# A simple unsupervised clustering analysis example using K-means

Godfrey Leung

There is a simple unsupervised clustering analysis example on some 2 or 3-dim datasets using K-means. The analysis is done using Python 3 and Jupyter notebook.

## 1 Exploratory data analysis (EDA)

A sample of 500 points generated from a 2D multivariate Gaussian mixture distribution is given in a csv file named 'sampledata_2D.csv'. The aim of this exercise is to perform clustering analysis on this sample dataset (and other variations of multivariate Gaussian mixture distributions) and categorise the data points into separate groups according. This is an example of unsupervised learning classification problem. A small subset of the sample are shown in Fig. 1.

| | x | y |
|---|---|---|
| **0** | 5.40230 | -7.1490 |
| **1** | -7.34510 | 15.2730 |
| **2** | 8.75070 | -5.9696 |
| **3** | -14.04800 | 17.6710 |
| **4** | 0.24255 | 6.8476 |
| **5** | 4.53560 | -3.6764 |
| **6** | 0.62931 | 5.6476 |
| **7** | 5.12890 | -4.0119 |

Figure 1: The first few data points in the sample.

The data are distributed in a 2D space as shown in the following plot, see Fig. 2. From the plot, we can see the data are roughly scattered in 3-4 clusters, with 5 outliners.

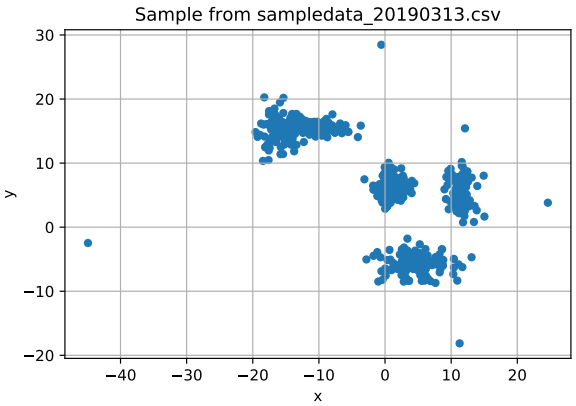


Figure 2: 2D plot of the sample dataset. From the plot, we can see there are roughly 4 clusters within the dataset, with a few outliners.

## 2 Clustering Analysis

This is a unsupervised learning problem since no labels are given for the data points. To group the sample data into appropriate clusters, **I applied the K-Means algorithm to find the corresponding clustering within the dataset.** The built-in KMeans module in the Python scikit-learn (sklearn) library is used here. A brief explanation on how the K-means algorithm works:

- Initially, randomly pick $N$ centroids in D-dimensional space ($D$ being the number of features). Ideally the centroids should be near the data but different from one another.
- Then assign each data point to the closest centroid.
- Move the centroids to the average positions of the data points which are assigned to it.
- repeat the previous two steps until convergence.

**K-means algorithm requires the number of clusters $N$ to be specified before running. From inspecting how the points are scattered in the 2D space, I chose to set this hyperparameter $N = 4$.** Under this algorithm, the data points are grouped into 4 clusters, each with the following size. To avoid overfitting and validate the clustering model in the case of new data, the sample is splitted into train and test dataset, where only the train dataset is used to fit the model and the test dataset is for cross-validation.

| cluster label | number of data |
|---|---|
| 0 | 110 |
| 1 | 192 |
| 2 | 107 |
| 3 | 91 |

Figure 3: Size of each cluster found using the K-means algorithm.

The centroids (means) of the corresponding clusters $\{0, 1, 2, 3\}$ are shown as below, see Fig. 4 and 5.

| cluster labels | x_0 | y_0 |
|---|---|---|
| 0 | -13.204885 | 15.184254 |
| 1 | 1.071899 | 6.118891 |
| 2 | 4.705526 | -5.858982 |
| 3 | 11.646568 | 5.764125 |

Figure 4: The $\{x, y\}$ coordinates of the cluster centroids.

### 2.1 Covariance matrix

Covariance matrix for each cluster is defined as

$$K_{X_i X_j} = \mathrm{cov}(X_i, X_j) \equiv E\left[(X_i - E[X_i])(X_j - E[X_j])\right] \tag{1}$$

where $i, j$ runs from 1 to 2 here, with $\{X_1, X_2\}$ denotes the $\{x, y\}$ coordinates of the data points within the cluster.

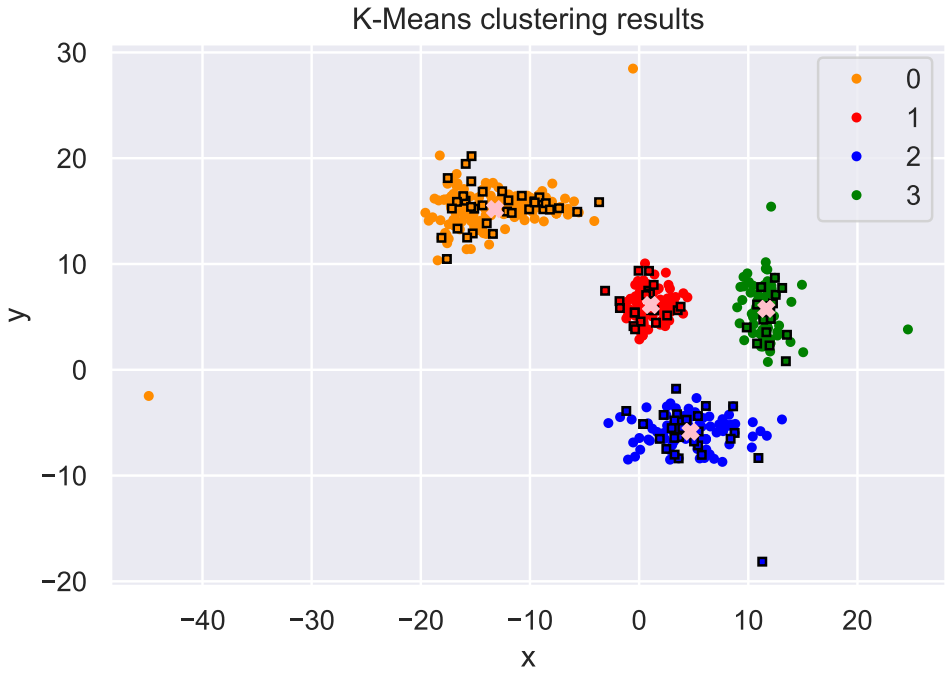


Figure 5: 2D plot of the sample dataset, classified into 4 clusters using the K-Means algorithm. Here 80% of the sample, denoted by the circular markers, are used as train dataset to build the K-means clustering model. The remaining data, denoted by the square markers, are the test dataset for cross-validation. The pink 'X' markers denote the centroids of each clusters.

