HW spectral clustering

Elena Nespolo and Giuseppe Mallo

January 2025

1 Introduction

Spectral clustering is a method used to find appropriate clusters to divide the points of a dataset. The particularity of this method lies in its use of linear algebra concepts, such as eigenvalues and eigenvectors. They are utilized to represent the points in a new vector space where clusters can be identified, and classical clustering algorithms can be applied.

2 Data

After loading the data into two distinct dataframes, *circles* and *spirals*, we can observe that the points are situated in the $\mathbb{R}^2$ space. Therefore, we start by discussing the *circles* dataset.

3 Circles dataset

From the image below, we can observe the arrangement of points in the two-dimensional space.

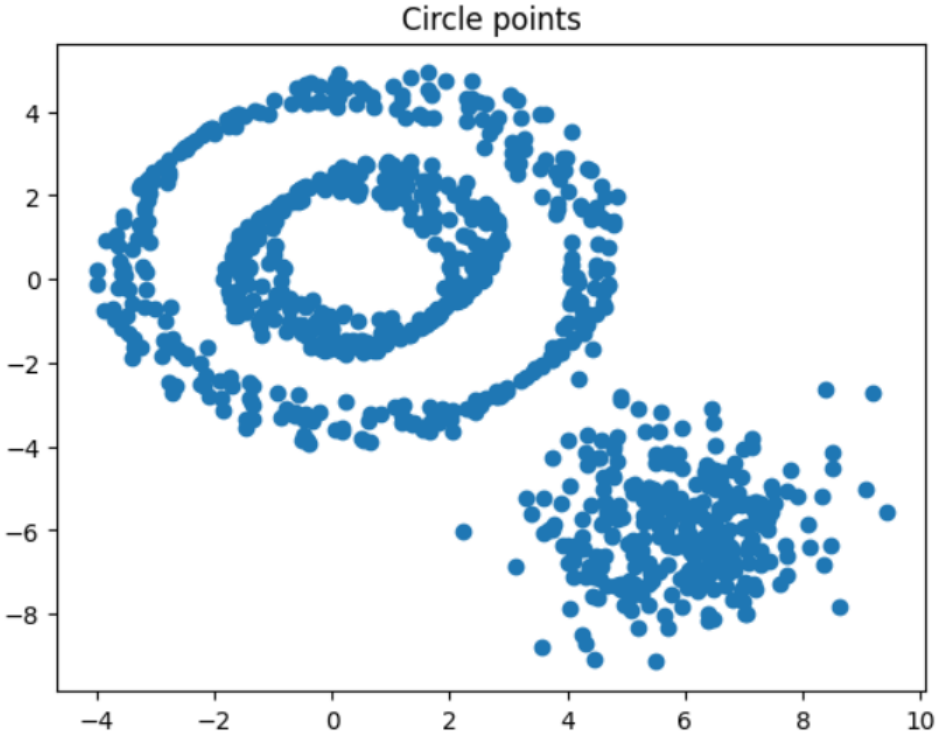


Figure 1: The dataset `Circle.csv`

We can assume that there will be three clusters (the two circles and the globular set of points).

To use spectral clustering, the first step is to construct the *similarity matrix* using the `compute_similarity_matrix` function. The similarity matrix is an $\mathbb{R}^{N\times N}$ matrix (N = number of points) where each entry (i, j) is calculated using a well-defined function, namely `compute_similarity`, which uses the following formula:

$$S(i,j) = e^{-\frac{\|x_i - x_j\|^2}{\sigma^2}}$$

This function returns a number indicating the similarity between points i and j. Note that the diagonal of the similarity matrix contains only zeros.

After constructing the similarity matrix, we calculate the adjacency matrix W using the `compute_adjacency_matrix` function. This function selects, for each row, the k highest similarity values, i.e., for each i-th point, it selects the k points most similar to it. We will examine the results for different values of $k = 10, 20, 40$.

The code shows that to construct W, an algorithm was applied in order to represents W as a sparse matrix using the Compressed Sparse Row (CSR) format. Three arrays are needed for this purpose:

- `data`: contains the non-zero elements,
- `row`: contains the corresponding row indices of each non-zero element,
- `col`: contains the corresponding column indices of each non-zero element.

Finally, using the `csr_matrix` function from the `scipy.sparse` package, W is transformed into sparse format.

Note: Occasionally, the dense version of some sparse matrices will be used to facilitate specific calculations.

However, we notice that when calculating W, we lose the property of symmetry, which is essential for further computations. Thus, using a simple algebraic operation described by the following formula:

$$W = \frac{W + W^\top}{2}$$

within the `compute_symmetric_adjacency_matrix` function, we can make W symmetric.

Finally, the previously described functions are used within the `compute_L_W_D` function. This function returns the matrix W, the degree matrix D (calculated with the `compute_degree_matrix` function), and the Laplacian matrix L, which is computed as follows:

$$L = D - W$$

This matrix is crucial because its eigenvalues and eigenvectors will be used for spectral clustering. Additionally, the sparse form of the Laplacian matrix is obtained using the `csr_matrix` function.

3.1 $k = 10$

We start by considering the property stating that the number of null eigenvalues of the Laplacian matrix equals the number of connected components. Thus, by calculating the dimension of the kernel of L using the relationship:

$$\ker(A) + \mathrm{Im}(A) = n \quad (\text{where } \mathrm{Im}(A) = \mathrm{rank}(A)),$$

we find that the number of connected components is 2. This implies that the graph obtained after choosing the 10-nearest-neighbors is a graph with two disjoint subgraphs, each internally connected.

Printing the eigenvalues of L, sorted in ascending order using the `np.linalg.eigh` function, we observe that the two smallest eigenvalues are of the order 10^{-15} or 10^{-16} (and can thus be considered as 0). The next smallest eigenvalues will correspond to an additional subgraph weakly connected to one of the two disjoint subgraphs.

Next, we select the first M eigenvectors corresponding to the smallest M eigenvalues and store them in the matrix $U_{\text{circles\_10}}$. This matrix, with dimensions $N \times M$, contains the M eigenvectors as its columns. Along its rows, we find the coordinates of each of the N points written in the vector space generated by the M eigenvectors. These eigenvectors are orthogonal since the Laplacian matrix is symmetric.

