Introduction to Machine Learning
Assignment 6

Group 06
Lucas Pereira (s4507983) & Matteo Wohlrapp (s4974921)

October 25, 2021

Introduction

In this practical, we experiment with Density-Based Spatial Clustering of Applications with Noise, short *DBSCAN*. *DBSCAN* is a method of cluster analysis which assigns points in dense regions to the same clusters, recognizing outliers in the process. The upside of using this approach is a general resistance to noise and no need to specify the number of clusters. To correctly identify dense regions and clusters, the algorithm depends mainly on two parameters: $minPts$ and ϵ. Using these two, which will be explained further in the following section, *DBSCAN* classifies points into 'core points', 'reachable points' and 'outliers' and gradually uses this classification to form clusters.

Methods

As mentioned before, the *DBSCAN* algorithm mainly depends on two parameters: $minPts$ and ϵ. ϵ describes the radius around a point x which is considered the neighbourhood of x. As seen in $Figure\,1$, the neighborhood of x consists of the green points, while the red points which are outside of the ϵ-radius are not considered to be part of it. $MinPts$ defines when the neighborhood around

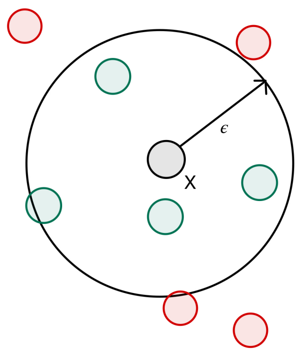

Figure 1: *ϵ-neighborhood of x*

a point x is considered dense. If, for example, $minPts = 3$, the area around x in $Figure\,1$ would be considered dense since 3 or more points are inside the ϵ-radius of x. When there are at least $minPts$ in the ϵ-neighborhood of x, x is classified as a 'core point'. If this condition is not given, but x is part of the ϵ-neighborhood of another point, it is considered a 'border point'. Otherwise, if none of the above mentioned conditions are met, x is considered an 'outlier' or 'noise'. The specific pseudo code is shown in $Listing\,1$, but the general idea of the algorithm can be described very simple. First, ϵ and $minPts$ needs to be defined, then the following steps are executed:

1. Find a point x in the data set which has not been visited yet and assign it to a new cluster, if there is no such point terminate

2. Find all $neighborhoodPts$ in the ϵ-neighborhood of x and mark it as visited

3. If $|neighborhoodPts| < minPts$ mark x as noise and return to 1., else the area around x is considered dense and we can continue

4. Now visit each point $y_1, .., y_n \in neighborhoodPts$ and find all unvisited points which are not part of any cluster yet

5. Continue this process until every 'core' and 'border' point is assigned to that cluster and start again with 1.

Since after the clustering, the silhouette score: $S_i = \frac{b_i - a_i}{max\{a_i, b_i\}}$, where $a_i = \frac{1}{|C_i|-1} \Sigma_{j \in C_i, i \neq j} d(i,j)$ and $b_i = min_{k \neq i} \frac{1}{|C_k|} \Sigma_{j \in C_k} d(i,j)$ needs to be measured, we made some tweaks to the actual implementation compared to the pseudo code in $Listing\,1$. To calculate the silhouette score, a built-in function from MATLAB was used. This function needs, besides an array for the actual data, also an array specifying the clustering. The clustering array needs to have the same length as the actual data, each cell in the clustering array having a value for the cluster the point in the cell at the same index in the data array is part of. Therefore, we immediately used such an array to not only show the affiliation of a point to a cluster, but also if the point has already been visited. As a result, the marking of a visited point is redundant since we assign it to a cluster right away. Furthermore we defined that the cluster number -1 is reserved for 'noise' and 'outliers'.

The bonus part of the assignment is implemented in very much a similar way, with the distance calculated in DBSCAN now containing 6 values for positioning instead of the two, (x, y), used in the original assignment. Since the bonus asks only for outlier detection, all groups that aren't outliers are unified in the value "1" while the outliers are classified as "0". Also, instead of calculating the silhouette score, we calculate the Precision, Recall and F-score by using the resulting classes (now simply divided between outliers and non-outliers) and the "real" classes given with the data-set.

Results

In this section we present the results of our program performing Density-Based Spatial Clustering of Applications with Noise. First we describe the k-nearest neighbour graph, showing the proposed epsilon values, afterwards clusters for $k = 3, 4, 5$ are shown. Furthermore, a table concerning the silhouette score is analyzed. In the end, we also present the results for the bonus.

0.1. K-nearest Neighbour

In this section the k-NN graph for $k = 3, 4, 5$ is discussed. For varying k, the functions in $Figure\,2$ show similar behaviour, with slow increase for a smaller index and exponential increase for higher indices. Each of the three plots has an 'elbow' point, which is estimated to be the best epsilon value. For $3 - NN$, this is at $\epsilon = 0.044209$, for $4 - NN$ at $\epsilon = 0.0549776$ and for $5 - NN$ at $\epsilon = 0.0576649$. The average is estimated at $\epsilon = 0.0523$ which was then also used for our calculations.

0.2. Clusters

In the following section, plots concerning the clustering of the data vectors, shown in $Figure\,3$, are described. We created graphs for $minPts = 3, 4, 5$ with $\epsilon = 0.523$. When considering $Figure\,4 - 6$, one can see, that the dense areas in the bottom right region of the plane, as well as in the center and mid-left are considered as seperate clusters for every value of $minPts$. The same goes for the outliers at the left and at the top, which are consistently not assigned to any cluster but -1 and therefore considered 'noise'. The biggest difference between the three clusters is the area of $x \in [0.3, 0.7]$ and $y \in [0.7, 0.9]$. For $minPts = 3$, this area is split into several clusters, while for increasing $minPts$, these clusters dissolve and the points in the area get considered as noise as well.

