Assignment 2

MIT 801

Connor McDonald
u16040725

Introduction To Machine And Statistical Learning

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science
University of Pretoria
5 July 2021

1 Hierarchical Clustering

Hierarchical clustering is a popular unsupervised machine learning algorithm which, like all clustering algorithms, was designed for the extraction or identification of natural groupings within data. Hierarchical clustering consists of two main approaches, namely; agglomerative clustering and divisive clustering. In agglomerative clustering each data point starts as it's own cluster, for each iteration of the algorithm the two nearest clusters are combined to form one cluster. This process is repeated until there is only one cluster remaining. A dendrogram figure 3 is often plotted along side this process and used to determine an optimal number of clusters once the algorithm has terminated. Divisive clustering works by starting with one cluster, which contains all the data, and recursively splits the data into new clusters with each iteration of the algorithm.

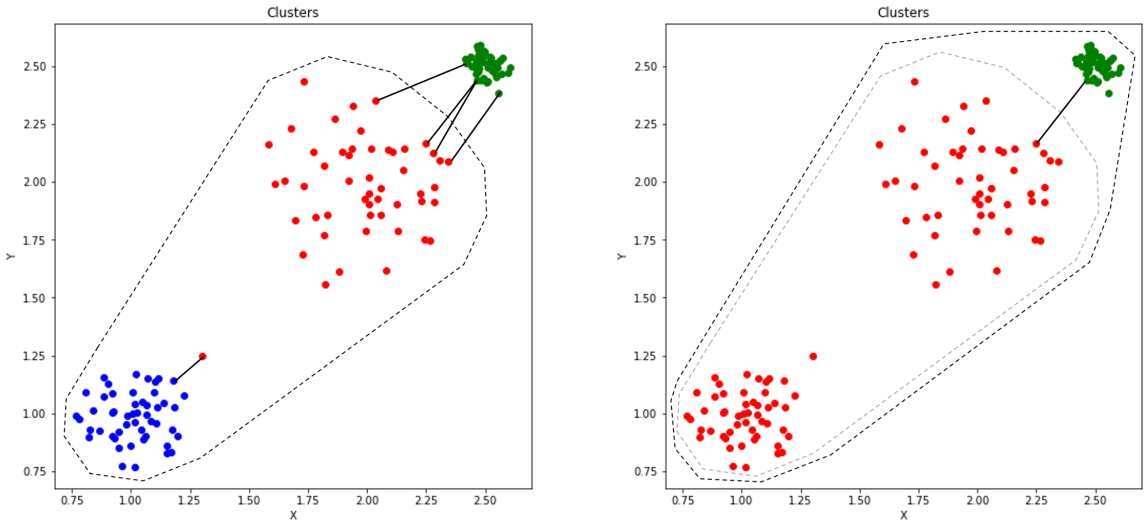
It is important to note that in order for these algorithms to work, a consistent distance metric needs to be defined. For example, it is easy to measure the distance between clusters when each of them is a single point, but what about when two clusters contain many points? How should one measure the distance between these two clusters now? To address this problem various methods have been created such as, single linkage, complete linkage, average linkage, the centroid method and Ward's method (Giudici & Figini 2009). Single linkage and complete linkage will be discussed further below.

1.1 Single Linkage Clustering

Single linkage clustering consists of combining clusters based on the distance between the two closest elements in each cluster. This can be defined by equation 1 shown below.

$$d_{single}(G, H) = \min_{i \epsilon G, j \epsilon H} d_{ij} \tag{1}$$

A major characteristic of single linkage clustering is that it can produce long chains of clusters, which could result in two points which are very far away from each other being classified in the same cluster. In some domains this is a useful trait, and in others it may adversely affect the quality of the clusters. A visual representation of single linkage clustering is shown in figure 1. In the first plot we can see that the red cluster has many points which are close to the green cluster, however a single outlier of this cluster is closer to a blue point than any of the green points. For demonstration purposes, distances were exaggerated, but nevertheless, this can still happen with single linkage clustering. The chain effect is visualised in the second plot. Now that the red and blue clusters have been combined, the next closest point is a green point, which results in a combination of the remaining clusters. This has allowed blue and green points to be clustered together even though they were initially very far apart. As stated before, this is not necessarily a bad thing as in some domains it may be useful.