Thus, we choose $M = 3$ to represent the points in $\mathbb{R}^3$. Visualizing these points:

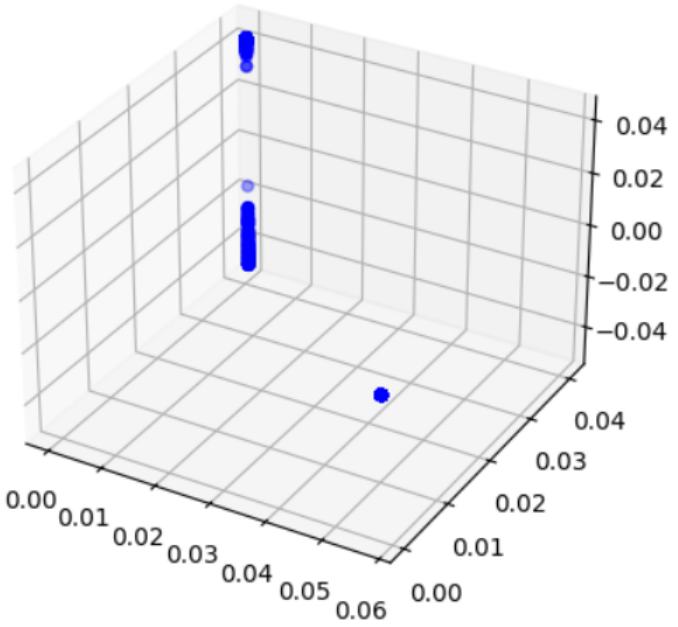
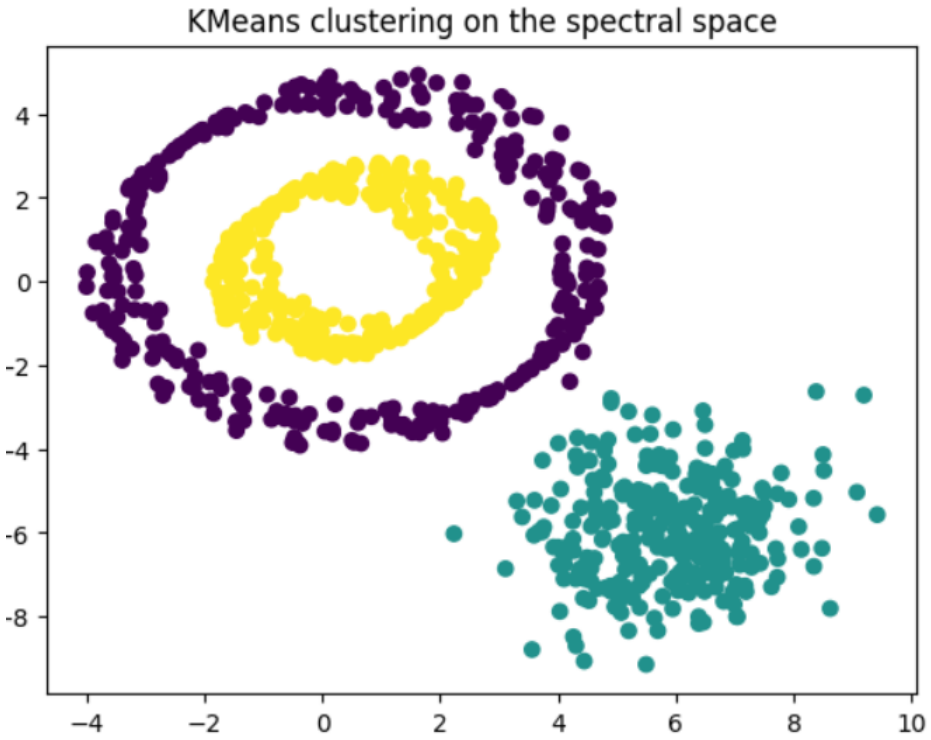


Figure 2: Points represented in the new space

We observe that the points form rather distinct clusters. Moreover, some points overlap because they share identical coordinates. We now apply clustering algorithms to these new points, assigning each one a label identifying its cluster, and compare the results to those obtained by applying the same algorithms to the original points in $\mathbb{R}^2$.

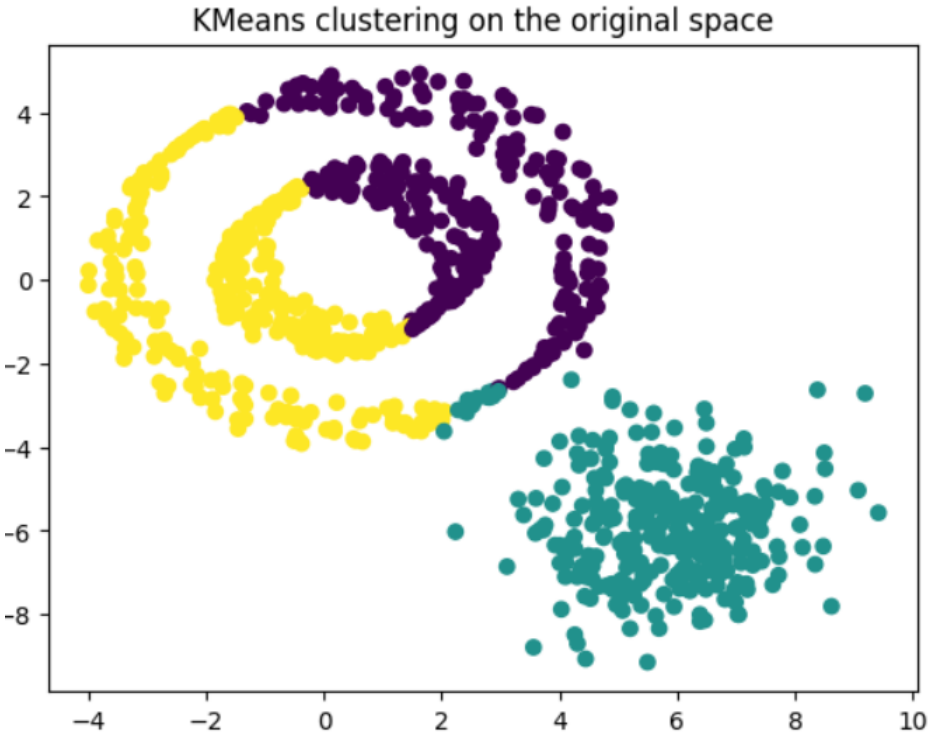
K-Means

We begin by applying K-Means to the points in the new space. Thanks to Figure 1, it is reasonable to set $n_{\text{clusters}} = 3$.



In this case, we observe that K-Means performs very well because the points are well-separated in the new space.

Applying K-Means with the same hyperparameters directly to the original *circles* points yields the following result:



Graphically, we observe less satisfactory results, as K-Means tends to favor globular cluster shapes.

Evaluating both clusterings with appropriate metrics, we obtain the following:

```
Silhouette score for KMeans clustering applied on the 3-dimensional points
    : 0.32326843682162765
```

```
Silhouette score for KMeans clustering applied on the original points:
    0.47444975618735247
Adjusted rand score: 0.4701072314954328
```