Covariance matrix is symmetric. The elements of each cluster covariance matrix denote the covariance of the data point within the cluster. Covariance is a measure of the joint variability of two random variables. The diagonal elements are simply the variances of the data points in the x and y directions respectively, which describe how much the data within each cluster are spread in the two directions. The off-diagonal terms on the other hand describe how changes in the x-direction are associated with changes in the y-direction for data within the cluster.

The covariance matrices of each cluster for this sample is given as

| | | x | y |
|---|---|---|---|
| **KMeans_labels** | | | |
| **0** | **x** | 18.785422 | 4.551088 |
| | **y** | 4.551088 | 4.807211 |
| **1** | **x** | 1.905480 | 0.100096 |
| | **y** | 0.100096 | 2.035818 |
| **2** | **x** | 9.722362 | -0.810570 |
| | **y** | -0.810570 | 3.511195 |
| **3** | **x** | 3.127399 | -0.645064 |
| | **y** | -0.645064 | 5.797291 |

Figure 6: The covariance matrices for each cluster found using the K-means algorithm.

## 2.2 Model evaluation

Contrary to supervised learning where we have the ground truth to evaluate the model's performance, clustering analysis does not have a solid evaluation metric that we can use to evaluate the outcome of different clustering algorithms in general. However, in the cluster-predict methodology, we can evaluate how well the models are performing based on different choices of the number of cluster hyperparameter $N$ since clusters are used in the downstream modeling. Two metrics that may give us some intuition about the optimal number of clusters are considered here, namely

- Elbow method
- Silhouette analysis

We might also compare the K-means clustering results with other clustering algorithms and benchmark models (For instance, assigning the data points into $N$ clusters in a completely random fashion.)

### Elbow method

Elbow method gives us an idea on what a good $N$ number of clusters would be based on the inertia/sum of squared distance (SSE) between data points and their assigned clusters' centroids. It can act as a validation of consistency of the clustering model. Ideally one should pick $N$ at the spot where SSE starts to flatten out and forming an elbow.

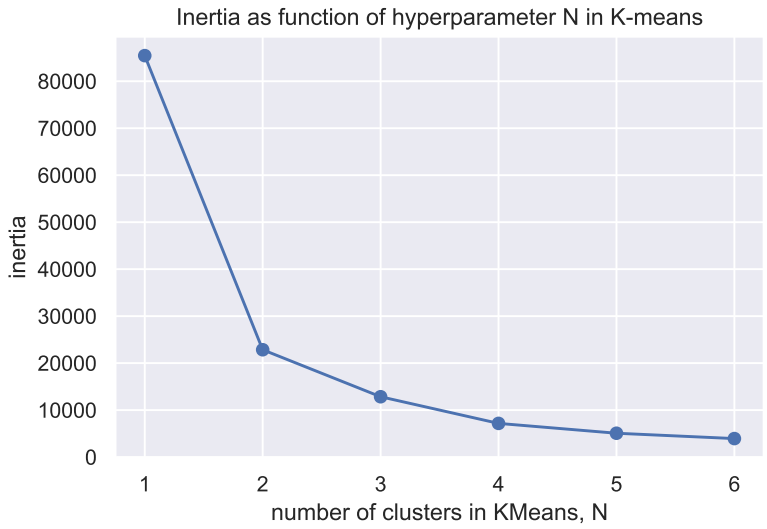


Figure 7: Inertia as a function of the hyperparameter $N$ of the K-means algorithm.

Fig. 7 shows the inertia of the clustering as a function of the hyperparameter $N$ of the K-means algorithm. From the plot, we can see the inertia starts to flatten out at around $N \approx 3 - 4$.

### Silhouette analysis

Next we perform the silhouette analysis. It can be used to determine the degree of separation between clusters. In general we would want the coefficients to be as big as possible and close to 1 to have a good clusters.

The Silhouette Coefficient is calculated using the mean intra-cluster distance $a$ and the mean nearest-cluster distance $b$ for each sample. The Silhouette Coefficient for a sample $i$ is given by

$$\textbf{Silhouette Coefficient} = \frac{b_i - a_i}{\textbf{max}(a_i, b_i)} \tag{2}$$

The best value is 1 and the worst value is -1. Values near 0 indicate overlapping clusters. Negative values generally indicate that a sample has been assigned to the wrong cluster, as a different cluster is more similar.

The silhouette score plot for each clusters for different choices of the number of clusters $N$ in the K-means algorithm is shown in Fig. 8 and 9. From the plots, we can see the choice $N = 4$ gives the best results, with the clusters of similar size (given by the thickness of the silhouette plot) and has the best average silhouette score of around 0.7. All clusters scores being above the average also shows that it is a good choice.

