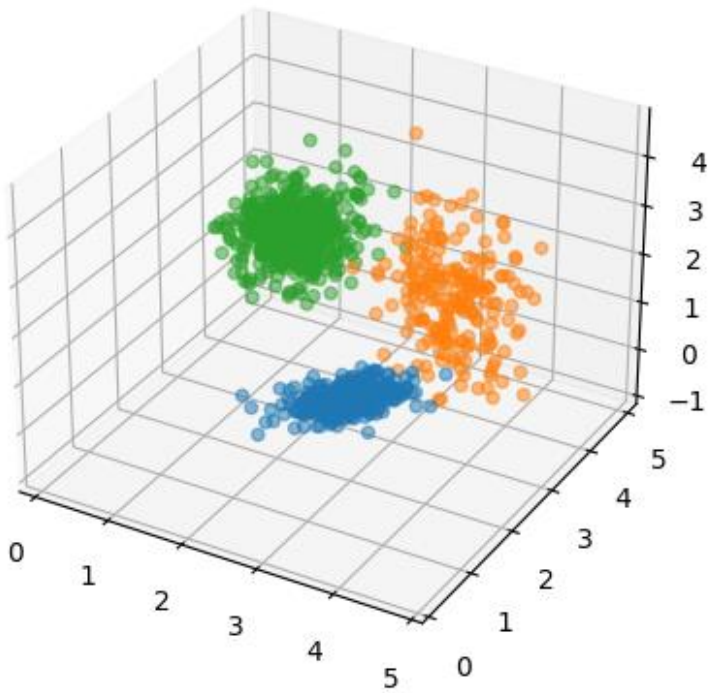
2nd Dataset: 3D data

here we will use a simple 3D dataset, and try to cluster it into 3 clusters. we will try the different distance metrics, and compare the results. different from before, we will see how the algorthm deals with 2 or more clusters that are close to each other, or even collide.

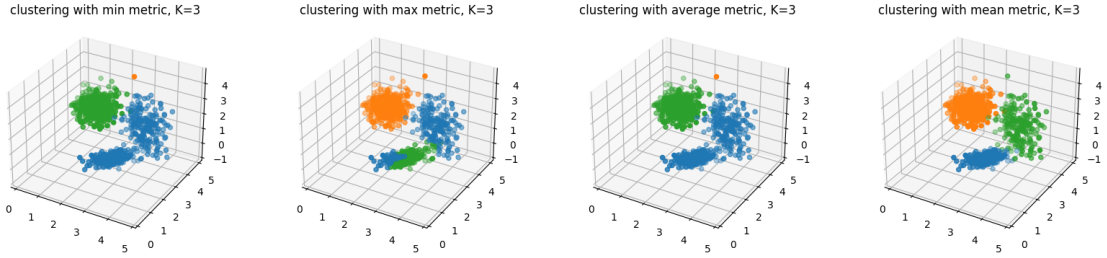
Generate the data

```
K = 3
d = 3
N = 1000
distance = 4
```