(a) Single Linkage Clustering

(b) Chain Effect

Figure 1

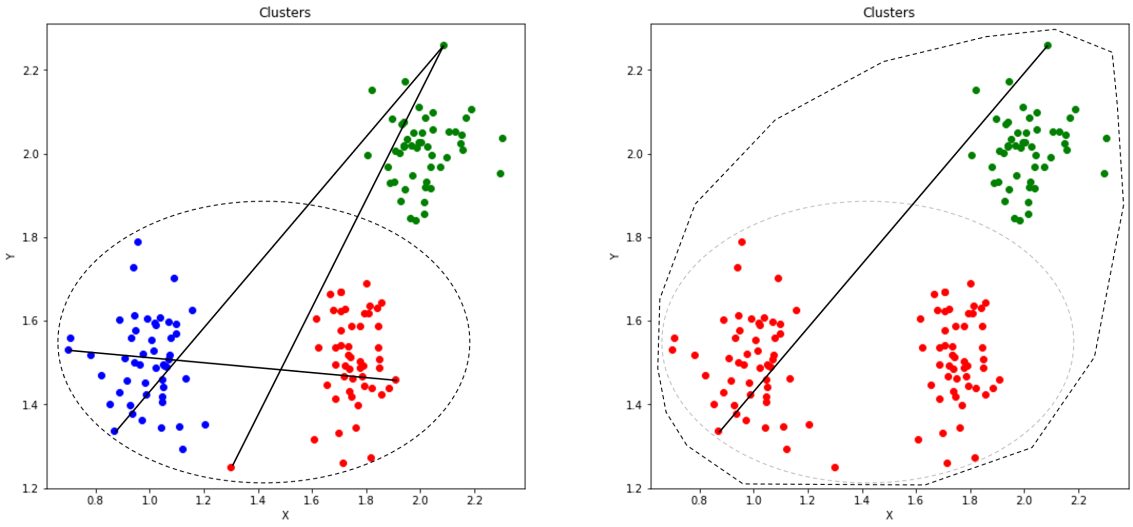
1.2 Complete Linkage Clustering

Complete linkage clustering works by combining clusters with the minimum distance between their furthest points. Unlike single linkage clustering which only ensures that two points within a cluster are close to each other, complete linkage ensures that all points are reasonably close to one another and essentially forces a "spherical" cluster with a consistent diameter. Complete linkage can be defined by equation 2

$$d_{complete}(G, H) = \max_{i \epsilon G, j \epsilon H} d_{ij} \tag{2}$$

Looking at the first plot in figure 2, if we calculate the distances between the furthest points of each cluster, we can see that this distance is the smallest between the red and blue clusters. We would now combine these two clusters, this creates a cluster where all of the internal points are relatively close together, unlike single linkage. This process is repeated in the second plot to combine the remaining two clusters.

Once again, neither method is "more correct" than another, they both perform well when applied to the correct domain. For example, single linkage may be better when applied to datasets that have different scales in their dimensions, which would result in longer and more linear clusters. Complete linkage may be better when all dimensions are within the same order of magnitude, and will create more spherical clusters. Lastly, there are numerous other methods which make use of different distance metrics that could be used in cases where single and complete linkage give insufficient results.



(a) (b)

Figure 2: Complete Linkage Clustering

1.3 Cluster Selection

Selecting the optimal amount of clusters is a challenge many researchers have faced. Too many clusters may not sufficiently reduce the dimensionality of the dataset, and too few clusters may not effectively show the natural grouping within the data. One solution to this problem is to plot a dendrogram (figure 3) of our clustering process. This method will be detailed on the following page. Please note that the dendrograms in figures 3 and 4 were created using Ward's method as it selects clusters with the least internal variance.