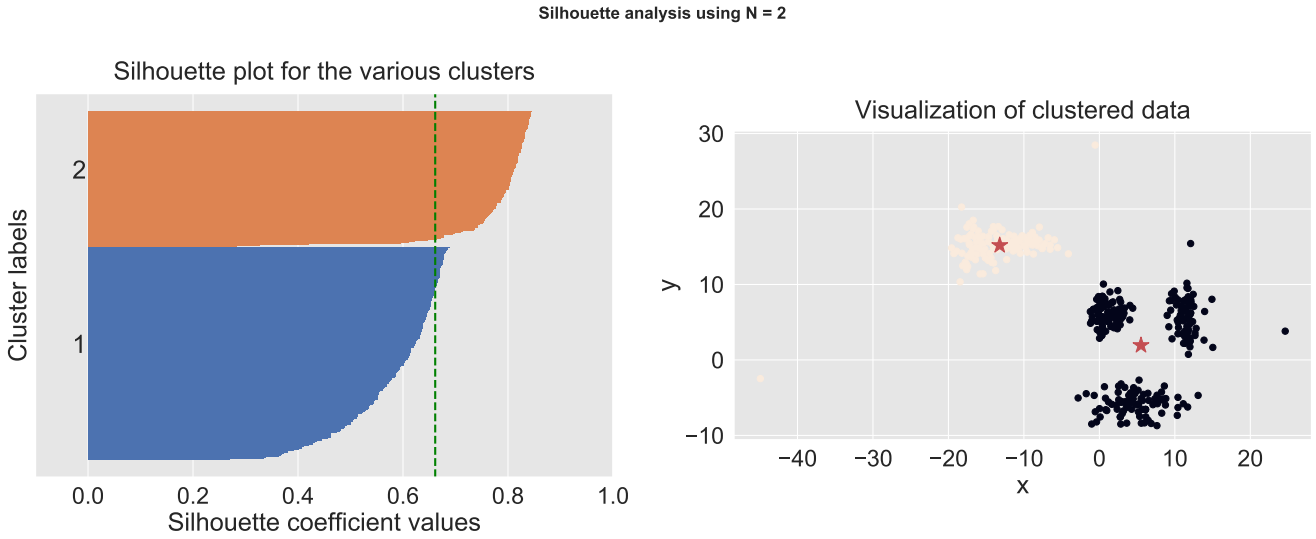


Figure 8: The silhoutette score for different choices of the number of clusters in the K-means algorithm.

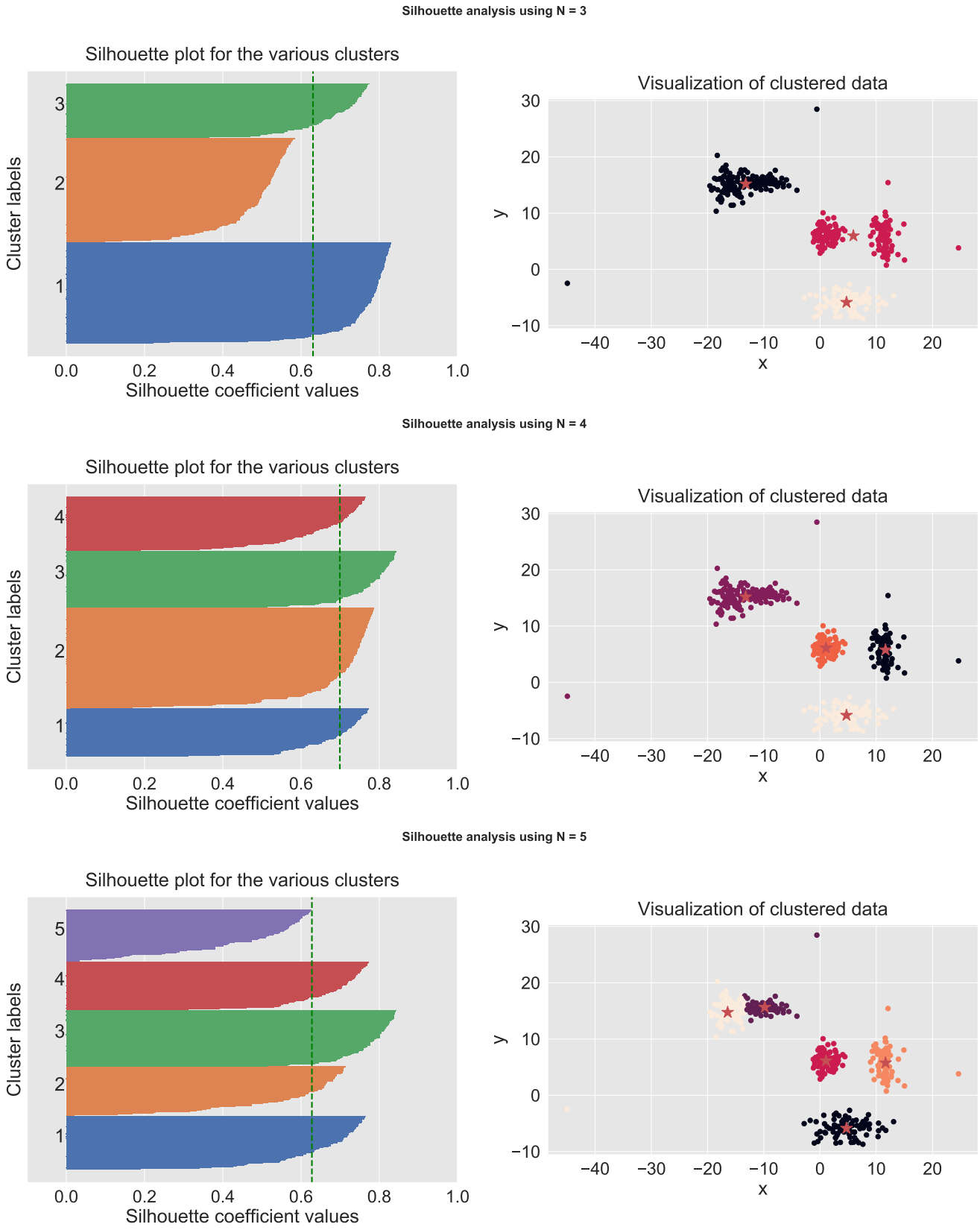


Figure 9: The silhoutette score for different choices of the number of clusters in the K-means algorithm. (con't)

## K-fold cross-validation

To validate how the clustering model would perform in the case of new data, a K-fold cross-validation test is performed. In the previous section, we already performed a simple version of a cross-validation test, see Fig.5. From the plot, we can see the test data are correctly assigned to the closest clusters. Here I give a more detailed 5-fold cross-validation, see Fig. 10. From the plots, we can see the model gives very similar clustering results for 5-fold cross-validation.

