K-means & DBSCAN Review

Christos Panourgias

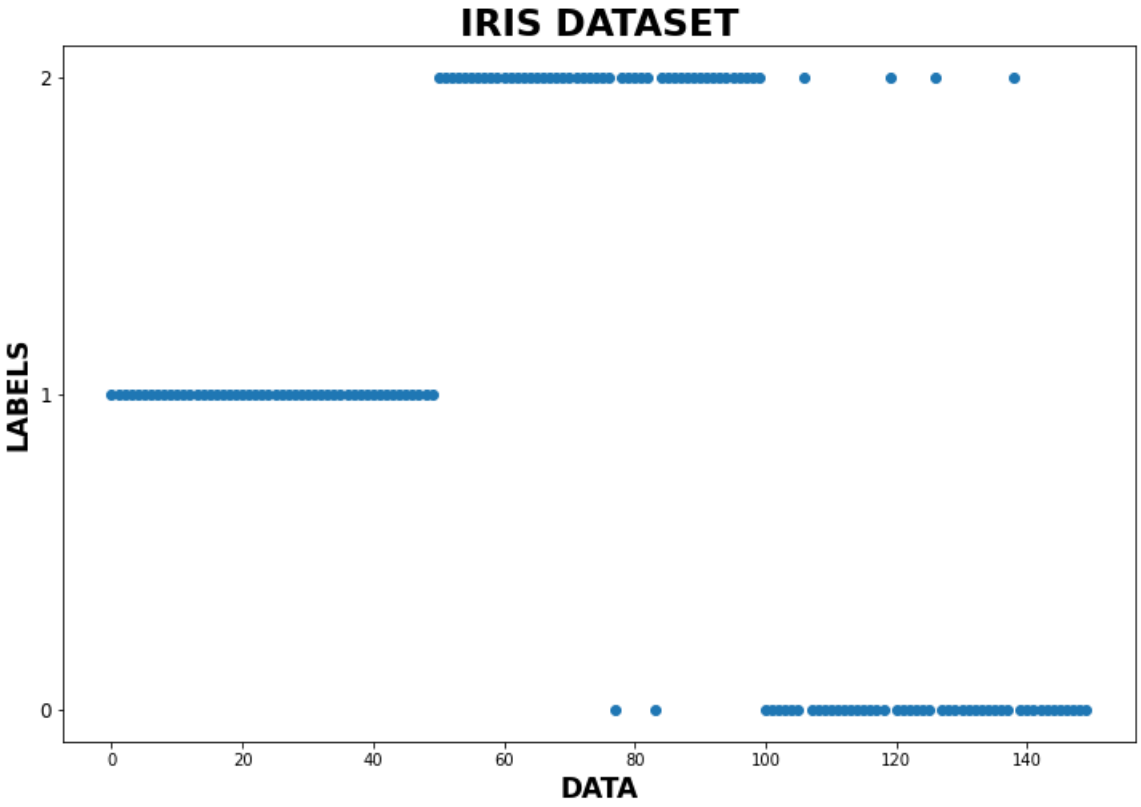
Contents

1 Clustering with k-means

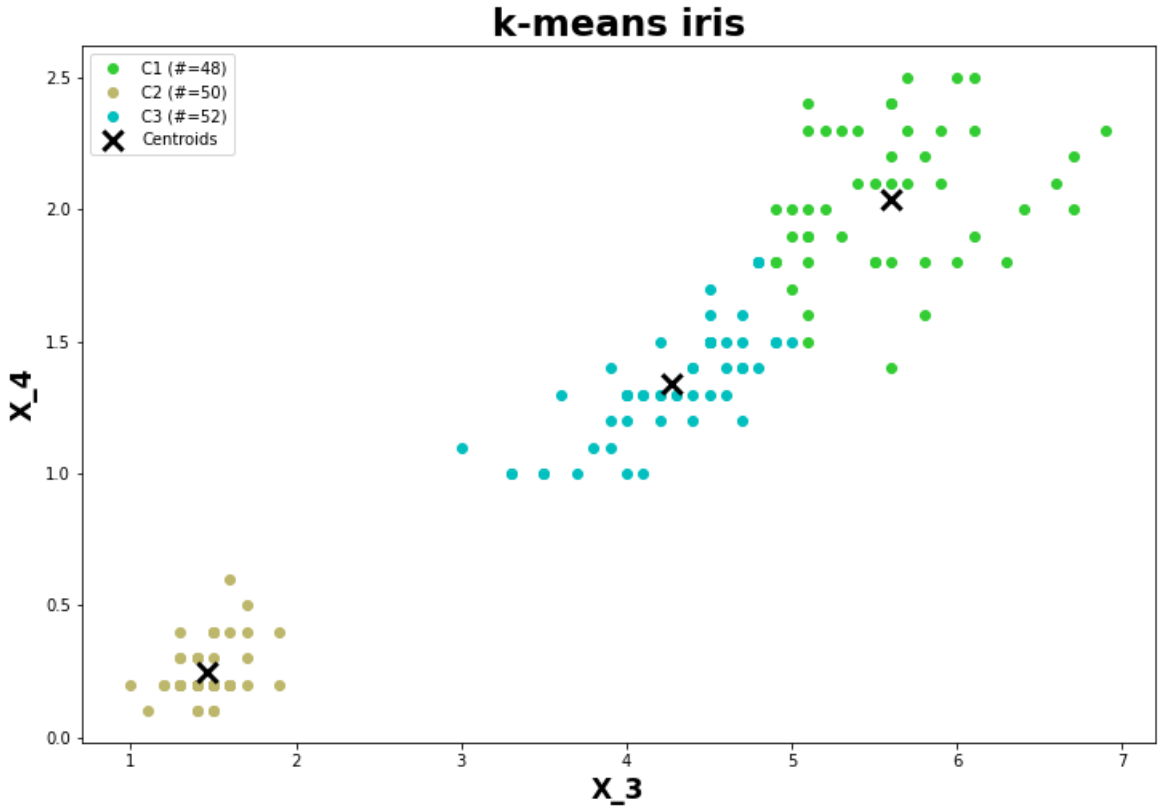
1.1 Application to the iris dataset

1.1.1 Introduction

The iris dataset consists of the three iris species, namely, setosa, versicolor and virginica, which are represented by classes 0, 1 and 2 respectively. The amount of data is 150 where the data is equally distributed in each class.



Knowing that the data classes are three and that petal length and width (features 3 and 4 of the data set) are most effective for classifying iris flowers, will apply the k-means clustering algorithm with 3 clusters for features 3 and 4.



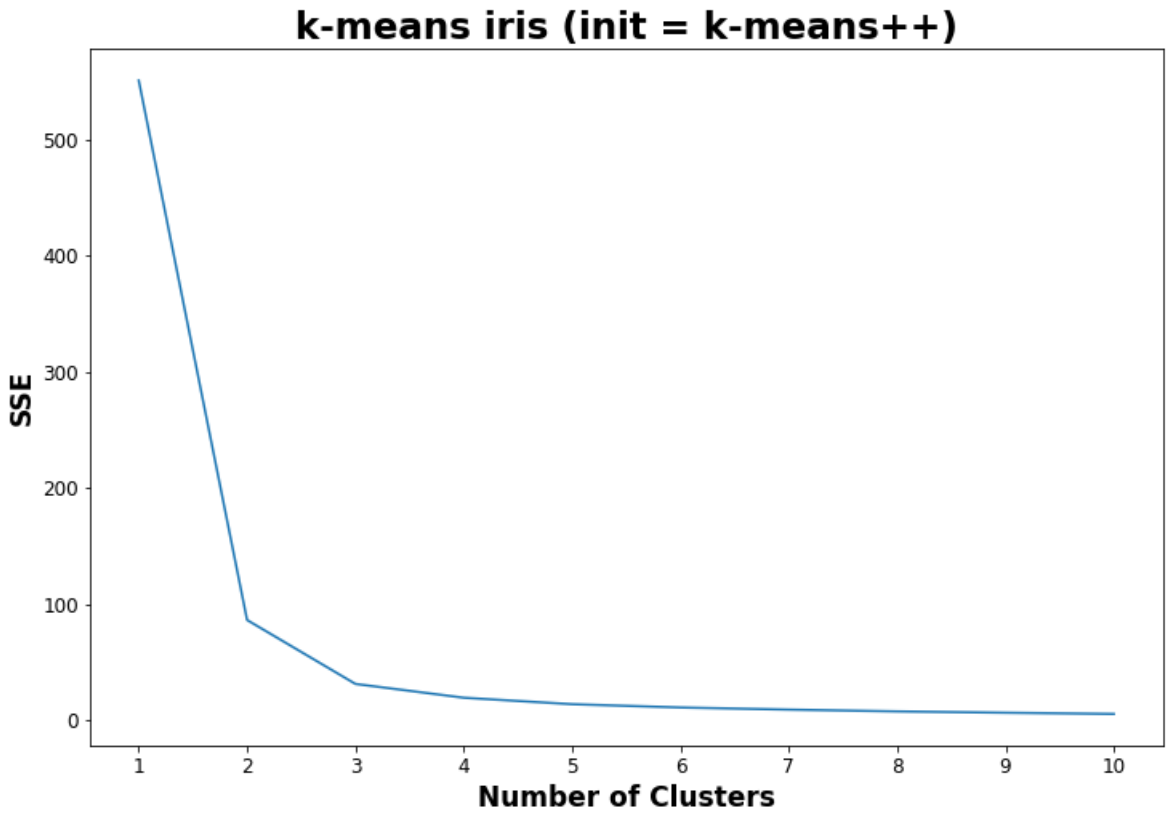
As shown in the graph above, the k-means algorithm has a good efficiency in separating the iris classes. The efficiency results will be investigated for various values of the k-means parameters and for different combinations of dimensions of the data table having as columns the length and width characteristics of the horseshoe.

1.1.2 Experiments with $Clusters \in \{2, 3, ..., 10\}$ and init = 'k-means++'

Using the 'k-means++' value of the init parameter better initializes the centers of the k-means algorithm so that faster convergence is achieved.

init = k-means++			
Clusters	SSE	Silhouette Score	Iterations
1	550.90	NaN	2
2	86.39	0.77	2
3	31.37	0.66	3
4	19.48	0.61	5
5	13.98	0.59	9
6	11.07	0.57	8
7	9.21	0.58	8
8	7.93	0.56	3
9	6.58	0.59	7
10	5.70	0.43	5

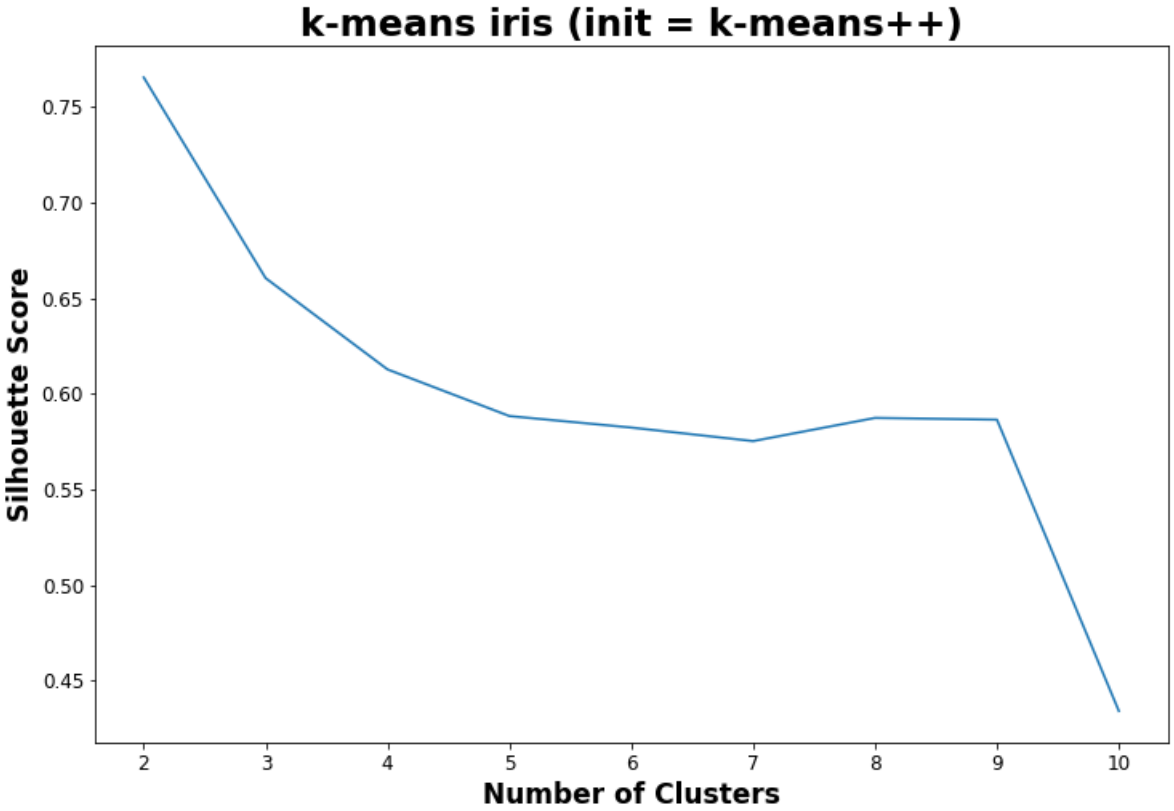
In the table above, for each selection of the number of clusters in the range {1,2,...,10}, the results of the measurements of SSE and Silhouette Score are presented as well as the number of iterations it took the k-means algorithm to converge. Below the visualization of the results is implemented so that valuable conclusions can be drawn.



