CSDA1010SUMA18 - LAB EXERCISE 3: Classification Problem

```
library(readr)
library(dplyr)
library(ggplot2)
library(rpart)
library(rpart.plot)
library(Amelia)
library(rattle)
library(RColorBrewer)
library(caret)
```

Nursery Data Set reference and short description

Source: http://archive.ics.uci.edu/ml/datasets/Nursery

```
| class values

not_recom, recommend, very_recom, priority, spec_prior

| attributes

parents:     usual, pretentious, great_pret.
has_nurs:    proper, less_proper, improper, critical, very_crit.
form:        complete, completed, incomplete, foster.
children:    1, 2, 3, more.
housing:     convenient, less_conv, critical.
finance:     convenient, inconv.
social:      nonprob, slightly_prob, problematic.
health:      recommended, priority, not_recom.
```

```
nursery_data <- read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/nursery/nursery.data

# nursery_data <- read.csv(file = "../data/nursery_data.csv")
```

Data Set exploration and cleaning

```
set.seed(77)
dim(nursery_data)
```

```
## [1] 12960     9
```

```
#head(nursery_data)
#str(nursery_data)
```

Coding for categorical variables

Reorder factors

Very often, especially when plotting data, we need to reorder the levels of a factor because the default order is alphabetical. A direct way of reordering, using standard syntax is as follows.

Current levels, need to be corrected to correspont to the dataset description

```
print (levels(nursery_data$parents))
```

```
## [1] "great_pret"  "pretentious" "usual"
```

```
print (levels(nursery_data$has_nurs))
```

```
## [1] "critical"    "improper"    "less_proper" "proper"      "very_crit"
```

```
print (levels(nursery_data$form))
```

```
## [1] "complete"   "completed"  "foster"     "incomplete"
```

```
print (levels(nursery_data$children))
```

```
## [1] "1"    "2"    "3"    "more"
```

```
print (levels(nursery_data$housing))
```

```
## [1] "convenient" "critical"   "less_conv"
```

```
print (levels(nursery_data$finance))
```

```
## [1] "convenient" "inconv"
```

```
print (levels(nursery_data$social))
```

```
## [1] "nonprob"       "problematic"   "slightly_prob"
```

```
print (levels(nursery_data$health))
```

```
## [1] "not_recom"   "priority"    "recommended"
```

```
print (levels(nursery_data$class))
```

```
## [1] "not_recom"  "priority"   "recommend"  "spec_prior" "very_recom"
```

Correction:

```
nursery_data$parents <- factor(nursery_data$parents,levels(nursery_data$parents)[c(3,2,1)])
nursery_data$has_nurs <- factor(nursery_data$has_nurs,levels(nursery_data$has_nurs)[c(4,3,2,1,5)])
nursery_data$form <- factor(nursery_data$form,levels(nursery_data$form)[c(1,2,4,3)])
nursery_data$children <- factor(nursery_data$children,levels(nursery_data$children)[c(1,2,3,4)])
nursery_data$housing <- factor(nursery_data$housing,levels(nursery_data$housing)[c(1,3,2)])
nursery_data$finance <- factor(nursery_data$finance,levels(nursery_data$finance)[c(1,2)])
nursery_data$social <- factor(nursery_data$social,levels(nursery_data$social)[c(1,3,2)])
nursery_data$health <- factor(nursery_data$health,levels(nursery_data$health)[c(1,3,2)])
nursery_data$class <- factor(nursery_data$class,levels(nursery_data$class)[c(1,3,5,2,4)])
```