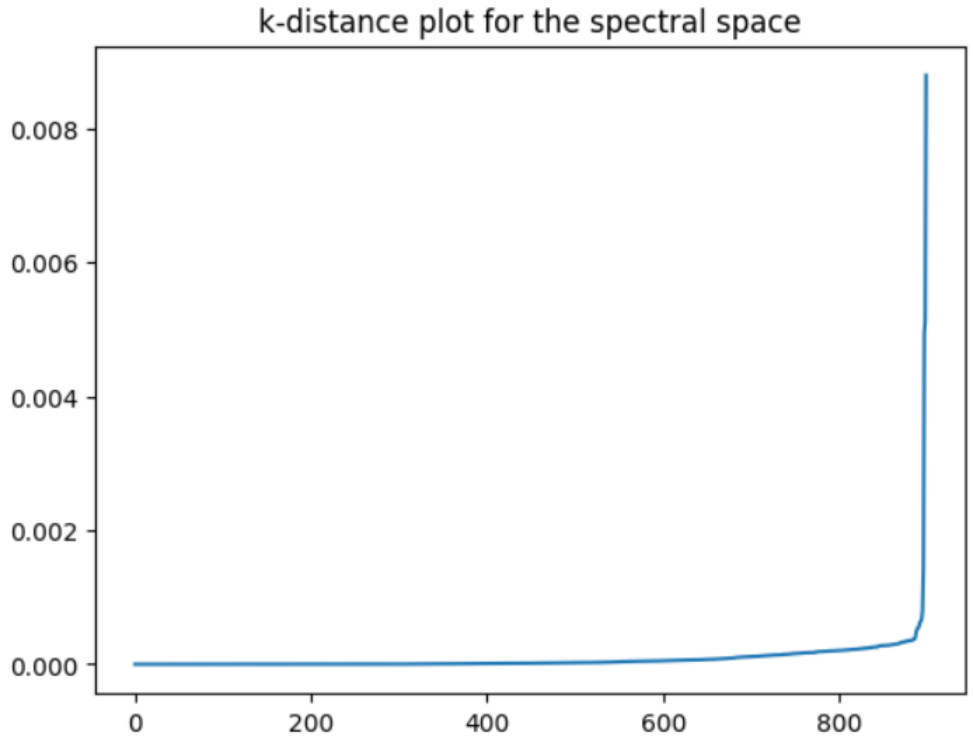
The silhouette score indicates that the best clusters are formed by applying K-Means to the points in the original space. Regarding the adjusted Rand score, considering `labels_km_spec` as ground truth and `labels_km_circles` as predictions, we find partial agreement between the two clustering results.

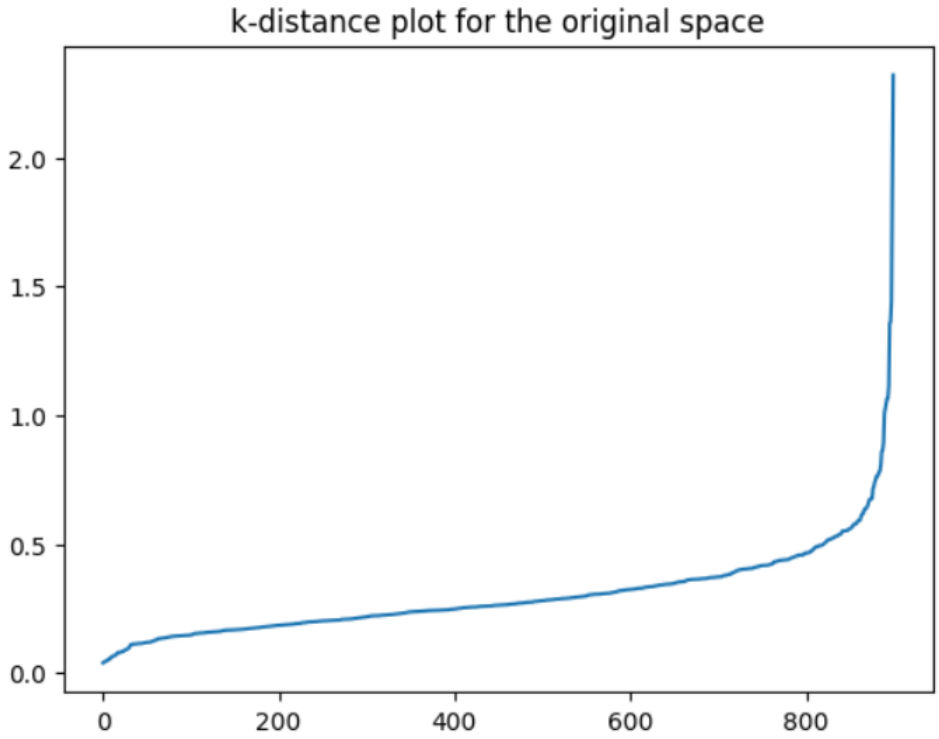
DBSCAN

Let's try applying DBSCAN on the points in the space $\mathbb{R}^M$. Recall that, given values for ϵ and min_samples, in DBSCAN, points are distinguished as follows:

- **Core points**: Points that have at least min_samples nearest neighbors within a radius of ϵ,
- **Border points**: Points that have fewer than min_samples nearest neighbors within a radius of ϵ, but are reachable from a core point,
- **Noise points**: Points that have fewer than min_samples neighbors within a radius of ϵ and are not reachable from any core point.

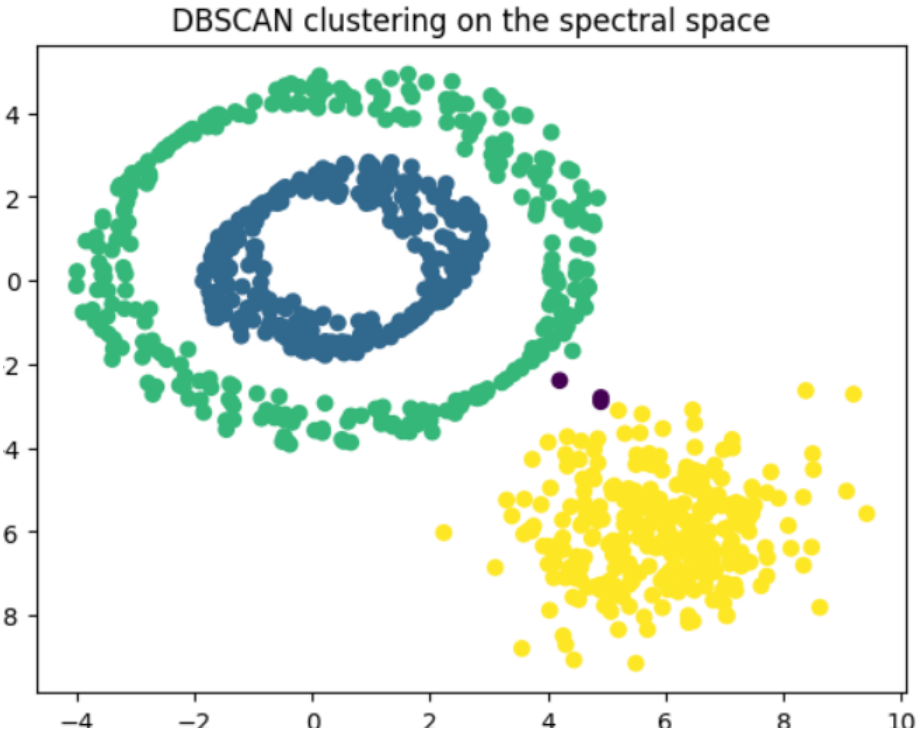
To determine the values of ϵ and min_samples, we can look at the plots below, which order the points based on their distance to the k-th nearest neighbor (in this case, $k = 10$).