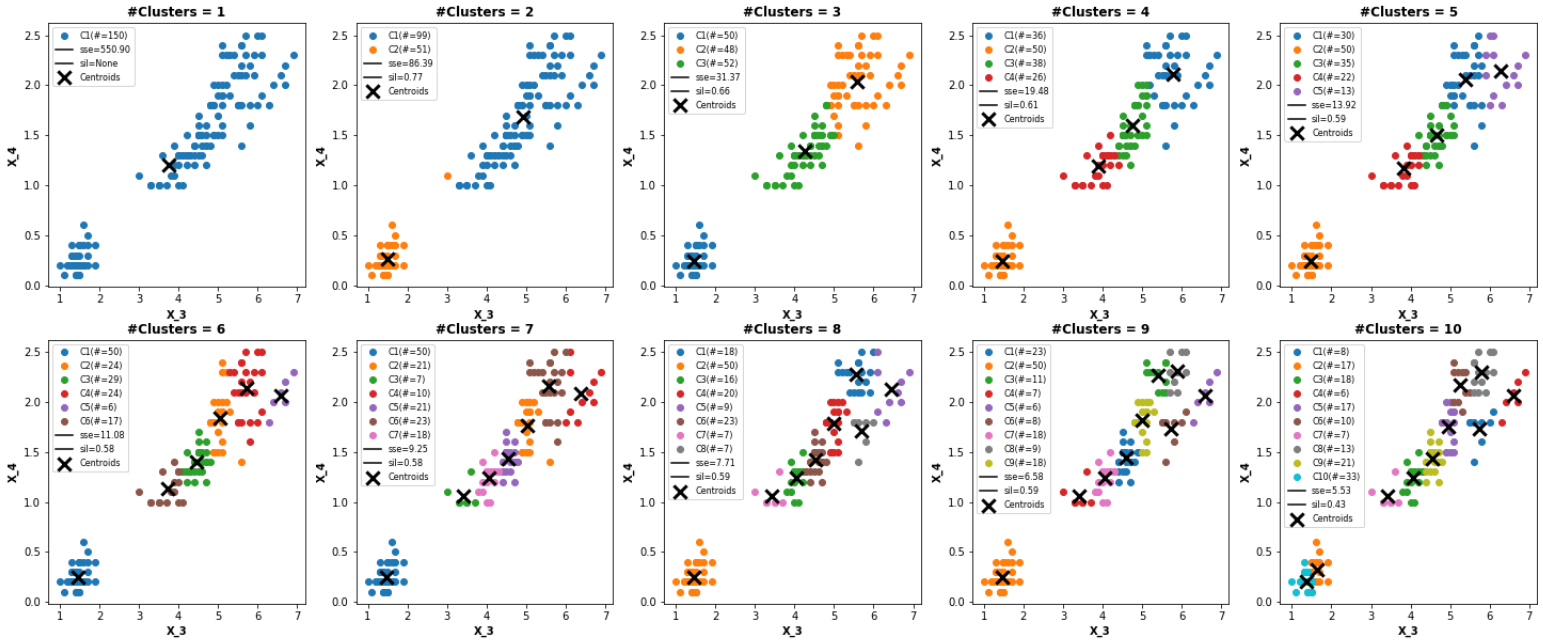
The elbow rule is usually used to find the ideal number of clusters. The elbow rule is based on the idea that the optimal number of clusters is the one that minimizes the sum of the squared errors within the clusters while still providing a satisfactory level of structure to the data. In the specific experiments the biggest difference of SSE is observed between the number of clusters 1 and 2 where the first ""elbow"" of the graph appears and the second big difference is between the number of clusters 2 and 3 where the second "elbow" of the graph appears there. The above observation indicates that the possible ideal number of clusters is 2 or 3.

However, it is important to note that the elbow rule is heuristic and may not always provide the optimal number of clusters, as is the case in the example dataset used.

In some cases, the SSE plot may not show a clear elbow point, or there may be multiple elbow points.



Silhouette Score is a measure of how well defined a cluster is in a data set. It is calculated for each sample in a data set and is based on the distance between the sample and other samples in the same cluster, as well as the distance between the sample and samples in the nearest cluster. Therefore, to calculate the Silhouette Score there needs to be at least 2 clusters, so its calculation was performed for the cluster count range {2,3,...,10}.
The graph above indicates that the best cluster quality is achieved when the cluster count is equal to 2 while the actual cluster count takes secondary value in terms of its cluster quality.



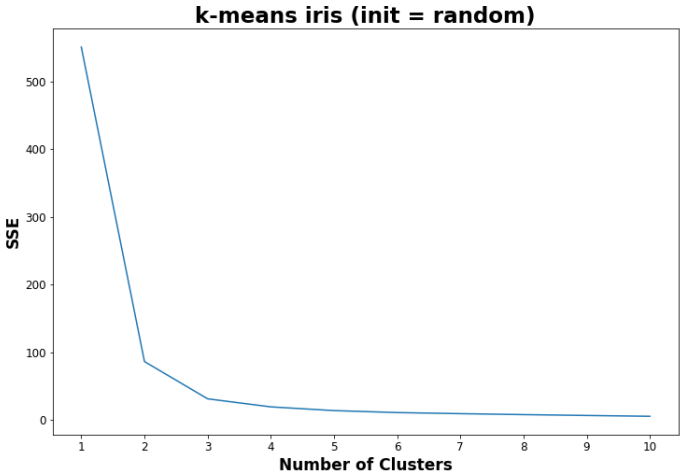
By visualizing the results of k-means, the results of Silhouette Score and SSE are confirmed, because for a number of clusters equal to 2, the best separability between clusters and the best cohesion within clusters. This means that the maximum petiole width and length values for the versicolor species are similar to the minimum petiole width and length values for the virginica species, which makes the separability between the clusters representing moderate these species.

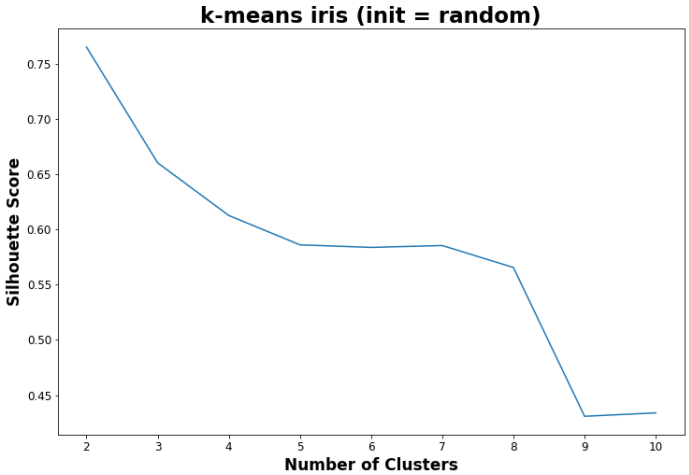
However, it is important to note that the actual number of clusters appeared as a secondary choice of the two given by the analysis of the SSE graph. Also, clustering based on the actual number of clusters resulted in clusters that approximated the true distribution of the data with an error of two data points.

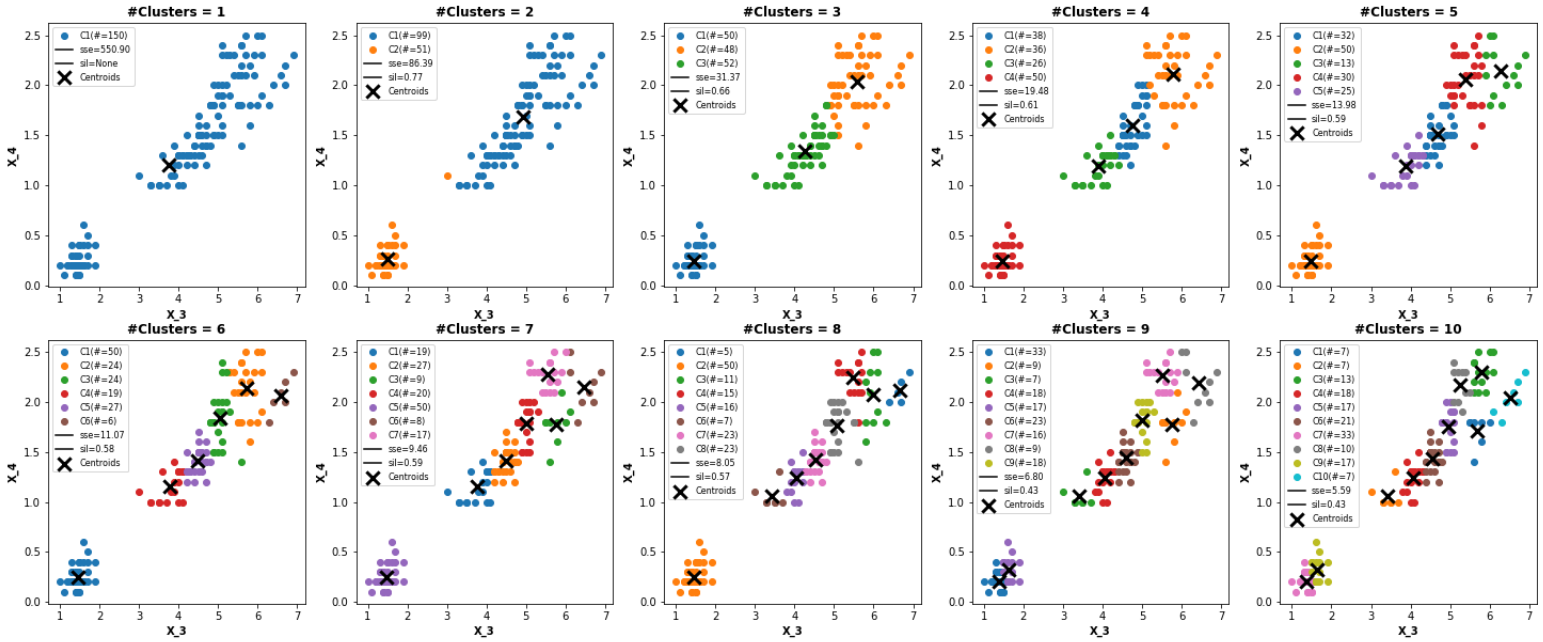
1.1.3 Experiments with $Clusters \in \{2, 3, ..., 10\}$ and init = 'random'

At this point the above procedure is repeated with the only difference in the init parameter of k-means where instead of 'k-means++' it receives the string 'random' .

init = random			
Clusters	SSE	Silhouette Score	Iterations
1	550.90	NaN	2
2	86.39	0.77	2
3	31.37	0.66	3
4	19.48	0.61	11
5	13.98	0.59	4
6	11.03	0.58	9
7	9.48	0.55	13
8	8.29	0.43	10
9	7.05	0.40	8
10	6.48	0.40	5