0.3. Silhouette score

In the following section, a table concerning the average silhouette scores of the data clustering is described. $Table\,1$ is comprised of the average silhouette score for $minPts = 3, 4, 5$ with $\epsilon = 0.523$. In general, the scores of the different amounts of $minPts$ are not far apart, however, $minPts = 4$ scores highest, followed by $minPts = 5$.

0.4. Bonus

For the bonus data set, the k-NN graph, $Figure\,7$ looks fairly similar, however, the 'elbow' point can be retrieved even clearer, since the graph follows more of an exponential curve. This time, the average ϵ is higher, with $\epsilon = 0.0089$ as the Time-Averaged-Epsilon.

The plots for the classes ($Figures\,9 - 11$) superficially look close to the ones found in the "real" classification ($Figure\,8$) however, looking at the table with the Precision, Recall and F-score values for every K ($Table\,4$), we can see resulting low values, with the highest F-score, for example, being 0.4026. It is important to note that, since this data set is $\in R^6$, we only represent the two first values in the plot as its x and y, meaning that a point that appears isolated in this plot and is not marked as an outlier may be much closer to other points thanks to the other values that constitute it's distance.

Discussion

The k-nearest neighbor function calculates the distance, ϵ, from a given point to its N^{st} neighbour, when doing $3 - NN$, for example, we are looking at the distance values for the third closest point from a given point in the data set. When points are reasonably close together this ϵ value is usually fairly small while those points much further away from others, usually the outliers, have bigger ϵ's. In our graph, the epsilon values are organized from lowest to highest, making the 'sudden jump' in the epsilon value easy to visualise. This point can then be considered as the best fitting one for this data set. As expected the ϵ value for $4 - NN$ is larger than for $3 - NN$ and the one for $5 - NN$ is larger than the other two, since we are measuring points which naturally further away.

When looking at the clusters, $Figure\,4 - 6$, one can see, that in $Figure\,4$, there are more clusters than in $Figure\,5$ and $Figure\,6$. Since the ϵ value is identical for all graphs, this can be traced back to the differing $minPts$ in the three graphs. While in $Figure\,4$, $minPts = 3$, it is higher for the other two graphs. This means that in $Figure\,4$ regions are faster considered to be dense which also causes the areas which are otherwise considered 'noise' in $Figure\,5$ and $Figure\,6$ to form own clusters. For the very dense regions, especially at the middle-right and bottom-middle sections of the plane, the algorithm produces equal solutions for every amount of $minPts$. This could imply that even though the $DBSCAN$ is especially applied on noisy data, the algorithm still struggles to divide correctly when the parameters are not set correctly.

Surprisingly, the silhouette score in $Table\,1$ is highest for $minPts = 4$, followed by $minPts = 5$. We would have expected $minPts = 3$ to result in a higher silhouette score, meaning better matching of the points to clusters. This could be a result of overfitting, since for $minPts = 3$, some areas are divided into separate clusters which don't necessarily seem to be very dense. Comparing the overall scores to the silhouette score to the silhouette scores of hierarchical clustering with the 'ward' linkage function, shown in $Table\,2$, on the same data set, we would expect equal performance. However, it appears that hierarchical clustering is better fitted to this particular data set which could be due to the fact, that there is simply not enough noise in the data.

In conclusion, when using $DBSCAN$ on this data set, $minPts = 4$ is the best option, since it yields the highest silhouette score. After setting $minPts$, also ϵ can be changed. Since the 'elbow' point in $Figure2$ for $k = 4$ is not exactly at the average $\epsilon = 0.0523$, switching to $\epsilon = 0.0549776$ could result in a higher overall silhouette score.

When it comes to the bonus, given the results seen in $Table\,4$, this method is not precise enough to detect outliers in the given dataset because the values are increasing when the number of minimum neighbours increases, especially due to a sharp increase in Recall. This is probably because it diminishes the total number of false positives and not because the method itself got much better as we can see by the marginal increase in precision itself. This might not be a fault of DBSCAN itself, though, possibly being a problem with the distance metric used. On the bonus, just as in the regular assignment, we used the squared euclidean distance, which works fine when doing the distance between two normalized dimensions, but the values in this case were most probably not normalized, variable 2 consisted of values mostly in the 10^{-2} to 10^{-3} range for example, or really suited for use in such a data set. So, given the seriousness of a thyroid problem diagnosis, this method would not be recommended.

Workload

The workload was split equally, with Lucas Pereira and Matteo Wohlrapp working both on code and report design. Issues and problems were solved in a collaborative manner, with both team partners contributing evenly.

A. Assignment 6

Listing 1: *Pseudocode for DBSCAN; Source: Lecture 7*

```
def DBSCAN (D, eps, MinPts):
    C=0
    for each unvisited point P in dataset D:
        mark P as visited
        NeighborPts = regionQuery(P, eps)
        if size (NeighborPts) < MinPts:
            mark P as NOISE
        else:
            C = next cluster
            expandCluster(P, NeighborPts, C, eps, MinPts)

def regionQuery(P, eps):
    return all points within P's eps-neighborhood


def expandCluster (P, NeighborPts, C, eps, MinPts):
    add P to cluster C
    for each point P* in NeighborPts:
        if P* is not visited:
            mark P* as visited
            NeighborPts* = regionQuery(P*, eps)
            if size(NeighborPts*)âL'ě MinPts:
                NeighborPts = NeighbotPts joined with NeighborPts*
        if P* is not yet member of any cluster:
            add P* to cluster C
```