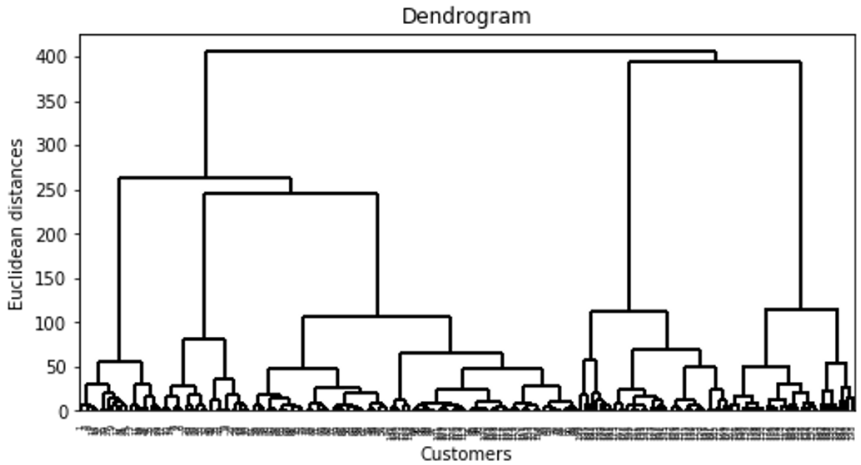
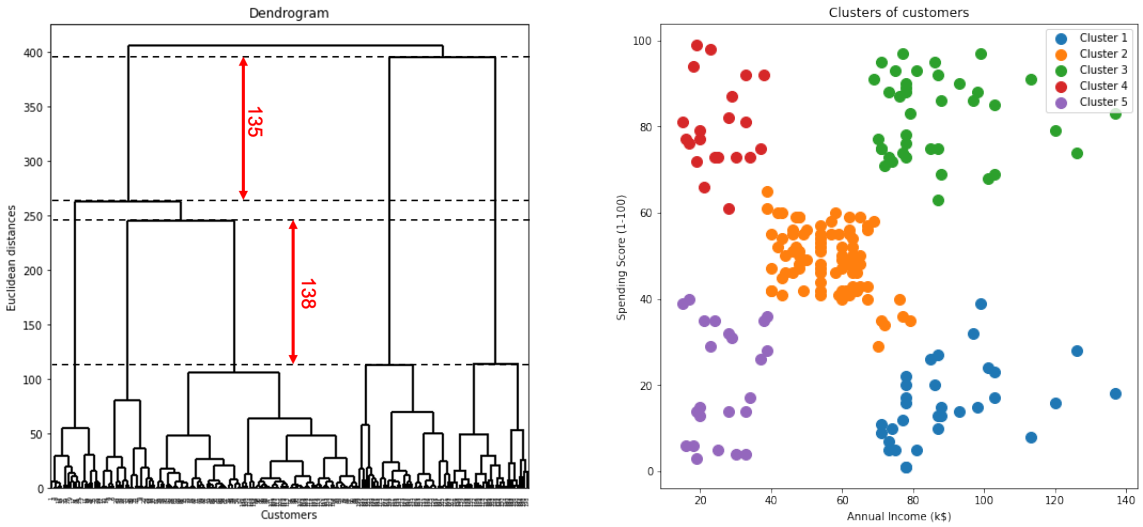


Figure 3: Dendrogram



(a) (b)

Figure 4: Optimal Cluster Selection

When using a dendrogram to select an optimal number of clusters we must either use a predetermined threshold of dissimilarity or we can measure the vertical distance between horizontal lines, and the section of the graph with the greatest distance between horizontal lines will contain the optimal number of clusters expressed as vertical lines. If we look at figure 4 we can see there are two major regions of vertical distance on the chart. The bottom region is slightly larger by three euclidean units, meaning that our optimal number of clusters is the number of vertical lines observed in this region, in this case we should group our data into five clusters. By visualising the scatter plot of the data we can see that five clusters proved to be sufficient in identifying the natural grouping within the data. When using a predetermined threshold one would plot a horizontal line on the dendrogram, and the number of vertical lines it crosses would equal the number of clusters to used. One reason to steer clear of using dissimilarity thresholds is because it can be difficult to predict what the dendrogram will look like which could result in a severe misplacement of this threshold.