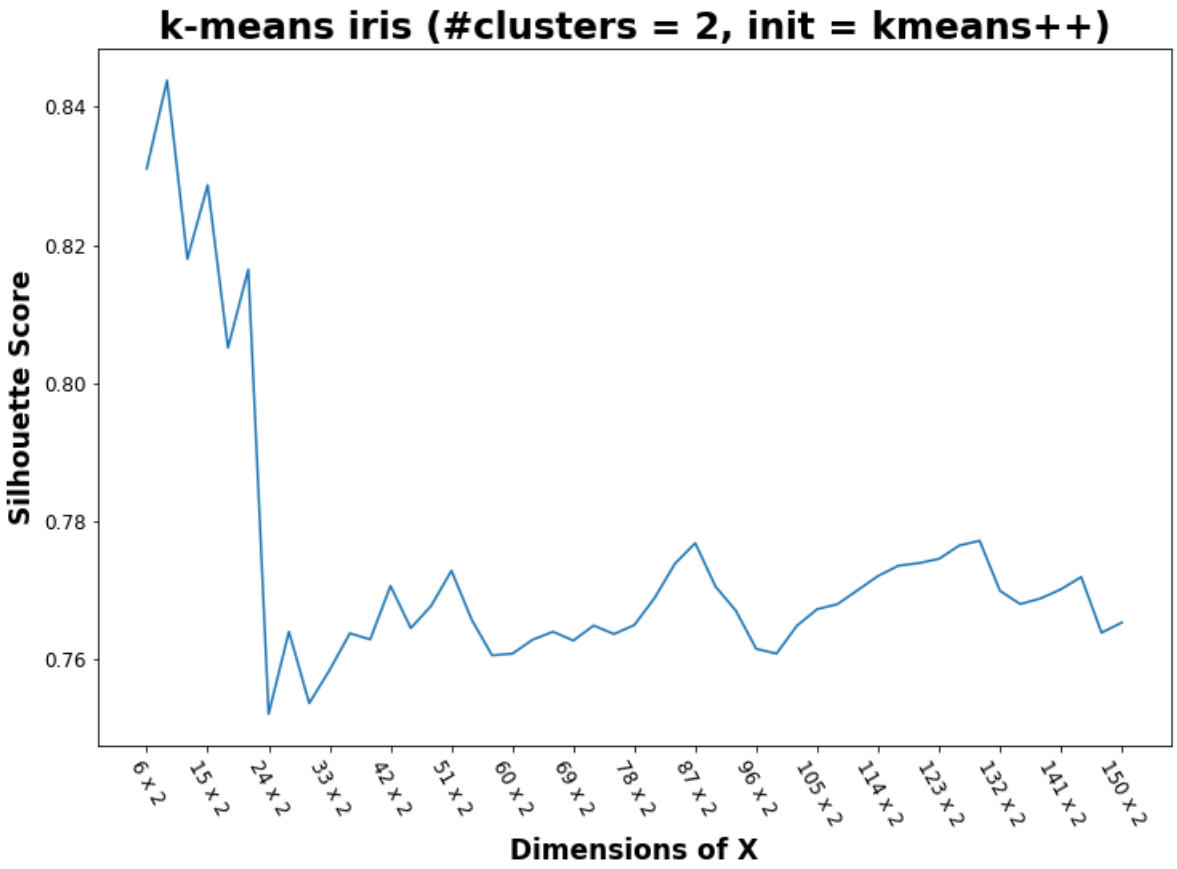
As shown above, changing this parameter had little influence on the results. The k-means algorithm, for the clusterings in which the centers were initialized at suboptimal positions, needed more iterations to converge. Also Silhouette Score showed a sharp decrease from the number of 8 clusters onwards.

1.1.4 Experiments with $Clusters = 2$, $dim(X) \in \{6 \times 2, 9 \times 2, ..., 150 \times 2\}$ and init = 'k-means++'

The table below shows, for each experiment, the results of SSE, Silhouette Score and the iterations the algorithm needed to converge. Experiments were constructed by starting with 2 data for each class and then adding to each class one data for each subsequent experiment.

clusters = 2 init = k-means++			
Dim(X)	SSE	Silhouette Score	Iterations
6 x 2	2.08	0.83	2
9 x 2	2.90	0.84	2
12 x 2	4.65	0.82	2
15 x 2	5.65	0.83	2
18 x 2	8.61	0.81	2
21 x 2	9.25	0.82	2
24 x 2	14.16	0.75	2
27 x 2	15.15	0.76	2
30 x 2	18.38	0.75	2
33 x 2	21.48	0.76	2
36 x 2	22.30	0.76	2
39 x 2	24.18	0.76	2
42 x 2	24.56	0.77	3
45 x 2	27.08	0.76	6
48 x 2	27.98	0.77	2
51 x 2	28.60	0.77	2
54 x 2	33.17	0.77	5
57 x 2	37.62	0.76	2
60 x 2	39.16	0.76	2
63 x 2	40.18	0.76	2
66 x 2	41.38	0.76	2
69 x 2	44.72	0.76	2
72 x 2	45.20	0.76	2
75 x 2	46.70	0.76	2
78 x 2	48.20	0.77	2
81 x 2	48.40	0.77	2
84 x 2	48.42	0.77	2
87 x 2	49.25	0.78	2
90 x 2	52.60	0.77	4
93 x 2	55.65	0.77	2
96 x 2	59.89	0.76	6
99 x 2	61.95	0.76	2
102 x 2	62.03	0.76	4
105 x 2	62.74	0.77	2
108 x 2	64.72	0.77	4
111 x 2	65.77	0.77	2
114 x 2	66.56	0.77	2
117 x 2	67.54	0.77	4
120 x 2	68.98	0.77	2
123 x 2	70.49	0.77	5
126 x 2	71.13	0.78	2
129 x 2	72.37	0.78	4
132 x 2	76.98	0.77	2
135 x 2	79.15	0.77	2
138 x 2	80.39	0.77	2
141 x 2	81.14	0.77	2
144 x 2	81.87	0.77	4
147 x 2	85.50	0.76	3
150 x 2	86.39	0.77	4

It stands to reason that as points are added to the categories the overall SSE will also increase because more distances will be added. However, the study of the Silhouette Score is more interesting as shown below.

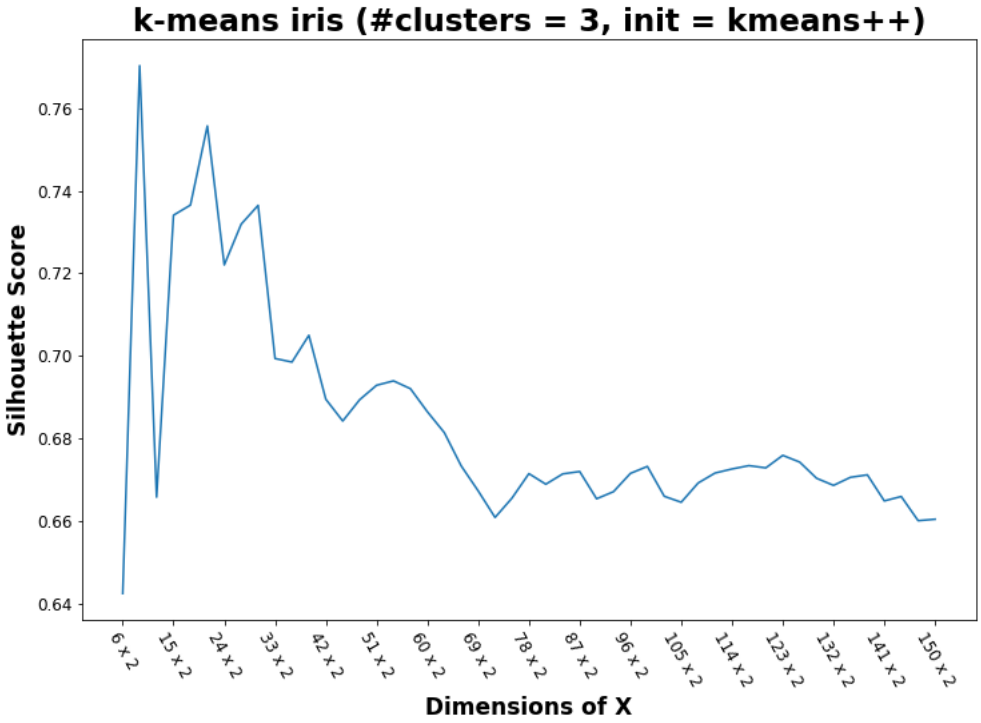


Visualizing the data of Silhouette Score it is observed that the graph marks a constant average after the dimensions $dim(X) = 42 \times 2$. This means that it is sufficient that each cluster consists of, at least, 14 data ($\frac{42}{3}$) so that a good approximation of the "real" separability between the clusters and the "real" cohesion within the clusters.

1.1.5 Experiments with $Clusters = 3$, $dim(X) \in \{6 \times 2, 9 \times 2, ..., 150 \times 2\}$ and init = 'k-means++'

The same procedure is repeated with the difference that instead of 2 clusters, 3 are selected.

clusters = 3 init = k-means++			
Dim(X)	SSE	Silhouette Score	Iterations
6 x 2	0.33	0.64	2
9 x 2	0.44	0.77	2
12 x 2	1.27	0.67	4
15 x 2	1.28	0.73	2
18 x 2	1.93	0.74	2
21 x 2	2.00	0.76	2
24 x 2	3.98	0.72	2
27 x 2	4.14	0.73	2
30 x 2	4.67	0.74	3
33 x 2	6.22	0.70	2
36 x 2	6.54	0.70	5
39 x 2	6.86	0.71	3
42 x 2	7.69	0.69	5
45 x 2	8.66	0.68	5
48 x 2	8.89	0.69	4
51 x 2	9.10	0.69	4
54 x 2	10.37	0.69	4
57 x 2	11.95	0.69	4
60 x 2	12.70	0.69	3
63 x 2	13.24	0.68	3
66 x 2	14.08	0.67	5
69 x 2	15.48	0.67	5
72 x 2	16.20	0.66	6
75 x 2	16.43	0.67	5
78 x 2	16.58	0.67	5
81 x 2	17.10	0.67	5
84 x 2	17.80	0.67	3
87 x 2	17.88	0.67	2
90 x 2	19.22	0.67	5
93 x 2	19.83	0.67	5
96 x 2	20.84	0.67	5
99 x 2	21.20	0.67	3
102 x 2	22.26	0.67	5
105 x 2	22.76	0.66	5
108 x 2	23.00	0.67	5
111 x 2	23.29	0.67	5
114 x 2	23.52	0.67	4
117 x 2	23.94	0.67	5
120 x 2	24.30	0.67	4
123 x 2	24.53	0.68	3
126 x 2	25.11	0.67	4
129 x 2	25.80	0.67	7
132 x 2	27.39	0.67	4
135 x 2	27.83	0.67	5
138 x 2	28.24	0.67	5
141 x 2	28.87	0.67	5
144 x 2	29.16	0.67	4
147 x 2	31.03	0.66	5
150 x 2	31.37	0.66	4

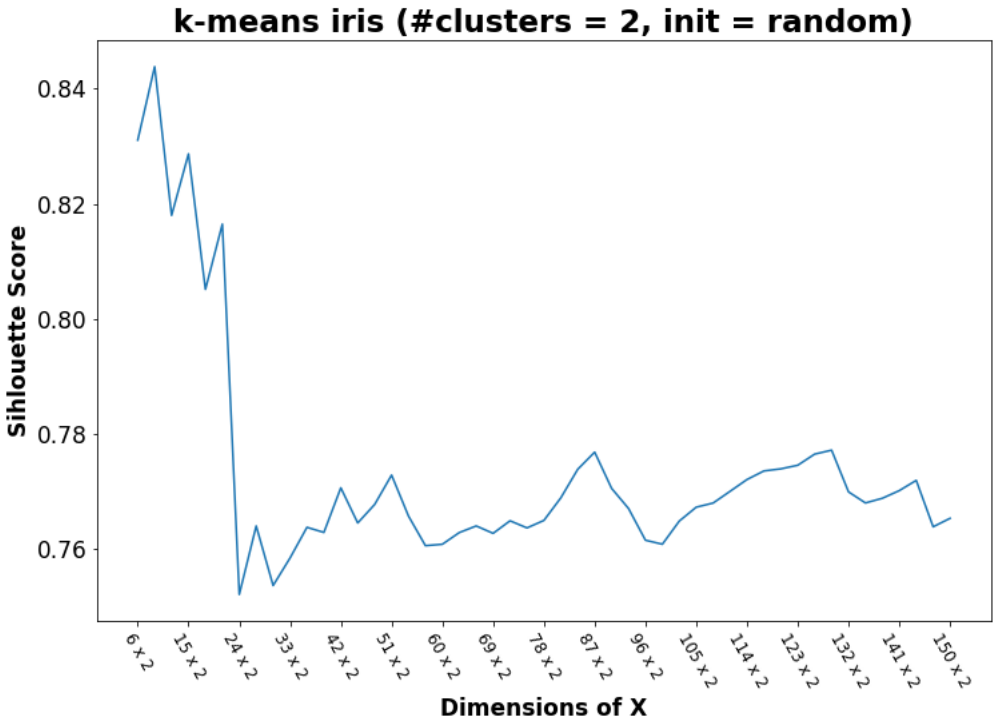


Here it is observed that the threshold, so as to achieve an approximation of the """real" parameters of the silhouette of the clusters, has been increased. Thus, to approximate the "real" parameters of the silhouette of the clusters, it is sufficient that each cluster consists of 26 data ($\frac{78}{3}$).