Corrected levels, now correspond to the dataset description

```
print (levels(nursery_data$parents))
```

```
## [1] "usual"       "pretentious" "great_pret"
```

```r
print (levels(nursery_data$has_nurs))
```

```
## [1] "proper"      "less_proper" "improper"    "critical"    "very_crit"
```

```r
print (levels(nursery_data$form))
```

```
## [1] "complete"   "completed"  "incomplete" "foster"
```

```r
print (levels(nursery_data$children))
```

```
## [1] "1"    "2"    "3"    "more"
```

```r
print (levels(nursery_data$housing))
```

```
## [1] "convenient" "less_conv"  "critical"
```

```r
print (levels(nursery_data$finance))
```

```
## [1] "convenient" "inconv"
```

```r
print (levels(nursery_data$social))
```

```
## [1] "nonprob"       "slightly_prob" "problematic"
```

```r
print (levels(nursery_data$health))
```

```
## [1] "not_recom"   "recommended" "priority"
```

```r
print (levels(nursery_data$class))
```

```
## [1] "not_recom"  "recommend"  "very_recom" "priority"   "spec_prior"
```

Convert to numbers in one step

[Ref] (https://stackoverflow.com/questions/47922184/convert-categorical-variables-to-numeric-in-r)

```r
data <- data.matrix(nursery_data)
head(data)
```

```
##      parents has_nurs form children housing finance social health class
## [1,]       1        1    1        1       1       1      1      2     2
## [2,]       1        1    1        1       1       1      1      3     4
## [3,]       1        1    1        1       1       1      1      1     1
## [4,]       1        1    1        1       1       1      2      2     2
## [5,]       1        1    1        1       1       1      2      3     4
## [6,]       1        1    1        1       1       1      2      1     1
```

Preparing scaled data and split into train and test

```r
index <- sample(1:nrow(data),round(0.75*nrow(data)))
#index <- createDataPartition(y= data$QLT, p=0.5, list = FALSE)
maxs <- apply(data, 2, max)
mins <- apply(data, 2, min)
scaled <- as.data.frame(scale(data, center = mins, scale = maxs - mins))
train_ <- scaled[index,]
test_ <- scaled[-index,]
```

The problem

Distribution of target value in the dataset

The target value class of the wine quality is not equally distributed. The Figure 1 demonstrates the distribution. As we can see, dataset covers mostly medium-quality wines with QLT between 5 and 7 well, low and high quality wines represented poorly.

```
prop.table(table(nursery_data$class))
```

```
##
##    not_recom   recommend  very_recom    priority  spec_prior
## 0.333333333 0.000154321 0.025308642 0.329166667 0.312037037
```

```
ggplot(data = nursery_data, mapping = aes(x = class)) + geom_bar()
```