plot the data



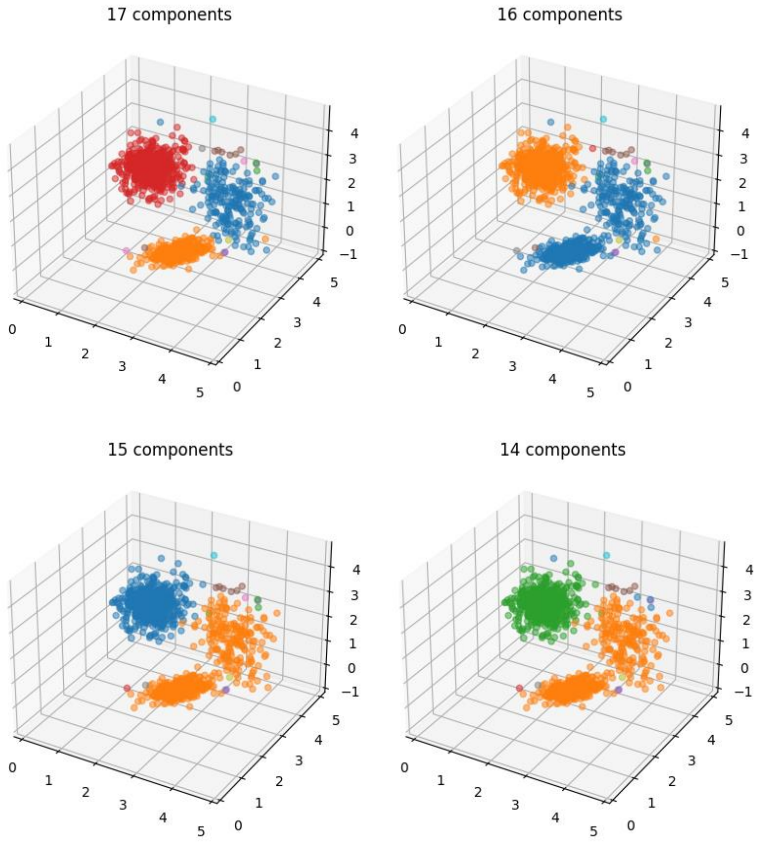
run the algorithm



what happend in the 'min' metric?

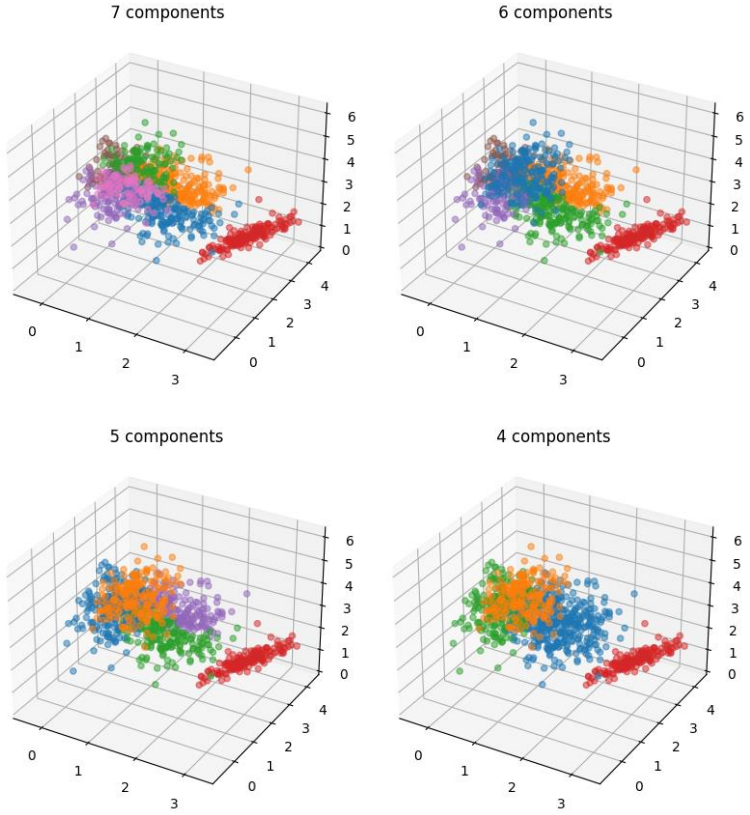
let's investigate the reason for the unification of the two clusters

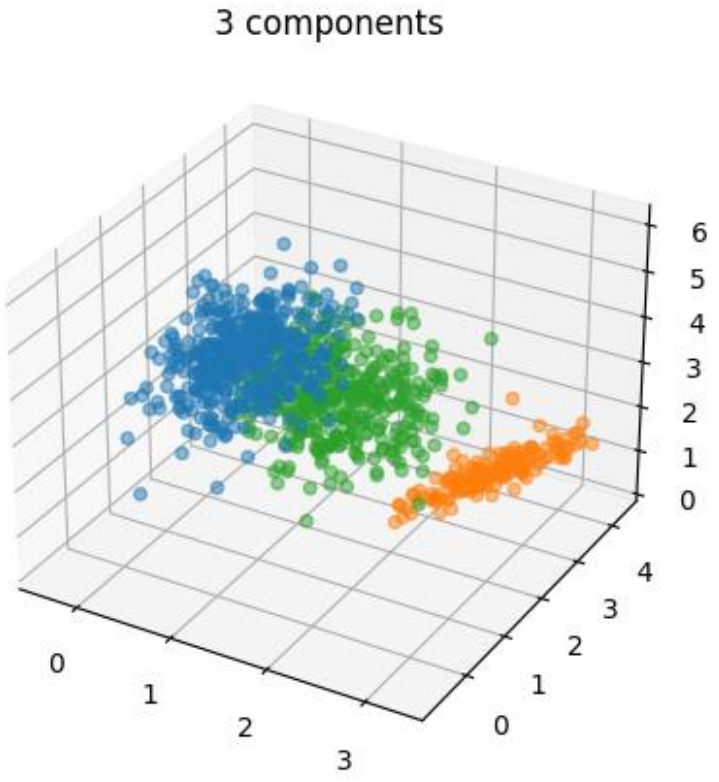
```
steps =  [17,16,15,14]
```



it seems that the two clusters are unified on the 17'th step (counting from the end) due to the proximity of two datapoints of the two, demonstrating the effect discussed earlier. outling points, however remain as isolated clusters