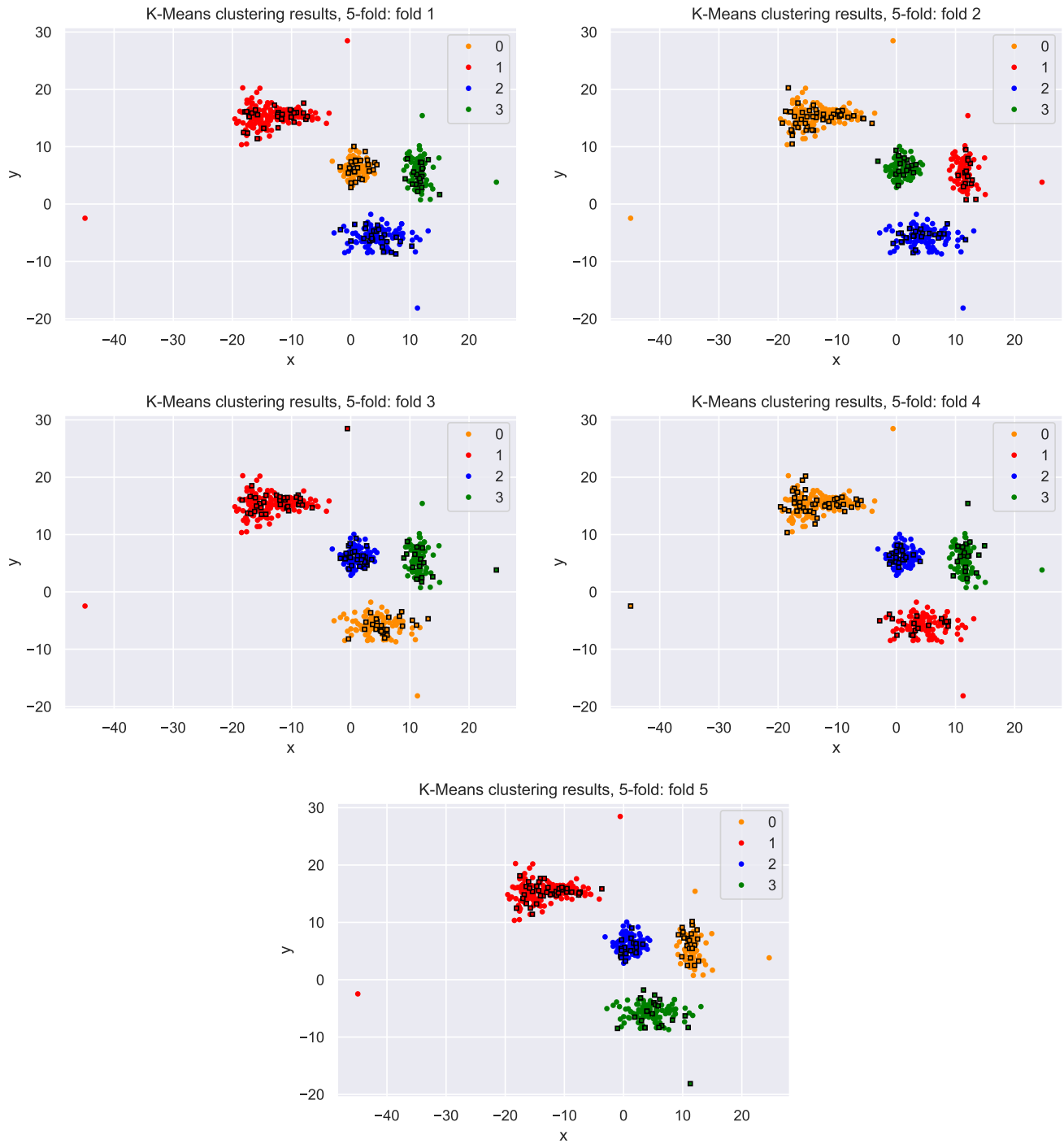


Figure 10: 5-fold cross-validation for clustering using K-means algorithm.

# 3 3D multivariate Gaussian distributions

Next we perform clustering analysis on some simulated 3D multivariate Gaussian distributions. To generate a sample of $M$ 3D multivariate Gaussian distributions, I use the make_blob module in the sklearn library.

## A mixture of two distributions

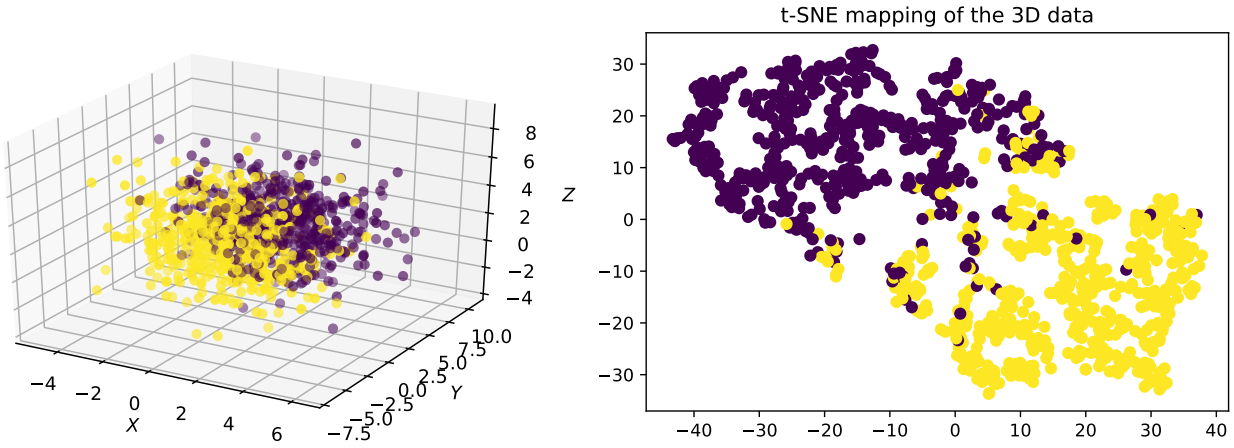


Figure 11: A mixture of 2 3D multivariate Gaussian distributions samples. **Left panel:** Visualisation in the 3D space; **Right panel:** t-SNE 2D mapping of the sample.