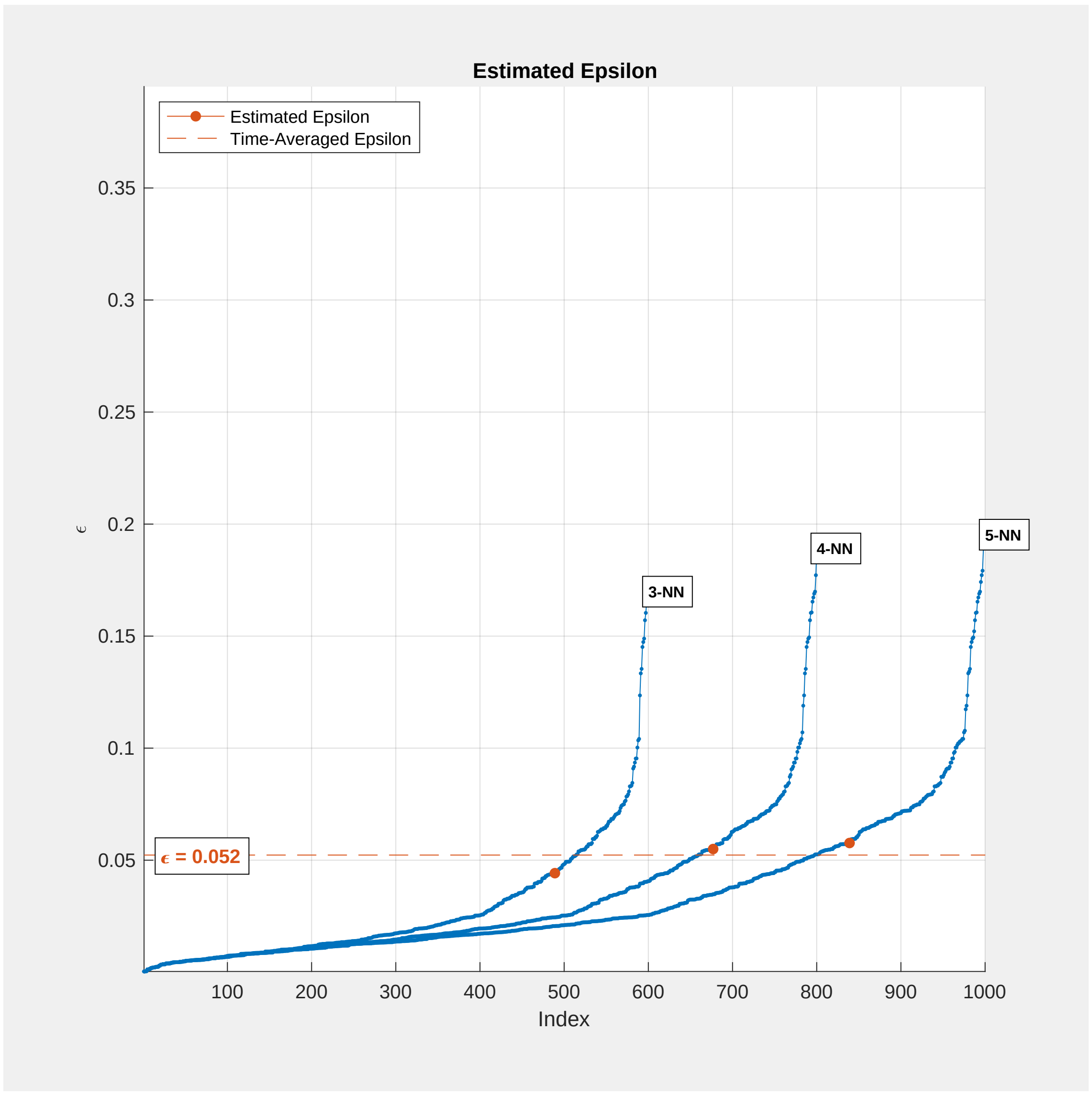

Figure 2: *k-NN graph for $k = 3, 4, 5$*

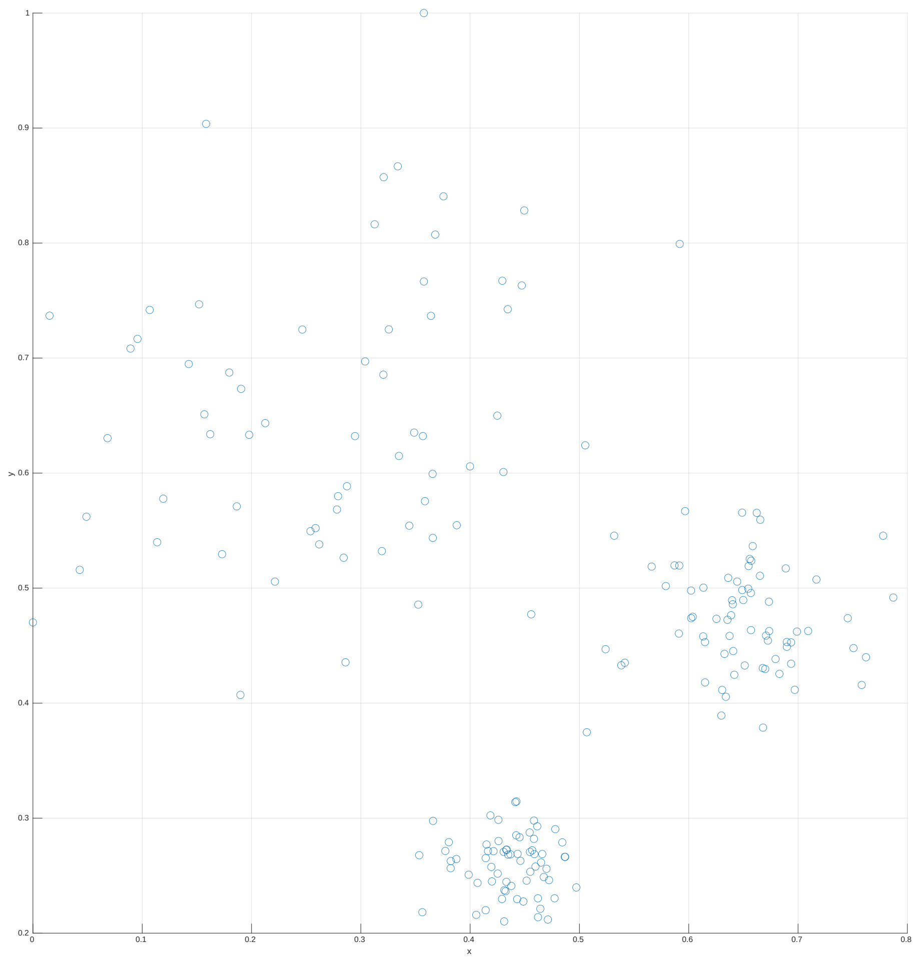

Figure 3: *Graph showing the distribution