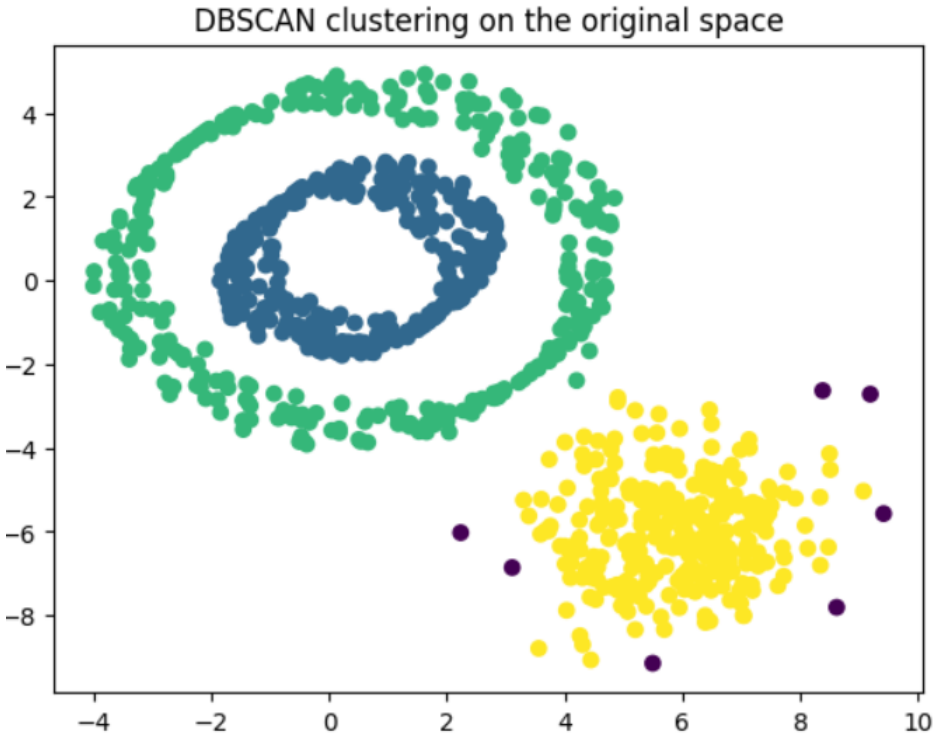




The curve represents these distances. Notice in the first plot that from around the 900th point onward, the k-th nearest neighbor is found at progressively higher distances.

From the plots, we can extract the values of the two hyperparameters, and the results obtained are as follows:






```
Silhouette score for DBSCAN clustering applied on the 3-dimensional points
    : 0.1612804537308867
Silhouette score for DBSCAN clustering applied on the original points:
    0.225795331124162
Adjusted rand score: 0.9837867296845562
```

For the clustering obtained with DBSCAN applied to the original points, the silhouette score is higher since more points around the yellow cluster are considered as noise compared to the first figure. Regarding external evaluation using the adjusted rand score, we observe nearly perfect agreement between the two clusters.

```
Adjusted rand score between the Kmeans clustering and the DBSCAN
    clustering on U_circles_10: 0.9966611990315535
```

Finally, when comparing the clusters obtained using KMeans and DBSCAN on the points in the new space, the result is that both clusterings show almost perfect agreement.

3.2 k = 20

Now, let's see what happens if, when constructing the adjacency matrix W, we consider the 20-nearest neighbours for each point. The number of null eigenvalues of the Laplacian matrix is 1, which means that the graph will have a single connected component. We choose 3 eigenvectors to generate the new space and place them into $U_{\text{circles\_20}}$.

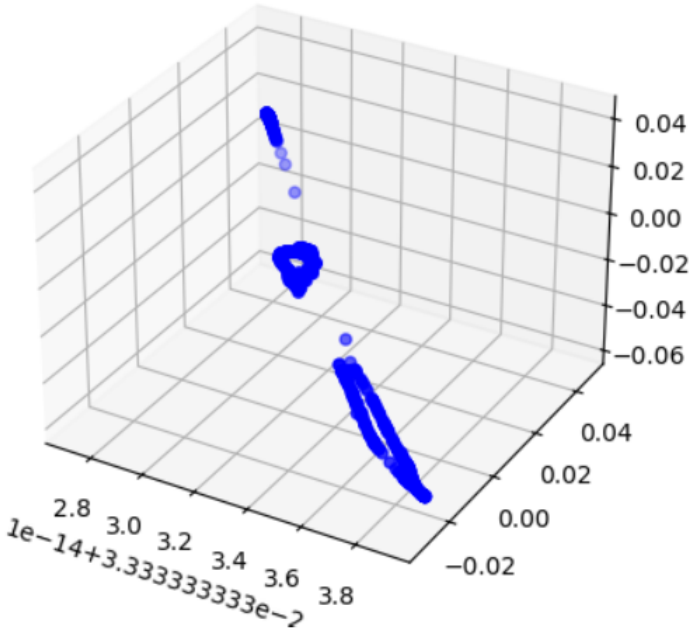
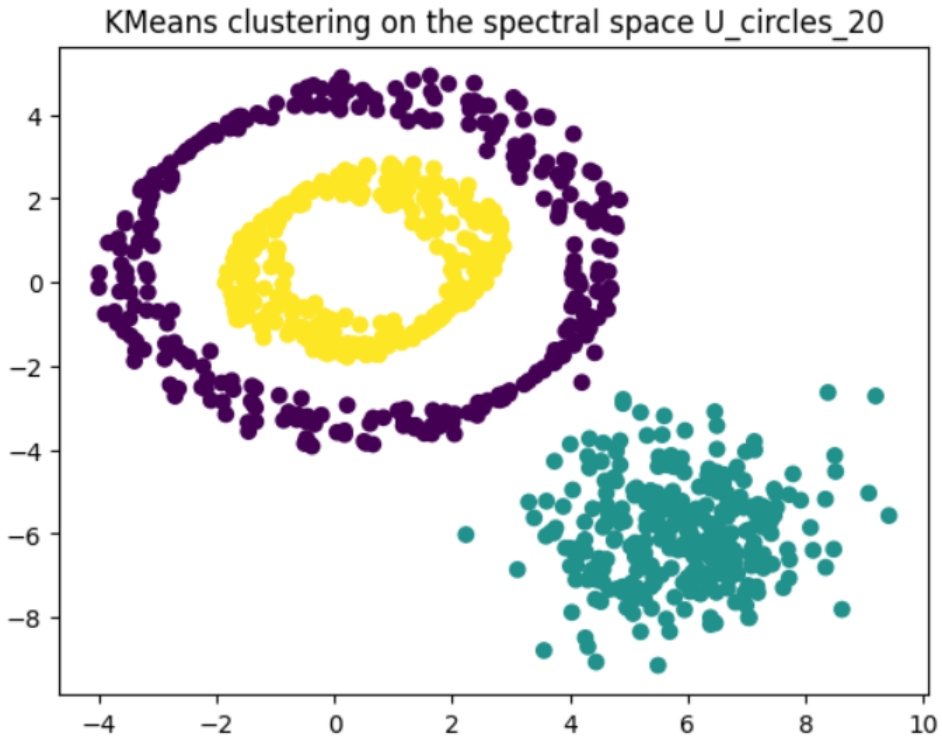


Figure 3: Points of Circles represented in the spectral space

KMeans

Applying KMeans with the hyperparameter $n_{\text{clusters}} = 3$ separately to the points in the new space and to those in the original space, we obtain the following clusterings.