Fig. 11 shows a sample of 1000 data points from a mixture of 2 3D multivariate Gaussian distributions, with a standard deviation of 2 for each cluster. As in the 2D case, I performed clustering analysis on this sample example using the K-means algorithm. The number of clusters is specified to be 2 to begin with. Since the data are generated from simulation with ground truth cluster labels, we can it to compare with the labels from the K-means algorithm directly to evaluate the model performance in this case, see Fig. 12. From it, we can the K-means algorithm correctly classifies most of the data ($> 90\%$) into the two clusters.

| KMeans_labels | 0 | 1 |
|---|---|---|
| true_labels | | |
| 0 | 469 | 31 |
| 1 | 31 | 469 |

Figure 12: A cross-tabulation comparing the ground truth cluster labels and that computed from using the K-means algorithm.

| cluster labels | x_0 | y_0 | z_0 |
|---|---|---|---|
| 0 | 0.895939 | 4.401688 | 1.936548 |
| 1 | 0.950456 | -1.776220 | 2.861765 |

Figure 13: The $\{x, y, z\}$ coordinates of the cluster centroids for the mixture of 2 3D multivariate Gaussian distribution sample example.

| KMeans_labels | | x | y | z |
|---|---|---|---|---|
| 0 | x | 3.773730 | -0.068305 | 0.126430 |
| | y | -0.068305 | 3.412390 | 0.058466 |
| | z | 0.126430 | 0.058466 | 3.578485 |
| 1 | x | 3.377151 | 0.198938 | -0.289019 |
| | y | 0.198938 | 3.121541 | 0.039996 |
| | z | -0.289019 | 0.039996 | 3.917282 |

Figure 14: The covariance matrice of each cluster for the mixture of 2 3D multivariate Gaussian distribution sample example.

## Model evaluation

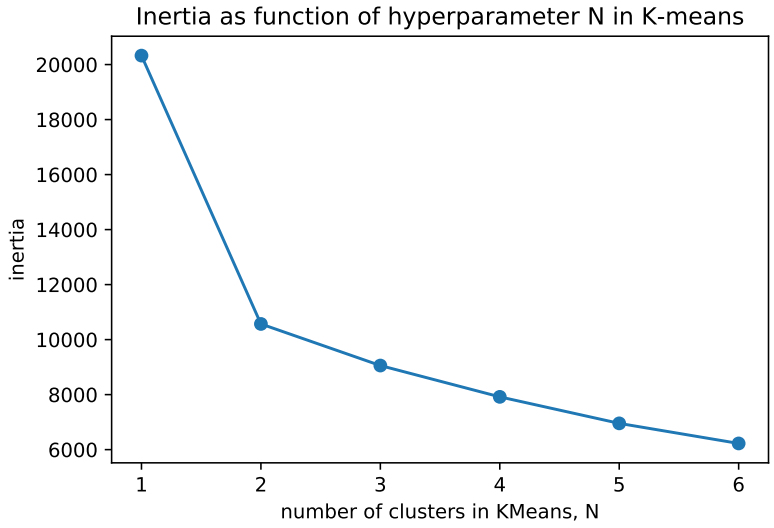


Figure 15: Inertia as a function of the hyperparameter $N$ of the K-means algorithm for the mixture of 2 3D multivariate Gaussian distributions sample example.

## Silhouette analysis

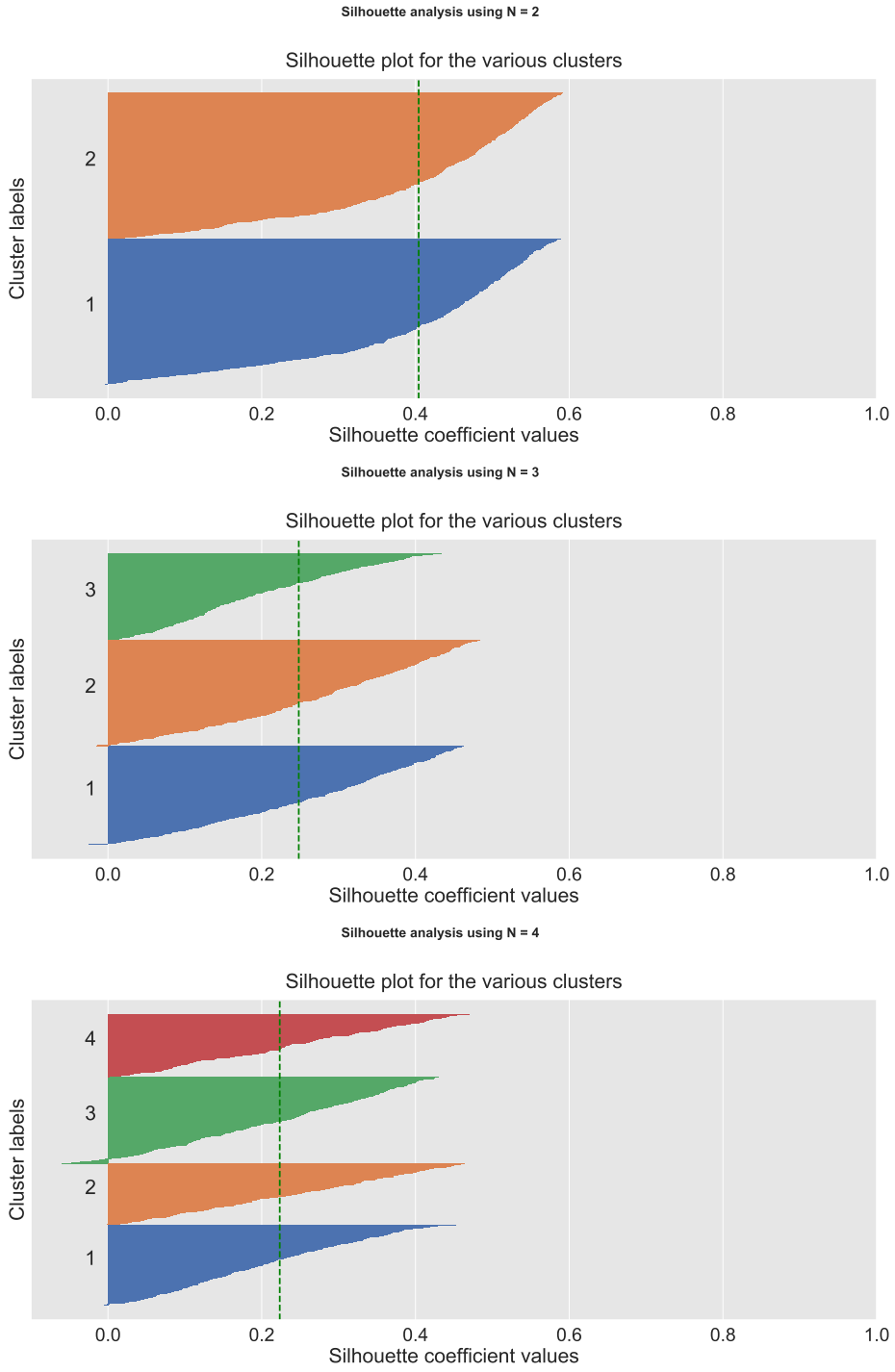


Figure 16: The silhoutette score for different choices of the number of clusters in the K-means algorithm for the mixture of 2 3D multivariate Gaussian distributions sample example.