2 Clustering By Mode Seeking

Mode seeking is a type of hierarchical clustering which, uses gradient ascent methods to find the mode(s) of the probability density function that describes a given dataset. However, the way in which this probability function is generated and ascended can differ. Mean-shift mode seeking is a popular algorithm among researchers that uses a kernel based density estimator to create a probability density function. The algorithm then analyses the convergence of the integral curve for each point in the dataset, all points which converge on the same mode will be considered part of the same cluster and will belong to the same "basin of attraction" (Myhre et al. 2018). A simulated example of this has been shown in figure 5, where the attraction basin is shown by the dashed lines, and the modes are shown by the points in the center of each cluster, lastly the solid lines represent estimated trajectories of various points. Please note that the figure below is purely for explanation purposes and may not be entirely accurate.

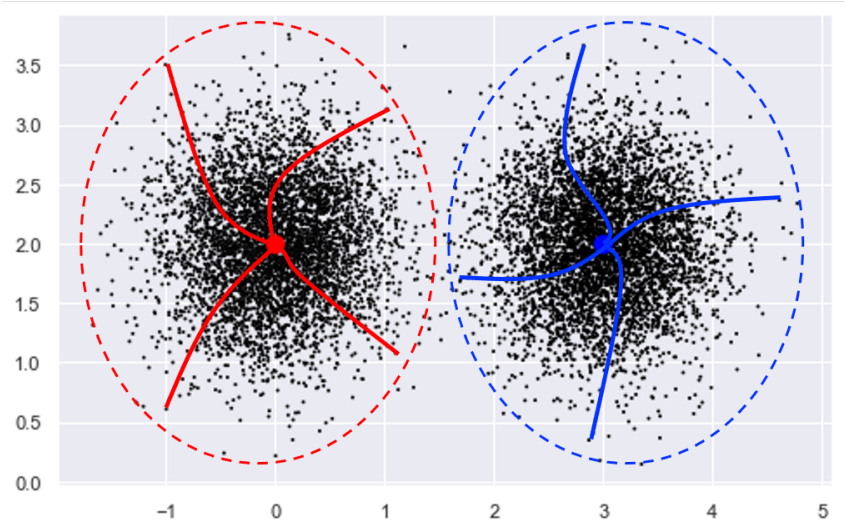


Figure 5: Mean-shift Mode Seeking Visualised

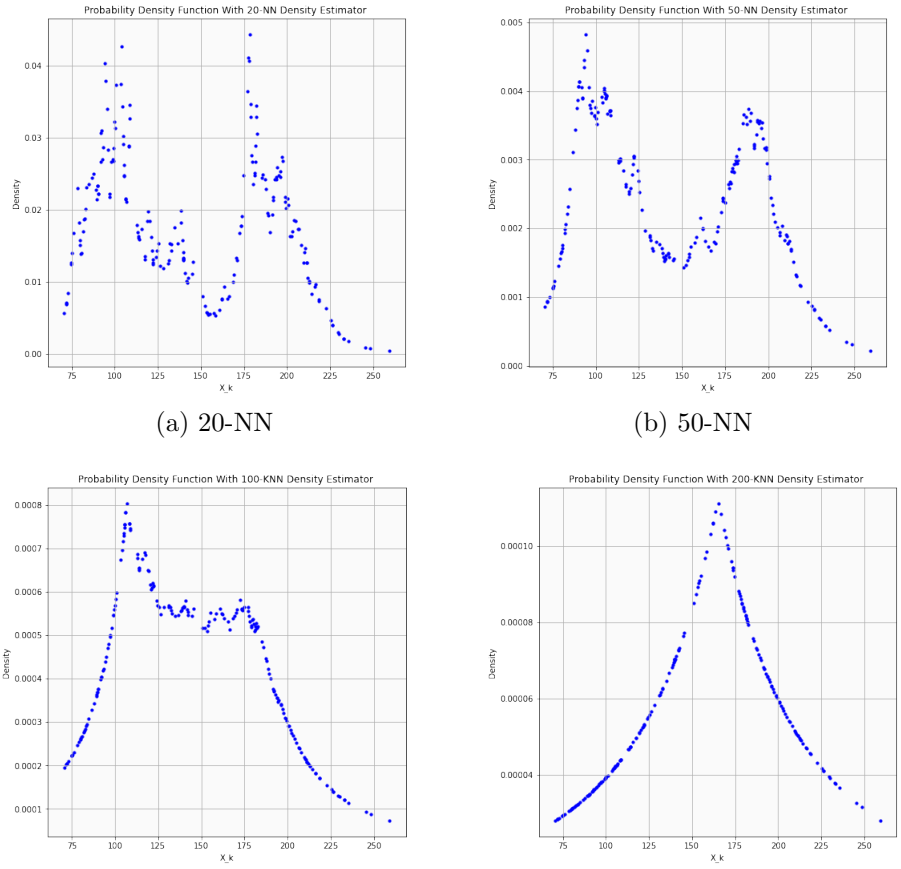
The mean-shift algorithm has seen great success when applied in various domains, such as image processing, computer vision, and object motion tracking. However, mean-shift is known to be relatively slow and computationally intensive, these drawbacks are further exaggerated when attempting to scale the process for high-dimensionality applications. Myhre et al. propose an alternative method of modal clustering known as k-Nearest Neighbor mode seeking. This algorithm produced comparable results to that of the mean-shift algorithm, however there was a notable improvement on run time and ability to scale. Myhre et al. have proposed new methods and techniques to address this problem. This algorithm is discussed in detail in the subsequent pages and will form the basis for the remained of this report.

2.1 K-NN Based Density Estimation

The k-NN based density estimation works in a similar fashion to that of k-NN clustering. The algorithm identifies the k-nearest neighbors to a given point measures the distance between the given point and it's k'th neighbor. The squared inverse of this distance is used as a density estimator. The formula for this density estimate is shown below in equation 3.