1.1.6 Experiments with $Clusters = 2$, $dim(X) \in \{6 \times 2, 9 \times 2, ..., 150 \times 2\}$ and init = 'random'

The same experiments are repeated, with the difference that the string "random" is entered in the value of the init parameter instead of "k-means++".

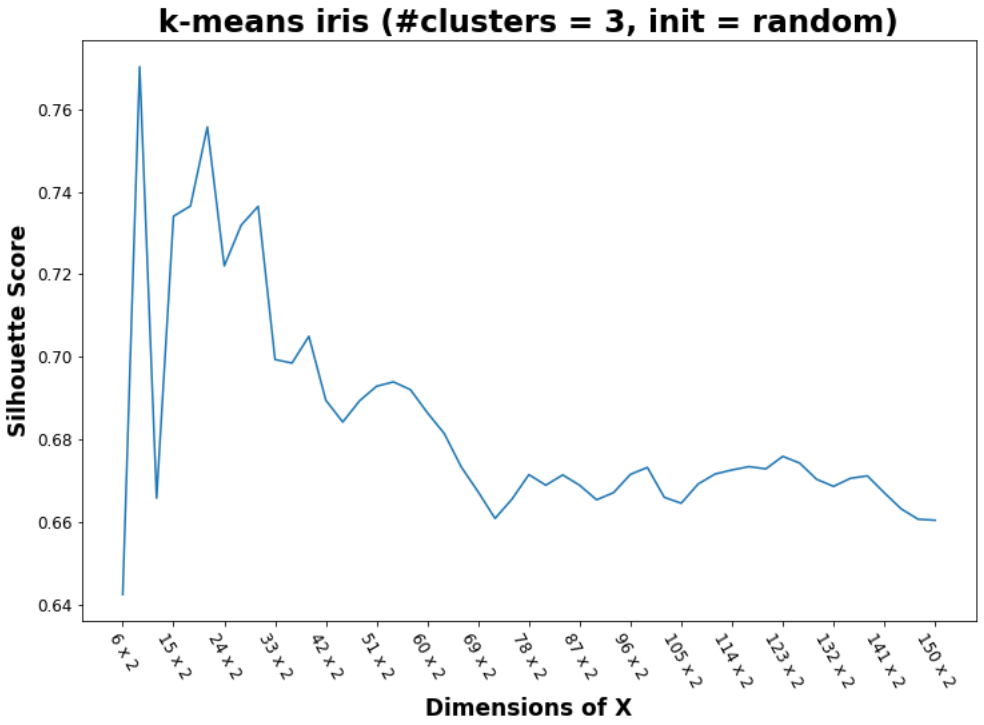
clusters = 2 init = random			
Dim(X)	SSE	Silhouette Score	Iterations
6 x 2	2.08	0.83	3
9 x 2	2.90	0.84	3
12 x 2	4.65	0.82	4
15 x 2	5.65	0.83	5
18 x 2	8.61	0.81	2
21 x 2	9.25	0.82	2
24 x 2	14.16	0.75	2
27 x 2	15.15	0.76	4
30 x 2	18.38	0.75	5
33 x 2	21.48	0.76	2
36 x 2	22.30	0.76	2
39 x 2	24.18	0.76	2
42 x 2	24.56	0.77	2
45 x 2	27.08	0.76	6
48 x 2	27.98	0.77	6
51 x 2	28.60	0.77	6
54 x 2	33.17	0.77	3
57 x 2	37.62	0.76	5
60 x 2	39.16	0.76	4
63 x 2	40.18	0.76	2
66 x 2	41.38	0.76	7
69 x 2	44.72	0.76	4
72 x 2	45.20	0.76	2
75 x 2	46.70	0.76	7
78 x 2	48.20	0.77	2
81 x 2	48.40	0.77	7
84 x 2	48.42	0.77	7
87 x 2	49.25	0.78	8
90 x 2	52.60	0.77	4
93 x 2	55.65	0.77	6
96 x 2	59.89	0.76	3
99 x 2	61.95	0.76	7
102 x 2	62.03	0.76	2
105 x 2	62.74	0.77	7
108 x 2	64.72	0.77	3
111 x 2	65.77	0.77	7
114 x 2	66.56	0.77	7
117 x 2	67.54	0.77	6
120 x 2	68.98	0.77	5
123 x 2	70.49	0.77	7
126 x 2	71.13	0.78	6
129 x 2	72.37	0.78	6
132 x 2	76.98	0.77	2
135 x 2	79.15	0.77	2
138 x 2	80.39	0.77	2
141 x 2	81.14	0.77	6
144 x 2	81.87	0.77	3
147 x 2	85.50	0.76	6
150 x 2	86.39	0.77	5



It is observed that changing the string in the variable init did not affect the results of SSE and Silhouette Score but increased the average of the algorithm iterations so that it converged.

1.1.7 Experiments with $Clusters = 3$, $dim(X) \in \{6 \times 2, 9 \times 2, ..., 150 \times 2\}$ and init = 'random'

clusters = 3 init = random			
Dim(X)	SSE	Silhouette Score	Iterations
6 x 2	0.33	0.64	2
9 x 2	0.44	0.77	3
12 x 2	1.27	0.67	3
15 x 2	1.28	0.73	4
18 x 2	1.93	0.74	4
21 x 2	2.00	0.76	2
24 x 2	3.98	0.72	4
27 x 2	4.14	0.73	4
30 x 2	4.67	0.74	2
33 x 2	6.22	0.70	6
36 x 2	6.54	0.70	5
39 x 2	6.86	0.71	6
42 x 2	7.69	0.69	5
45 x 2	8.66	0.68	3
48 x 2	8.89	0.69	4
51 x 2	9.10	0.69	4
54 x 2	10.37	0.69	2
57 x 2	11.95	0.69	5
60 x 2	12.70	0.69	7
63 x 2	13.24	0.68	5
66 x 2	14.08	0.67	4
69 x 2	15.48	0.67	10
72 x 2	16.20	0.66	4
75 x 2	16.43	0.67	9
78 x 2	16.58	0.67	13
81 x 2	17.10	0.67	6
84 x 2	17.80	0.67	3
87 x 2	17.88	0.67	2
90 x 2	19.22	0.67	8
93 x 2	19.83	0.67	13
96 x 2	20.84	0.67	8
99 x 2	21.20	0.67	12
102 x 2	22.26	0.67	7
105 x 2	22.76	0.66	8
108 x 2	23.00	0.67	5
111 x 2	23.29	0.67	7
114 x 2	23.52	0.67	7
117 x 2	23.94	0.67	5
120 x 2	24.30	0.67	11
123 x 2	24.53	0.68	6
126 x 2	25.11	0.67	7
129 x 2	25.80	0.67	7
132 x 2	27.39	0.67	5
135 x 2	27.83	0.67	5
138 x 2	28.24	0.67	6
141 x 2	28.87	0.67	7
144 x 2	29.16	0.67	7
147 x 2	31.03	0.66	8
150 x 2	31.37	0.66	5



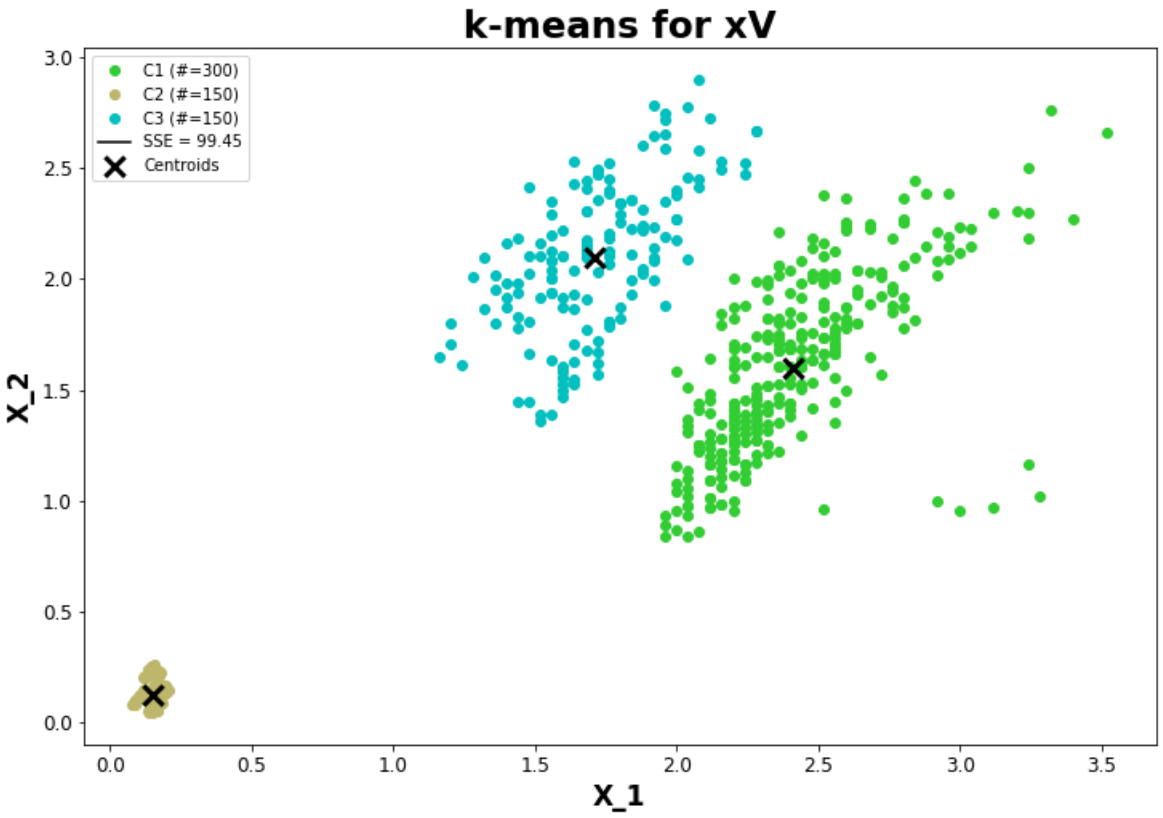
The observations mentioned in experiments 1.1.6 apply here as well.

