Exercise:
K-Means Clustering

Machine Learning and Prediction Modelling

Exercise 1: Wine quality data

The wine quality data set summarizes various properties of different wines. The variables include amongst others the amount of alcohol, the ph-value, the density, the residual sugar and a quality-score (ranging from 0-10). There are two data sets, one for red wines and one for white wines.

a) Load the two data sets contained in the `wine_data.rda` file using the `load` command. Inspect the two data sets to get a first impression of the data. You can also draw pairs plots for the two data sets to get a visual impression.

b) We want to add a `colour` column to both data sets. For the white wine data, this column should only have the value `"white"` and for the red wine data `"red"`. (**Hint**: `wine_red$colour <- ...`)

c) Combine the two data sets into one big data set called `wines` using the `rbind` command. The `rbind` command can take multiple data frames as arguments and combines them rowwise. Check what the dimensions of the new data set are to make sure the combination worked.

d) Turn the `colour` variable in the `wines` data frame into a factor.

e) Although the `colour` variable will be of use later, for k-means clustering we can only use numerical variables. Create a copy of the data frame with the name `wines_nocol` which does not include the `colour` column. (**Hint** `wines_nocol$... <- NULL`)

f) The `wines_nocol` data frame only includes numeric variables. We want to scale this data before we apply any cluster algorithm to it. The reason is that the variables are on rather different scales, and we don't want variables with larger variance to dominate any measures of distance between points. To scale all columns of a data frame, we can use the `scale` function, which takes a data frame as an argument. Because the `scale` function automatically changes the data into a matrix, we will apply the `data.frame` function to the result again, so that we retain the data as a data frame. (**Hint**: `data.frame( scale(...) )`)

g) Apply k-means clustering to the scaled data looking for two clusters. Do not set the random seed before applying the `kmeans` function. To get an idea of how good the found cluster pattern is, create a silhouette plot of the found solution. Interpret the result. Now run the used code a couple of times again. Do the results always turn out the same way? Why not?

h) We can force the `kmeans` function to try out multiple random starting positions and choose among the achieved solutions the best one. This can be done with the `nstart` option. Apply again k-means

clustering to the scaled data looking for two clusters, but trying out 10 random starting positions (**Hint**: `kmeans(..., nstart=10)`). Draw again a silhouette plot to inspect the solution.

i) To investigate whether a different number of clusters fits our data better, we want to calculate the within-cluster SSQ for different values of k. Visualize the resulting SSQ when using the values 1 to 6 for k. Use again multiple random starting positions when applying `kmeans`. Interpret the resulting figure.

j) Based on the previous exercise, apply k-means clustering to the data looking for three clusters and create the associated silhouette plot.

k) We would like to get a visual impression of the data to see what is going on. Perform a dimensionality reduction by applying a PCA to the data and plot the first two PCs (since we have scaled the data before, it doesn't make a difference if we apply the PCA with or without scaling).

l) Create again the same plot but this time colour the points according to whether they represent a white or red wine. (**Hint**: `wines$colour`).

m) Create the plot again but now colour the points according to the assigned cluster numbers of the two-cluster solution found with k-means in exercise h).

n) Visualize in the same way the three-cluster solution from exercise j).