Exercise: Desnity-based Clustering (DBSCAN)

The problem

We would like to analyze kernels of wheat to find a method through which we can group kernels by size. We will first consider clustering by just the size of the kernel (kernel length and width), and then we will look at creating clusters based on all numerical variables in the dataset. In this notebook, we will load and explore 'seeds.txt' in R, which is a dataset containing measurements of geometrical properties of kernels belonging to three different varieties of wheat.

The DBSCAN algorithm combines data points that are closely packed together into clusters based on the parameters **eps** and **minPts**.

The **eps** parameter corresponds to the size of the neighborhood.

The **minPts** parameter corresponds to the minimum number of points that must exist in this neighborhood to define it as a dense area.

This exercise covers:

1. Installing and loading the DBSCAN library and the data in R
2. Creating a subset containing kernel length and width from the datatset
3. Using the dbscan function to obtain clusters
4. Converting the clusters to factors
5. Attaching the clusters to the measurements
6. Visualizing the results
7. Considering all numerical variables
8. Conclusion

Installing and loading the DBSCAN library and the data in R

We begin by installing the dbscan package and loading it along with the dataset .

```
# Installing the library 'dbscan'
install.packages("dbscan", dependencies = TRUE)
library('dbscan')
```

```
# Load data
seeds <- read.csv("https://ibm.box.com/shared/static/c1aw37ex3sx99pb9q2l8fwz643wnbeo6.txt", sep = " ")
head(seeds)
```

Creating a subset from the dataset

```
# Creating the subset: The subset contains the kernel width and the kernel length
seeds.sub <- subset(seeds, select = c(width,length))
head(seeds.sub)
```

Using the dbscan function to obtain clusters

Use the dbscan function to obtain clusters and store them in clusters_assignments1

For this exercise we will use eps = 0.8 and minPts = 4

Note: You can change the values to see the formation of different clusters

Noise points are assigned to cluster 0 by default

```
## Your Answer Code Here: ##
clusters_assignments1 <- dbscan(seeds.sub, eps = .08, minPts = 4)
clusters_assignments1
```

Converting the clusters to factors

```
## Your Answer Code Here: ##
# Clusters must be converted to factor before plotting in different colors
clusters_assignments1$cluster <- as.factor(clusters_assignments1$cluster)
head(clusters_assignments1$cluster)
```

Attaching the clusters to the measurements

```
## Your Answer Code Here: ##
# Combine the cluster assignments with the subset
seeds.sub$cluster_no <- clusters_assignments1$cluster
head(seeds.sub)
```

Visualizing the results

```
## Your Answer Code Here: ##
# Visualize results (noise is shown in black) using a simple plot
plot(seeds.sub$width, seeds.sub$length, col = clusters_assignments1$cluster, pch = 16, main = "Scatterplot Displaying Clusters", xlab = "Kernel
legend(x = 2.6, y = 6.75, legend = levels(clusters_assignments1$cluster), col = c(1:5), pch = 16, title = "Clusters")
```

Considering all numerical variables

```
## Your Answer Code Here: ##
# Create subset including all numerical variables only.
# Note: The variable 'type' is categorical.
seeds.submain <- seeds[1:7] # We will run the dbscan algorithm with eps = 0.9 and minPts = 4.
# Note: Using the previous value of eps = 0.08 will create only one cluster containing all datapoints.
clusters_assignments2 <- dbscan(seeds.submain, eps = .9, minPts = 4)
clusters_assignments2 # Clusters must be converted to factor before plotting in different colors
clusters_assignments2$cluster <- as.factor(clusters_assignments2$cluster) # Linking the assigned cluster to each station
```

```
seeds.submain$cluster_no <- clusters_assignments2$cluster # Visualize results using a pairs plot (noise is shown in black)
pairs(seeds.submain, col = clusters_assignments2$cluster, pch = 16, main = "Scatterplot Displaying Clusters")
```

Click here for the solution

``` seeds.submain <- seeds[1:7] # We will run the dbscan algorithm with eps = 0.9 and minPts = 4. # Note: Using the previous value of eps = 0.08 will create only one cluster containing all datapoints. clusters_assignments2 <- dbscan(seeds.submain, eps = .9, minPts = 4) clusters_assignments2 # Clusters must be converted to factor before plotting in different colors clusters_assignments2$cluster <- as.factor(clusters_assignments2$cluster) # Linking the assigned cluster to each station seeds.submain$cluster_no <- clusters_assignments2$cluster # Visualize results using a pairs plot (noise is shown in black) pairs(seeds.submain, col = clusters_assignments2$cluster, pch = 16, main = "Scatterplot Displaying Clusters")

```

Aside:

A pairs plot produces a matrix of scatterplots for each variable of the input dataset. To interpret a pairs plot consider it as a matrix with rows and columns. The variable stated in a row represents the variable that is used for the y-axis of all plots in that row. The variable stated in a column is the variable used for the x-axis of all plots in that column.

Conclusion

The results from both implementations of the DBSCAN algorithm provide somewhat similar clusters when considering the geometric size of the kernel (kernel length and width). In both cases, cluster 1 represents kernels of a smaller geometric size, which comprises more than 60% of the datapoints. However, the implementation using all the numerical variables has fewer noise points (represented as cluster 0).

Thank you for completing this exercise!

This notebook is part of the ML with R course on Cognitive Class

Notebook created by: Dominique Warren

References:

https://en.wikipedia.org/wiki/DBSCAN
https://cran.r-project.org/web/packages/dbscan/dbscan.pdf
https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf
