Wrocław University
of Science and Technology

Research phase

Comparison of clustering methods

Authors: *Piotr Kędra, Wiktor Kilijan, Bauyrzhan Marat*
Subject: *Advanced Topics in Artificial Intelligence*
Guided by: *MEng Artur Zawisza*

Contents

1. Research plan

I. Motivation

Clustering is a solution to many different problems concerning numerous different areas of today's world. Cluster analysis is widely used in market research for a better understanding of relationships between different groups of consumers. What is more, it supports medicine in case of making critical decisions [11], as well. It is also beneficial for social science, for instance, it is used to identify areas where there are greater incidences of particular types of crime, therefore, managing law enforcement resources can be more effective. That is why we decided to explore different clustering methods for the purpose of determining their potential.

II. Goal

Our goal is to implement and then test several different clustering methods.

III. Plan

In this project, we are planning to implement 6 clustering methods: K-Means Clustering, DBSCAN, Mean-shift, Ward's Method, Single-link clustering, Complete link clustering. Moreover, our team will compare this method of how they do with different datasets. Afterward, we promise to compare our clustering methods with different parameters and against sklearn implementation. Finally, we will execute our implementation and sklearn, record time, and make conclusions with the results of our experiments.

2. Datasets

For the purpose of experiments we used 4 different datasets. We wanted to evaluate how each clustering method is dealing with diverse data. In this paper we will use three two-dimensional datasets and one multidimensional:

1) Aggregation dataset [6] (***Fig.*** *Aggregation)*
   It contains 788 numbers of samples, each sample consists of x (range 2 to 38) and y (range 1 to 29) features. Both these features are float values.

2) Compound dataset [6] (***Fig.*** *Compound)*
   It contains 399 numbers of samples, each sample consists of x (range 2 to 44) and y (range 6 to 22,5) features. Both these features are float values.

3) R15 dataset [6] (***Fig.*** *R15)*
   It contains 600 numbers of samples, each sample consists of x (range 3 to 17,5) and y (range 2,5 to 15,5) features. Both these features are float values.

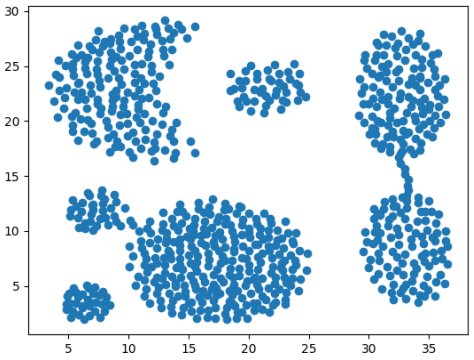


Fig. *Aggregation*

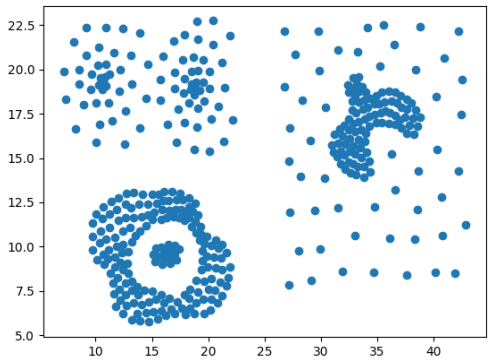


Fig. *Compound*

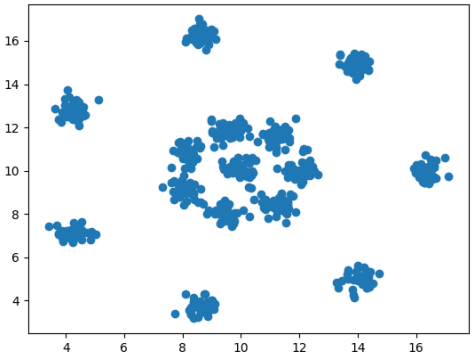


Fig. *R15*

4) Wine dataset
This dataset was found on Kaggle [12]. It contains 13 features (***Table.*** *Wine dataset*). There are no missing values.

Table. *Wine dataset*

Feature	Min value	Max value	Data type
Alcohol	11	14,8	Float
Malic_Acid	0,74	5,8	Float
Ash	1,36	3,23	Float
Ash_Alcanity	10,6	30	Float
Magnesium	70	162	Integer
Total_Phenols	0,98	3,88	Float
Flavanoids	0,34	5,08	Float
Nonflavanoid_Phenols	0,13	0,66	Float
Proanthocyanins	0,41	3,58	Float
Color_Intensity	1,28	13	Float
Hue	0,48	1,71	Float
OD280	1,27	4	Float
Proline	278	1680	Integer

3. Research environment

Below you can find information regarding what tools have we used in order to complete this paper.

1) Programming Language
   We decided to choose Python because of its simplicity, implementing algorithms in this language feels like writing pseudocode. In addition python provides us with lots of useful libraries listed in the next subsection.

2) Libraries
   Our methods were implemented without much help from listed libraries, we just used numpy and math packages for simple calculations, because of its efficiency. We also used pandas DataFrame objects for reading data from csv files. Tools that u can find in this subsection were mainly used for drawing the results and for comparing if ours implementations work correctly in comparison with sklearn[4] implementations of the same methods. Below you can find packages that we used:

   - Matplotlib
   - Pandas
   - NumPy
   - Sklearn
   - Math

3) Integrated development environment (IDE)
   Pycharm is more user-friendly, powerful, and configurable as compared to other IDEs. It provides integration with git, has its own terminal and python console, provides support for various handy plugins, and a lot of useful keyboard shortcuts.

4. Measures

I. Davies–Bouldin index [7]

Davies–Bouldin index is an internal evaluation method to calculate how well a dataset has been clustered. The domain of values is above 0 and the closest to 0 value, the better clustering.

That index can be calculated with a following formula:

$$DB = \frac{1}{n}\sum_{i=1}^{n} max_{i \neq j}\left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right)$$

where:

n - is the number of clusters

c_i - is a centroid of a cluster

σ_i - is an average distance of all elements in a cluster to its centroid

$d(c_i, c_j)$ - is a distance between two cluster centroids

II. Silhouette score [10]

Silhouette Score is a metric used to calculate the goodness of a clustering technique. Its value ranges from -1 to 1.

1: Means clusters are well apart from each other and clearly distinguished.
0: Means clusters are indifferent, or we can say that the distance between clusters is not significant.
-1: Means clusters are assigned in the wrong way.

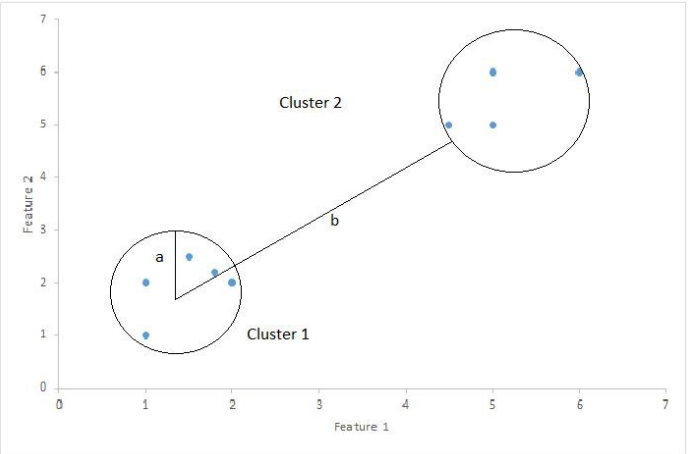


Fig. Silhouette score [10]

Silhouette Coefficient = (b-a)/max(a,b), where a = average intra-cluster distance i.e the average distance between each point within a cluster and b = average inter-cluster distance i.e the average distance between all clusters.

III. Jaccard index [2, 7]

The Jaccard index (Jaccard similarity coefficient) is a statistic used for sets comparison. Jaccard's coefficient measures the similarity between two sets and it is defined as the quotient of the intersection and union of sample sets (*A* and *B*):

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

The values given by the Jaccard rate are values of numeric numbers between 0 and 1. If the result is close to 1, the sample sets are relative to each other, while when it is close to 0, the sets are different.

5. Overview of chosen methods

I. K-means [1, 9]

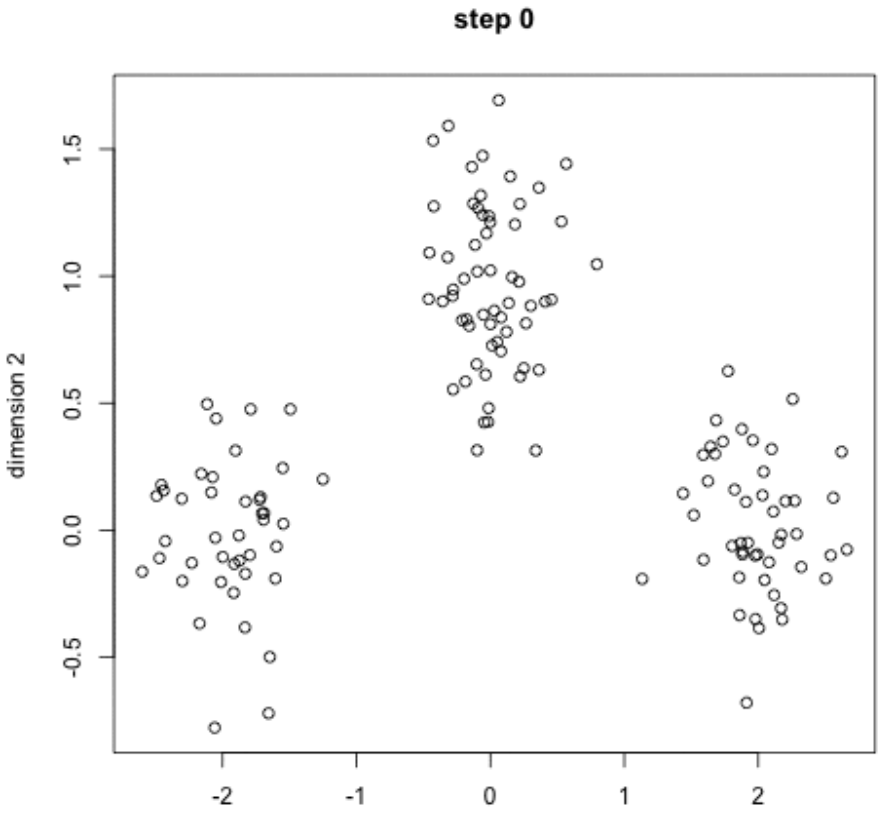
K-means is one of the most widely used centroid-based clustering methods. It is so popular mostly because of its simplicity and relatively good results. It is also very fast because most of the calculations are just counting distances.

K-means clustering algorithm consists of three main steps:

1. Starting cluster centers initialization - random selection of given number of cluster centers.
2. Computing the distance between points and cluster centers - for a given point, the assigned cluster is the one that cluster center is the closest.
3. Recomputing the cluster centers - the new cluster center is the mean of all points belonging to the cluster.
   Steps 2 and 3 are repeated until there are no changes.

Despite of the mentioned pros, there are also some disadvantages of K-means so it shouldn't be used in every case:

- a need to have a given number of clusters
- random clusters centers - the results may differ
- works poorly with irregular shaped clusters



Animation. *K-means [9]*

II. DBSCAN [1, 9]

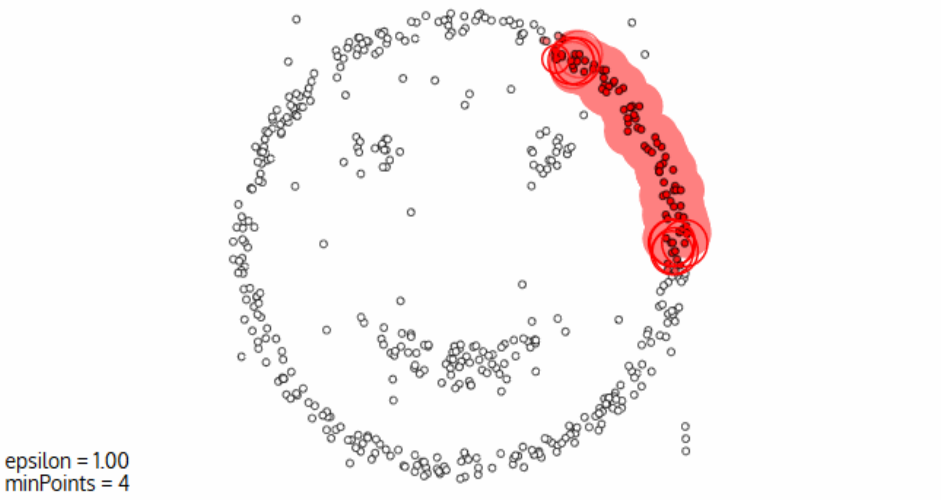
Density-based spatial clustering of applications with noise (DBSCAN) is a density-based clustering method. It is density-based in contrast to k-means so it does not use cluster centers but the idea is based on density - areas where the density of points is high become clusters.

1. Firstly not visited yet data point is selected.
2. If the number of neighbours is sufficient (higher than a given value) - it becomes a new cluster, else - is labeled as “noise”.
3. Points close to the selected point are also included to the cluster - this step is repeated until all possible points are added to the cluster.

Steps 1-3 are repeated until all points belong to a specific cluster or are labeled as noise.

A big advantage of DBSCAN is that it does not need a given number of clusters. Another one is that there are no points that are forced to belong to some cluster that they don’t fit in.

There are also some disadvantages of DBSCAN, such as a need to have at least two parameters given - number of neighbours not to be labeled as noise and the distance - maximal distance to another point when it may be counted as a neighbour. Another problem is that it has worse performance when clusters in a given dataset don’t have similar density.

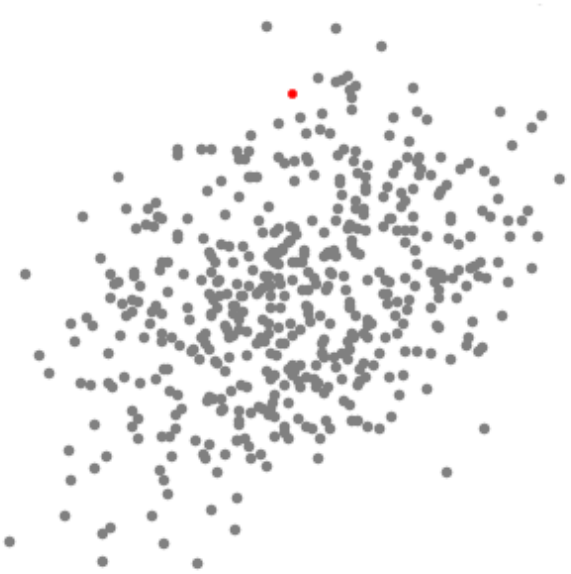


Animation. *DBSCAN [9]*

III. Mean-shift [8]

Mean shift clustering is a sliding-window-based algorithm that attempts to find dense areas of data points. It is a centroid-based algorithm, which works by updating candidates for center points to be the mean of the point within the sliding-window:

1. At every iteration, the window is shifted towards the region of higher density, calculated by the mean of points within the window
2. This step is repeated until there is no direction in what window can accommodate more points inside.
3. During the last iteration, near-duplicates are eliminated, and then each data point is assigned to the proper centroid.



Animation. *Mean-shift [9]*

IV. Ward's method

Ward's is a hierarchical clustering method. It is distinguished from the others by the use of an analysis of variance approach to estimate the distance between clusters. It aims to minimize the sum of squared deviations of any two clusters that may be formed at any stage. It is considered to be very effective, although it aims at creating clusters of small size. This method can be described in four steps:

1. We merge each pair of the cluster and estimate a centroid for these clusters (***Fig.*** *Ward's centroid*).
2. For each pair, we calculate the sum of the squared deviations of all the points in this centroid.
3. We pick the smallest deviations and we join these 2 clusters.
4. Repeat those 3 steps until 1 cluster remains.

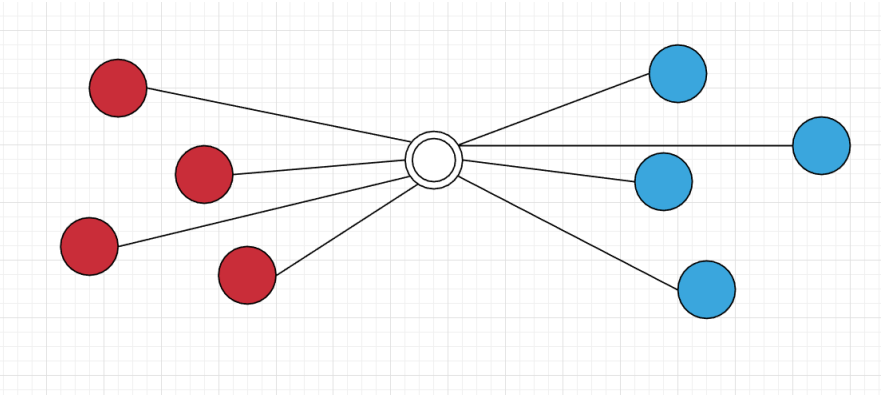


Fig. *Ward's centroid*

V. Single linkage [13]

Single linkage is one of the methods of hierarchical clustering methods. Single-linkage is the shortest distance between a pair of observations in two clusters. It can sometimes produce clusters where observations in different clusters are closer together than to observations within their own clusters. These clusters can appear spread-out.

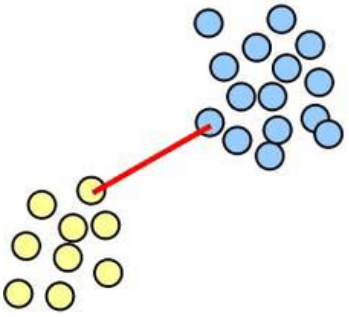


Fig. *Single linkage [14]*

VI. Complete linkage [13]

Complete linkage is one of the methods of hierarchical clustering methods. Complete-linkage is where distance is measured between the farthest pair of observations in two clusters. This method usually produces tighter clusters than single-linkage, but these tight clusters can end up very close together. Along with average-linkage, it is one of the more popular distance metrics.

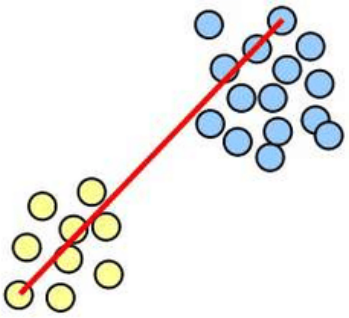


Fig. *Complete linkage [14]*

6. Comparison of methods

In this section you can see 3 different experiments. In the first section we evaluated each method with different parameters using our 4 datasets. Afterward, there are results of comparison of our implementations against sklearn ones. In the last section, there are information referring to execution time of our and sklearn methods.

I. Comparison based on different datasets

1. K-means

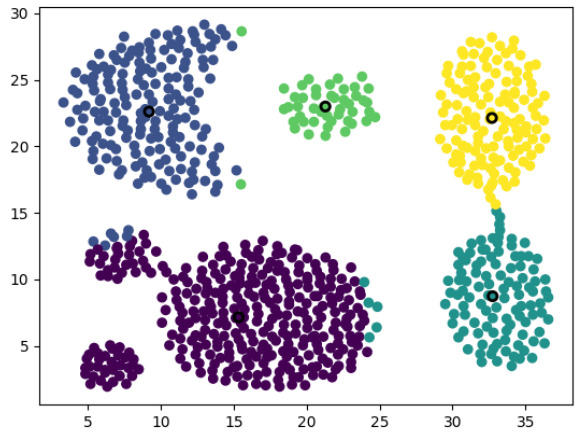


Fig. *K-means(k=5)*

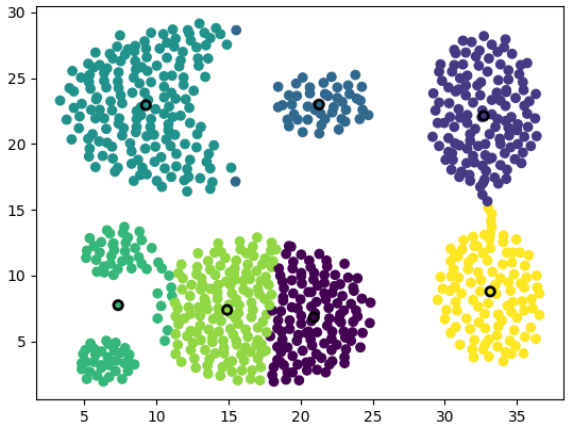


Fig. *K-means(k=7)*

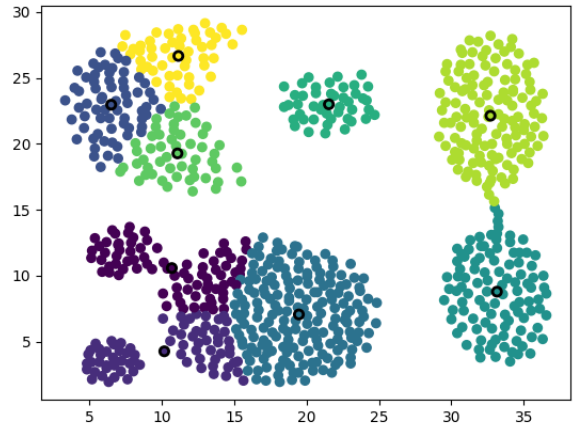


Fig. *K-means(k=9)*

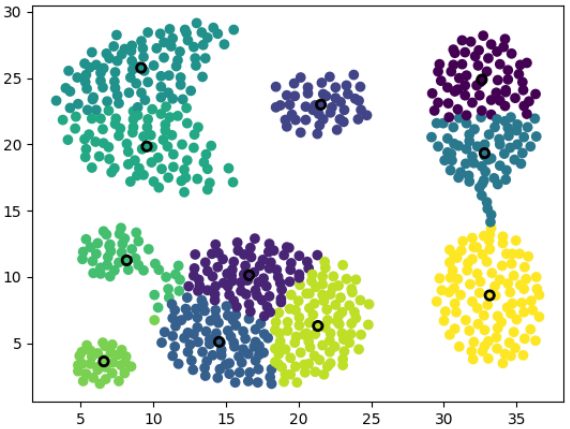


Fig. *K-means(k=11)*

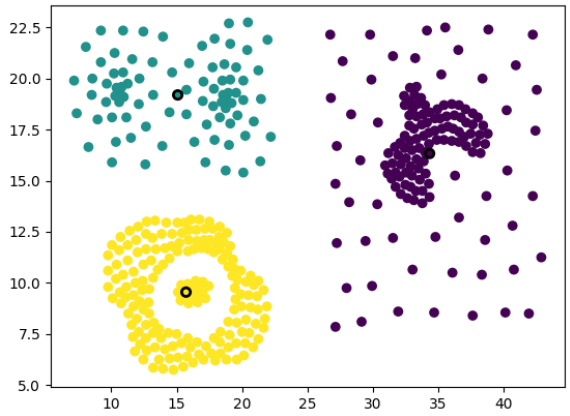


Fig. *K-means(k=3)*

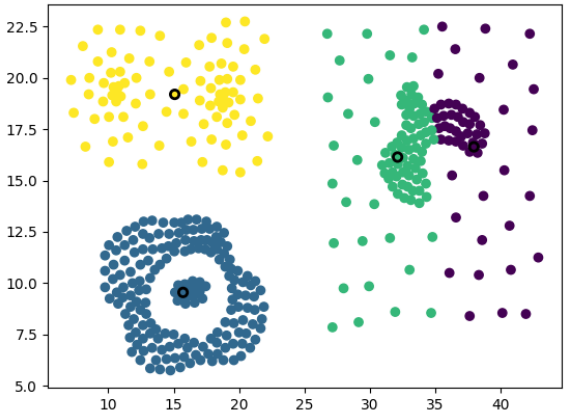


Fig. *K-means(k=4)*

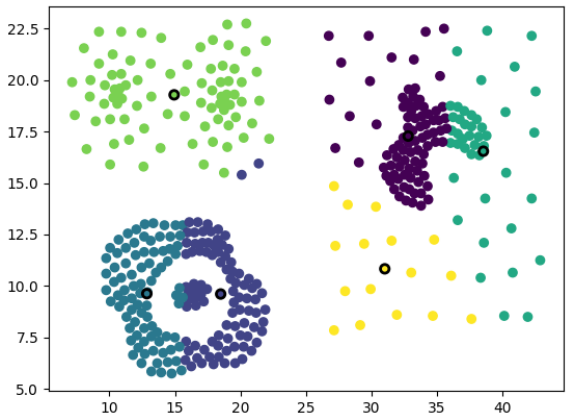


Fig. *K-means(k=6)*

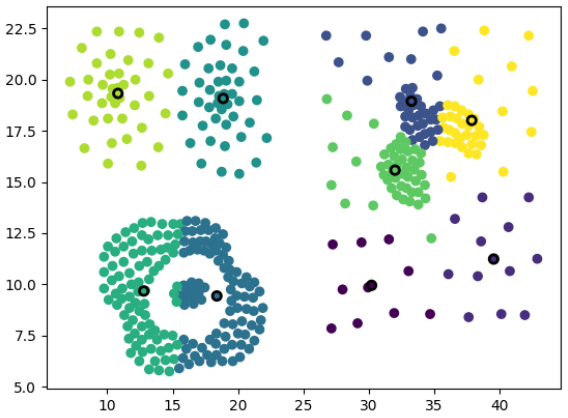


Fig. *K-means(k=9)*

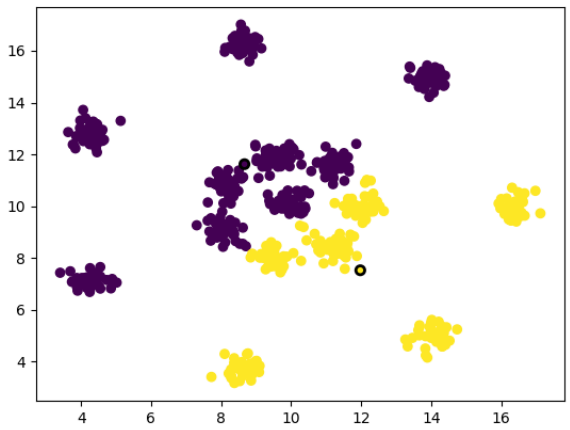


Fig. *K-means(k=2)*

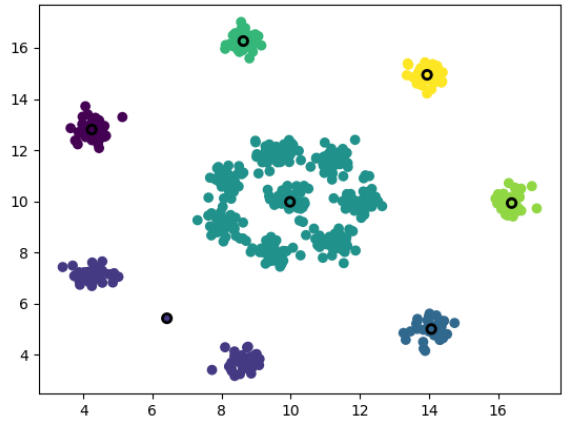


Fig. *K-means(k=7)*

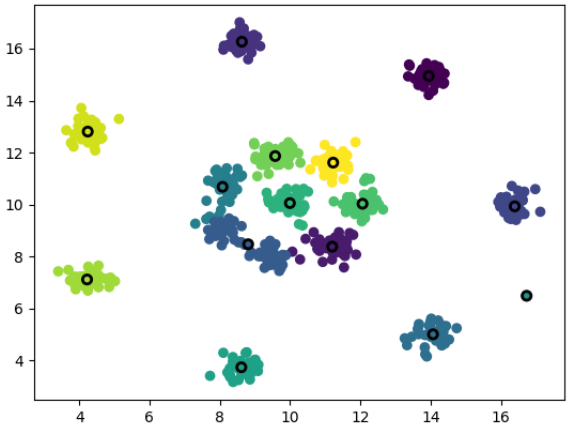


Fig. *K-means(k=15)*

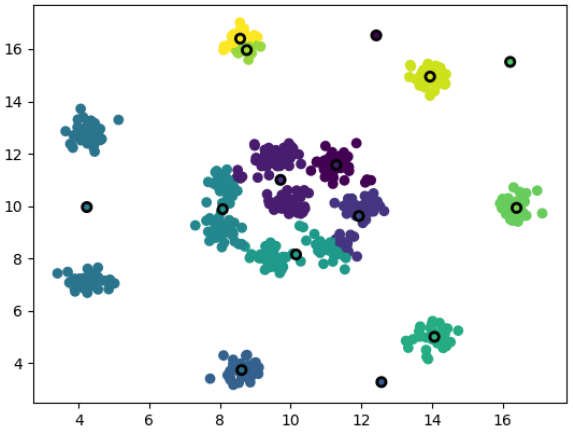


Fig. *K-means(k=15) 2*

Conclusions:

- In K-means it is very important to give a proper number of clusters because it may give very good results with right k and very bad results with not suitable one.
- Because of random initialization of cluster centers the results differ between runs with the same number of clusters given.

Selecting parameters:

K-means was run many times with different k parameters. After each run Davies–Bouldin and Silhouette indexes were measured to see which value of k is the best for each dataset. The results are shown on the following diagrams:

1)

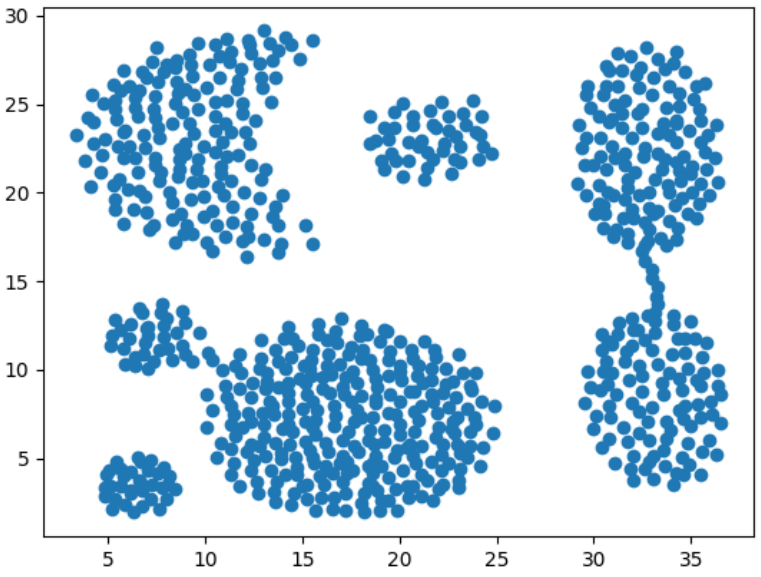


Fig. *Aggregation dataset*

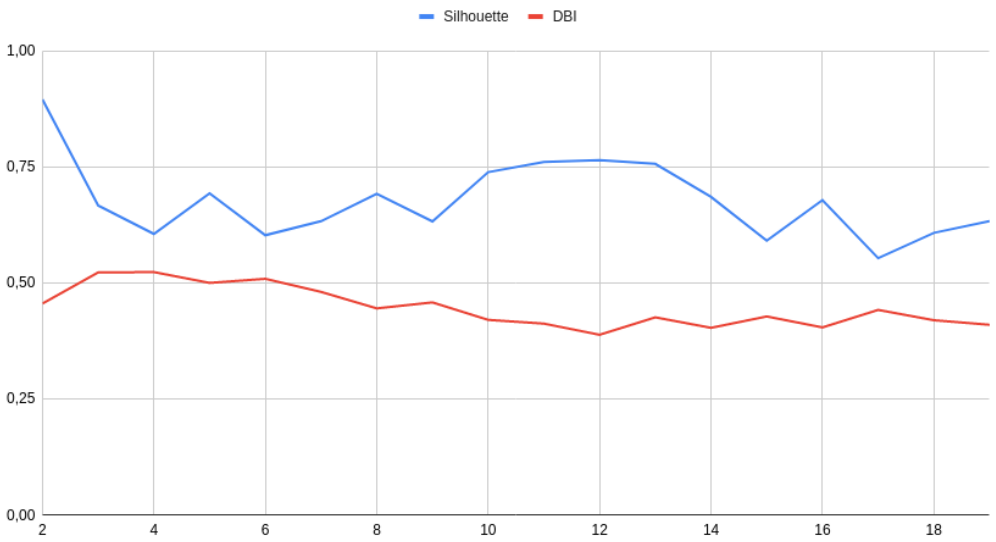


Fig. *Parameters score Aggregation*

Taking into consideration the fact that for value of Davies–Bouldin index lower is better and the best value of Silhouette index is the closest to 1, we can see that for this dataset the optimal number of clusters should be 12.

2)

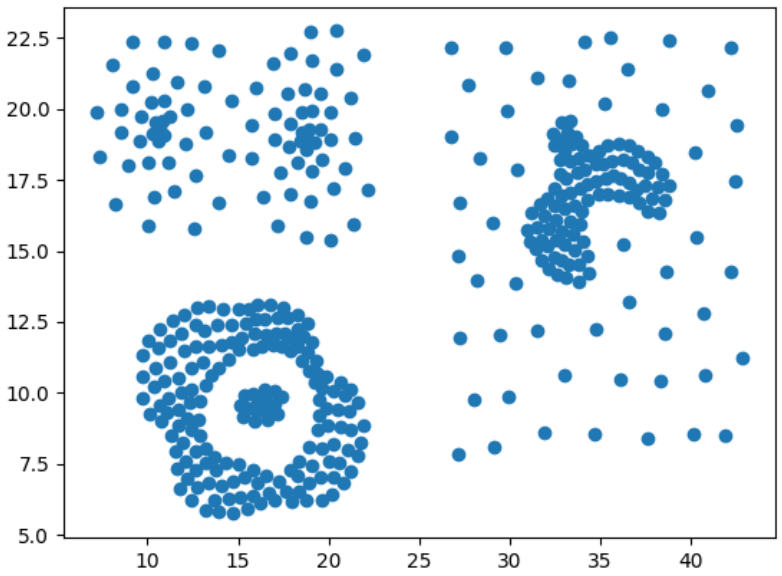


Fig. *Compound dataset*

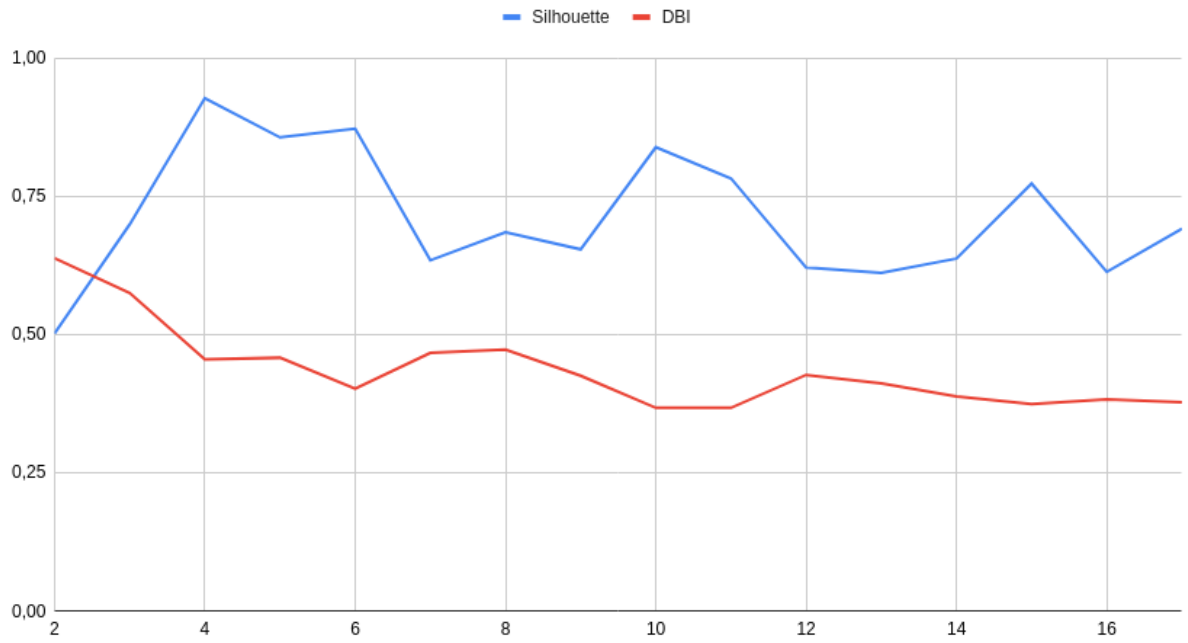


Fig. *Parameters score Compound*

In this case we can have a couple of candidates, such as 4, 6, 10 or 15.

3)

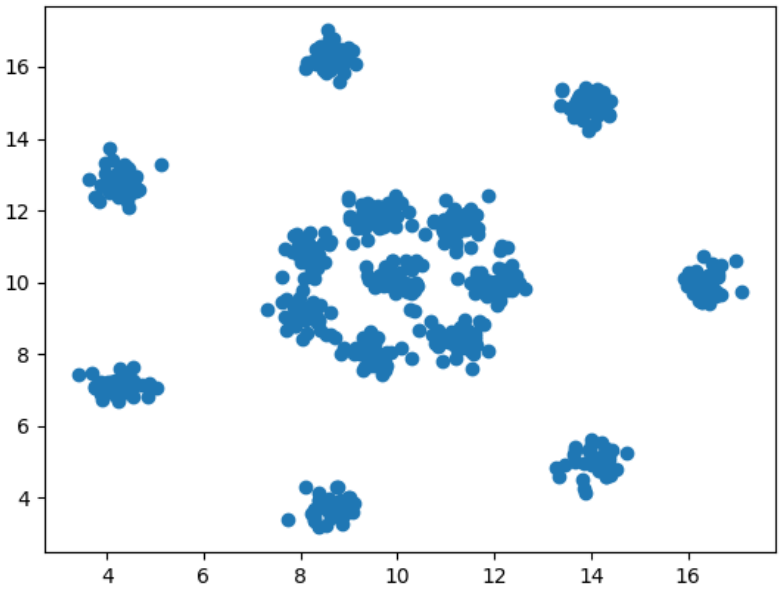


Fig. *R15 dataset*

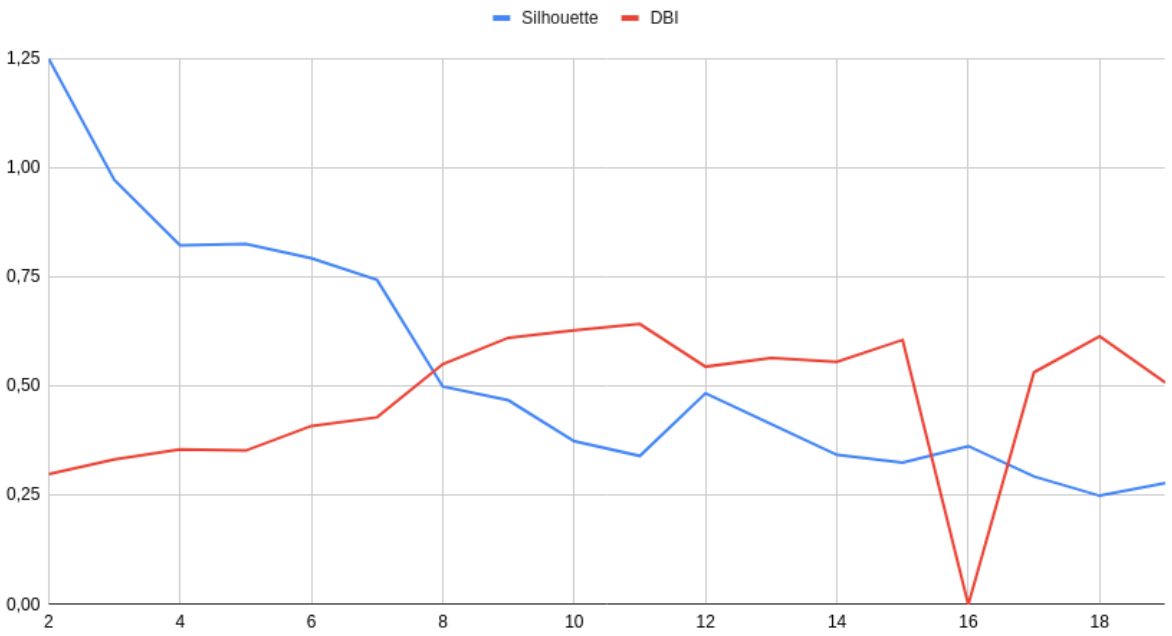


Fig. *Parameters score R15*

Here the possible candidates could be: 11, 15 or 18.

When the number of clusters is 16, Silhouette value seems to be equal to 0 - this is only our assumption, because in that case there was a cluster with only 1 point and to calculate the Silhouette index each cluster has to have at least 2 elements. We decided that in such cases it would be set to 0.

Conclusions:

- Using indexes like Davies–Bouldin or Silhouette may be helpful during selection of the values of parameters but they are not perfect.
- Such measurements should be used together with knowledge of domain experts or with visualisation of the data.

Selecting parameters on multidimensional dataset:

While working with two dimensional datasets we can visualize the values and more or less estimate how many clusters there should be or if the clustering worked well, but when one works with a multidimensional dataset it is not so easy. In such cases there is a must to base on indexes.

In our work we have a multidimensional dataset wine dataset and in the following section we are going to show how to select proper parameters when we cannot see the results on a diagram.

In this part sklearn implementation of K-means was used because: a) as was proven in “Our implementation vs sklearn” section our implementation gives similar results and b) sklearn’s performance is much better than ours.

We run K-means on the Wine dataset with k in range from 2 to 15:

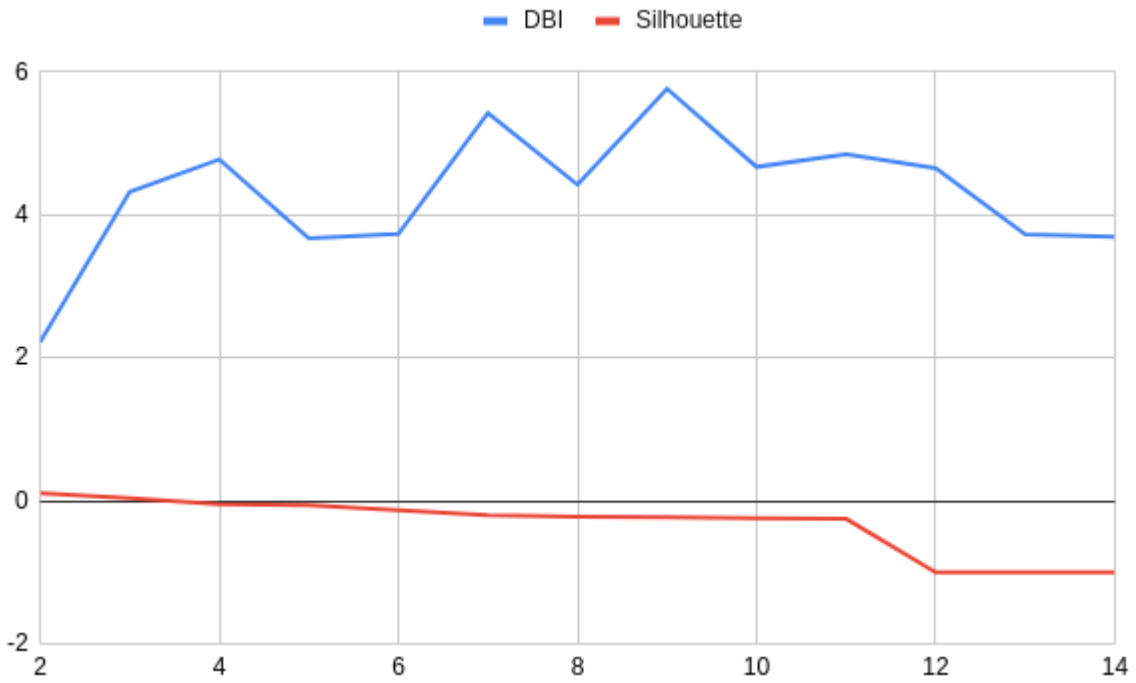


Fig. *Parameters score Wine dataset*

Based only on the Silhouette index we could say that it is the best to use the least possible number of clusters. Davies–Bouldin index also has the lowest (best) value when k=2 so if one wanted to divide the dataset into 2 parts, he should select this value. If more clusters are needed, the number of clusters equal to 5, 6, 8 could be selected.

2. DBSCAN

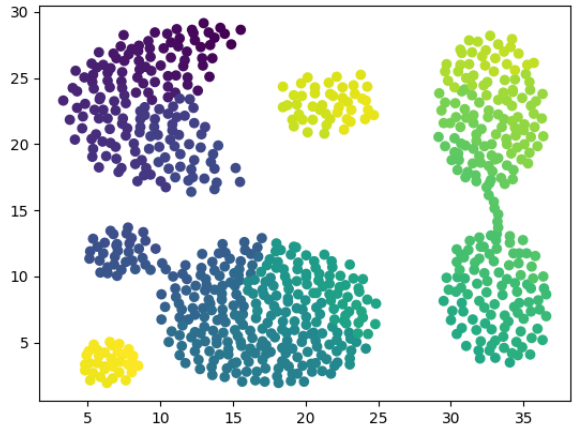


Fig. *DBSCAN (eps = 0.5)*
min_neighbours = 1
(clusters: 618, labeled as noise: 0)

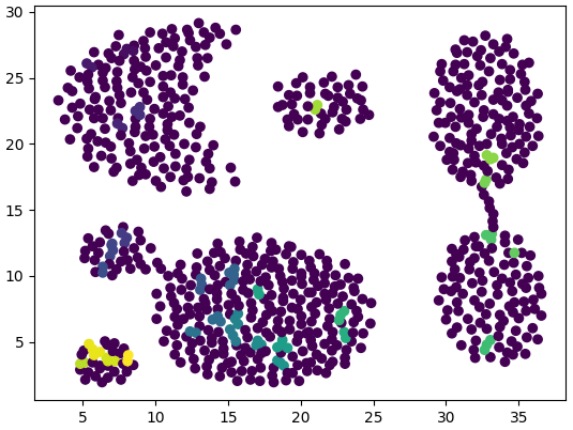


Fig. *DBSCAN (eps = 0.5)*
min_neighbours = 3
(clusters: 30, labeled as noise: 708)

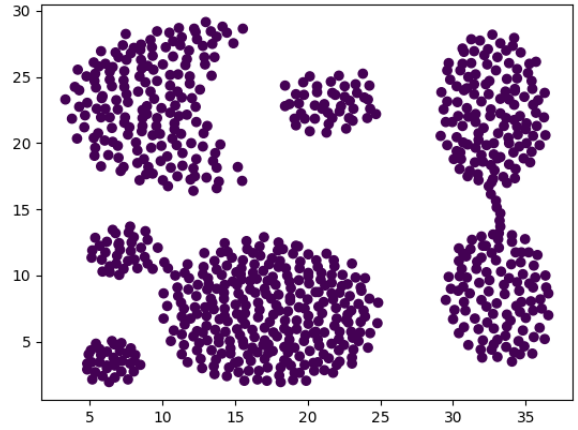


Fig. *DBSCAN (eps = 5)*
min_neighbours = 3
(clusters: 1, labeled as noise: 0)

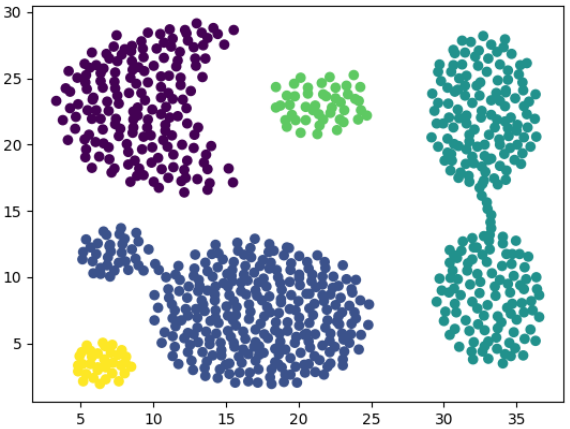


Fig. *DBSCAN (eps = 2)*
min_neighbours = 3
(clusters: 5, labeled as noise: 0)

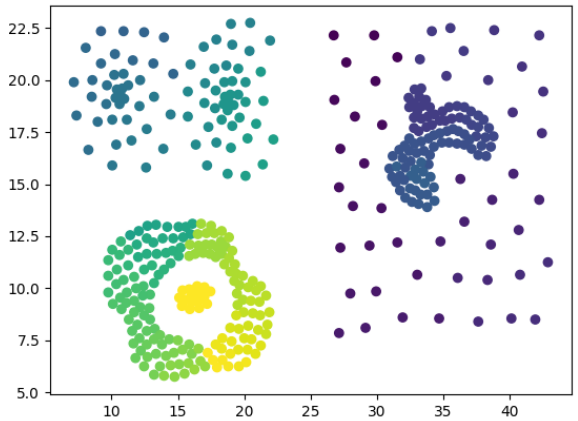


Fig. *DBSCAN (eps = 0.5)*
min_neighbours = 1
(clusters: 280, labeled as noise: 0)

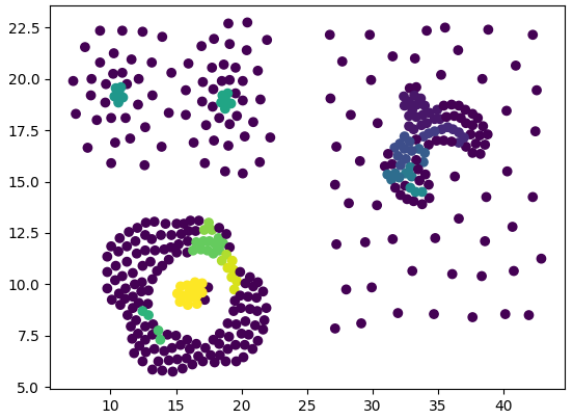


Fig. *DBSCAN (eps = 0.5)*
min_neighbours = 3
(clusters: 18, labeled as noise: 297)

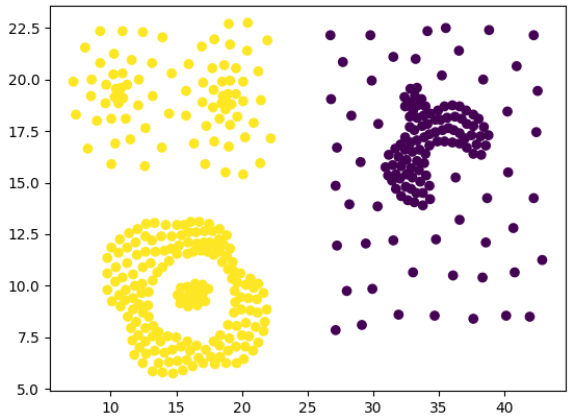


Fig. *DBSCAN (eps = 4)*
min_neighbours = 3
(clusters: 2, labeled as noise: 0)

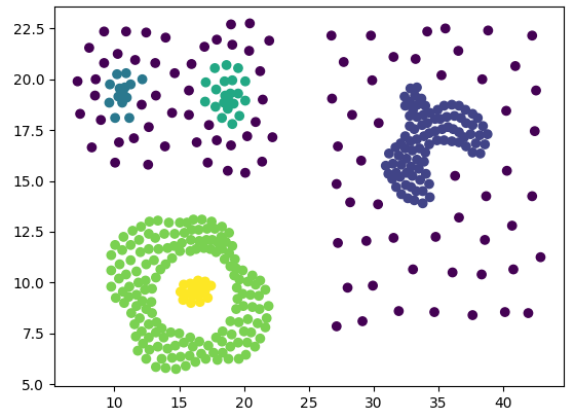


Fig. *DBSCAN (eps = 1)*
min_neighbours = 3
(clusters: 6, labeled as noise: 98)

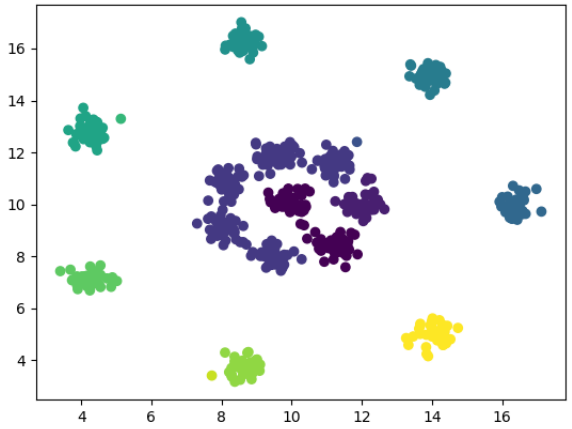


Fig. *DBSCAN (eps = 0.5)*
min_neighbours = 1
(clusters: 13, labeled as noise: 0)

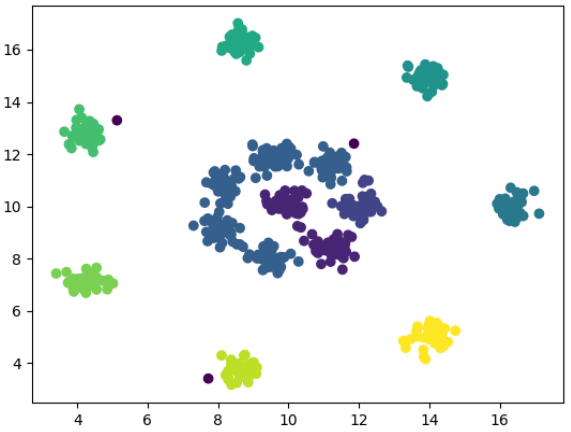


Fig. *DBSCAN (eps = 0.5)*
min_neighbours = 3
(clusters: 11, labeled as noise: 3)

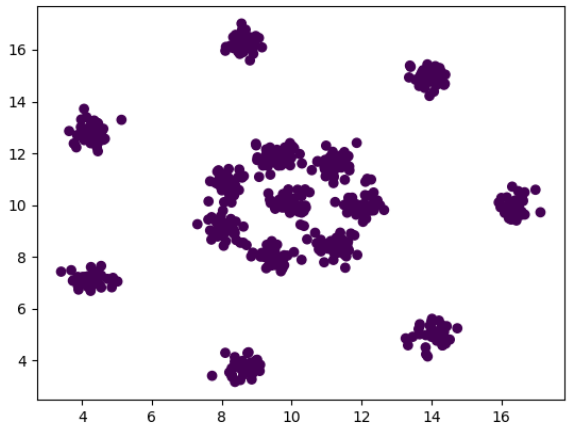


Fig. *DBSCAN (eps = 4)*
min_neighbours = 3
(clusters: 1, labeled as noise: 0)

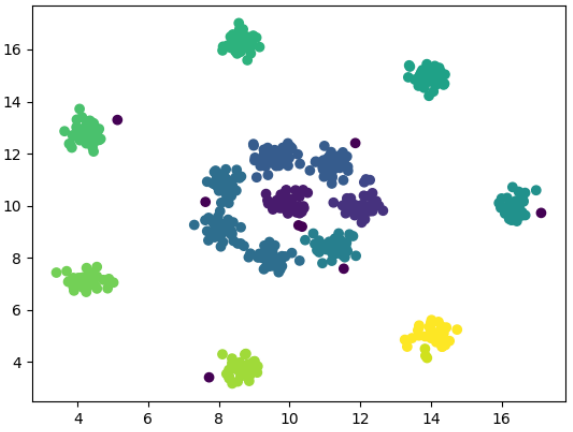


Fig. *DBSCAN (eps = 0.4)*
min_neighbours = 3
(clusters: 15, labeled as noise: 8)

Conclusions:

- With a higher minimum number of neighbors, the number of points labeled as noise increases.
- When the value of epsilon is high, there is a low number of clusters.
- To get good results, given values must fit the given set.

Selecting parameters on the multidimensional dataset:

Since in DBSCAN there are 2 parameters, it is more complicated than in the previous case. Similarly, we run DBSCAN on the wine dataset with different values and visualized the results below.

The range of eps was between 1 and 30. The minimal number of neighbors (min_neigh) was from 1 to 8.

The first diagram shows the Davies-Bouldin index depending on the mentioned parameters.

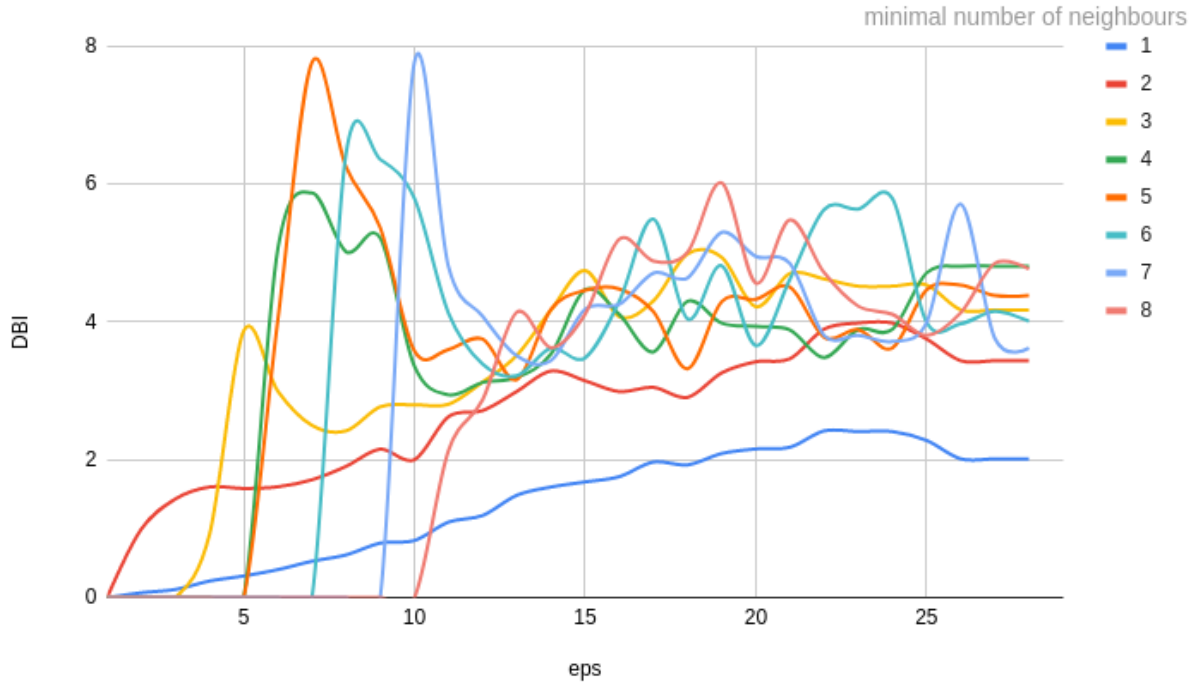


Fig. *Davies-Bouldin index*

As we can see best results are when the minimal number of neighbors is low or (if we need a higher minimum number of neighbors) when epsilon is higher than 12.

Below there is a similar diagram but with Silhouette index instead of Davis-Bouldin:

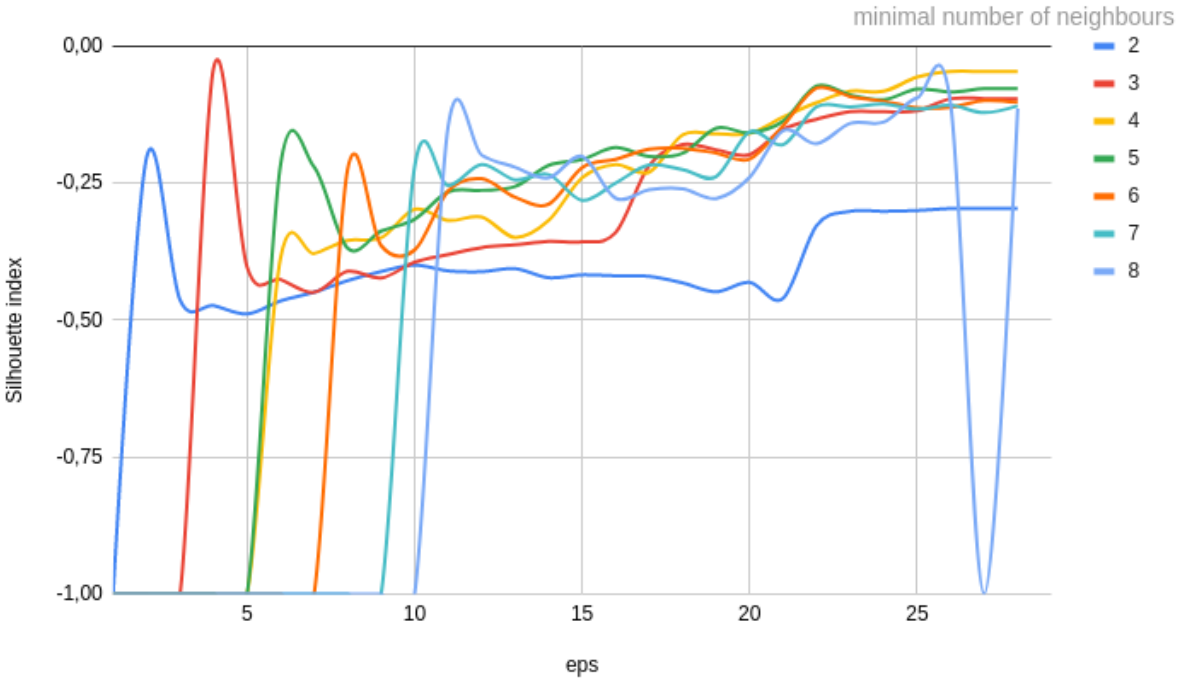


Fig. *Silhouette index*

As we can see there are several top values:

- eps=3, min_neigh=4
- eps=11, min_neigh=8
- eps=7, min_neigh=5

Also when the value of epsilon becomes high, the Silhouette index gives better results.

Conclusion:

- Selecting parameters for clustering is very important as well as a very difficult task.
- Using indexes such as used here may be helpful but does not make it trivial.

3. Mean-shift

Firstly we are going to test mean-shift with 3 2D data sets and then on a Wine data set. For each data set, we will try to find a proper bandwidth parameter based on the data (in the case 2D on visualization, in the case of Wine, just on the data). In the second approach, we will use the bandwidth that we got using sklearn function cluster.estimate_bandwidth (we decided to use this function because it is even hard to pick proper bandwidth using visualization, but in the case of the multidimensional datasets it is almost not possible). Below each Mean-shift figure, you can find used bandwidth (figures on the left show bandwidth generated by the mentioned sklearn function), as well as numbers of clusters returned by our implementation.

For the Wine dataset sklearn function gave us bandwidth 189.344, while we decided to try it also with bandwidth 100 (it was a guess that was based on the data).

Each experiment was run 3 times, and each time the results were exactly the same.

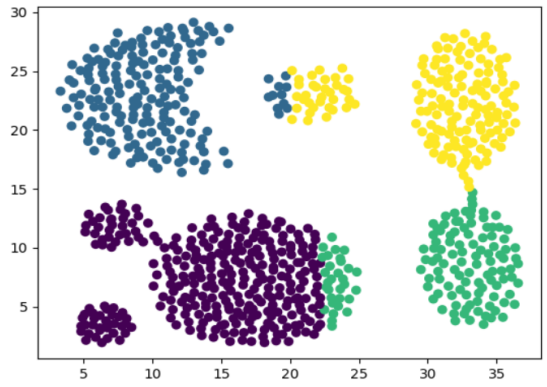


Fig. *Mean-shift 1.*
(bandwidth=11,72, clusters=4)
Aggregation dataset

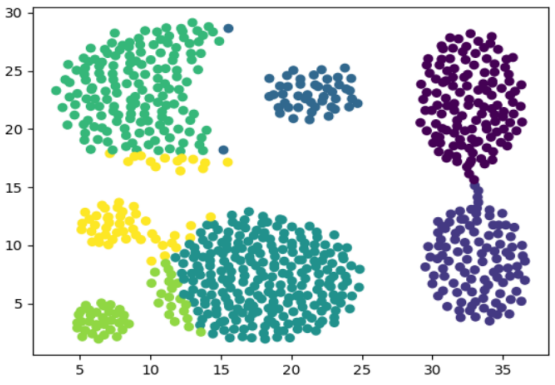


Fig. *Mean-shift 2.*
(bandwidth=7, custers = 7)
Aggregation dataset

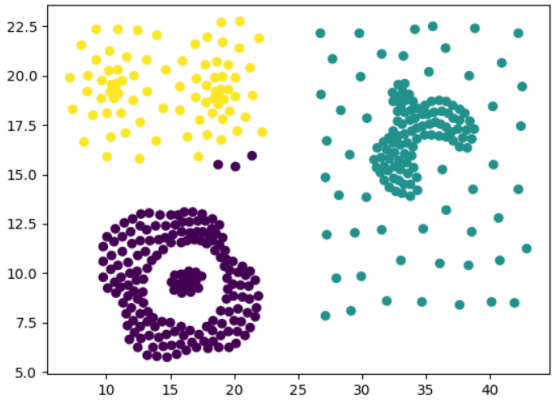


Fig. *Mean-shift 3.*
(bandwidth=7,67, clusters 3)
Compound dataset

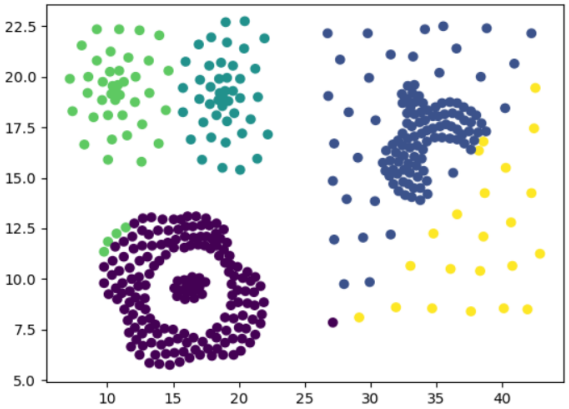


Fig. *Mean-shift 4.*
(bandwidth=6, clusters 5)
Compound dataset

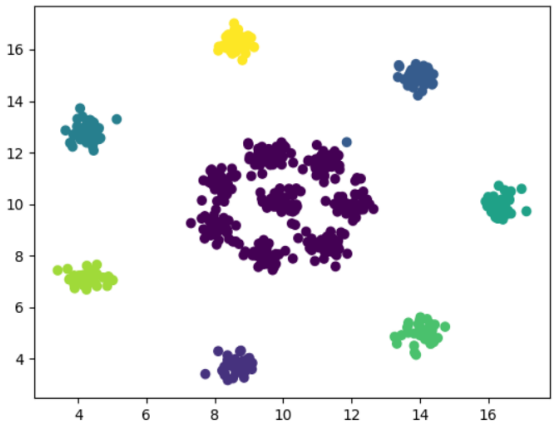


Fig. *Mean-shift 5.*
(bandwidth=4,13, clusters 8)
R15 dataset

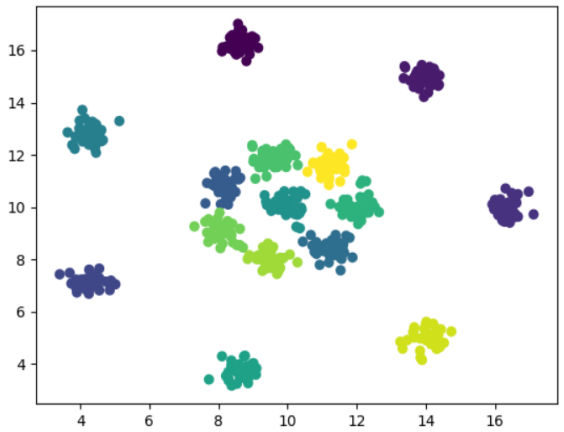


Fig. *Mean-shift 6.*
(bandwidth=1, clusters 15)
R15 dataset

We visualized results for two-dimensional datasets, but in the case of a Wine dataset, we can only rely on the measurement methods. Scores for each experiment can be found in the table below.

Table. *Mean-shift results*

	DBI	Silhouette Score
Mean-shift 1	0.6056	0.4886
Mean-shift 2	0.6478	0.5054
Mean-shift 3	0.7684	0.4667
Mean-shift 4	0.6853	0.5744
Mean-shift 5	0.2076	0.7527
Mean-shift 6	0.3012	0.6516
Wine 100	0.4611	0.5583
Wine 189.344	0.4848	0.5557

Conclusion:

- Referring to the Aggregation dataset we can clearly see that mean-shift was struggling to aggregate clusters, however, DBI and Silhouette scores are not the worst ones compared to the rest.
- When it comes to the Compound dataset we can see in figure *Mean-shit 3* that it returned pretty good results, what is interesting here DBI gives much better results than the Silhouette score, it is because data points in the blue cluster are not accumulated close to each other.

- Mean-shift did well with the R15 dataset, what we can see on the figures, what can be noticed on DBi score
- Regarding the Wine dataset, the scores are averaged, the 189,344 bandwidth did slightly better.

4. Ward's method

For each dataset, we will use two different numbers of clusters. Cluster number in the case of the Wine dataset will be defined as 3 and 5 based on the labels on in the dataset.

Each experiment was run 3 times, and each time the results were exactly the same.

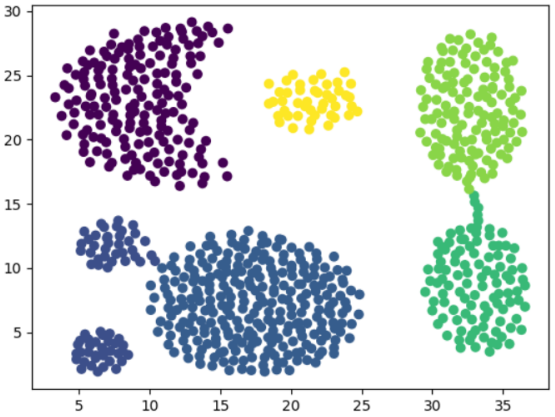


Fig. *Ward 1. (n_clusters=6)*
Aggregation dataset

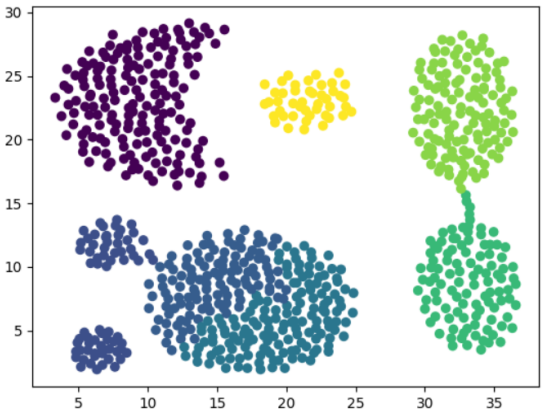


Fig. *Ward 2. (n_clusters=7)* .
Aggregation dataset

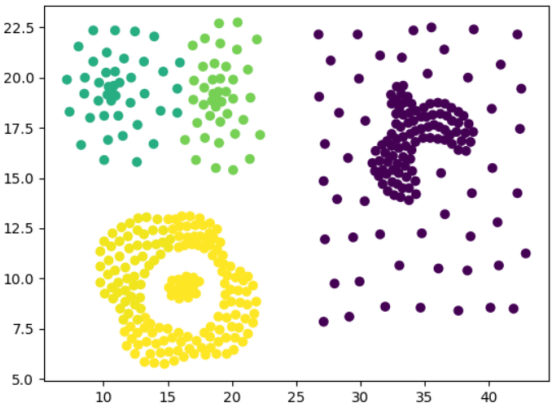


Fig. *Ward 3. (n_clusters=5)*
Compound dataset

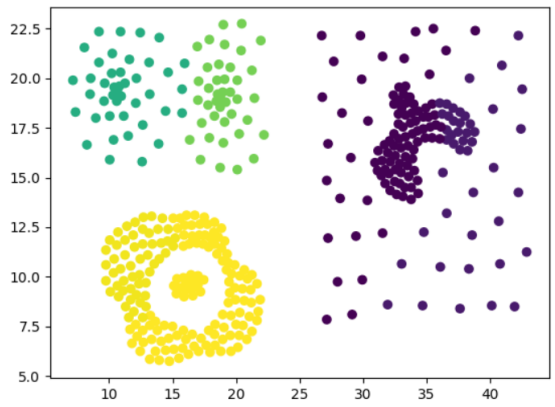


Fig. *Ward 4. (n_clusters=6)* .
Compound dataset

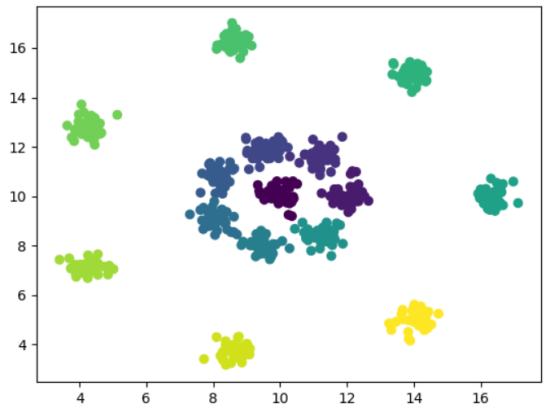


Fig. *Ward 5. (n_clusters=15)*
R15 dataset

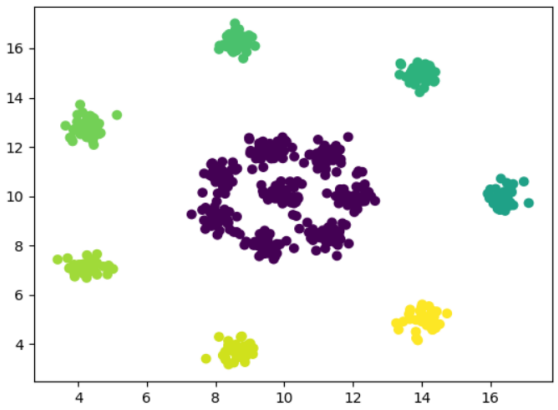


Fig. *Ward 6. (n_clusters=8)* .
R15 dataset

In the table below you can find the scores for Ward's results. Wine 3 refers to 3 numbers of clusters and Wine 5 respectively.

Table. *Ward's method results*

	DBI	Silhouette Score
Ward 1	0.6147	0.5141
Ward 2	0.7684	0.4637
Ward 3	0.6814	0.4883
Ward 4	0.8990	0.3922
Ward 5	0.3149	0.7521
Ward 6	0.3487	0.6530
Wine 3	0.5612	0.5418
Wine 5	0.5691	0.5102

Conclusion:

- Ward's method handles the Aggregation datasets quite well, especially with numbers of clusters set to 6.
- In the case of the Compound dataset, Ward's method could not separate the two cluster properly that was classified as a yellow one
- R15 was done perfectly
- Referring to a multidimensional dataset, this method reached pretty good scores

5. Single linkage

The numbers of clusters were chosen exactly the same as it was in Ward's method. Each experiment was run 3 times, and each time the results were exactly the same.

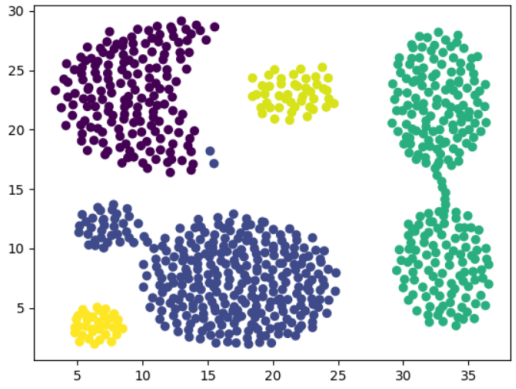


Fig. *Single Linkage 1.*
(n_clusters=6)
Aggregation dataset

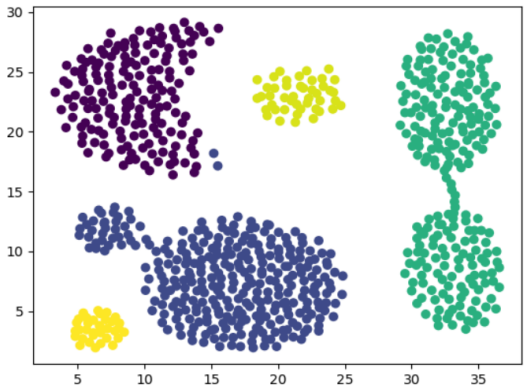


Fig. *Single Linkage 2.*
(n_clusters=7)
Aggregation dataset

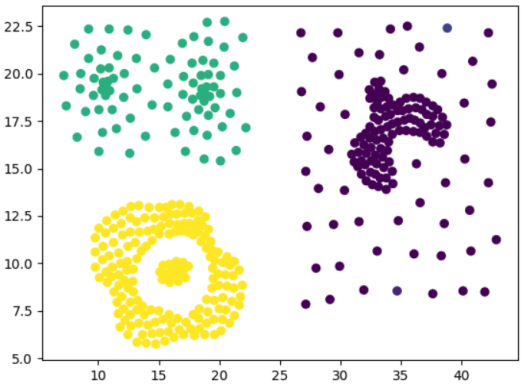


Fig. *Single Linkage 3.*
(n_clusters=5)
Compound dataset

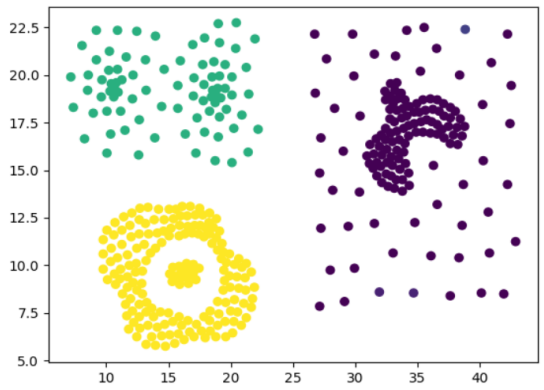


Fig. *Single Linkage 4.*
(n_clusters=6)
Compound dataset

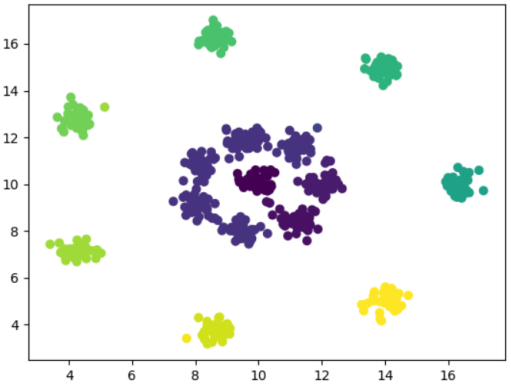


Fig. *Single Linkage 5.*
(n_clusters=15)
R15 dataset

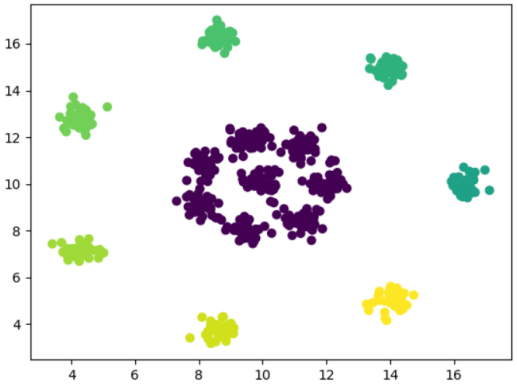


Fig. *Single Linkage 6.*
(n_clusters=8)
R15 dataset

Table. *Single linkage results*

	DBI	Silhouette Score
Single linkage 1	0.6211	0.3187
Single linkage 2	0.6000	0.2828
Single linkage 3	0.6449	0.3719
Single linkage 4	0.6158	0.3621
Single linkage 5	0.8304	0.3990
Single linkage 6	0.3487	0.6530
Wine 3	0.3081	0.4907
Wine 5	0.5874	-0.0267

Conclusion:

- In the Aggregation, single linkage did well, because each iteration connects the closest data points.
- In the Compound dataset, it was similar to different methods, but on the other hand, the R15 dataset with a number of clusters set to 15 gave the wrong result.
- When it comes to the Wine dataset this method did not do quite well with 5 clusters, because the Silhouette score was below 0, but with 3 clusters it reaches one of the best DBI scores.

6. Complete linkage

Here as well numbers of clusters were chosen like in previous methods. Each experiment was run 3 times, and each time the results were exactly the same.

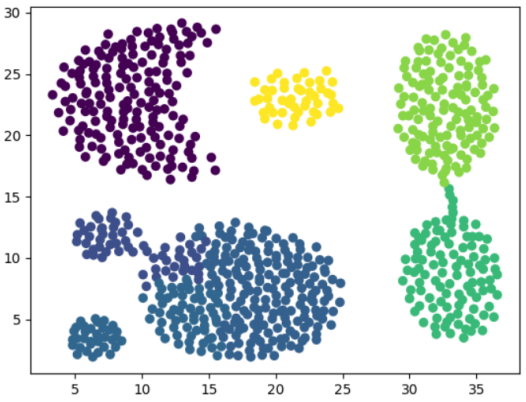


Fig. *Complete Linkage 1. (n_clusters=6) Aggregation dataset*

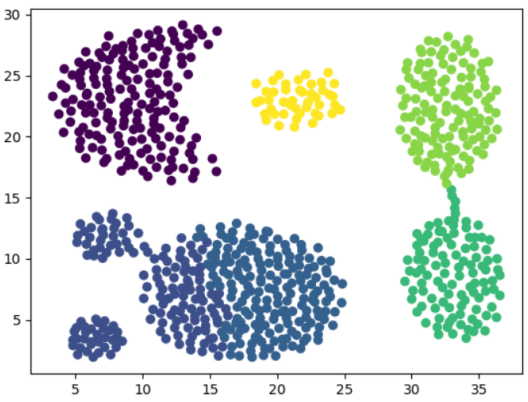


Fig. *Complete Linkage 2. (n_clusters=7) Aggregation dataset*

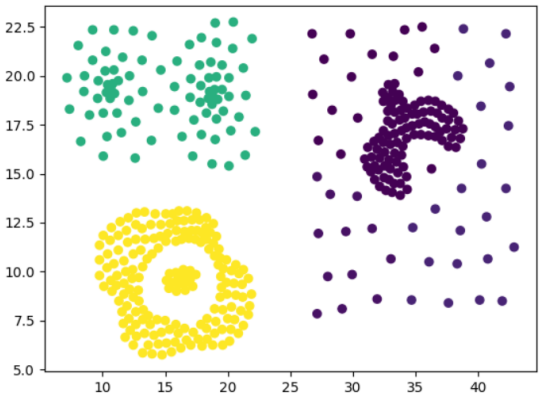


Fig. *Complete Linkage 3.*
(n_clusters=5)
Compound dataset

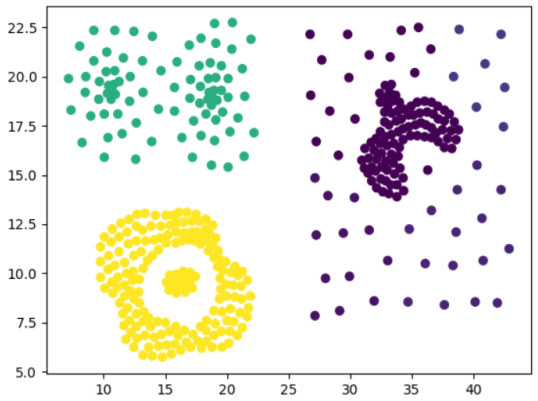


Fig. *Complete Linkage 4.*
(n_clusters=6)
Compound dataset

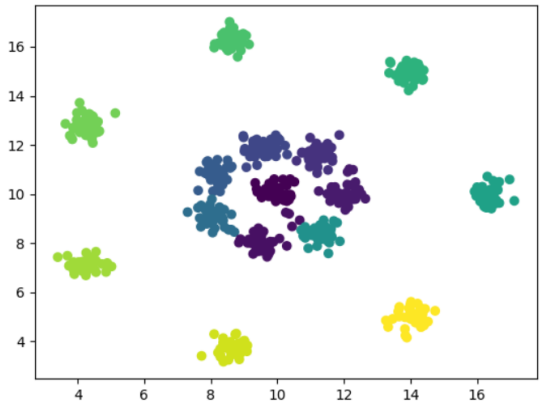


Fig. *Complete Linkage 5.*
(n_clusters=15)
R15 dataset

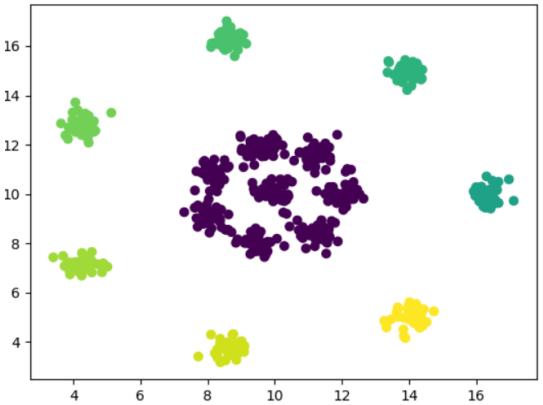


Fig. *Complete Linkage 6.*
(n_clusters=8)
R15 dataset

Table. *Complete linkage results*

	DBI	Silhouette Score
Complete linkage 1	0.6464	0.5000
Complete linkage 2	0.7261	0.4786
Complete linkage 3	0.9372	0.4589
Complete linkage 4	0.7614	0.4532
Complete linkage 5	0.3259	0.7472
Complete linkage 6	0.3487	0.6530
Wine 3	0.5418	0.5612
Wine 5	0.5359	0.4806

Conclusion:

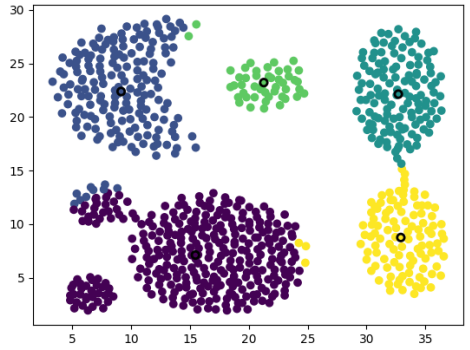
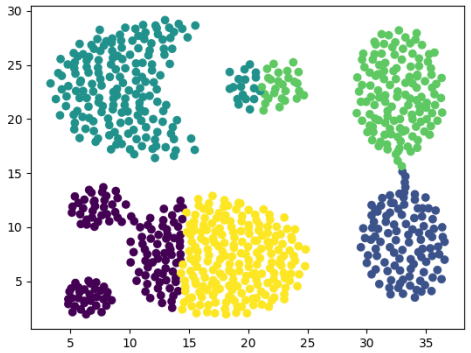
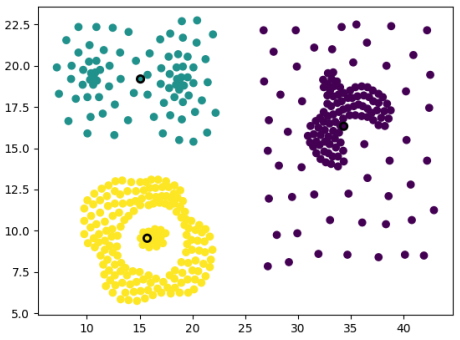
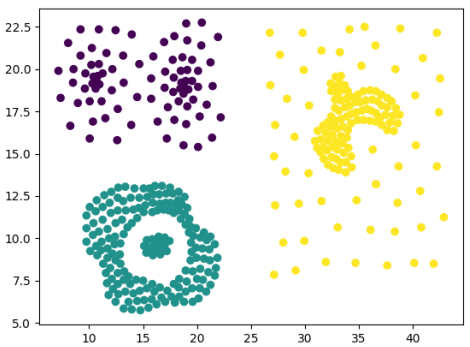
- In the case of 2d datasets, complete linkage gives us proper results compared to different methods.
- Referring to the Wine dataset its result was similar to Ward's

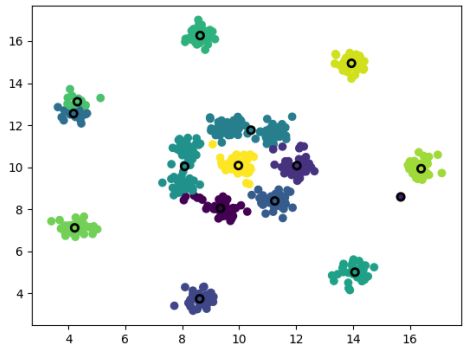
II. Our implementation vs sklearn

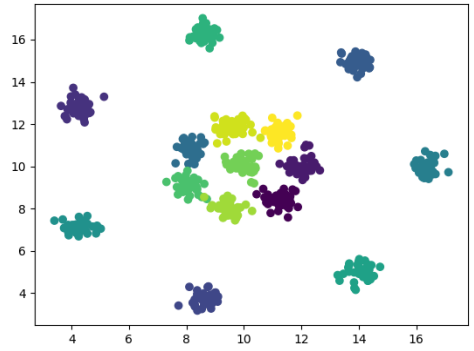
In this experiment, we will compare our implementation against sklearn. For each method, we will use parameters that in the previous phase gave the best results. Will will do it using 4 datasets Aggregation, Compound, R15, and Wine. For every run DBI, Silhouette, and Jaccard scores will be calculated.

The purpose of this test is to see how our implementations differ from the sklearn.

Table. *Kmean (our vs sklearn)*

<table>
<tr><td>parameters</td><td>our implementation</td><td>sklearn</td></tr>
<tr><td rowspan="2">k = 5</td><td> 
Davies–Bouldin index: 0.5458
Silhouette index: 0.5217</td><td> 
Davies–Bouldin index: 0.6677
Silhouette index: 0.4985</td></tr>
<tr><td colspan="2">Jaccard index: 0.6820</td></tr>
<tr><td>k = 3</td><td> 
Davies–Bouldin index: 0.6991
Silhouette index: 0.5755</td><td> 
Davies–Bouldin index: 0.6986
Silhouette index: 0.5755</td></tr>
</table>

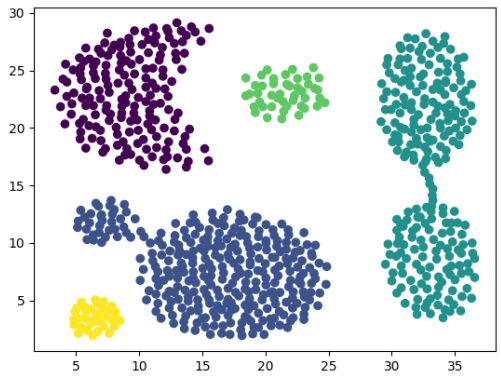
<table>
<tr><td></td><td colspan="2">Jaccard index: 1.0</td></tr>
<tr><td rowspan="2">k = 15</td><td> 

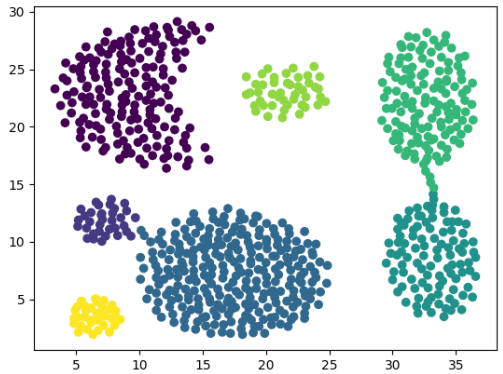
Davies–Bouldin index: 0.2602
Silhouette index: 0.6541</td><td> 

Davies–Bouldin index: 0.2421
Silhouette index: 0.7499</td></tr>
<tr><td colspan="2">Jaccard index: 0.8464</td></tr>
</table>

Table. *DBSCAN (our vs sklearn)*

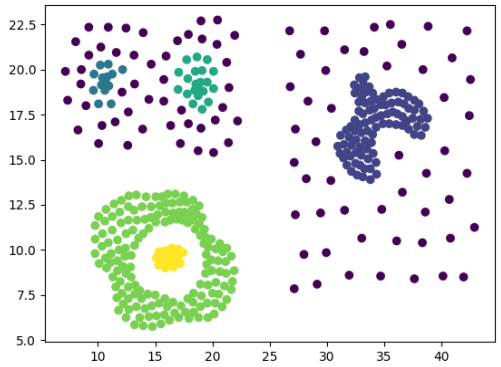
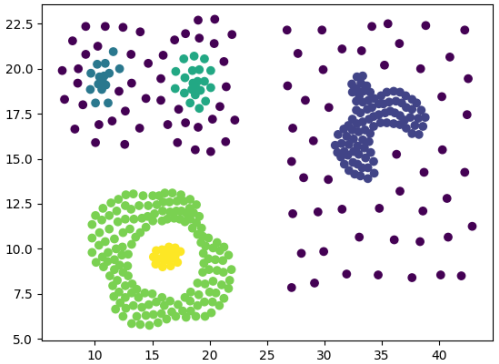
<table>
<tr><td>parameters</td><td>our implementation</td><td>sklearn</td></tr>
<tr><td rowspan="2">eps = 2

min_neigh
bours = 3</td><td> 

Davies–Bouldin index: 0.6193
Silhouette index: 0.4120</td><td> 

Davies–Bouldin index: 0.4947
Silhouette index: 0.4925</td></tr>
<tr><td colspan="2">Jaccard index: 0.8899</td></tr>
</table>

<table>
<tr><td>eps = 1

min_neigh bours = 3</td><td> 
Davies–Bouldin index: 2.2551
Silhouette index: 0.0302</td><td> 
Davies–Bouldin index: 2.2467
Silhouette index: 0.0338</td></tr>
<tr><td></td><td colspan="2">Jaccard index: 0.9871</td></tr>
<tr><td>eps = 0.4

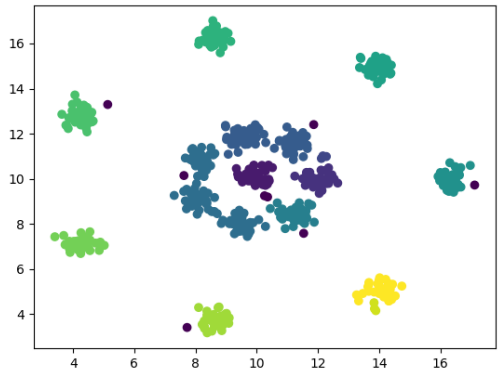
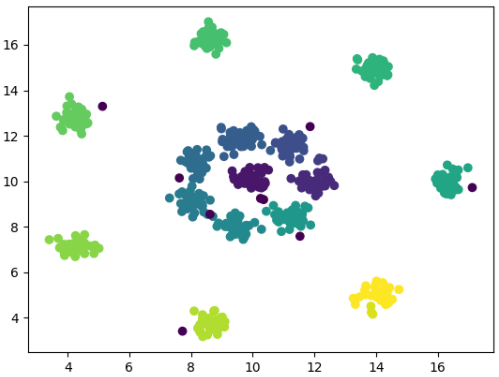
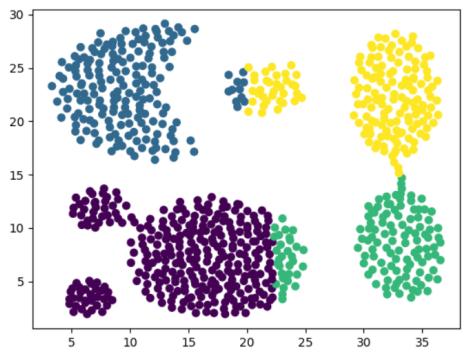
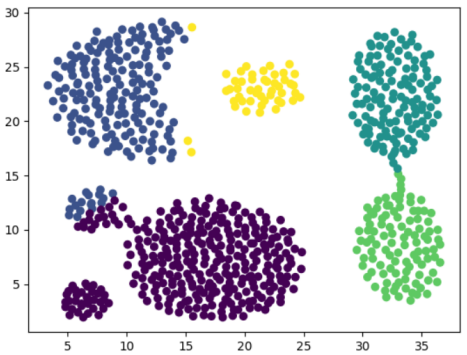
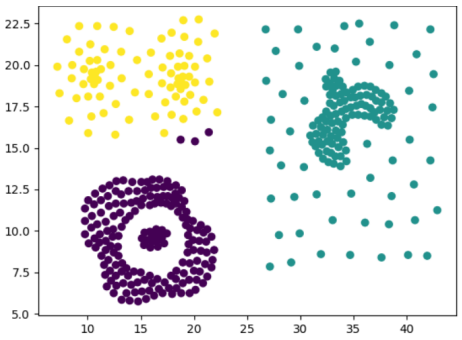
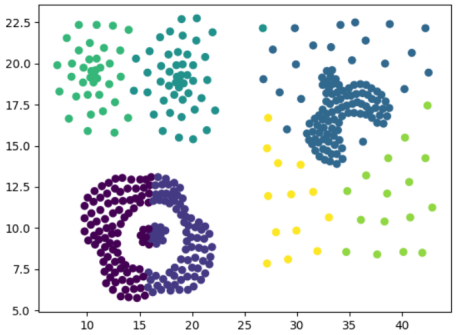
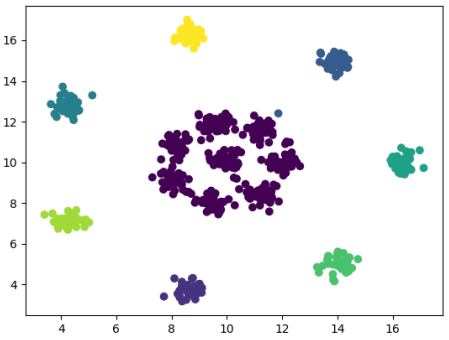
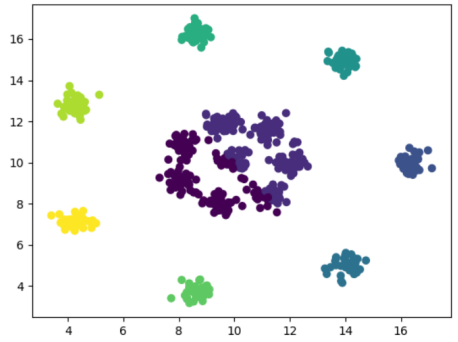
min_neigh bours = 3</td><td> 
Davies–Bouldin index: 0.9363
Silhouette index: 0.5485</td><td> 
Davies–Bouldin index: 0.7652
Silhouette index: 0.6855</td></tr>
<tr><td></td><td colspan="2">Jaccard index: 0.9154</td></tr>
</table>

Table. *Mean-shift (our vs sklearn)*

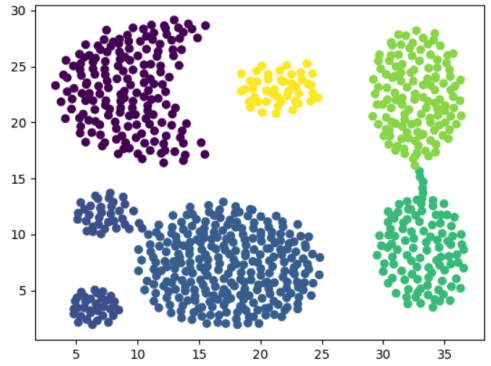
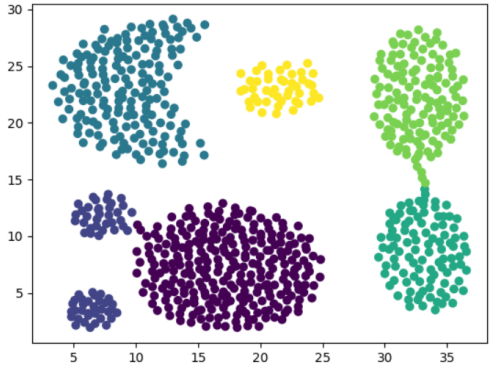
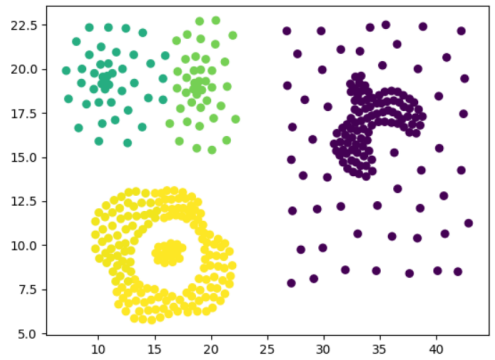
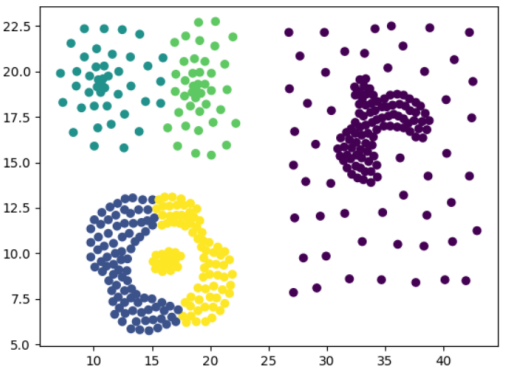
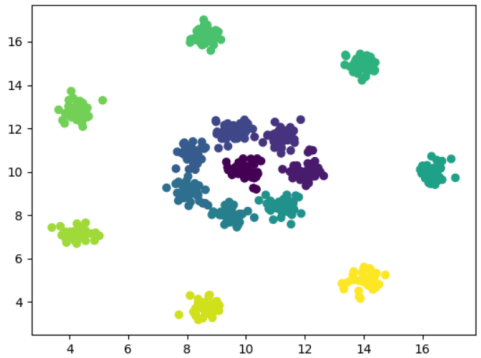
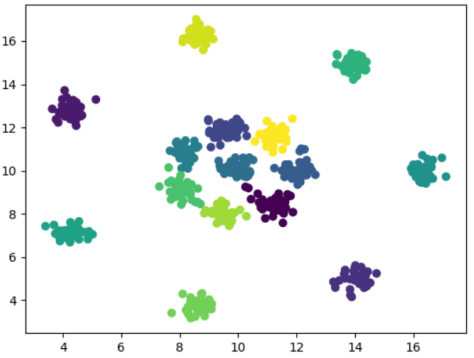
<table>
<tr><td>parameters</td><td>our implementation</td><td>sklearn</td></tr>
<tr><td>bandwidth =11,72</td><td> 
Davies–Bouldin index:0.6501
Silhouette index: 0.5054</td><td> 
Davies–Bouldin index: 0.5569
Silhouette index: 0.5205</td></tr>
</table>

<table>
<tr><td></td><td colspan="2">Jaccard index: 0.8092</td></tr>
<tr><td rowspan="2">bandwidth
=7,67</td><td> 
Davies–Bouldin index: 0.6968
Silhouette index: 0.5744</td><td> 
Davies–Bouldin index: 0.5186
Silhouette index: 0.6069</td></tr>
<tr><td colspan="2">Jaccard index: 0.6049</td></tr>
<tr><td rowspan="2">bandwidth
=4,13</td><td> 
Davies–Bouldin index: 0.9363
Silhouette index: 0.5485</td><td> 
Davies–Bouldin index: 0.5186
Silhouette index: 0.6069</td></tr>
<tr><td colspan="2">Jaccard index: 0.9364</td></tr>
<tr><td rowspan="2">bandwidth
=189.344</td><td>Wine dataset

Davies–Bouldin index: 0.4848
Silhouette index: 0.5557</td><td>Wine dataset

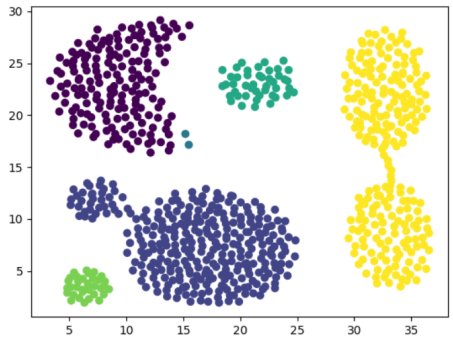
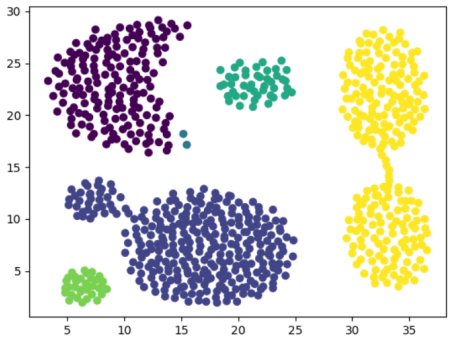
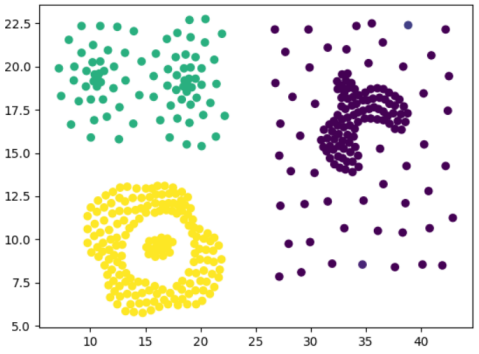
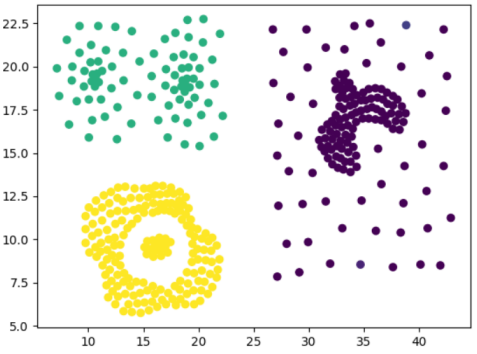
Davies–Bouldin index: 0.4119
Silhouette index: 0.5306</td></tr>
<tr><td colspan="2">Jaccard index: Jaccard index: 0.8092</td></tr>
</table>

Table. *Ward's method (our vs sklearn)*

parameters	our implementation	sklearn
n_clusters =6	 Davies–Bouldin index: 0.6147 Silhouette index: 0.5141	 Davies–Bouldin index: 0.6132 Silhouette index: 0.5124
	Jaccard index: 0.9642	
n_clusters =5	 Davies–Bouldin index: 0.6814 Silhouette index: 0.4883	 Davies–Bouldin index: 0.6927 Silhouette index: 0.5099
	Jaccard index: 0.9101	
n_clusters =15	 Davies–Bouldin index: 0.3149 Silhouette index: 0.7521	 Davies–Bouldin index: 0.3197 Silhouette index: 0.7495
	Jaccard index: 0.9530	

n_clusters =3	Wine dataset Davies–Bouldin index: 0.5612 Silhouette index: 0.5418	Wine dataset Davies–Bouldin index: 0.5357 Silhouette index: 0.5644
	Jaccard index: 0.8926	

Table. *Single linkage (our vs sklearn)*

parameters	our implementation	sklearn
n_clusters =6	 Davies–Bouldin index: 0.6211 Silhouette index: 0.3187	 Davies–Bouldin index: 0.6211 Silhouette index: 0.3187
	Jaccard index: 1.0	
n_clusters =5	 Davies–Bouldin index: 0.6449 Silhouette index: 0.3719	 Davies–Bouldin index: 0.6449 Silhouette index: 0.3719
	Jaccard index: 1.0	

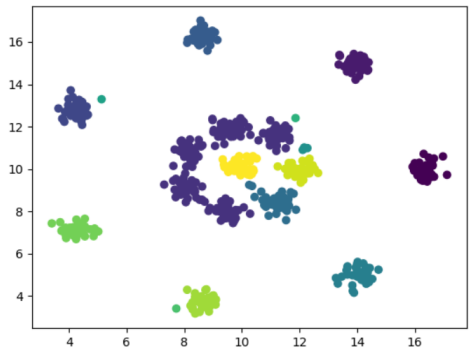
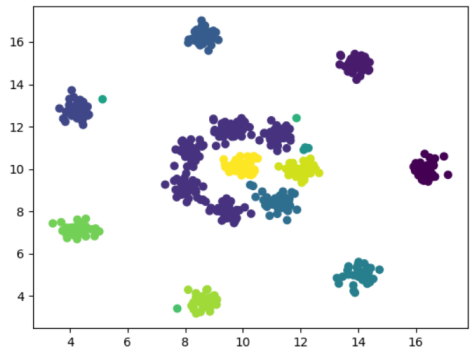
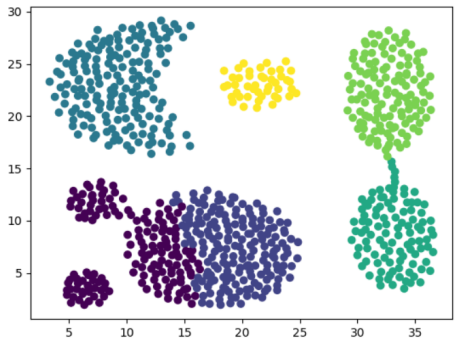
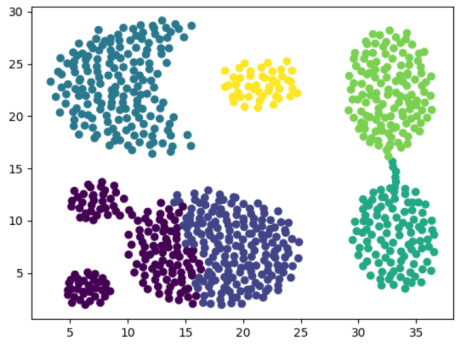
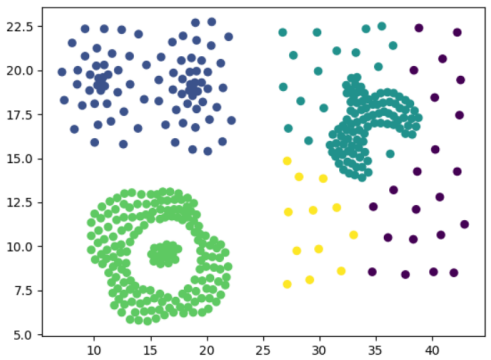
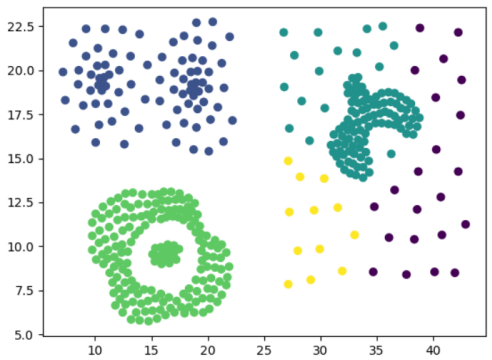
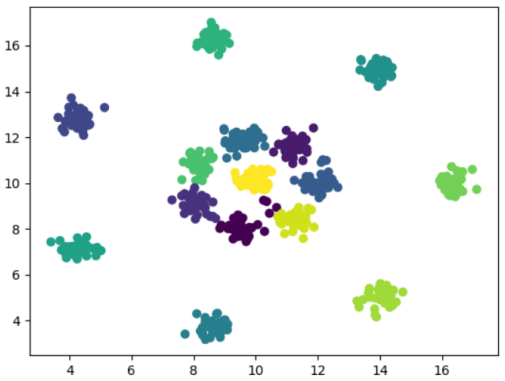
n_clusters =15	 Davies–Bouldin index: 0.8304 Silhouette index: 0.3990	 Davies–Bouldin index: 0.8304 Silhouette index: 0.3990
	Jaccard index: 1.0	
n_clusters =3	Wine dataset Davies–Bouldin index: 0.3081 Silhouette index: 0.4907	Wine dataset Davies–Bouldin index: 0.3081 Silhouette index: 0.4907
	Jaccard index: 1.0	

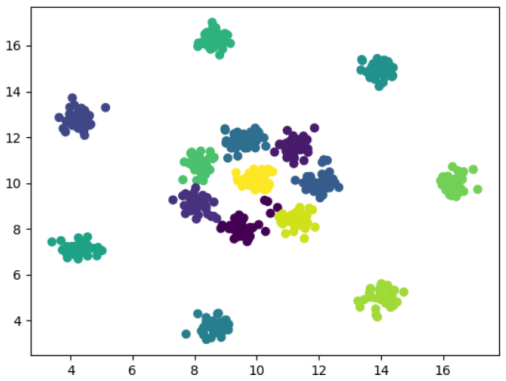
Table. *Complete linkage (our vs sklearn)*

parameters	our implementation	sklearn
n_clusters =6	 Davies–Bouldin index: 0.6464 Silhouette index: 0.5000	 Davies–Bouldin index: 0.6464 Silhouette index: 0.5000
	Jaccard index: 1.0	

<table>
<tr><td rowspan="2">n_clusters =5</td><td> 

Davies–Bouldin index: 0.9372
Silhouette index: 0.4589</td><td> 

Davies–Bouldin index: 0.9372
Silhouette index: 0.4589</td></tr>
<tr><td colspan="2">Jaccard index: 1.0</td></tr>
<tr><td rowspan="2">n_clusters =15</td><td> 

Davies–Bouldin index: 0.3259
Silhouette index: 0.7472</td><td> 

Davies–Bouldin index: 0.3259
Silhouette index: 0.7472</td></tr>
<tr><td colspan="2">Jaccard index: 1.0</td></tr>
<tr><td rowspan="2">n_clusters =3</td><td>Wine dataset

Davies–Bouldin index: 0.5612
Silhouette index: 0.5418</td><td>Wine dataset

Davies–Bouldin index: 0.5612
Silhouette index: 0.5418</td></tr>
<tr><td colspan="2">Jaccard index: 1.0</td></tr>
</table>

For Kmeans and DBSCAN we did not use the Wine dataset since it takes a long time to calculate the result, but using 2D datasets will be enough to determine if implementations are similar to sklearn ones.

Conclusion:

- Referring to the K-means we can see that besides the Aggregation Jaccard index is very high, with a score of 1.0 in the case of Compound. It means that implementations are similar. The difference in Aggregation can be probably explained by randomly initializing starting points.
- In the case of DBSCAN almost every time above 0.9, so we can conclude the implementations are very similar.

- Mean-shift gives us a Jaccard score on average around 0.8, it is still good, but the cause of why it is not higher may be because of different application of bandwidth parameters.
- Almost all Jaccard indexes are above 0,9 in the case of Ward. Also when we look at other indexes each time they are very similar, so we can conclude that implementation give almost the same solutions
- In the case of Single and Complete linkage we can see that all rates are the same comparing our and sklearn implementations, Jaccard index is always 1.0, so it means that both methods give exactly the same results. It could be achieved there because the calculations in these methods are not very tough to apply.

IV. Execution time

For each of the 2D datasets, each method was run multiple times on the same computer. The values in the following table are the average of all measurements in microseconds:

Table. *Execution time comparison*

dataset no	1) Aggregation		2) Compound		3) R15	
	our impl	sklearn	our impl	sklearn	our impl	sklearn
K-means	291,608	40,256	105,681	29,562	209,240	23,629
DBSCAN	3,211,048	14	801,730	17	1,988,836	18
Mean shift	5,665,543	2,978,314	1,508,986	1,282,019	3,063,123	1,434,831
Ward	2,577,752,561	15,734	174,083,962	4,923	949,788,711	8,497
Complete linkage	58,728,985	9,779	10,250,828	3,290	28,557,270	5,095
Single linkage	56,041,251	5,146	9,251,433	2,461	27,262,678	3,584

Conclusions:

- In each dataset and each method sklearn has much better performance so if there is such a possibility, one should use available libraries.
- From selected methods the fastest ones are K-means (our implementation) and DBSCAN (sklearn's implementation)
- The best implementation from our ones is mean shift - its performance is the closest to the one from sklearn

7. Summary of research

Regarding the ‘implementation’ part of our goal, we can clearly state that our implementations are correct. Looking at the Jaccard score in the comparison of our methods against sklearn, we can notice that all scores are very high. Mean-shift and K-means achieved the lowest rate, approximately 0.8, which still is a good result. That difference between our implementation and sklearn can be explained in the case of K-means with random initialization of starting points, while in the case of Mean-shift cause is probably in the different way of initializing starting points, which may result in different solutions. Rest of the methods reach very good scores, single and complete linkage got 1.0 Jaccard index, which means that our implementation works exactly like sklearn ones.

To summarize the ‘comparison’ part of our goal. Each method was tested with different parameters and using different dataset. In the case of Aggregation dataset DBSCAN, Ward’s method and Single linkage achieved the best results. Referring to the Compound dataset DBSCAN got the most satisfying outcome. When it comes to the R15 dataset, all of the methods manage to group this data correctly, but the perfect one was Ward and Complete linkage. Considering the Wine dataset we can only rely on the DBI and Silhouette scores, looking at the first one the best score achieved Single linkage around 0.3, and in the case of Silhouette index the leading method was Mean-shift. Looking at this statistics it is very hard to say which method is the best one. The only conclusion that can be drawn from this experiment is that each method is suitable for different datasets, and in order to pick the greatest one you have to properly analyze the data that you are working with.

All things considered, we have fulfilled the requirements of our set goal, therefore in the future, when we will need to use such clustering methods, we will know how to act.

8. References

[1] Manimaran (2019) Clustering Evaluation strategies
[2] Fletcher & Islam (2018) Comparing sets of patterns with the Jaccard index, Australasian Journal of Information Systems
[3] Xu, D., Tian, Y. A (2015) Comprehensive Survey of Clustering Algorithms
[4] Sklearn (2020) Documentation
[5] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze (2008) Introduction to Information Retrieval
[6] P. Fränti and S. Sieranoja (2018) K-means properties on six clustering benchmark datasets
[7] Wikipedia Website, Lots of articles were taken from wikipedia
[8] Konstantinos G. Derpanis (2005) Mean Shift Clustering
[9] George Seif (2018) The 5 Clustering Algorithms Data Scientists Need to Know
[10] Ashutosh Bhardwaj (2020) Silhouette Coefficient
[11] Dilts, David & Khamalah, Joseph & Plotkin, Ann. (1995). Using Cluster Analysis for Medical Resource Decision Making
[12] Harry Wang (2020) Wine Dataset for Clustering (Cluster wines based on their chemical constituents)
[13] John Clements (2019) Introduction to Hierarchical Clustering
[14] Pamela Beatriz Guevara Alvez. (2011) Inference of a human brain fiber bundle atlas from high angular resolution diffusion imaging.

GitHub Repository:
Our implementation can be found on github:
github.com/PiotrKedra/ClusteringMethods