$$f(x_i) = \frac{1}{dist(x_i - x_k)^2} \tag{3}$$

However, our density estimate shows extreme sensitivity to k-values, where low values can cause a high number of local modes resulting in a "spikey" appearance of the probability density function. Furthermore, high values of k can cause excessive smoothing in the probability density function. This is seen in figure 6 where a k-NN density estimator was used to create a probability density function for 200 points with various values of k. One should aim to find a balance between smoothness and identifying an adequate number of modes.



(a) 20-NN

(b) 50-NN

(c) 100-NN

(d) 200-NN

Figure 6

2.2 Mode Seeking With k-NN Density Estimator

Mode seeking is fundamentally the same for the mean-shift and k-NN algorithm, both algorithms find the local maximums (modes) of the probability density function through an iterative gradient ascent method. When looking at the k-NN approach, we must first note that the "neighborhood" size/population is the same for each point, because of this property, the only factor that can influence the density of a point is how far away it is from it's k'th neighbor. Another property worth noting is that points may fall in more than one neighborhood, this is important when searching for modes, as a point may not contain a mode in it's neighborhood, but one of its neighbors may contain a mode within it's respective neighborhood. This property is also useful in clustering as we can track the path that each point takes when converging on a particular mode. As stated previously, a low value of k will cause a "spikey" probability density function, this in turn will cause a lot of modes to be found in the mode searching algorithm, which can negatively impact the effectiveness clustering points based on the mode they converge on. By using the same dataset we used to generate figure 6, we can see the number of modes found has an inversely proportional relationship to the value of k (Table 1). This was only for a dataset of 200 observations and we are already seeing up to 6 modes (which would later become clusters). With larger datasets the number of modes for low k-values can be drastically higher, which creates an unnecessarily high number of clusters that dilute the value of clustering in the first place.