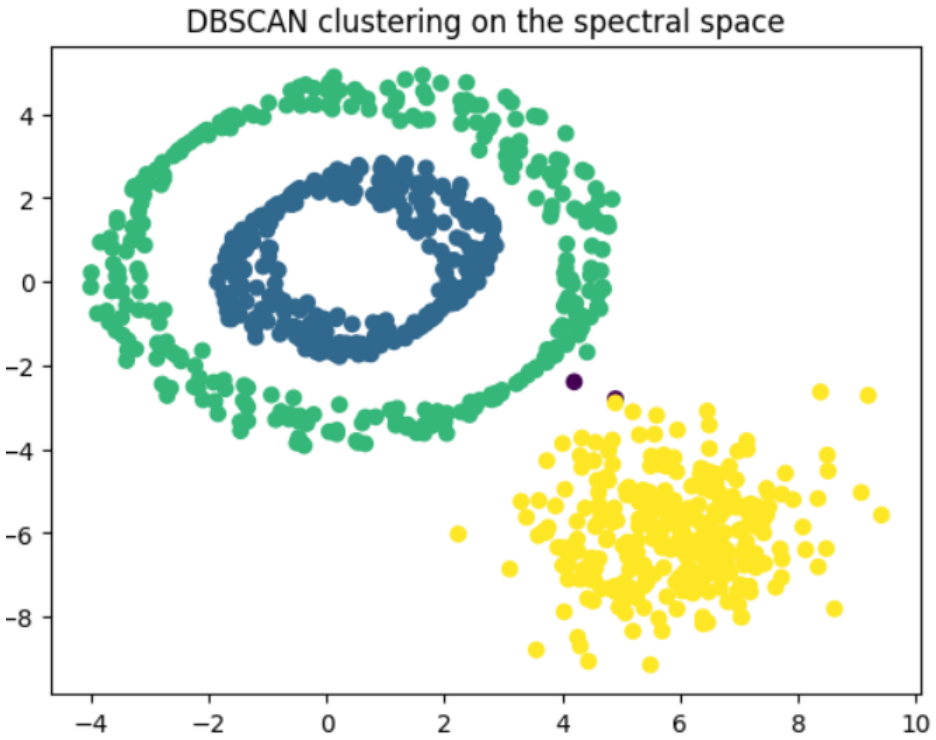


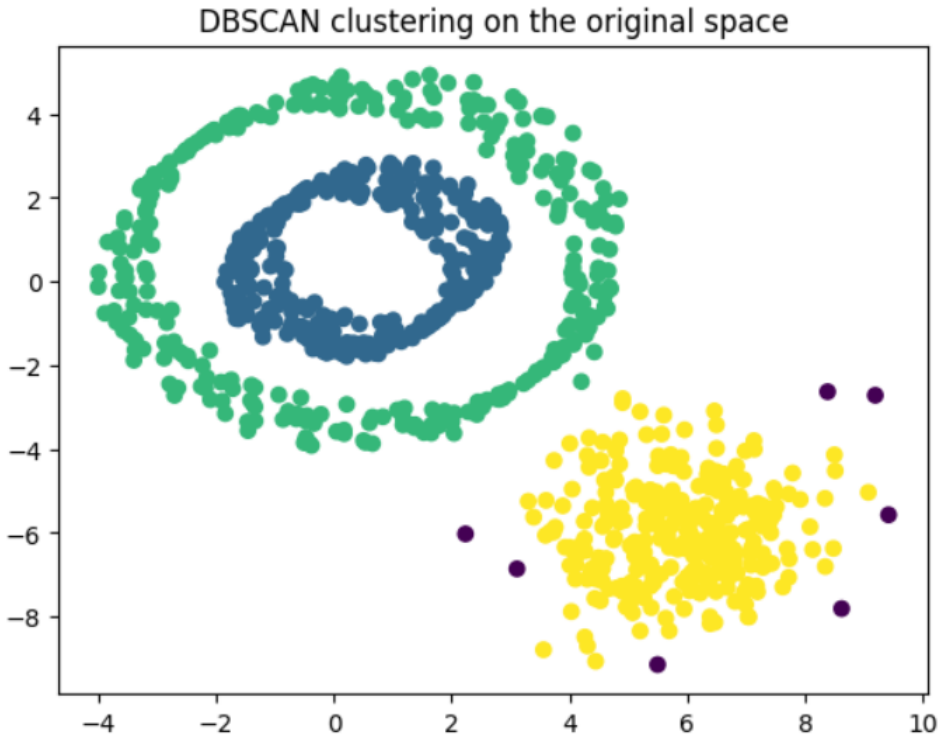

```
Silhouette score for KMeans clustering applied on the 3-dimensional points
    : 0.32326843682162765
Silhouette score for KMeans clustering applied on the original points:
    0.47444975618735247
Adjusted rand score: 0.4701072314954328
```

Although the silhouette scores are slightly different, the adjusted rand score indicates a partial agreement between the two clusterings.

DBSCAN

Plotting the results obtained by applying DBSCAN separately to the points in the new space and the original space, the resulting clusters are as follows.





```
Silhouette score for DBSCAN clustering applied on the 3-dimensional points
    : 0.16110984675031592
Silhouette score for DBSCAN clustering applied on the original points:
    0.225795331124162
Adjusted rand score: 0.9853681854685423
```

As usual, the silhouette score for the second clustering is slightly higher, and the adjusted rand score returns a value close to 1.

```
Adjusted rand score between the Kmeans clustering and the DBSCAN
    clustering on U_circles_20: 0.9966611990315535
```

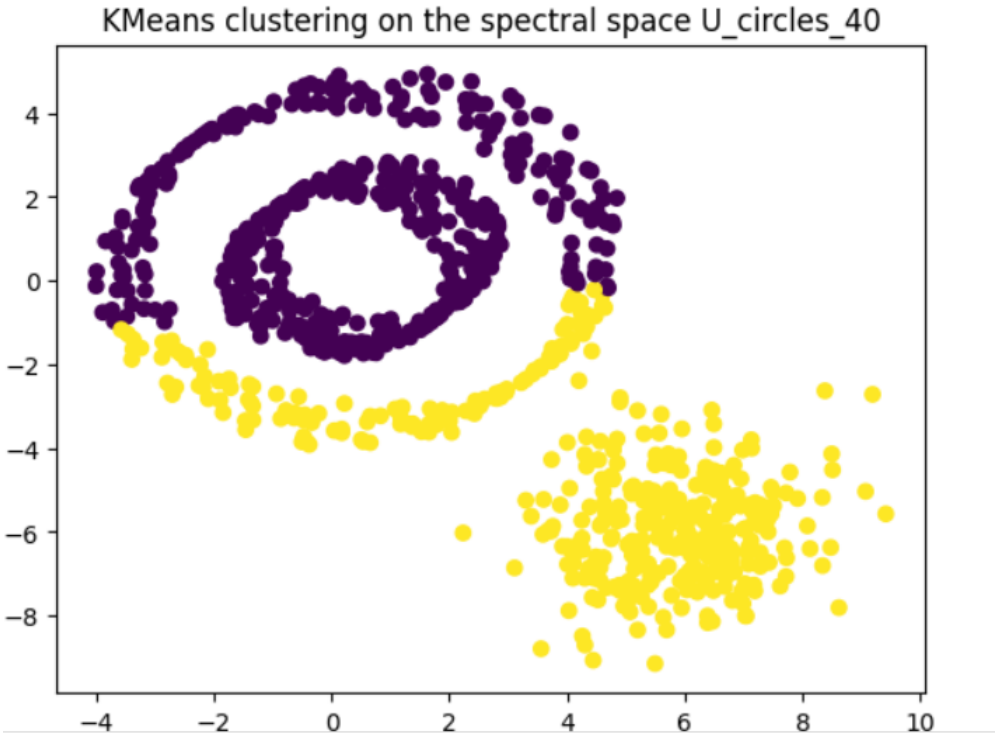
Finally, when comparing the clusters obtained using KMeans and DBSCAN on the points in the new space, the result is that both clusterings show almost perfect agreement.