the progress of the mean metric



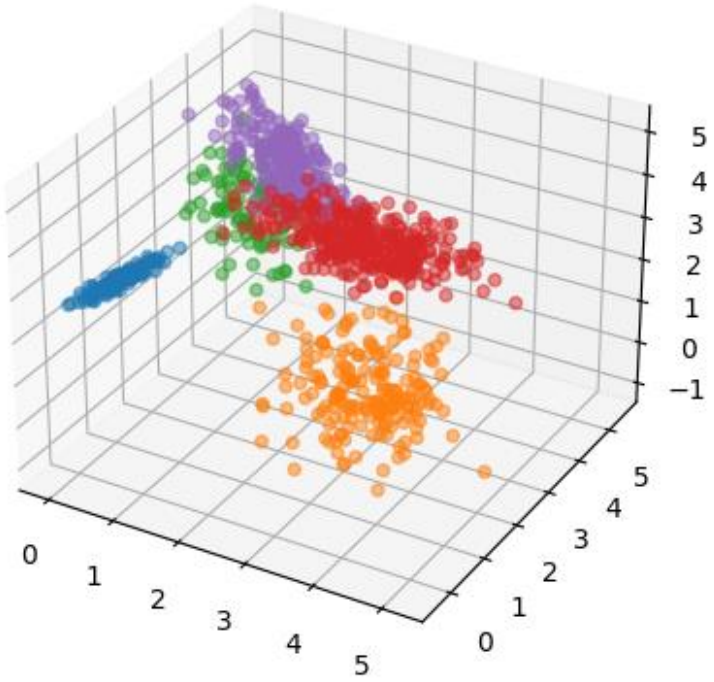


results

we see that in opposed to the min metric, in this case clusters remain balanced in size and shape, allowing the algorithm to converge to almost the exact original clustering

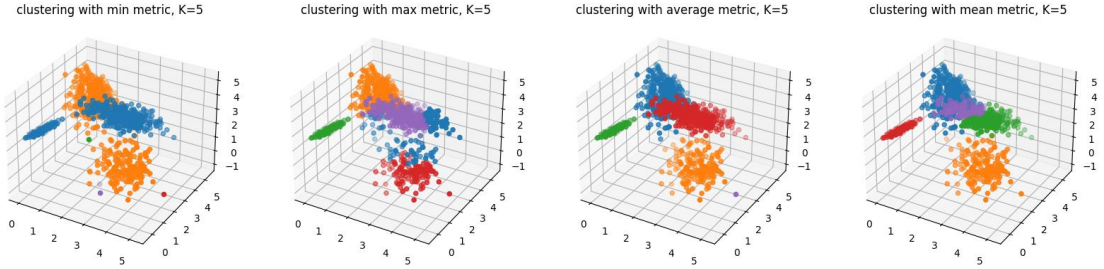
3rd Dataset: dense clusters of various sizes

```
K = 5
d = 3
N = 1000
distance = 4
```



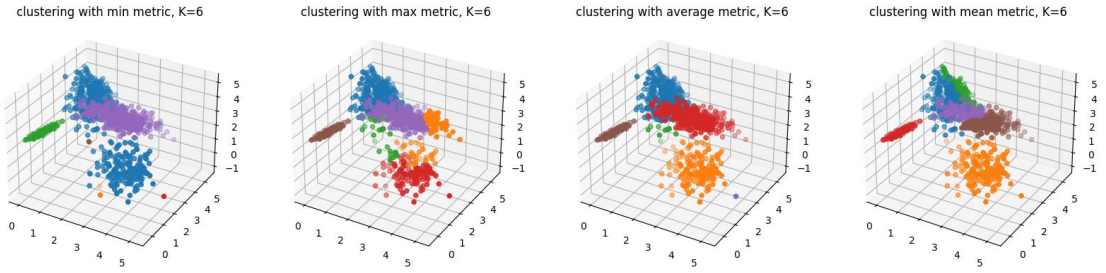
run the algorithm

```
compare_linkage_metrics(data3,K)
```



False unifications, False splits

here we see the trade-off between false unifications and false splits is a fundamental aspect in hierarchical clustering algorithms, including single linkage, complete linkage, and Ward's linkage. It revolves around the sensitivity of each linkage method to different cluster shapes and structures in the data. The choice of linkage method in hierarchical clustering involves a trade-off between false unifications and false splits. Single linkage, being more sensitive to chaining, is more likely to merge clusters that should remain separate (false unifications). In contrast, complete linkage and Ward's linkage, which focus on compact and spherical clusters or minimizing variance, respectively, may encounter false splits.