of the original data*

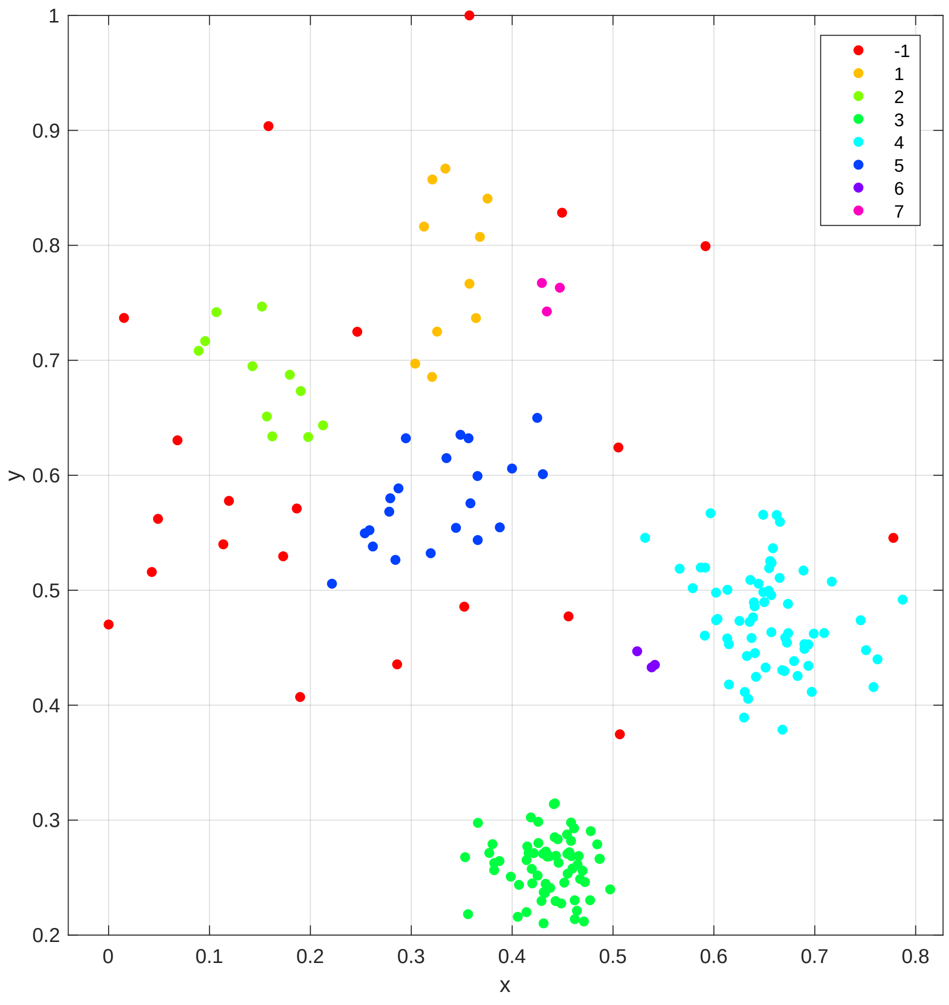

Figure 4: *Resulting cluster after the DBSCAN with* $\epsilon = 0.523$ *and* $minPts = 3$