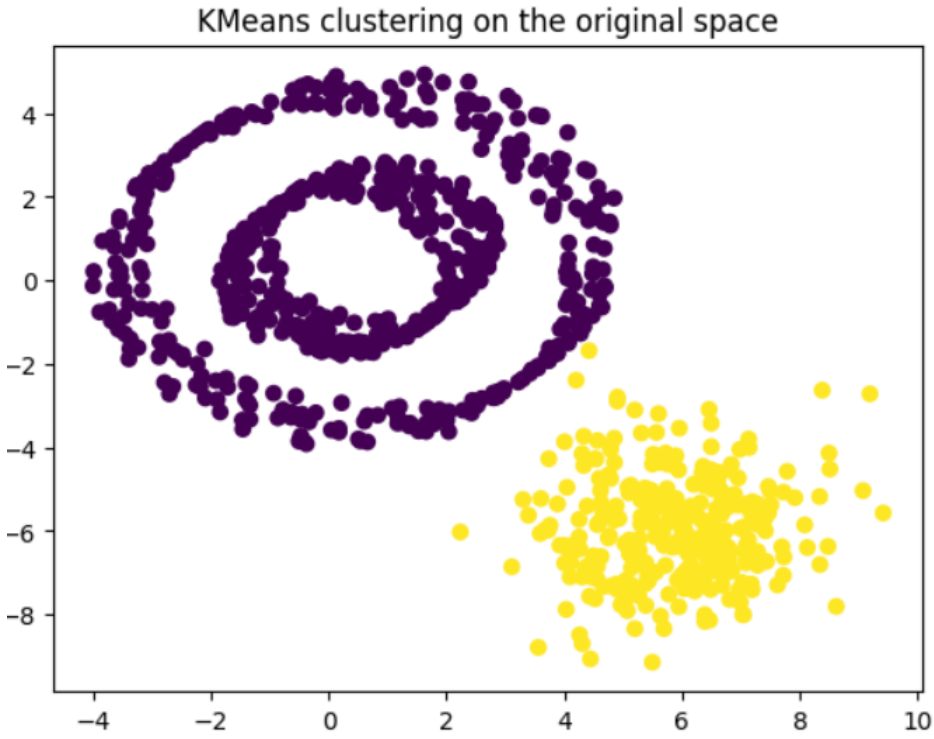
3.3 k = 40

In this case as well, the number of connected components is 1. After some trials, we choose the first 4 eigenvectors, which will generate the new space where we will represent the points from Circles.csv.

KMeans

Attempting to apply KMeans with $n_{\text{clusters}} = 3$, we do not obtain satisfactory results. Therefore, when trying $n_{\text{clusters}} = 2$, we obtain more appropriate clusters.



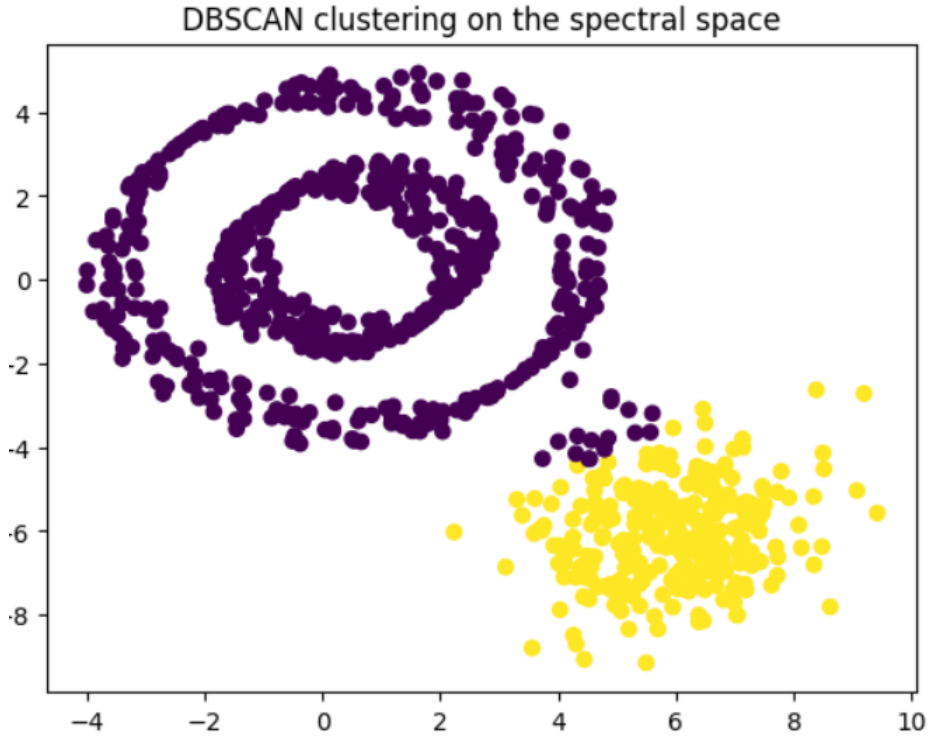


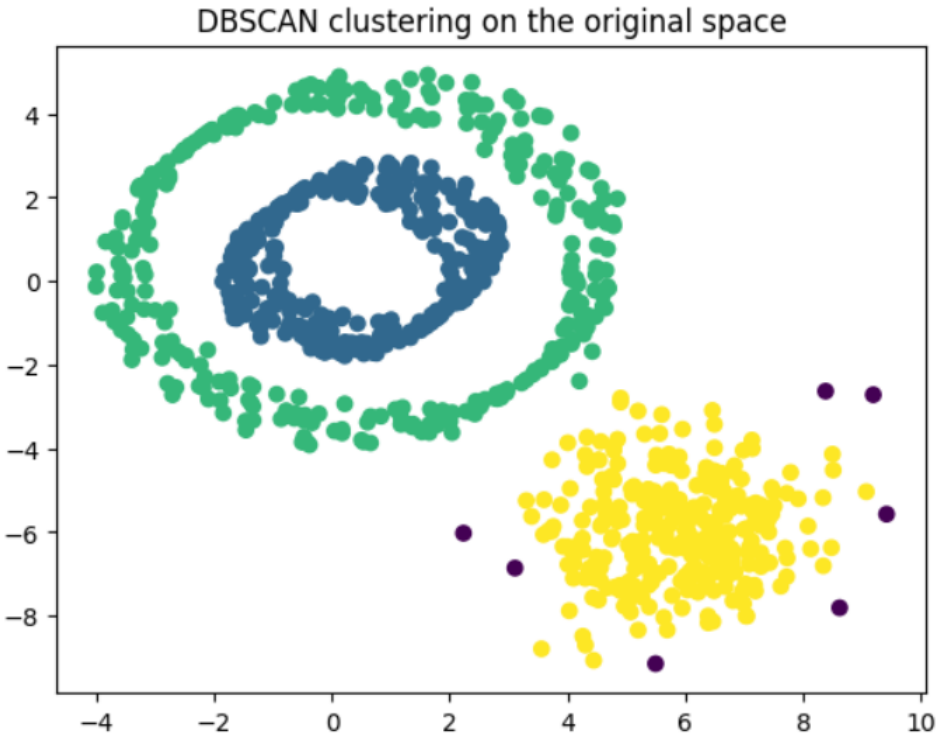

```
Silhouette score for KMeans clustering applied on the 4-dimensional points
    : 0.48917425960742955
Silhouette score for KMeans clustering applied on the original points:
    0.48917425960742955
Adjusted rand score: 0.5241588587178106
```

Indeed, the silhouette scores increase.