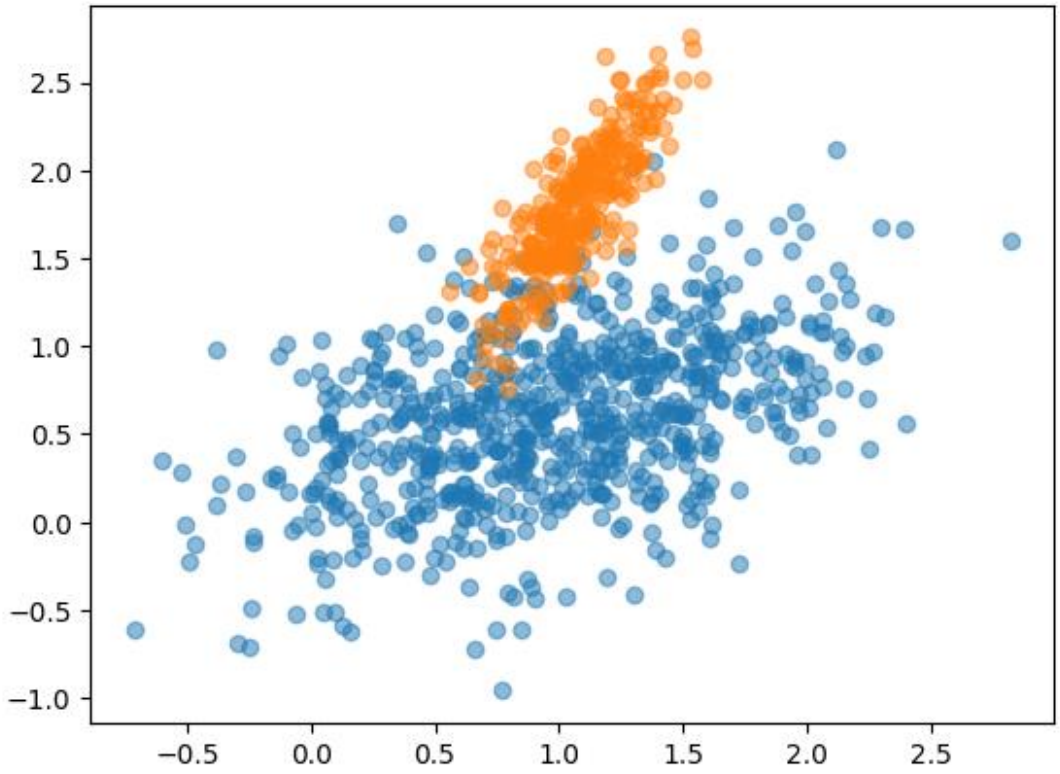


results

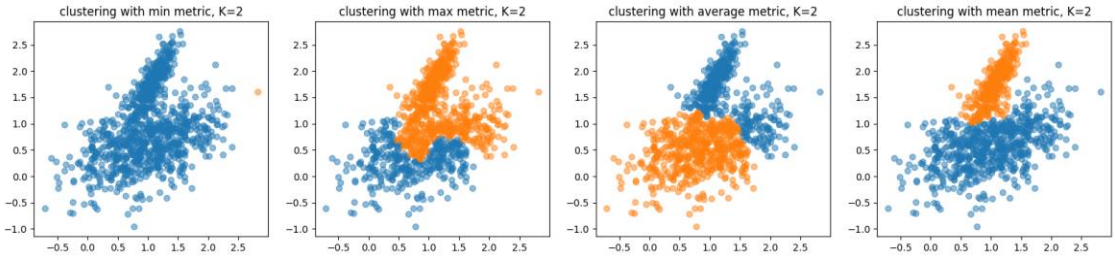
however, trying a larger k value might reveal some of the false unifications, the more improbable the unification is, the further on it will happen in the algorithm, and the easier it is to detect.