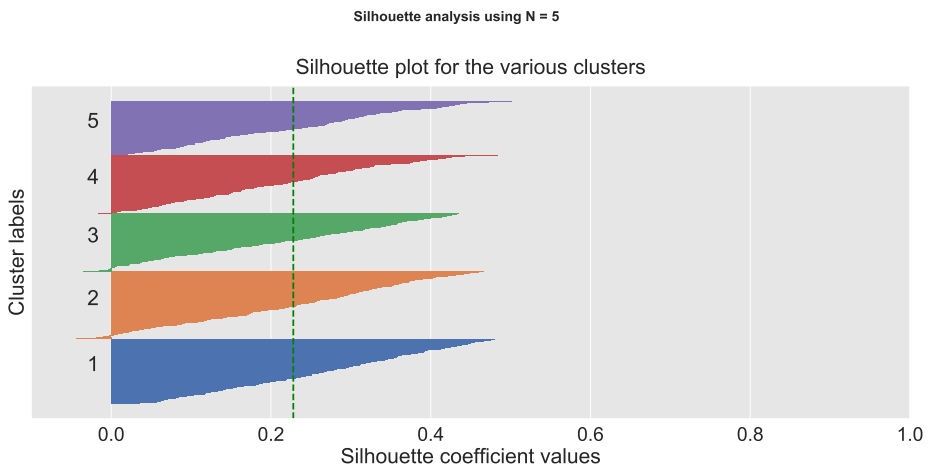


Figure 17: The silhoutette score for different choices of the number of clusters in the K-means algorithmfor the mixture of 2 3D multivariate Gaussian distributions sample example. (con't)

## A mixture of $M$ distributions

Here shows another example of a sample of 1000 data points from a mixture of M 3D multivariate Gaussian distributions, where $M = 5$.

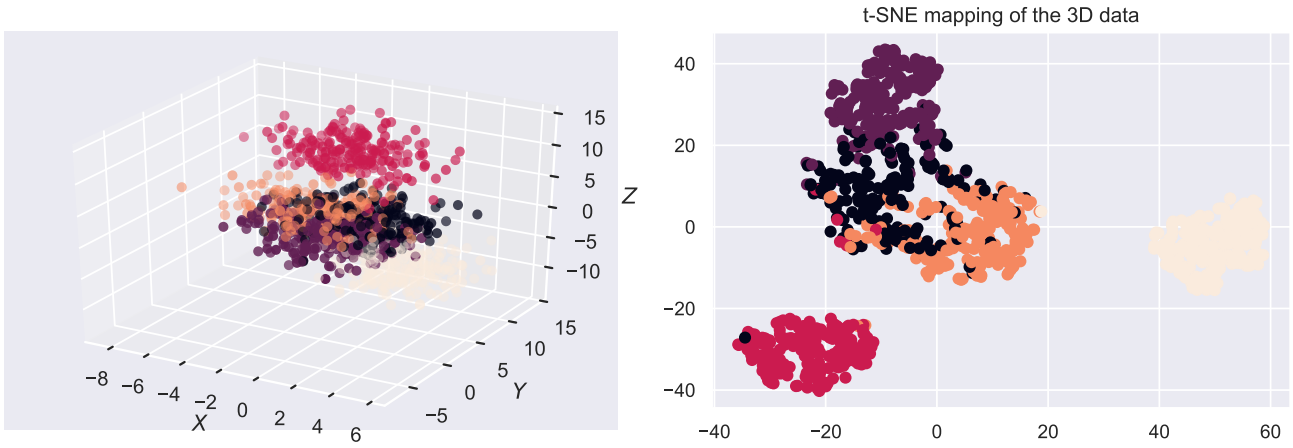


Figure 18: A mixture of ($M = 5$) 3D multivariate Gaussian distributions samples. **Left panel:** Visualisation in the 3D space; **Right panel:** t-SNE 2D mapping of the sample.

## Model evaluation

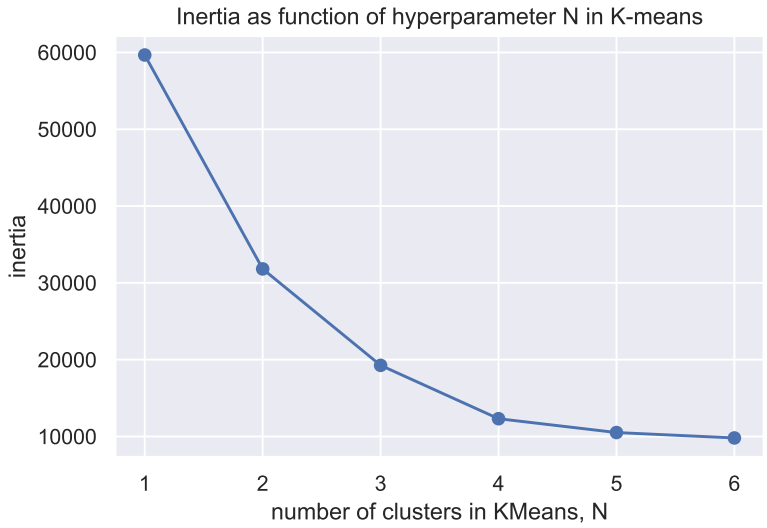


Figure 19: Inertia as a function of the hyperparameter $N$ of the K-means algorithm for the mixture of ($M = 5$) 3D multivariate Gaussian distributions sample example.