1.2 Application to dataset xV

1.2.1 Introduction

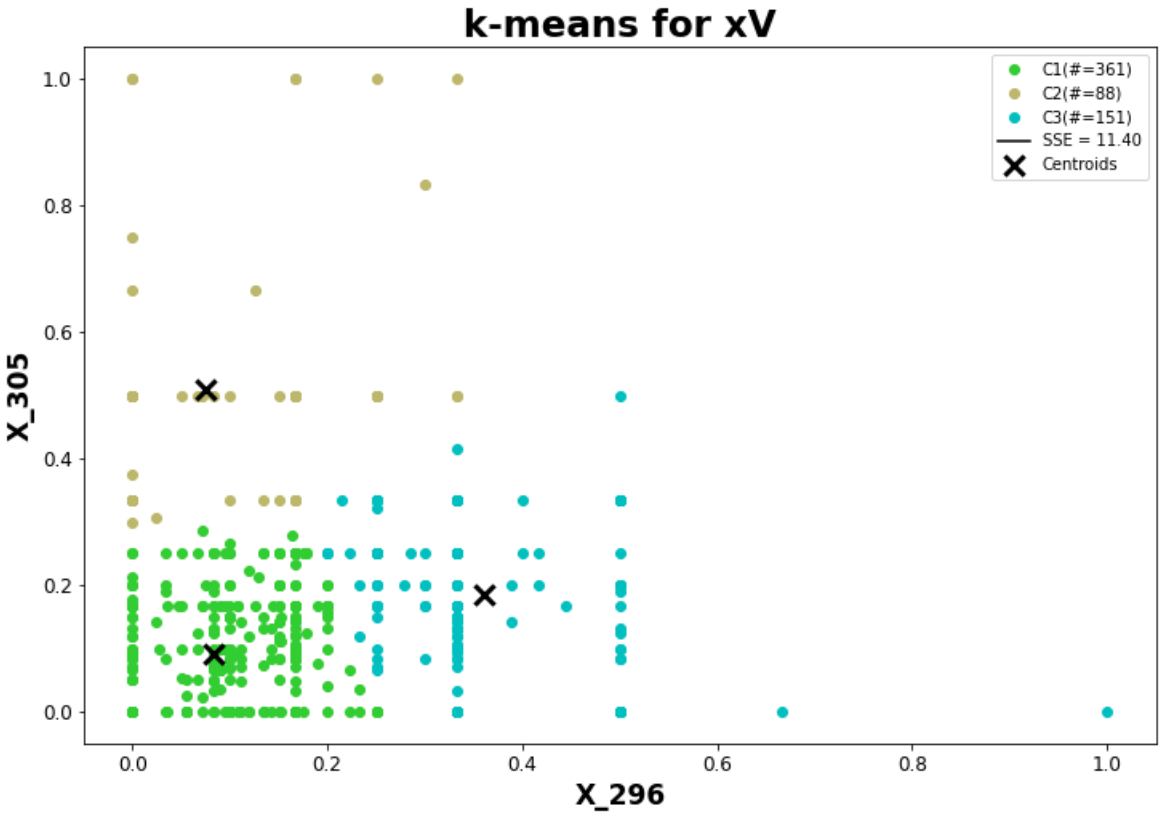
The xV dataset consists of 469 features and 600 samples. In the implementation of k-means only features 1, 2, 175, 205, 296, 305, 468, 469 will be used. For each pair to which k-means is applied, 3 clusters will be used and SSE will be calculated.

1.2.2 k-means with $Clusters = 3$ and cols = [1, 2]



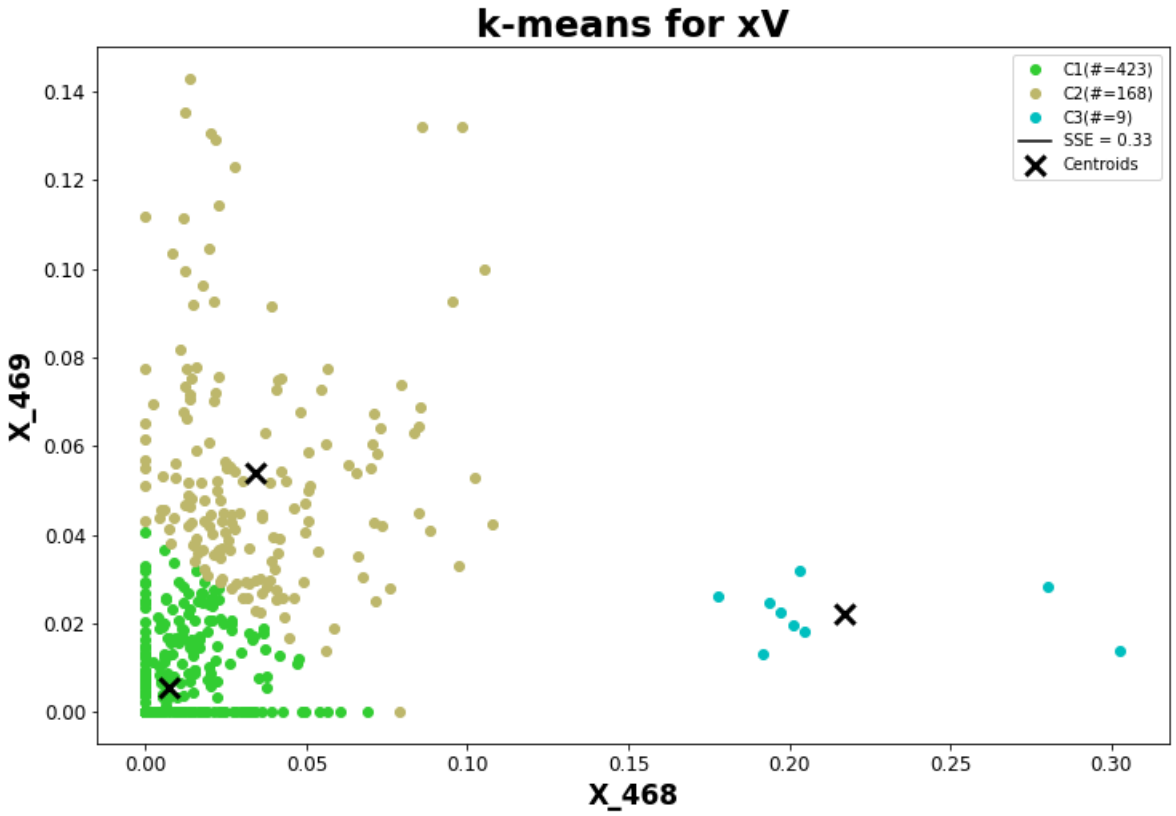
In the above clustering, SSE = 99.45 was calculated and visually a good separation of the second class from the other two is observed, but also a relatively satisfactory separation between classes 1 and 3.

1.2.3 k-means with $Clusters = 3$ and cols = [296, 305]



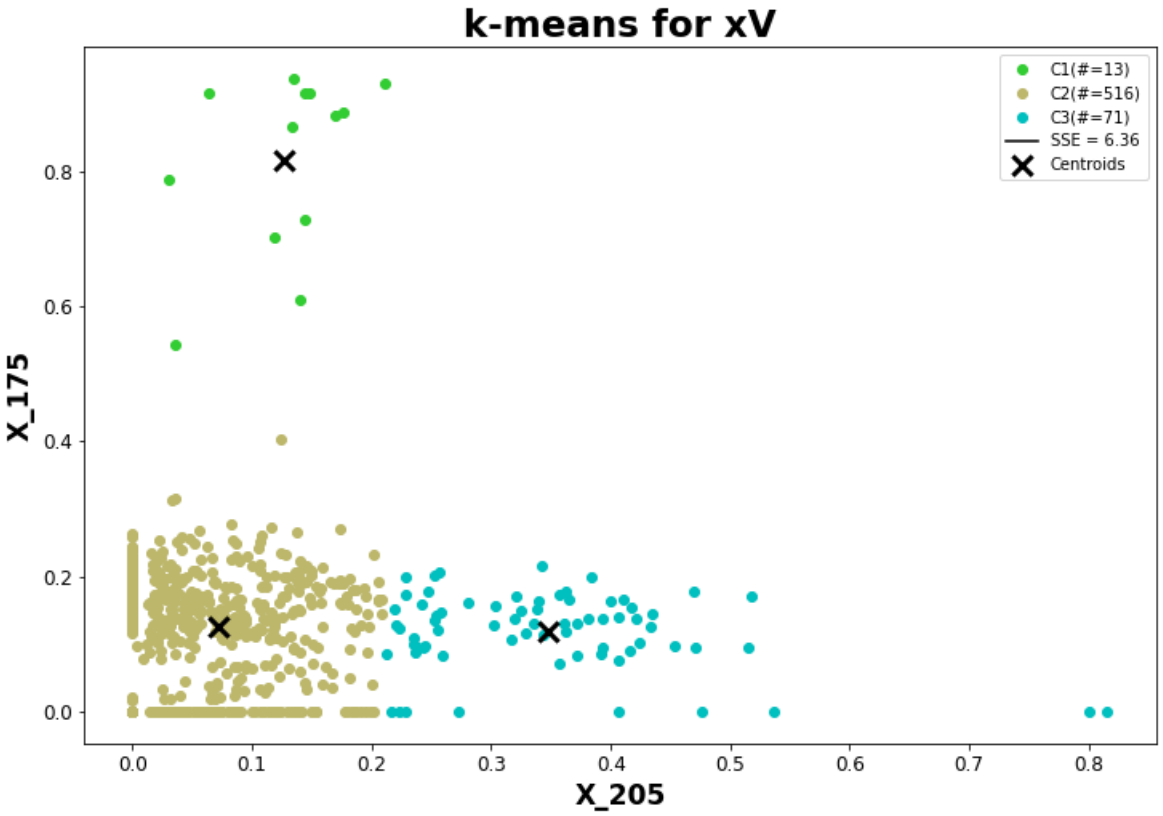
In this clustering, SSE = 11.40 was calculated and visually a moderate separation is observed between classes 1 and 2, while class 3 shows a little better separation.

1.2.4 k-means with $Clusters = 3$ and cols = [468, 469]



In this clustering SSE = 0.33 was calculated and it is observed that in classes 1 and 2 there is confusion while in class 3 there is a good separation from the other two. However, it is worth noting that class 3 consists of 9 data, which may mean that these data are outliers (outliers) and that the actual number of clusters is 2 or 1.

1.2.5 k-means with $Clusters = 3$ and cols = [205, 175]



In this clustering SSE = 8.35 was calculated and it is visually observed that there is confusion between clusters 1 and 2.

1.2.6 Comparison of results for k-means with $Clusters = 3$ and cols = [1, 2], cols = [468, 469], cols = [205, 175]

Initially, in order to compare the clusters, the x and y axes, for each cluster, should belong to the same value range so that a comparison of the SSE calculations can be made. A normalization that equates the x and y axes is not appropriate because it will affect the results of the k-means algorithm which implies different results of the SSE and Silhouette Score.
The aim is to implement a normalization which will preserve the ratio of the distances between the data.
The procedure implemented to achieve this goal is explained below.

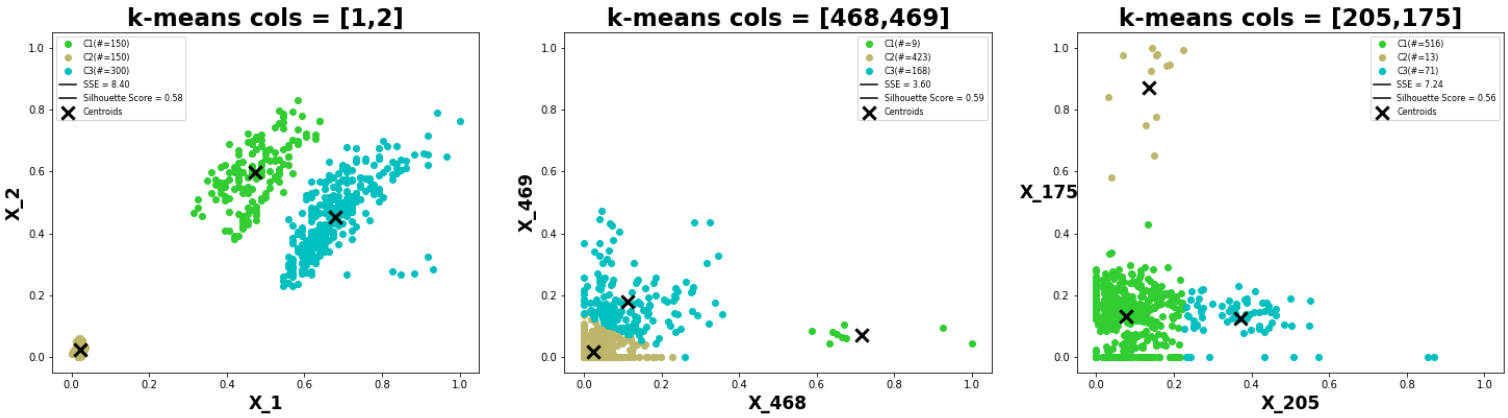
Initially, $[0, 1]$ is chosen as the set to which the values of x and y axes will belong. Then a variant of the generalized min-max normalization will be applied ($x_i \coloneqq a + \frac{(x_i - x_{min})(b-a)}{x_{max} - x_{min}}$) for the x and y axis separately so that the data distances are preserved.