4th Dataset: 2D data with 2 clusters

generate the data



```
compare_linkage_metrics(data4,K)
```

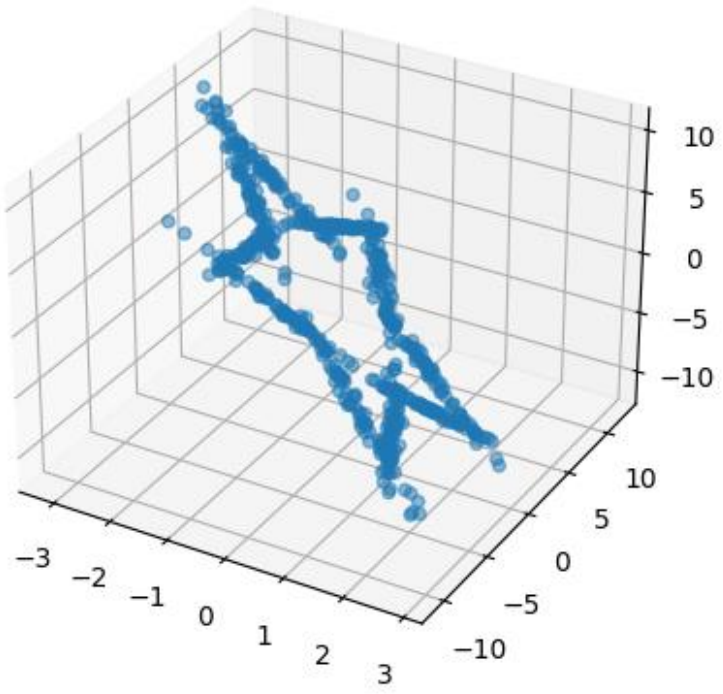


results

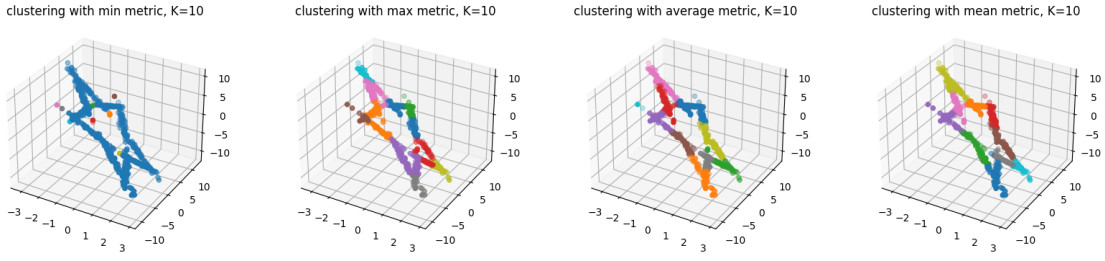
we tried all of the metrics on another dataset, one of the dofficulties with the avarage metric can be ssen, approx 1/5 of the points of the "big" cluster, are on avarage closer to the points of the other cluster, making the algorithm assign them with the wrong label

5th Dataset: star shaped data

let's try our star shaped data from the first notebook, with 10 line-shaped gaussians, forming a polygon



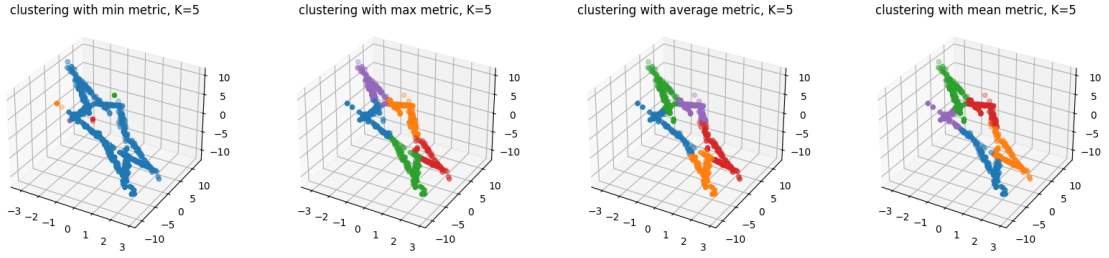

```
compare_linkage_metrics(d,10)
```



results

we could, at this point, expect the min metric to assign almost the whole dataset to the same cluster, and assign their own clusters to each of the outliers due to the linkaing effect, but interesting results emerge whith the other metrics , we see the 'max' metric has a tendecy to pick on th edges of the star, as distinct clusters, and the average/ mean metrics take lines as clusters.

we know the data consists of 10 original clusters, but an interesting rwesult emerges when we lower the number of clusters down to $K \leftarrow 5$



the sides of the star get clustered almost exactly to their original assignments, but in pairs, according to their location on the star sides a surprising accurate result for all of the metrics, especially for the max metric, which clusterd the original clusters (although in pairs) in astounding accuracy