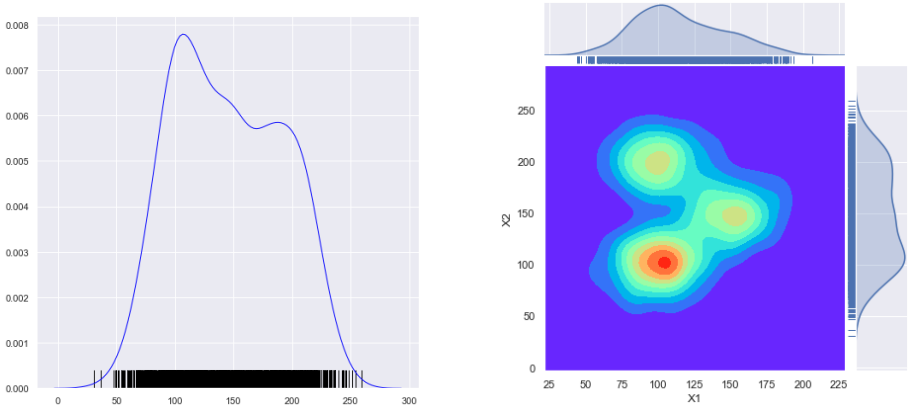
In essence, a mode is found by starting at an initial point i, and finding the highest density point within the $i's$ neighborhood. If the highest density point in the neighborhood is i, then a local max is found and we can categorize i as a mode. If not then we move to point $i+1$ and repeat the process, until we have found all the points that have the highest density within their own respective neighborhoods.

Table 1: Number of Modes Found at Different Values of K

K	**Modes Found**
20	6
50	2
100	2
200	1

2.3 Univariate & Bivariate Clustering

When implementing the k-NN mode seeking algorithm, a kernel density plot was created as a bench mark for both the *Mode1.xlsx* and *Mode2.xlsx* files. This is visualised in figure 7, we can see that *Mode1.xlsx* (the univariate case) shows two distinct modes, one could even argue that there is a less prominent third mode between these two modes. Looking at the bivariate case, it is obvious that there are three distinct modes within the data.

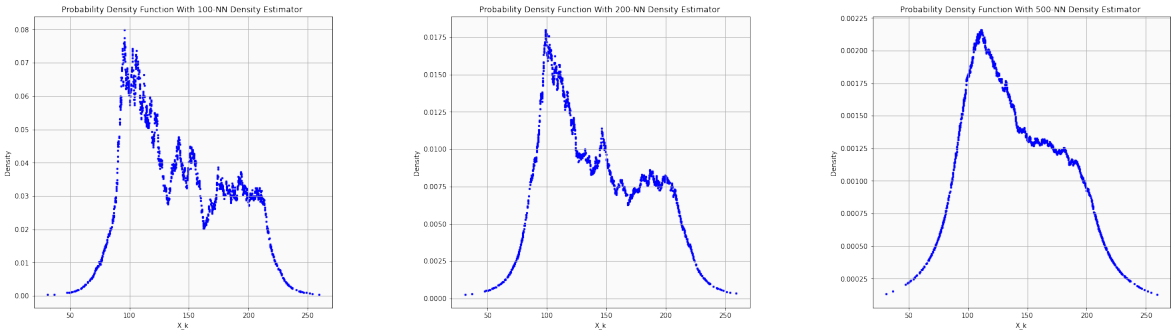


(a) Univariate Kernel Density Estimate

(b) Bivariate Kernel Density Estimation

Figure 7

To effectively utilise the the k-NN density estimation algorithm we need to select an adequate value of k. Looking at figure 8, we can see that using a value of 100 may create too many local modes, whilst a value of 500 may "oversmooth" the density function causing us to miss the subtle mode between to two clear ones. Using a k-value of 200 produced favorable results for this dataset as it achieves a relatively smooth estimate while clearly identifying the smallest mode in the middle. This value was later used in the clustering process.