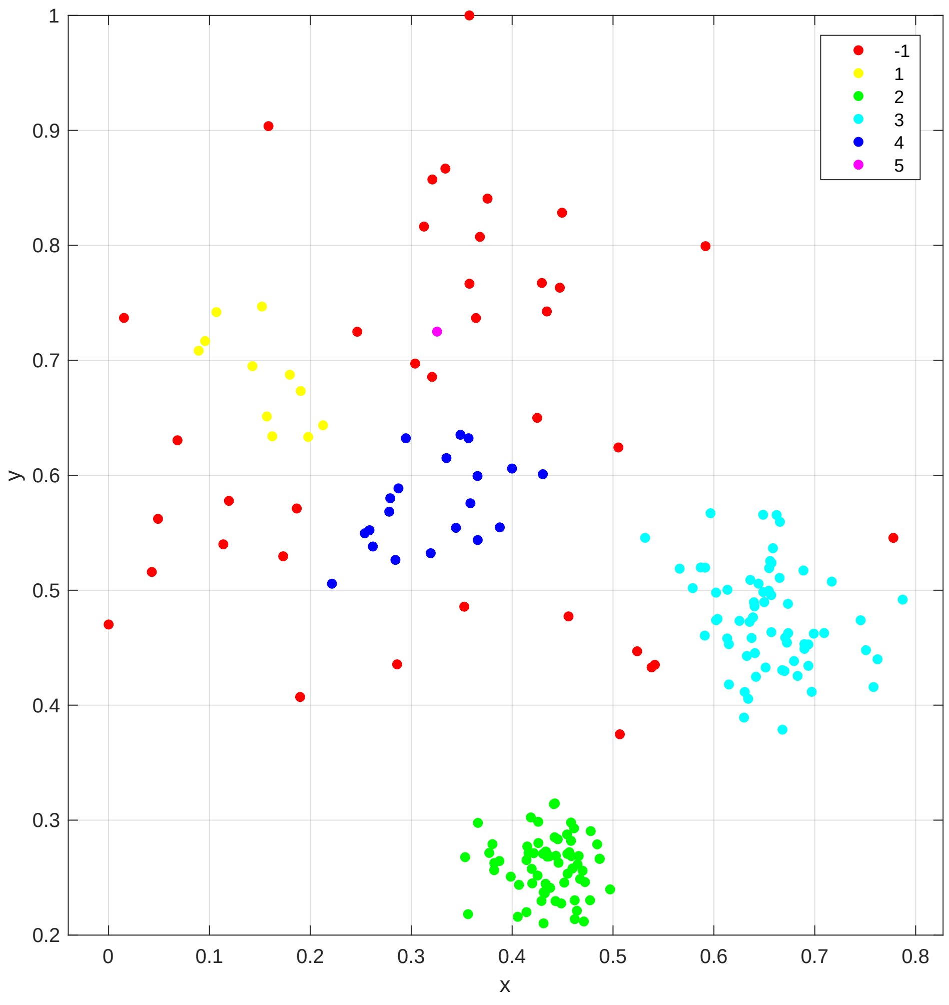

Figure 5: *Resulting cluster after the* $DBSCAN$ *with* $\epsilon = 0.523$ *and* $minPts = 4$

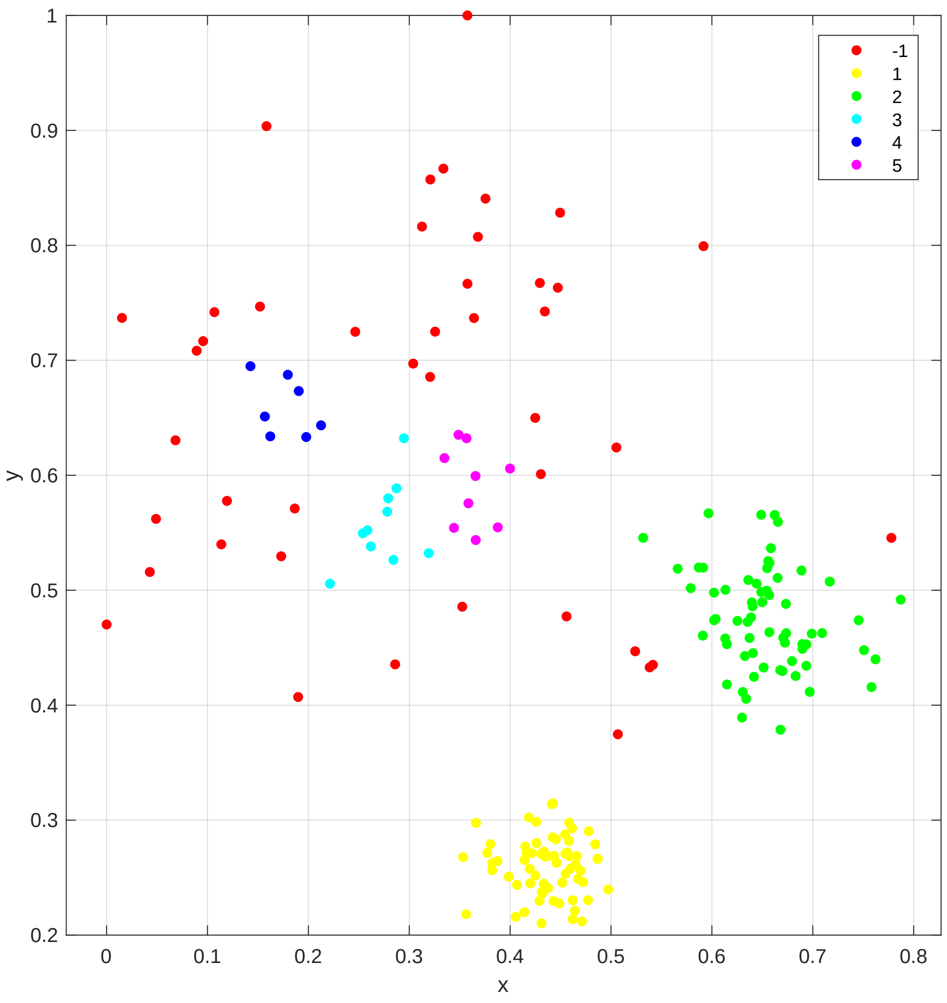

Figure 6: *Resulting cluster after the DBSCAN with* $\epsilon = 0.523$ *and* $minPts = 5$

minPts x epsilon	0.523
minPts = 3	0.5239
minPts = 4	0.5673
minPts = 5	0.5492

Table 1: *Table of Silhouette scores for* $minPts = 3, 4, 5$ *and* $\epsilon = 0.523$