1. The maximum distance between abscissas $dist_x \coloneqq x_{max} - x_{min}$ and the maximum distance between ordinates $dist_y \coloneqq y_{max} - y_{min}$ are calculated.

2. Since the set chosen to belong to the values of the axes is $[0, 1]$, the ratio of the distances will be chosen so that it is less than unity. So $r \coloneqq min(\{\frac{dist_x}{dist_y}, \frac{dist_y}{dist_x}\})$
   Note: The data do not show negative values on the axes.

3. Using the variant of generalized min-max normalization, the axis on which the data exhibits the smallest maximum distance will be mapped to the set $[0, r]$ while the other axis will be mapped to the set $[0, 1]$.
   That is, if the axis on which the data shows the smallest maximum distance is x, then the normalization will be used :
   $(x_i, y_i) = (\frac{(x_i - x_{min}) \cdot r}{x_{max} - x_{min}}, \frac{y_i - y_{max}}{y_{max} - y_{min}})$

Below are the results of k-means clustering after applying the above procedure.



Observing the graphs, the intuitive conclusion is that the first clustering is the best because a good separation and cohesion of the clusters is noted, also due to these features in visualizing the data before applying k-means, it is logical to choose three clusters.

The second clustering seems, intuitively, not to have a good separation because the data of class 3 and the data of class 1 overlap, but there seems to be a satisfactory coherence between the clusters. In contrast to the first clustering, in the visualization of the data before applying k-means it seems to make more sense to use two clusters instead of three. Furthermore it could be assumed that class 2 is a result of noise and thus the entire data set would consist of one cluster.

The third clustering marks an overlap of the data between classes 1 and 2, however, as in the second clustering, the coherence of the clusters is satisfactory, but with the difference that here there are excellent points at greater distances than in the second clustering , which is expected to negatively affect cluster cohesion.

Contrary to the intuitive conclusions above, using the measurements of SSE and Silhouette Score, it is concluded that the second clustering is the best, this is concluded because the smallest value of SSE ($= 3.60$) and at the same time the largest value of Silhouette Score ($= 0.59$). The explanation for this phenomenon is that the data is quite dense in the $[0, 0.4] \times [0, 0.5]$ window of the plane, and so the distances overall sum to a smaller number than in the other clusterings, also because of this of gathering the data, the value of the cohesion factor increases which in combination with the separation of the second class causes an increase in the value of Silhouette Score.

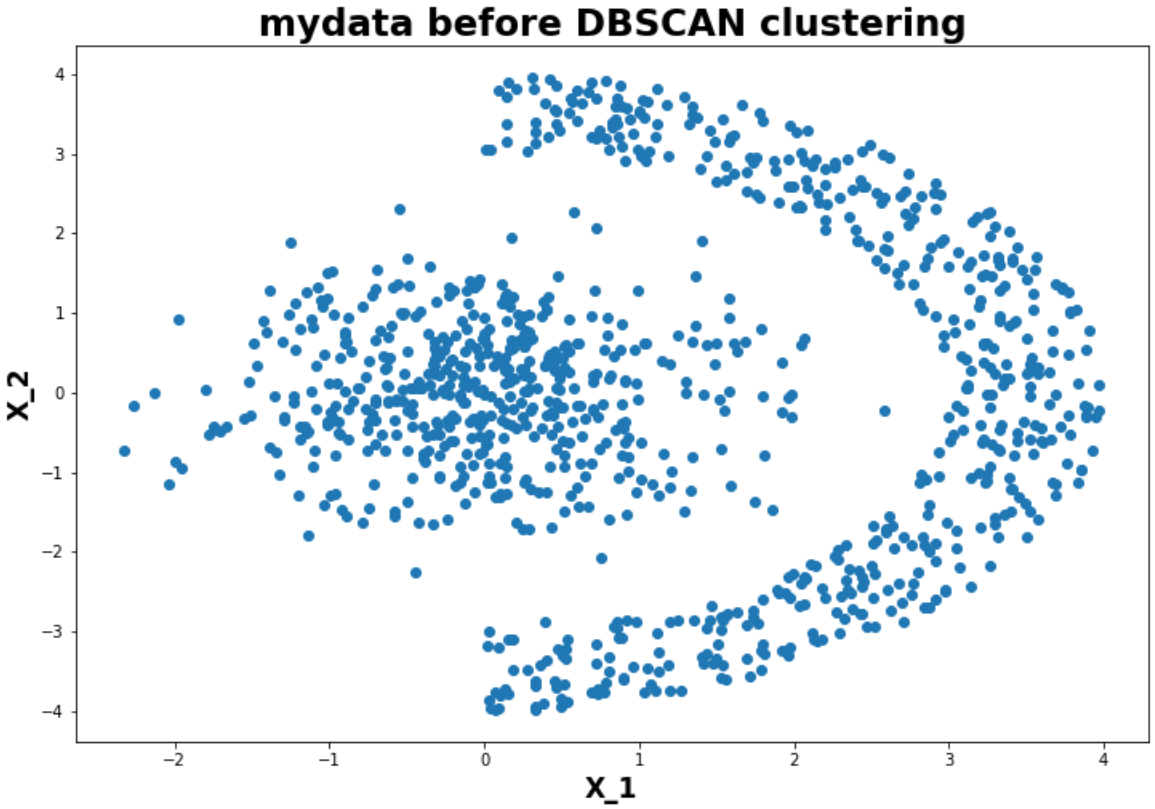
Comparing the third clustering with the first, it is observed that the third clustering has a better result in calculating SSE than the first ($3rd = 7.24 < 8.40 =$1st) but the first clustering has a better result in calculating Silhouette Score from third ($3rd = 0.56 < 0.58 =$1st). This is because unlike the data of the first clustering, which covers much of the space $[0, 1] \times [0, 1]$, the data of

the third clustering mainly covers the window $[0, 0.6]$ $times[0, 0.4]$ which implies that the data of the third clustering have, on average, a smaller distance between them and as before this results in a reduction of the total sum and this implies a lower value of SSE. However, because there are more outstanding data in the third cluster than in the second, and because the window occupied by the data of the third cluster is larger than that of the second, there is no significant increase in Silhouette Score so that exceed the Silhouette Score of the first clustering.
As for the first clustering, because the data occupy a large subset of the layer, the distances of the data between them are on average larger compared to the other clusterings, and thus the first clustering has the largest SSE value. The coherence of the second class is quite good and combined with the satisfactory coherence of the other classes, the Silhouette Score of the first clustering has a moderate result.

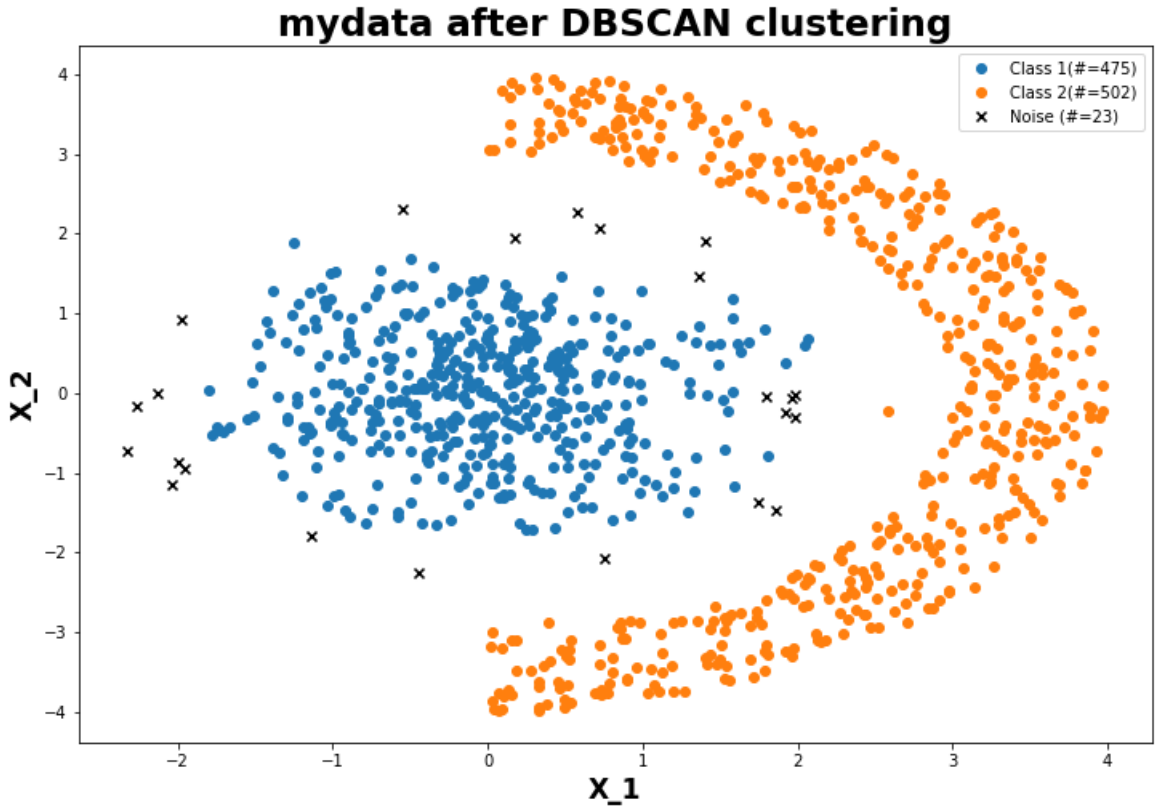
2 Density-based clustering with DBSCAN

2.1 Apply to dataset mydata

2.1.1 DBSCAN(=0.5, MinPts=15)



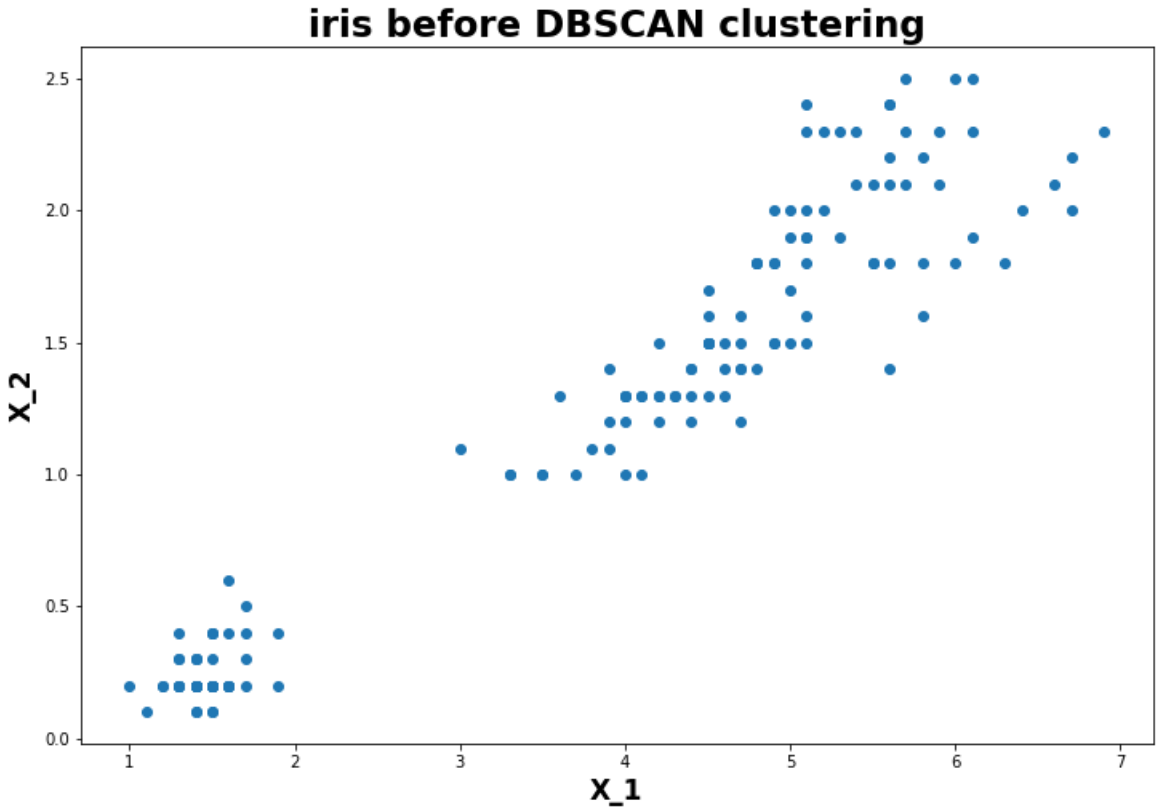
Clustering will be performed using the DBSCAN algorithm so as to identify the data of the circle described in the center and the "'crescent"' located to its right.