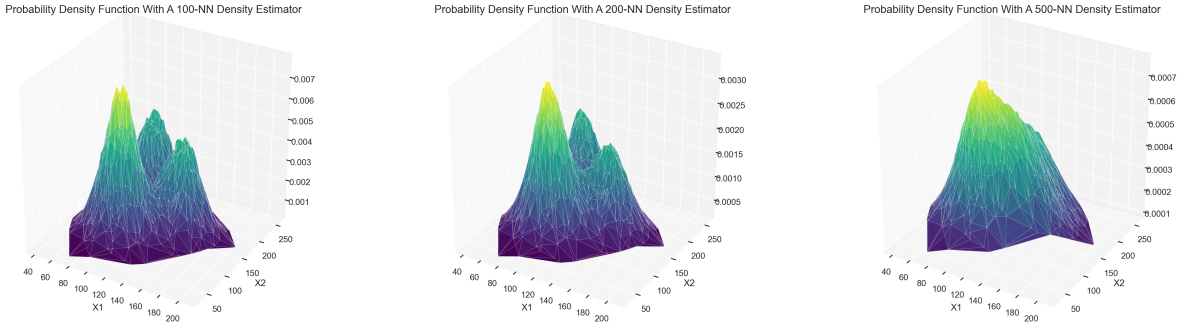
(a) 100-NN

(b) 200-NN

(c) 500-NN

Figure 8

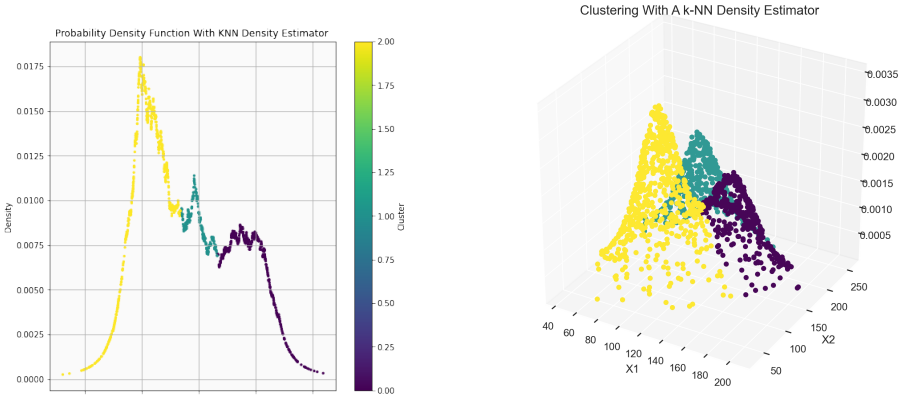
As for the bivariate case, we must once again investigate different values of k, using the same k values as in the univariate case we get figure 9. Here we can see that a k value of 500 has "oversmoothed" the density function and there is only one clear mode. However, using 100 and 200 produced comparable results with the only difference being that a k-value of 100 took noticeably longer to converge when searching for modes, this may be due to the fact that the neighborhood size is smaller and therefore has less of a chance of containing a mode. For the reasons stated above a k-value of 200 was also selected for the bivariate data when clustering the data.



(a) 100-NN (b) 200-NN (c) 500-NN

Figure 9

Once we begin clustering each dataset with a k-value of 200, we see that the algorithm was successfully able to group the points into three distinct clusters for both datasets (figure 10). This aligns with the number of modes seen in figure 7, which made use of the kernel density estimate.



(a) Univariate Clustering With k-NN Density Estimate

(b) Bivariate Clustering With k-NN Density Estimate

Figure 10

References

Giudici, P. & Figini, S. (2009), *Applied data mining for business and industry*, Wiley Online Library.

Myhre, J. N., Mikalsen, K. Ø., Løkse, S. & Jenssen, R. (2018), 'Robust clustering using a knn mode seeking ensemble', *Pattern Recognition* **76**, 491–505.