DBSCAN

For DBSCAN, we notice that between the two cases, we obtain a different number of clusters.






```
Silhouette score for DBSCAN clustering applied on the 4-dimensional points
    : 0.5678063871957475
Silhouette score for DBSCAN clustering applied on the original points:
    0.225795331124162
Adjusted rand score: 0.49609865320153373
```

Consequently, the adjusted rand score no longer returns a value close to 1.

```
Adjusted rand score between the Kmeans clustering and the DBSCAN
    clustering on U_circles_40: 0.5067274605273342
```

Similarly, the adjusted rand score between the labels obtained with KMeans and DBSCAN on the points in the new space is also lower.

4 Spirals dataset

The file `Spirals.csv` contains data for 312 points belonging to a 2-dimensional space. Additionally, each point is assigned a label that classifies it into one of the 3 classes. The original classification is illustrated in the Figure 4.

The 3 classes have a particular behaviour: their shape is not globular and their points are more distant from each other as they approach the figure's edges. Indeed, it is not surprising to observe bad clustering results from the K-means method, as shown in Figure 5(a), since it typically struggles with clusters of irregular shapes. Meanwhile, the DBSCAN method achieves perfect clustering, as demonstrated in the following code snippet.

```
Adjusted rand score between the DBSCAN clustering on the original points
    and the original labels: 1.0
```

Then, our aim is to focus on the K-means method applied after the spectral clustering in order to get a better approximation of the labels with respect of the original classes.

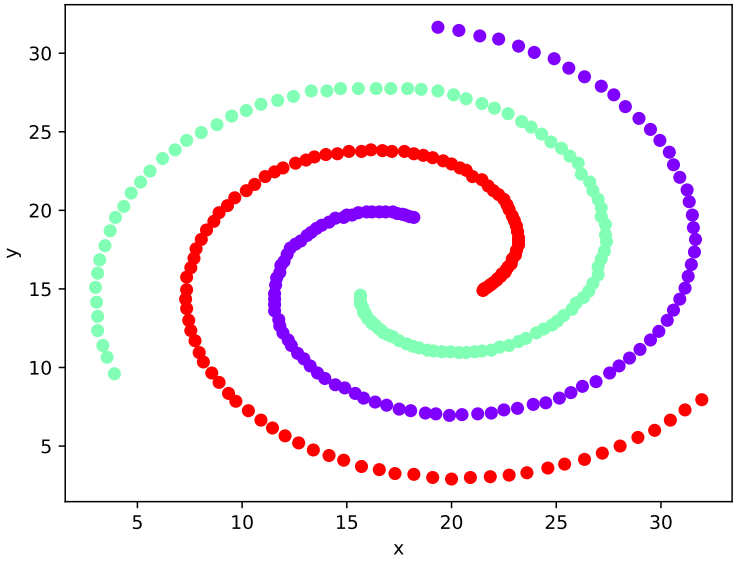


Figure 4: The dataset `Spirals.csv`

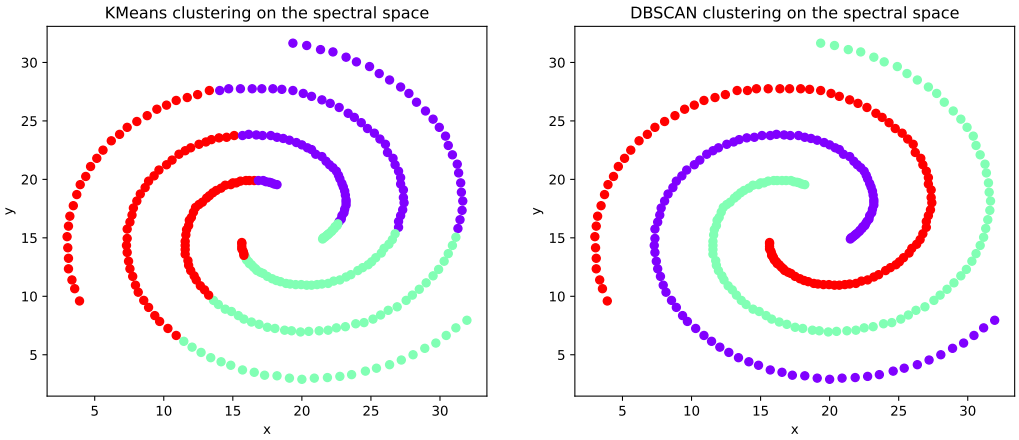


Figure 5: K-means and DBSCAN method on `Spirals.csv`

4.1 $k = 10$

First, we consider the 10-nearest neighbourhood adjacency matrix `W_spiral_10` and generate the degree matrix `D_spiral_10` and the Laplacian matrix `L_spiral_10 = D_spiral_10 - W_spiral_10`.

This matrix has only 1 null eigenvalue, hence one connected component in its graph and then we choose $M = 3$, that represent the number of the smallest eigenvalues of `L_spiral_10`.

When we apply the K-means with `n_cluster=3` as hyperparameter, the adjusted Rand score with respect of the original labels is not so high. Indeed, in the Figure 6 it is possible to observe that the behaviour in the centre of the plot is really good while it gets worse in the edges.

```
Adjusted rand score between the Kmeans clustering on U_spiral_10 with 3
    columns and the original labels: 0.48587064446954364
```

Then we try to increase the M value to 11. When applying the same procedure as before, the result is Figure 7 doesn't seem to improve but we try a different way.