Both the elbow and silhouette analysis on evaluating the performance of the K-means models seem to suggest 4 clusters, $N = 4$, is the best choice instead of 5. This might be due to the fact that some of the simulated clusters are being too close to one another, as seen in Fig. 18.

## Silhouette analysis

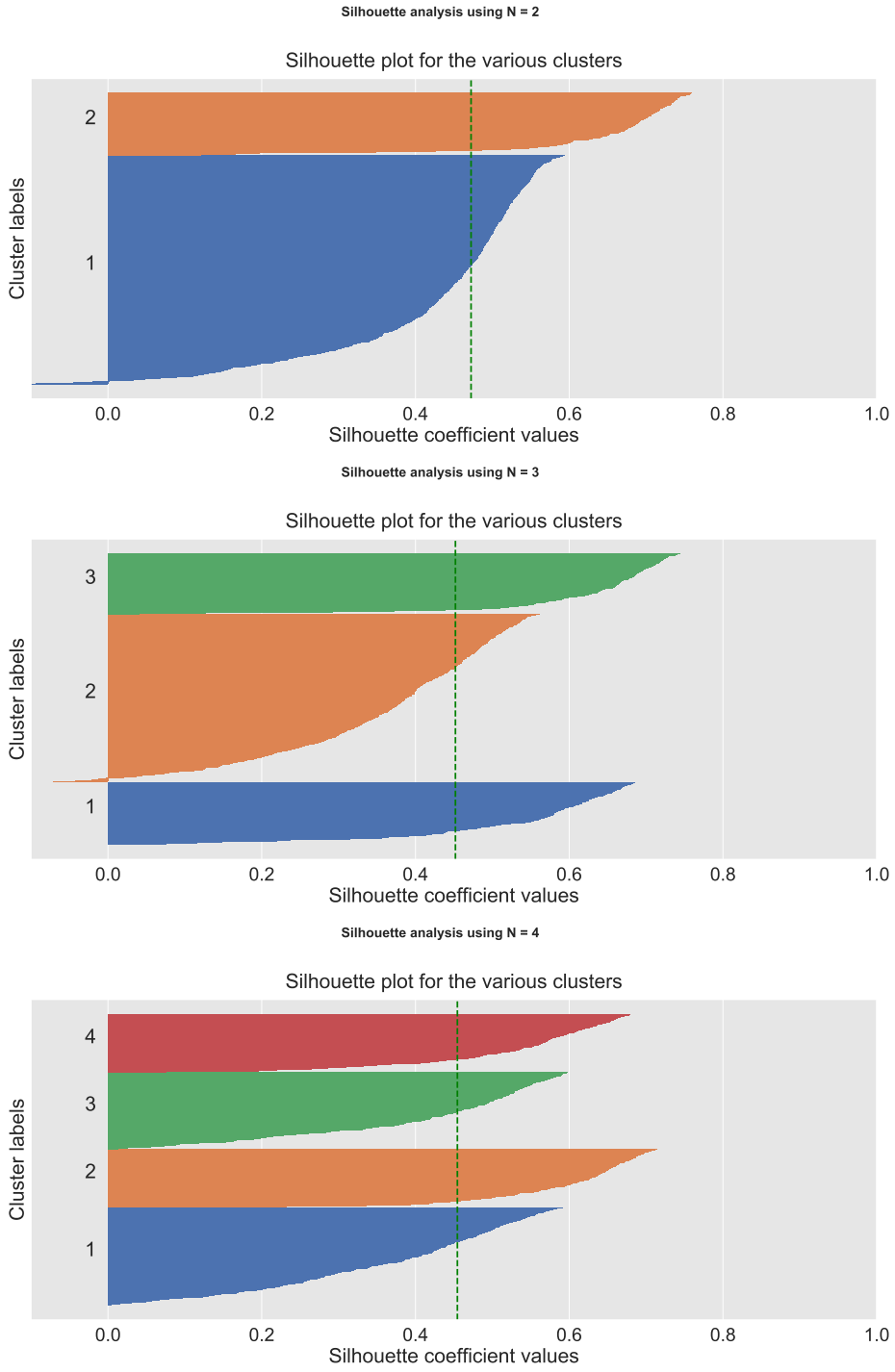


Figure 20: The silhoutette score for different choices of the number of clusters in the K-means algorithm for the mixture of 5 3D multivariate Gaussian distributions sample example.