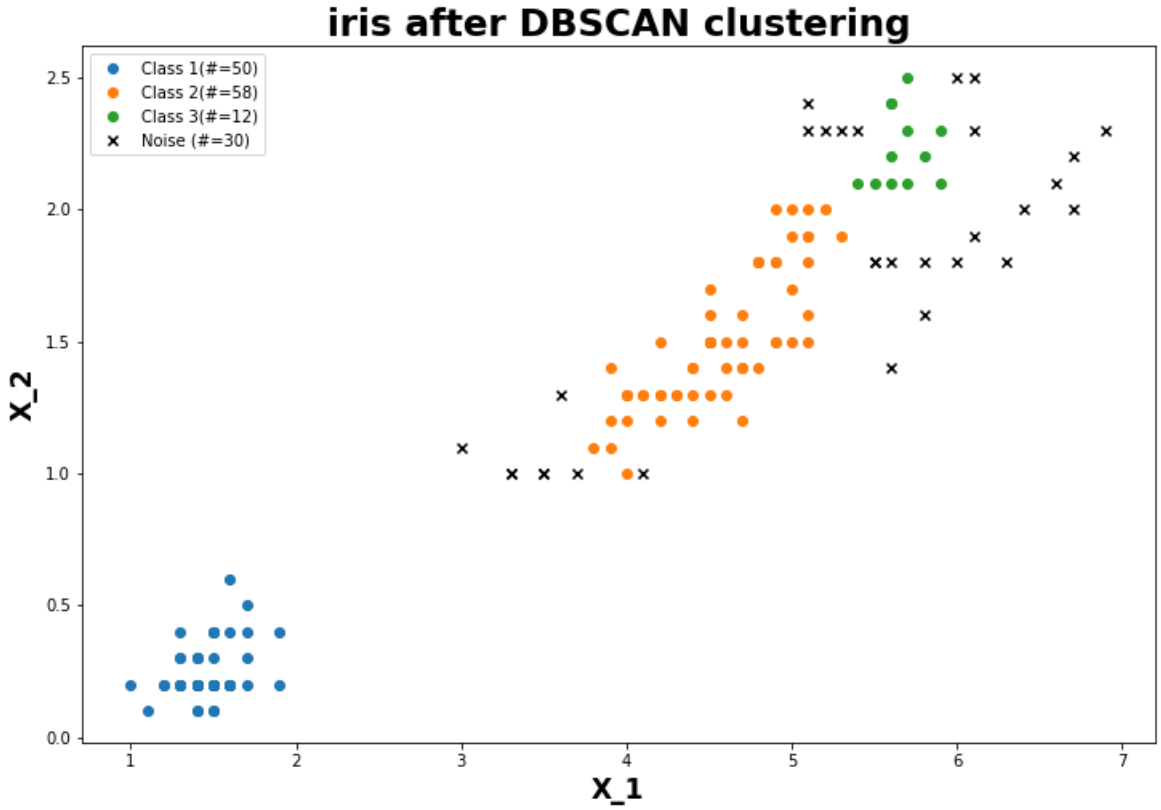
After the application of the DBSCAN algorithm with a radius of 0.5 and a number of points (density) of 15, a successful separation of the two shapes is observed, where the noise constitutes 3.83% of the data and is noted only on the outline of the circle.

2.2 Application to the iris dataset

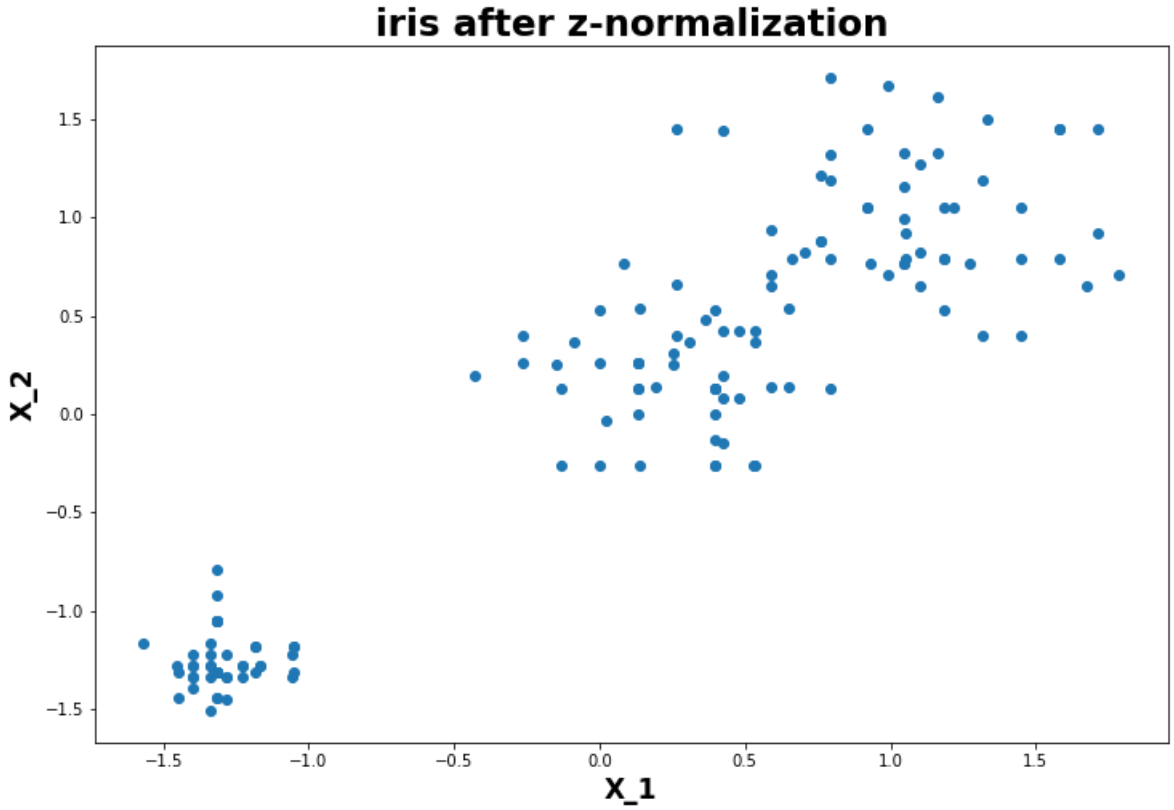
2.2.1 DBSCAN(=0.2, MinPts=5), cols=[3, 4]



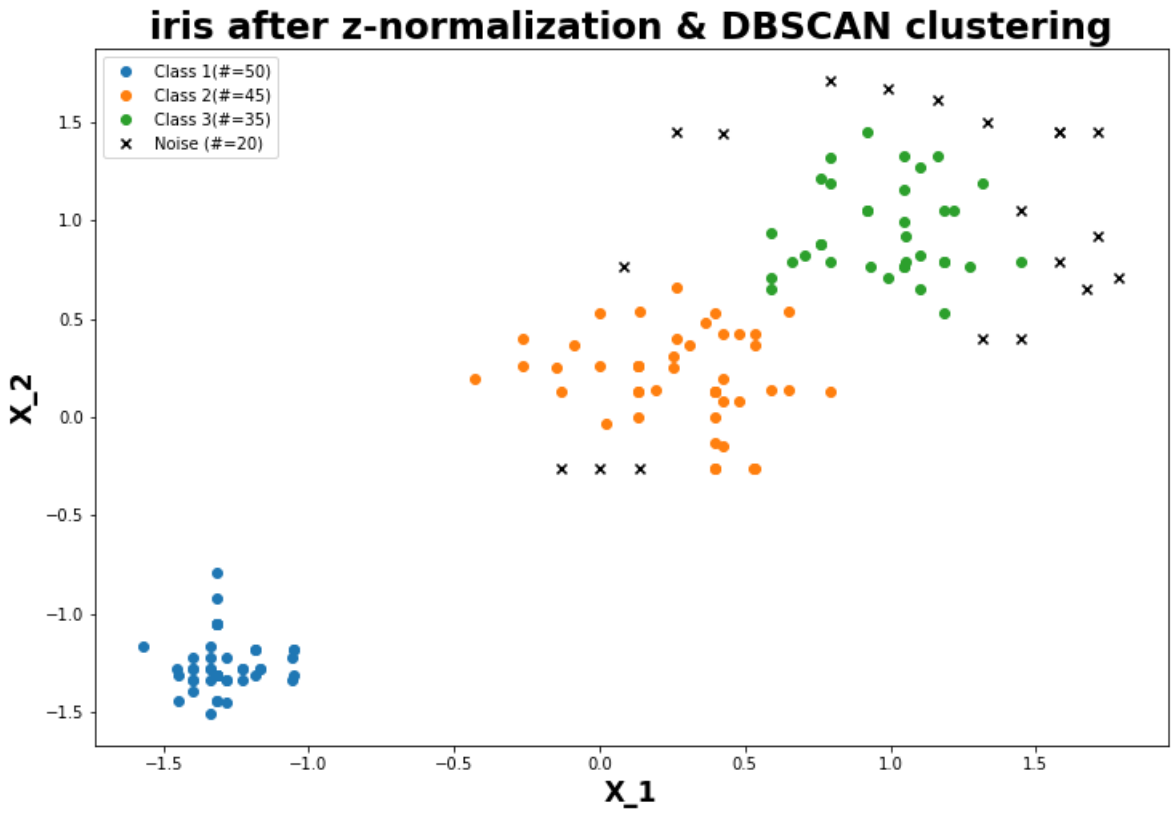
This figure shows the data of the iris dataset without the flower classes being labeled. For the clustering of the three classes, the DBSCAN algorithm will be used with a radius of 0.2 and a density of 5.



After performing the clustering, the DBSCAN algorithm completely separated the first class without introducing noise. In the second and third clusters, there is confusion because we know that actually the eight highest data in the second cluster belong to the third class. Also shown is noise which covers a total of 20% of the data, where 4% appears in the second cluster and the remaining 16% appears in the third cluster.