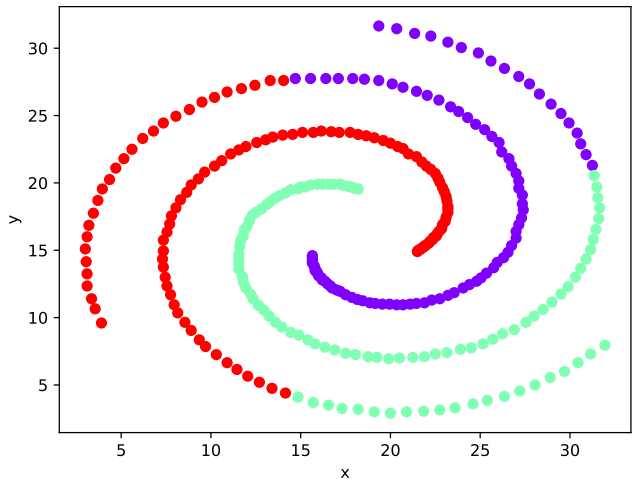


Figure 6: K-means clustering on `U_spiral_10` with 3 columns

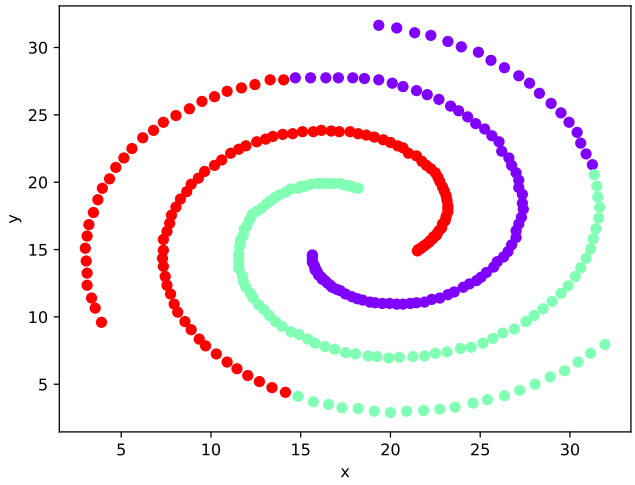


Figure 7: K-means clustering on `U_spiral_10` with 11 columns

After checking how many clusters we need to get the higher adjusted Rand score with respect to the original labels, we choose a value of 7 for the `n_cluster` and so perform another time the K-means method.

```
Adjusted rand score between the Kmeans clustering on U_spiral_10 with 7
    columns and the original labels: 0.4288653467614152
```

Even if the adjusted rand score is lower than previously and there are more clusters than the necessary ones, it is possible to merge them and get a better approximation of the original labels.

4.2 $k = 20, 40$

Then, the clusterings originated from the 20-nearest neighbourhood and the 40-nearest neighbourhood adjacency matrix have been tested, but none of them has given a satisfactory result, despite testing various values of M and numbers of clusters as done before.

We report only the results on `U_spiral_20`, which has 7 columns, and `U_spiral_40`, which

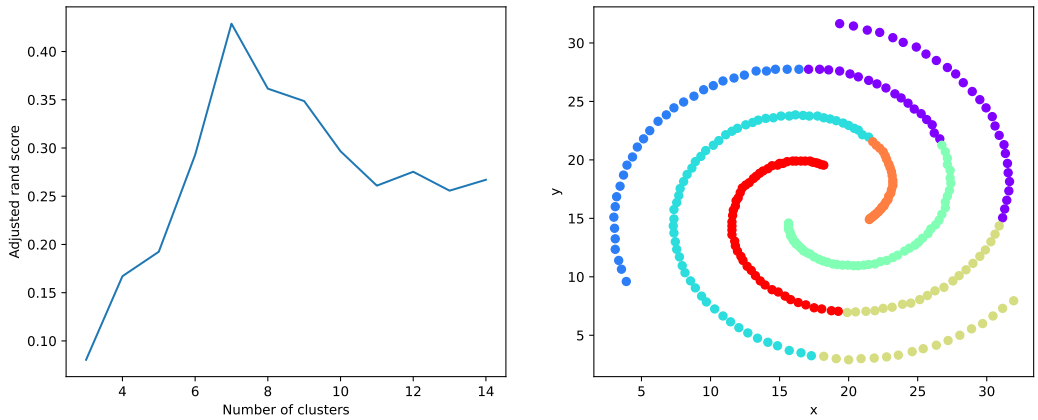


Figure 8: K-means clustering on `U_spiral_10` with 11 columns and 7 clusters

has 6 columns, as the others appear to be of little relevance.

```
Adjusted rand score between the Kmeans clustering on U_spiral_20 and the
    original labels: 0.19055933577097697

Adjusted rand score between the Kmeans clustering on U_spiral_40 and the
    original labels: 0.031259480223890246
```

The low Rand Index scores indicates some very poor clusterings, as it is possible to notice in Figure 9.

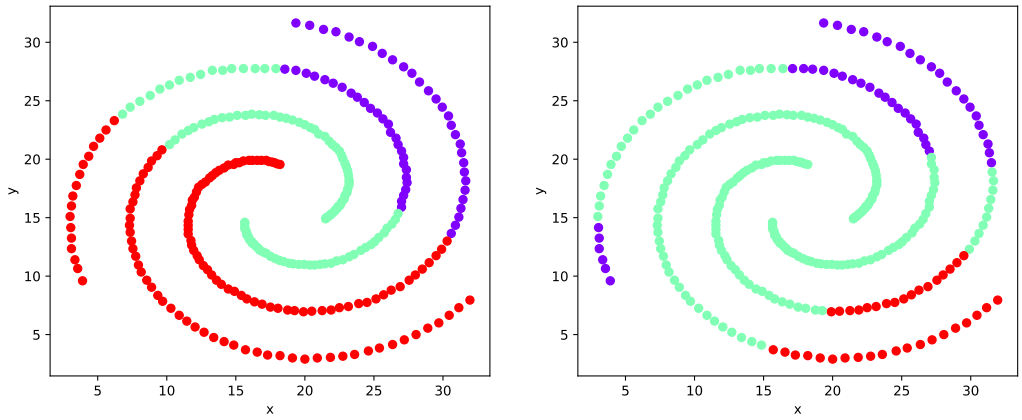


Figure 9: K-means clustering on `U_spiral_20` and `U_spiral_40`

The performance of K-means on `U_spiral_20` is poor, as evidenced by the low adjusted Rand score relative to the original labels. Similarly, the clustering based on the Laplacian matrix `L_spiral_40` yields unsatisfactory results. These findings suggest that using 20 or more nearest neighbours for each point leads to ineffective clustering.

4.3 $k = 8$

Since we have understood that it is better to lower the parameter k instead of increasing it, we try $k = 8$. The Laplacian matrix `L_spiral_8` obtained from the 8-nearest neighbourhood adjacency matrix has again 1 null eigenvalue, that is the number of connected components

in the Laplacian graph. After looking at the smallest eigenvalue of the Laplacian matrix we choose $M = 3$.

The K-means method results in a better clustering than all the other cases analized thanks to a quite high value for the adjusted Rand score, as displayed in the following code snipped.

```
Adjusted rand score between the Kmeans clustering on U_spiral_8 and the
    original labels: 0.7862289401393443
```

In the Figure 10 we can notice that the three clusters are quite well divided with the only exception of the tails. This seems the best clustering we could find on the dataset `Spirals.csv` using the K-means method, and it enhances also the K-means clustering on `U_spiral_10` in Figure 6.

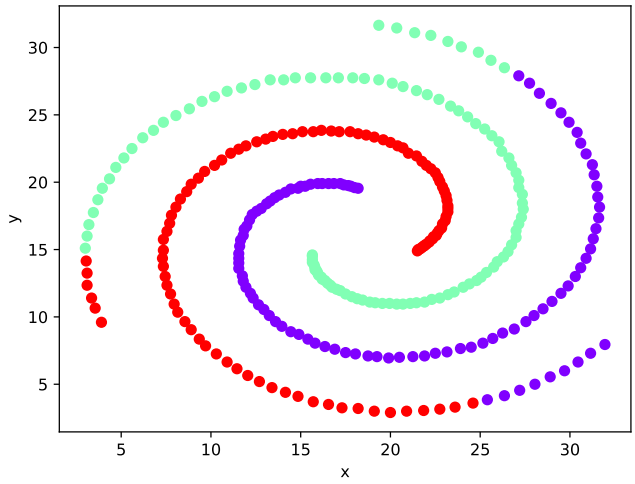


Figure 10: K-means clustering on `U_spiral_8`