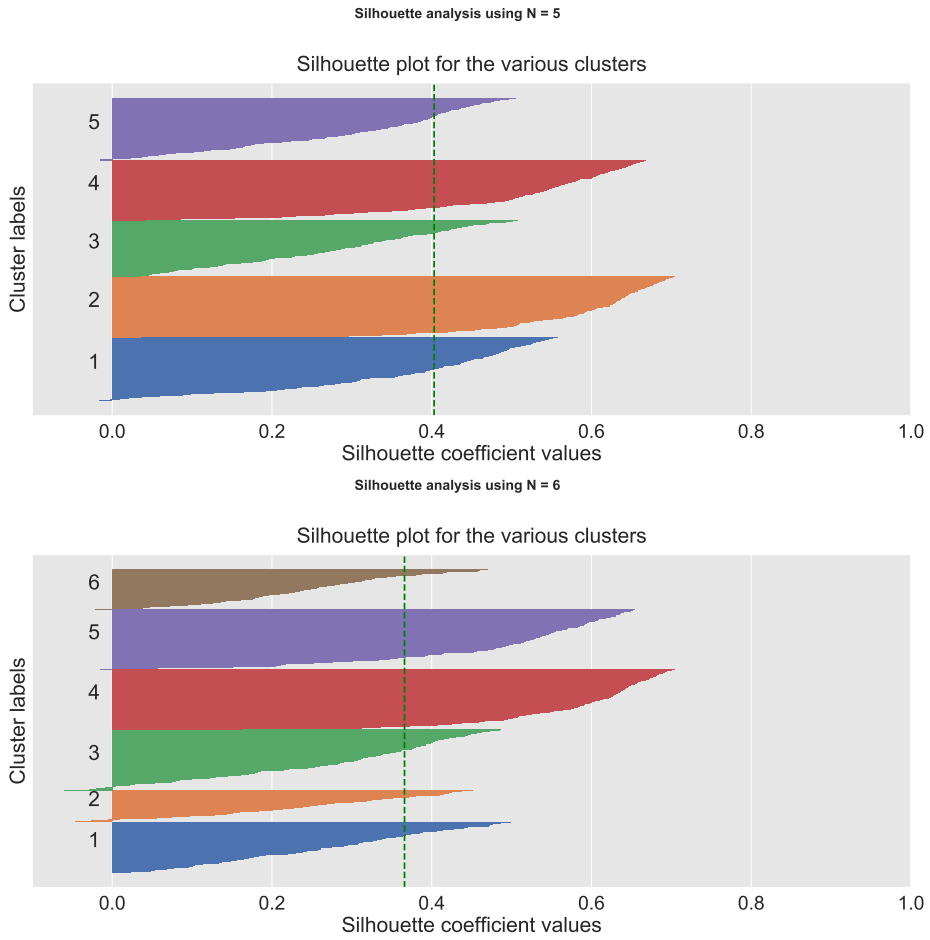


Figure 21: The silhoutette score for different choices of the number of clusters in the K-means algorithmfor the mixture of 5 3D multivariate Gaussian distributions sample example. (con't)

# 4 Assumptions and drawbacks of K-means algorithm

As discussed, one main assumption of the k-means algorithm is that the number of clusters $N$ have to be explicitly specified. The other implicit assumptions of the K-means algorithm are

- the clusters are assumed to be roughly isotropic
- the clusters are linearly separable
- the clusters have roughly equal variance

For the cases of data sampled from a uniform distribution or a radial distribution, these assumptions would be violated and the k-means algorithm would produce unintuitive and unexpected results. In such cases, **one might use single linkage hierachical clustering or spectral clustering instead**. In the following, I will illustrate these using an example of radial distribution sample.

## 4.1 Radial distribution examples

Here examples of dataset sampled from a radial distribution of some complicated geometric shapes such as moons and circles within each other are shown. The results of the clustering analysis done using the K-means and spectral clustering algorithms differ significantly, as shown in Fig. 22. Alternatively, it is possible K-means algorithm might also work by transforming the data into polar coordinates in the case of radial distribution samples.

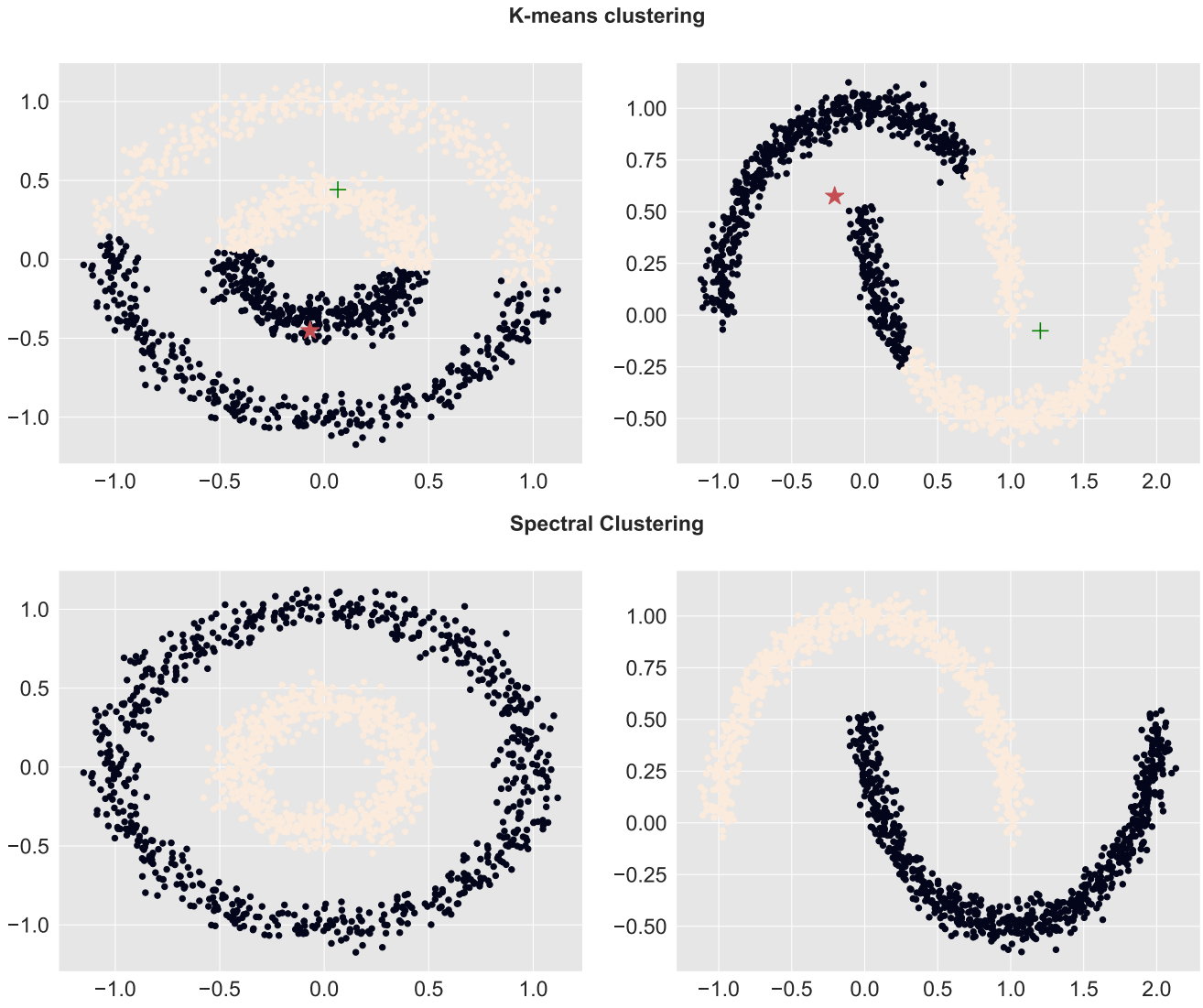


Figure 22: Examples of dataset sampled from some radial distributions (**Left panel:** circular rings; **Right panel:** moons). The data are grouped into clusters using the K-means and spectral clustering algorithms.