K x Linkage Type	Ward
K = 2	0.7207
K = 3	0.8027
K = 4	0.7744

Table 2: *Table of Silhouette values for hierarchical clustering with the 'ward' linkage and varying K values*

minPts x epsilon Bonus	0.089479
minPts = 3	0.0832604
minPts = 4	0.0894866
minPts = 5	0.0956901

Table 3: *Table of ϵ values for $k = 3, 4, 5$ and resulting average $\epsilon = 0.089479$*

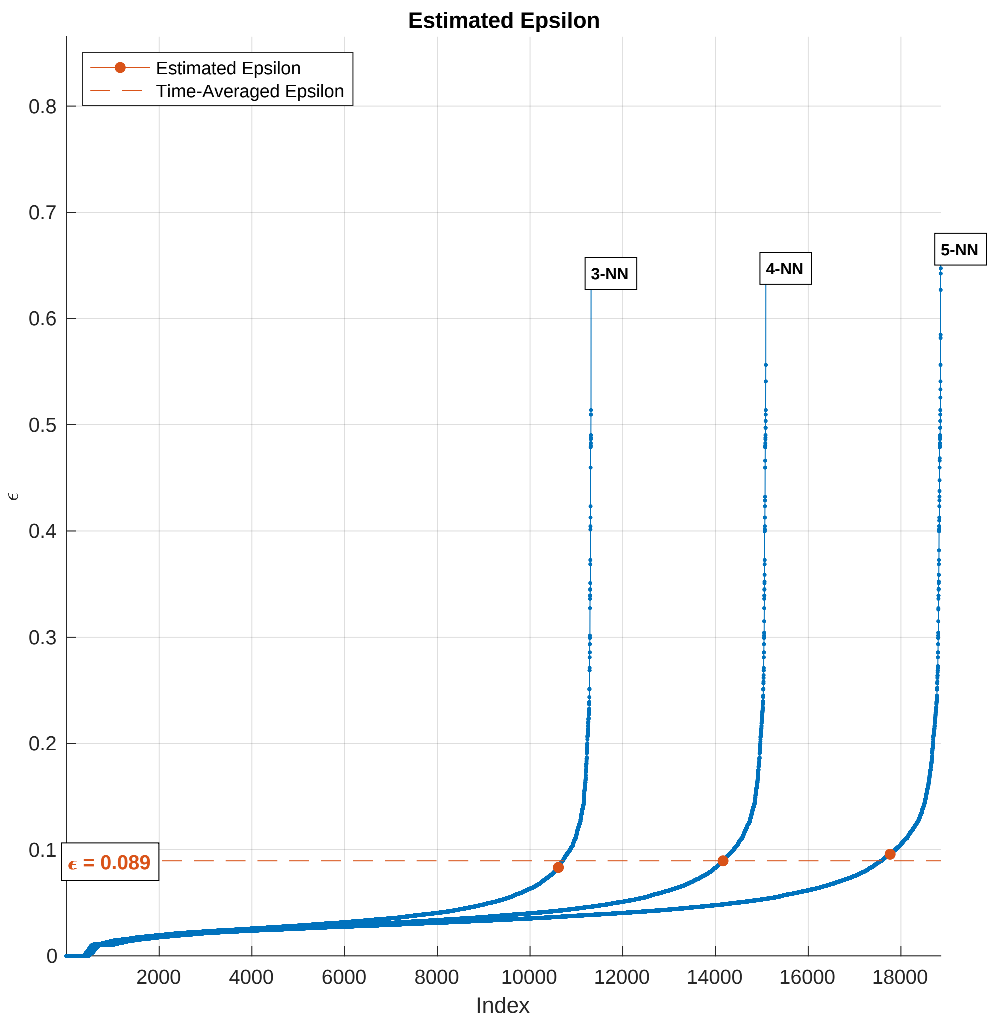

Figure 7: *K-NN for* $\epsilon = 0.089479$ *and* $k = 3, 4, 5$

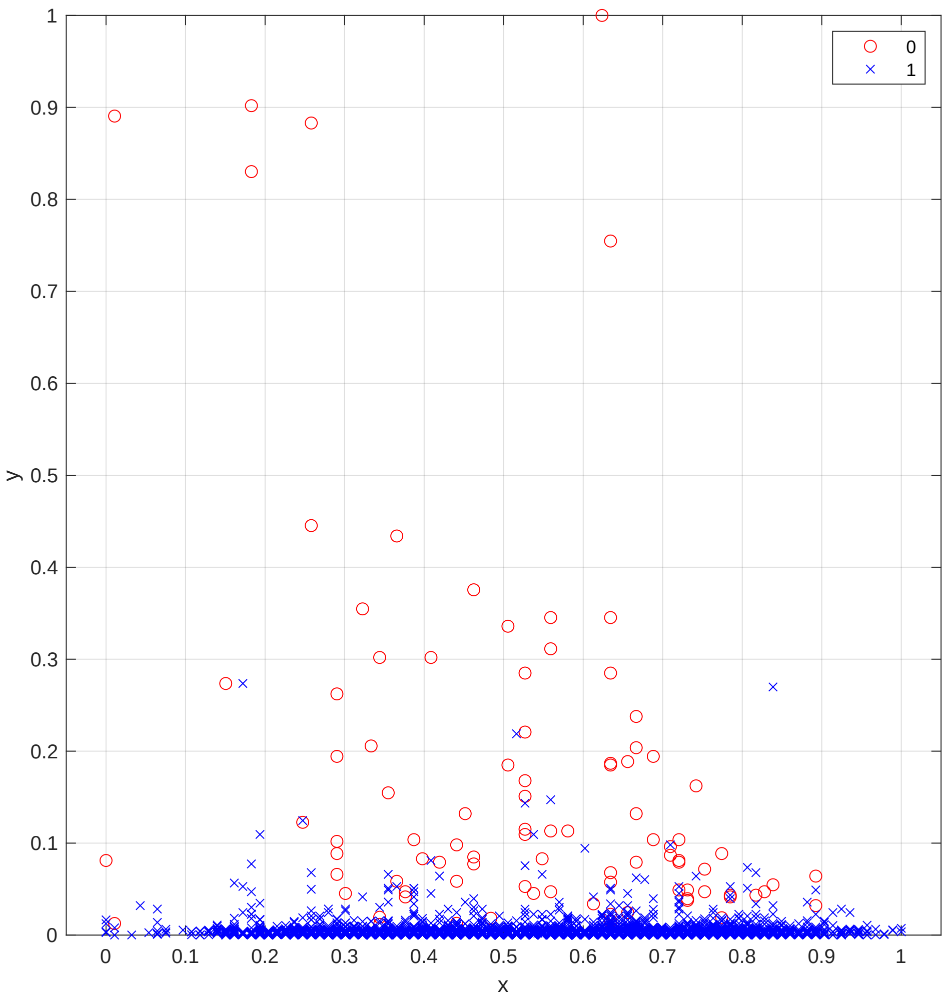

Figure 8: *Real Classification of outliers. Outliers are represented by the red circles.*

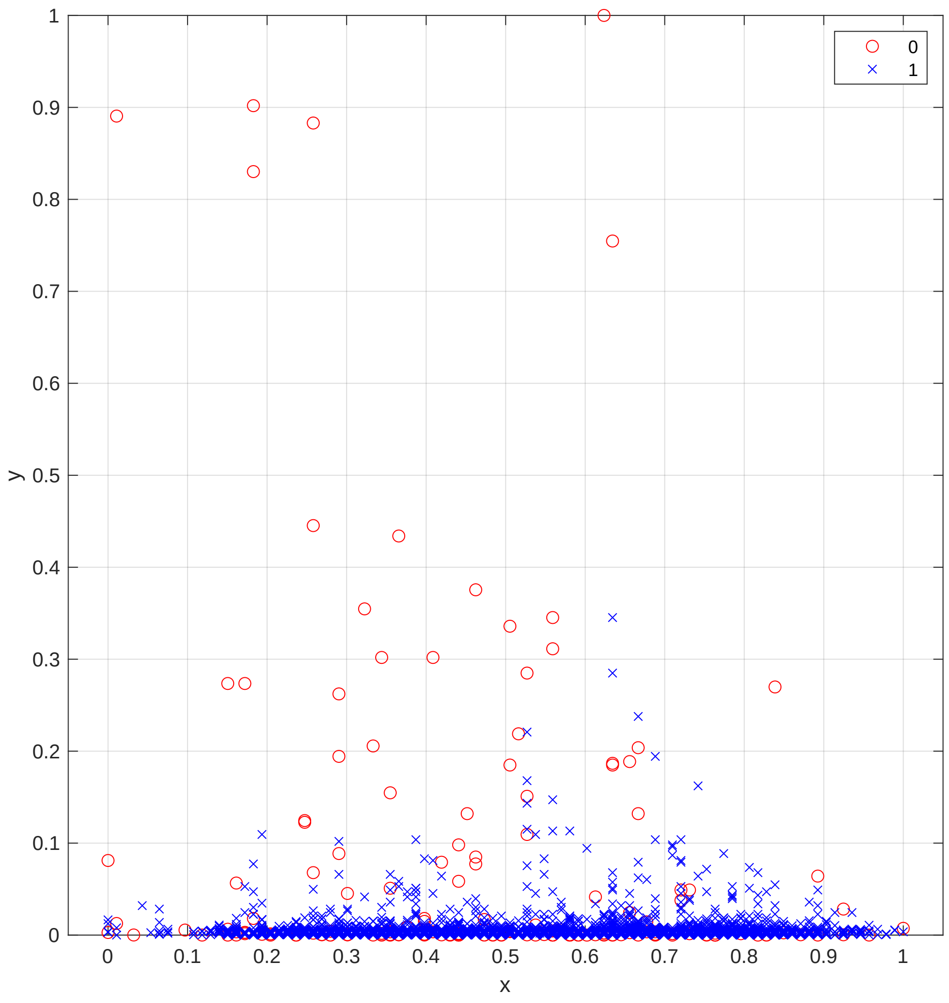

Figure 9: *Classification of outliers for $minPts = 3$. Outliers are represented by the red circles.*