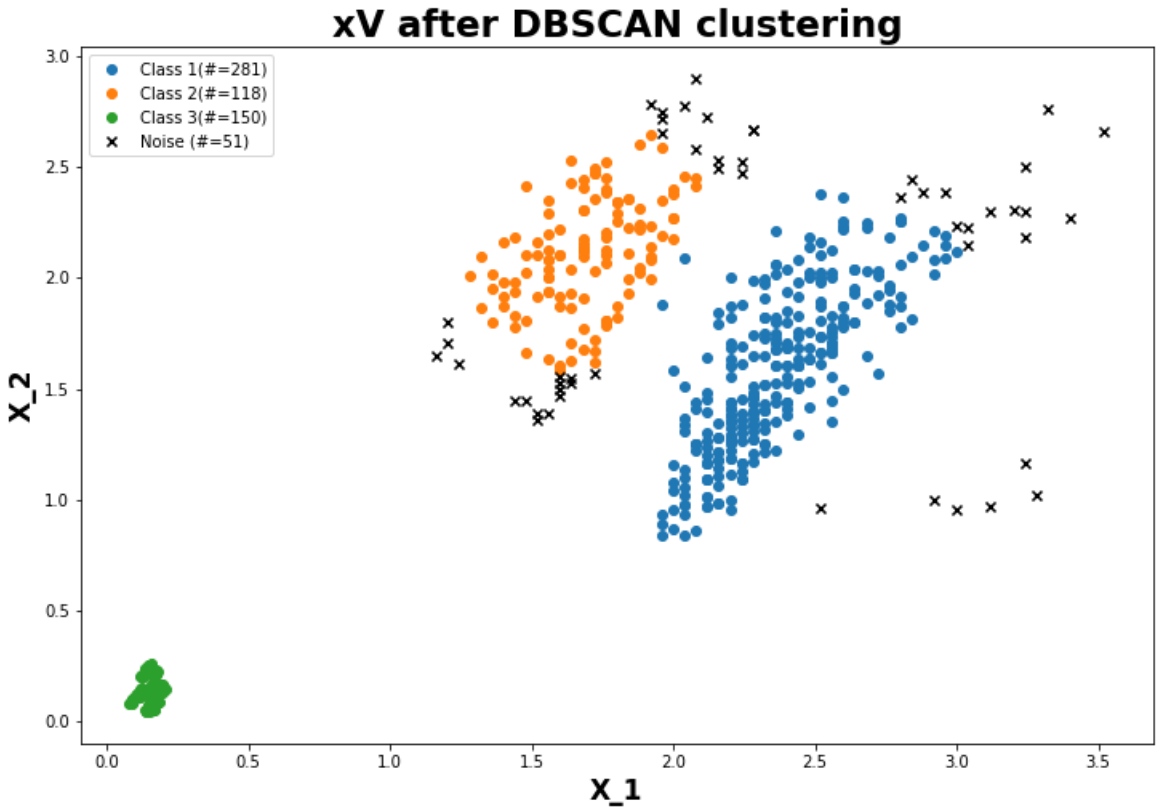
The graph above shows the data of the iris set after applying z-normalization and before applying the DBSCAN algorithm



Applying normalization and then clustering with DBSCAN , using the same values for the density and radius parameters, a noise reduction of 33.3% is observed compared to the previous clustering and at the same time a better clustering is noted in classes 2 and 3 . Specifically, noise covers 13.3% of the data, where 3.3% appears in the second cluster and the remaining 10% appears in the third cluster. Also class 2 now contains only one element from class 3.

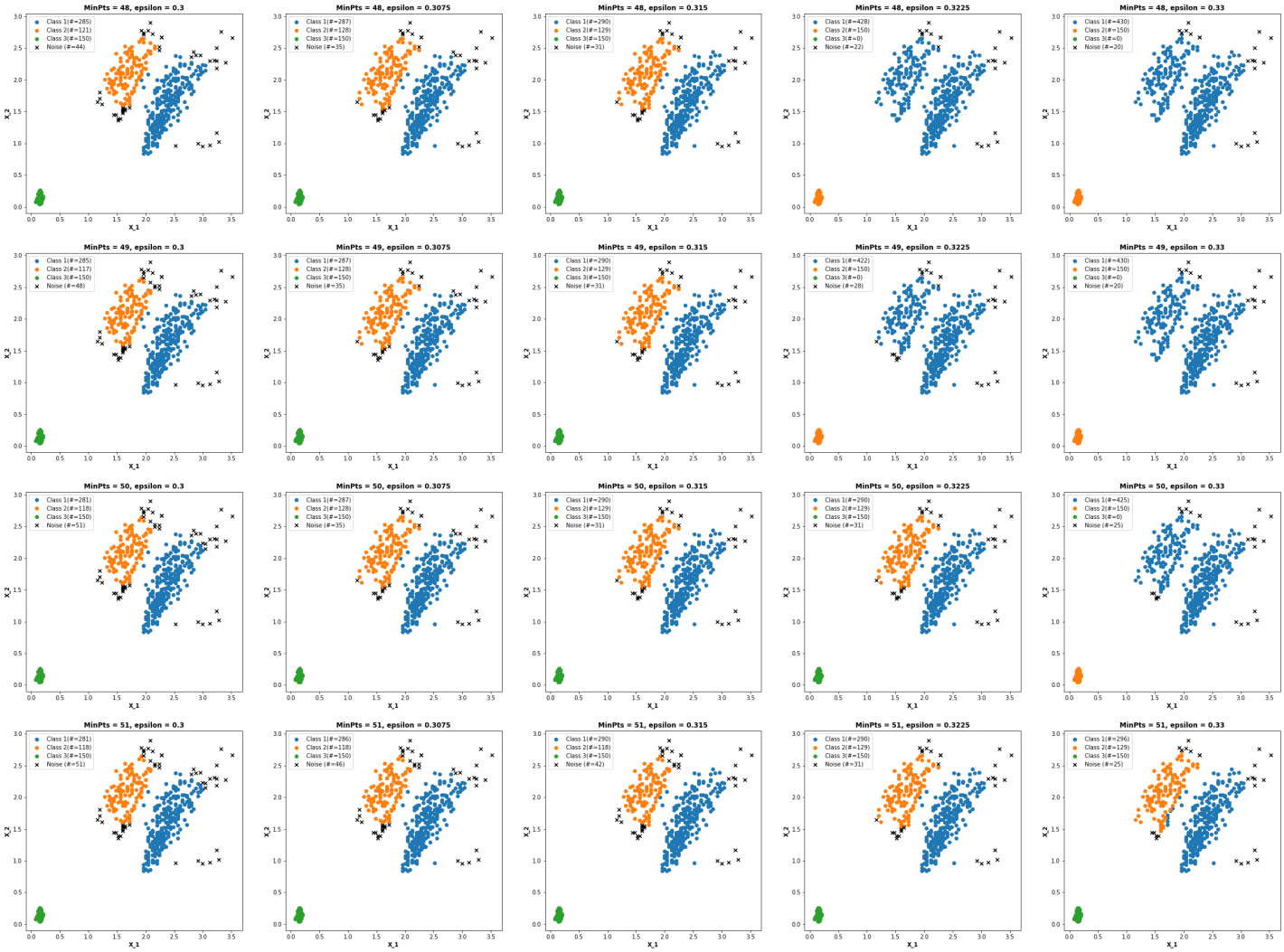
2.3 Application to dataset xV

2.3.1 DBSCAN(=0.3, MinPts=50), cols=[1, 2]



After applying the DBSCAN algorithm with a radius of 0.3 and a density of 50, an ideal clustering of the third class is observed, and a fairly satisfactory clustering of classes 1 and 2. Specifically the noise covers 8.5% of the data where 5% appears in class 2 and 3.5% appears in class 1. In addition it is observed that two data have been included in the second cluster, which normally should have belonged to the first cluster.

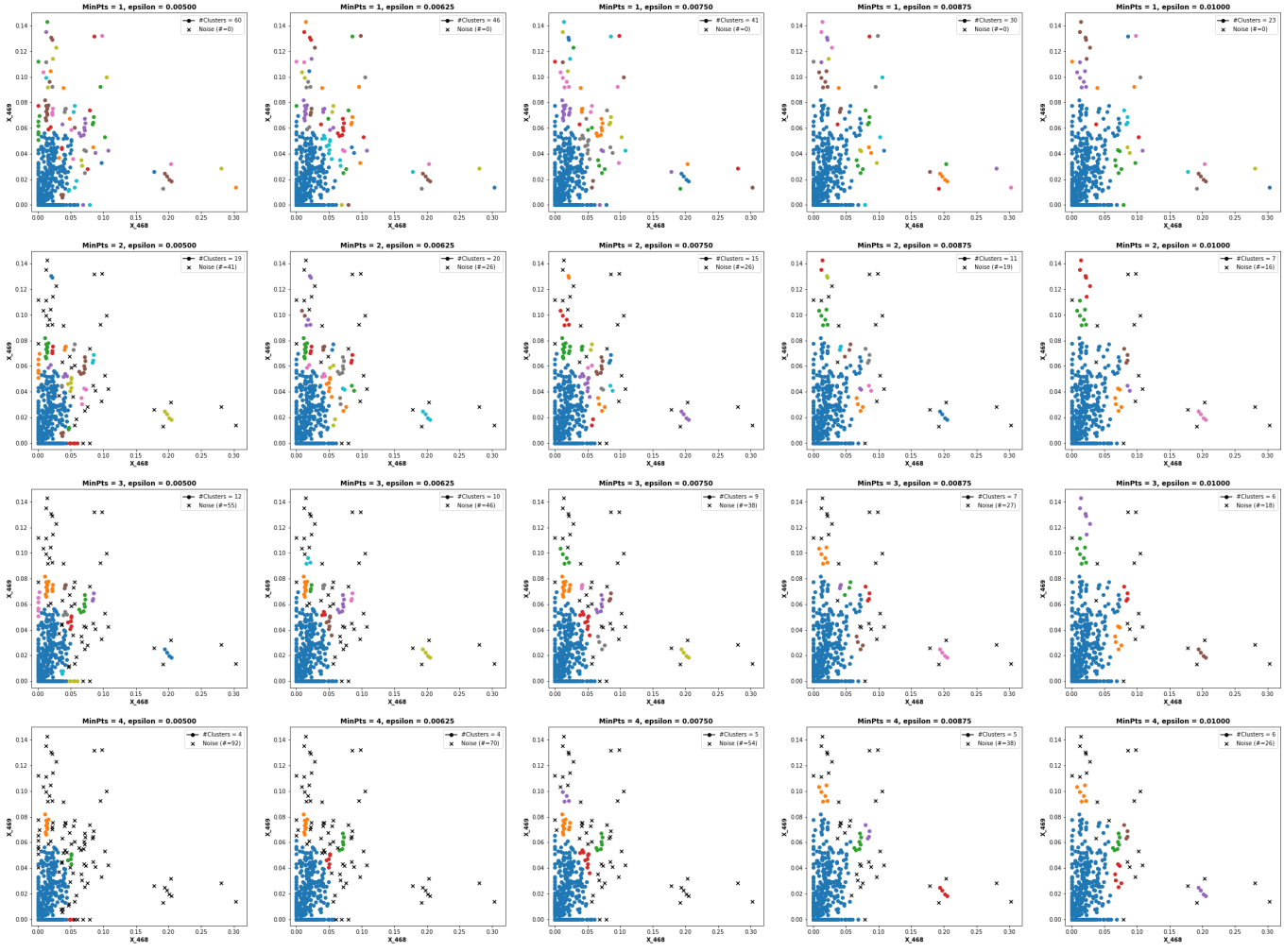
2.3.2 DBSCAN(e, MinPts), cols=[1, 2] with (e, MinPts) ∈ np. linspace(0.3, 0.33, 5) × [48, 49, 50, 51, 52]



From the above experiments it is observed that the DBSCAN algorithm is unable to distinguish between classes 1 and 2 when the radius is greater than 0.3225 and when the radius is equal to 0.3225 with the density simultaneously taking the value 48 or 49 (That is for total
(e, MinPts) $\in \{[0.33, \infty) \times \{48, 49, ..., \infty\} \bigcup \ 3.225\} \times \{48, 49\}$).

However for density MinPts $\in \{49, 50\}$ and radius $= 0.315$ an improvement of results because, compared to the previous application, a noise reduction of 39.2% is noted. In particular, nine data that were marked as noise in the previous application and one data that should normally belong to the second cluster were added to the first cluster. The second cluster also had twelve data flagged as noise in the previous application added to it and, as mentioned above, had one data removed that normally should not have been removed.

2.3.3 DBSCAN(e, MinPts), cols=[1, 2] with (e, MinPts) ∈np. linspace(0.005, 0.01, 5) × [1, 2, 3, 4, 5]



Due to the shape and density of the last two features of the xV data set, it is impossible for the DBSCAN algorithm to single out clusters that approximate the true classes of the set.