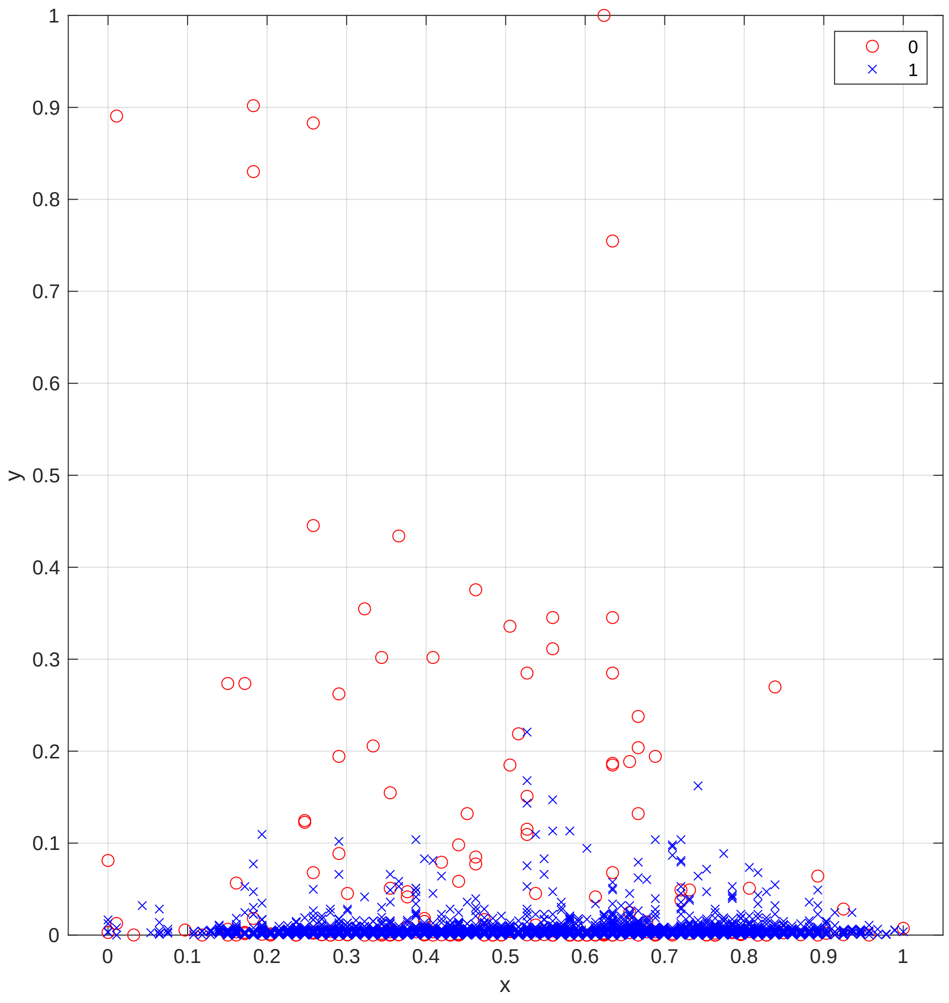

Figure 10: *Classification of outliers for $minPts = 4$. Outliers are represented by the red circles.*

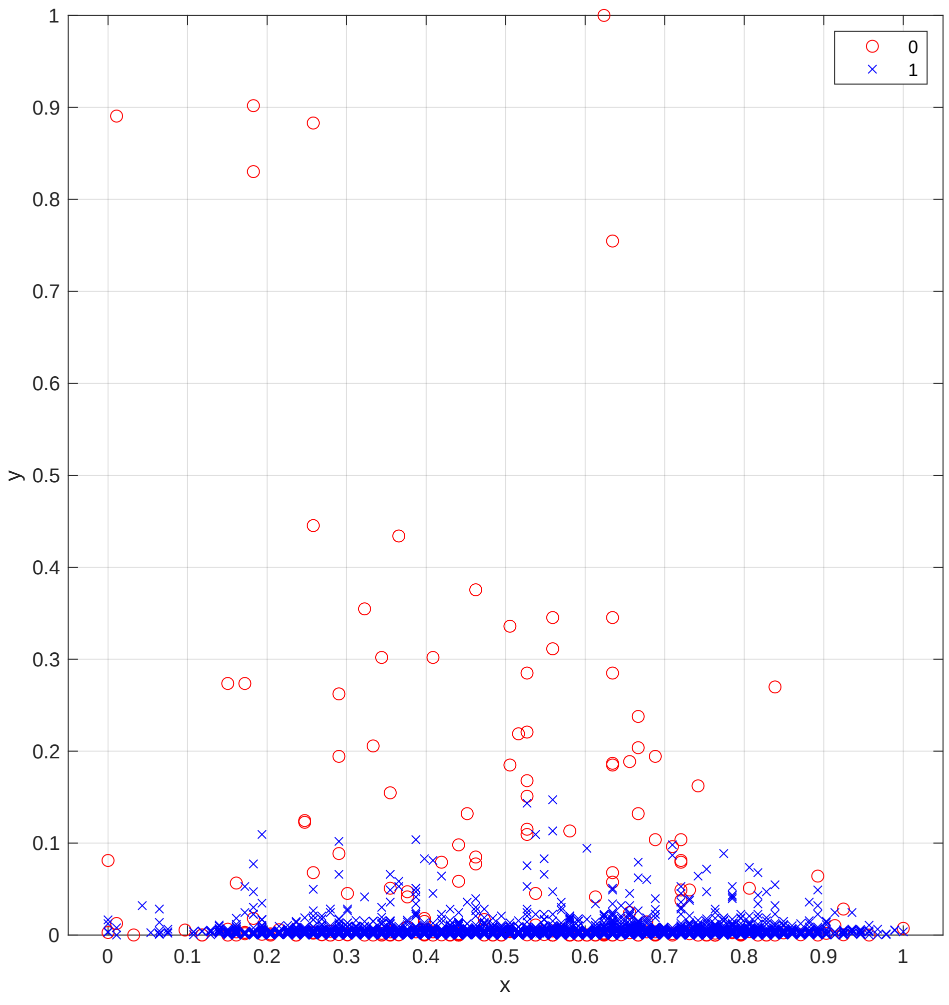

Figure 11: *Classification of outliers for* $minPts = 5$. *Outliers are represented by the red circles.*

K	Precision	Recall	F-score
3	0.2619	0.4731	0.3372
4	0.2650	0.5699	0.3618
5	0.2864	0.6774	0.4026

Table 4: *Values for Precision, Recall and P-score for each value of* K