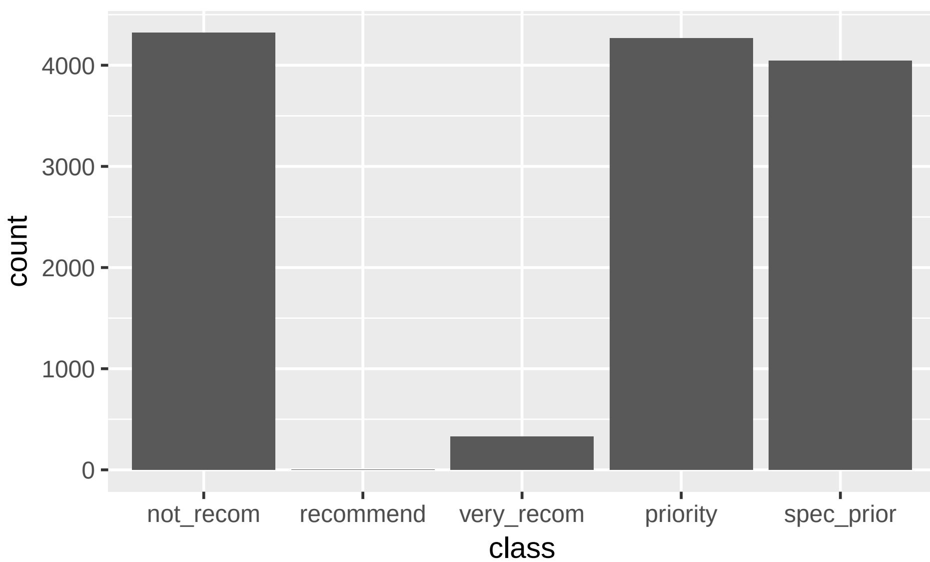


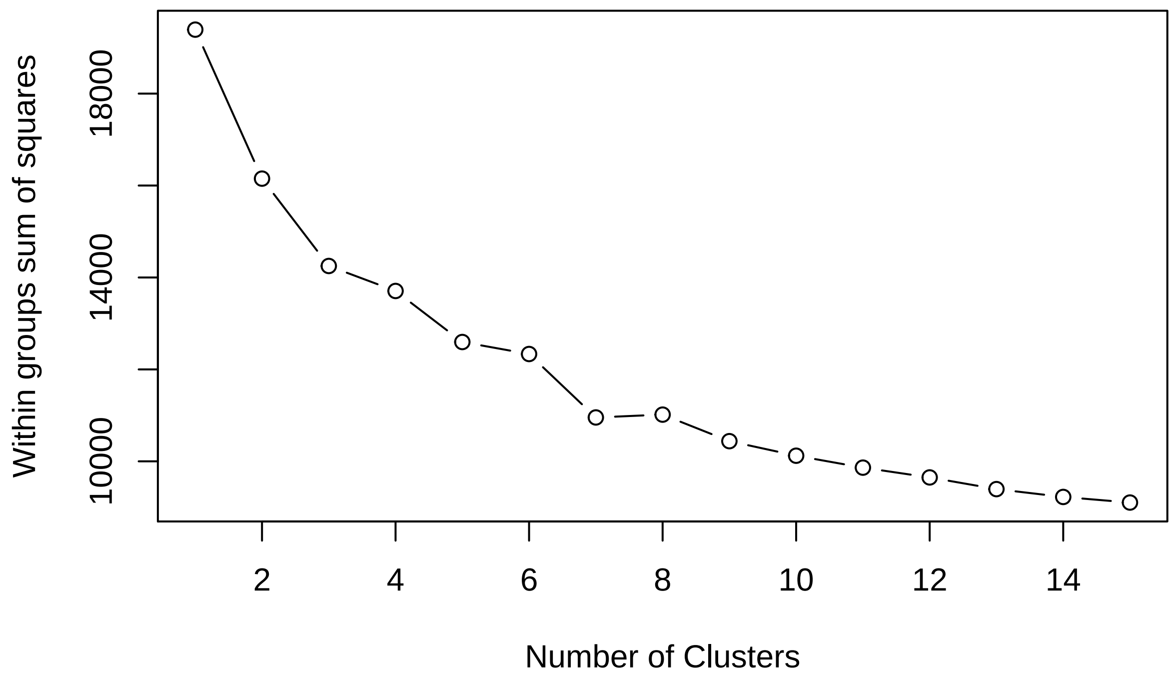
Figure 1: Distribution the Target 'class' Attribute in the Nursery Dataset

Clustering

A fundamental question is how to determine the value of the parameter k. If we looks at the percentage of variance explained as a function of the number of clusters: One should choose a number of clusters so that adding another cluster doesn't give much better modeling of the data. More precisely, if one plots the percentage of variance explained by the clusters against the number of clusters, the first clusters will add much information (explain a lot of variance), but at some point the marginal gain will drop, giving an angle in the graph. The number of clusters is chosen at this point, hence the 'elbow criterion'.

```
wssplot <- function(data, nc=15, seed=1234){
  wss <- (nrow(data)-1)*sum(apply(data,2,var))
  for (i in 2:nc){
    set.seed(seed)
    wss[i] <- sum(kmeans(data, centers=i)$withinss)}
  plot(1:nc, wss, type="b", xlab="Number of Clusters",
       ylab="Within groups sum of squares")}
```

```r
wssplot(scaled, nc=15)
```



Clustering using K-means method

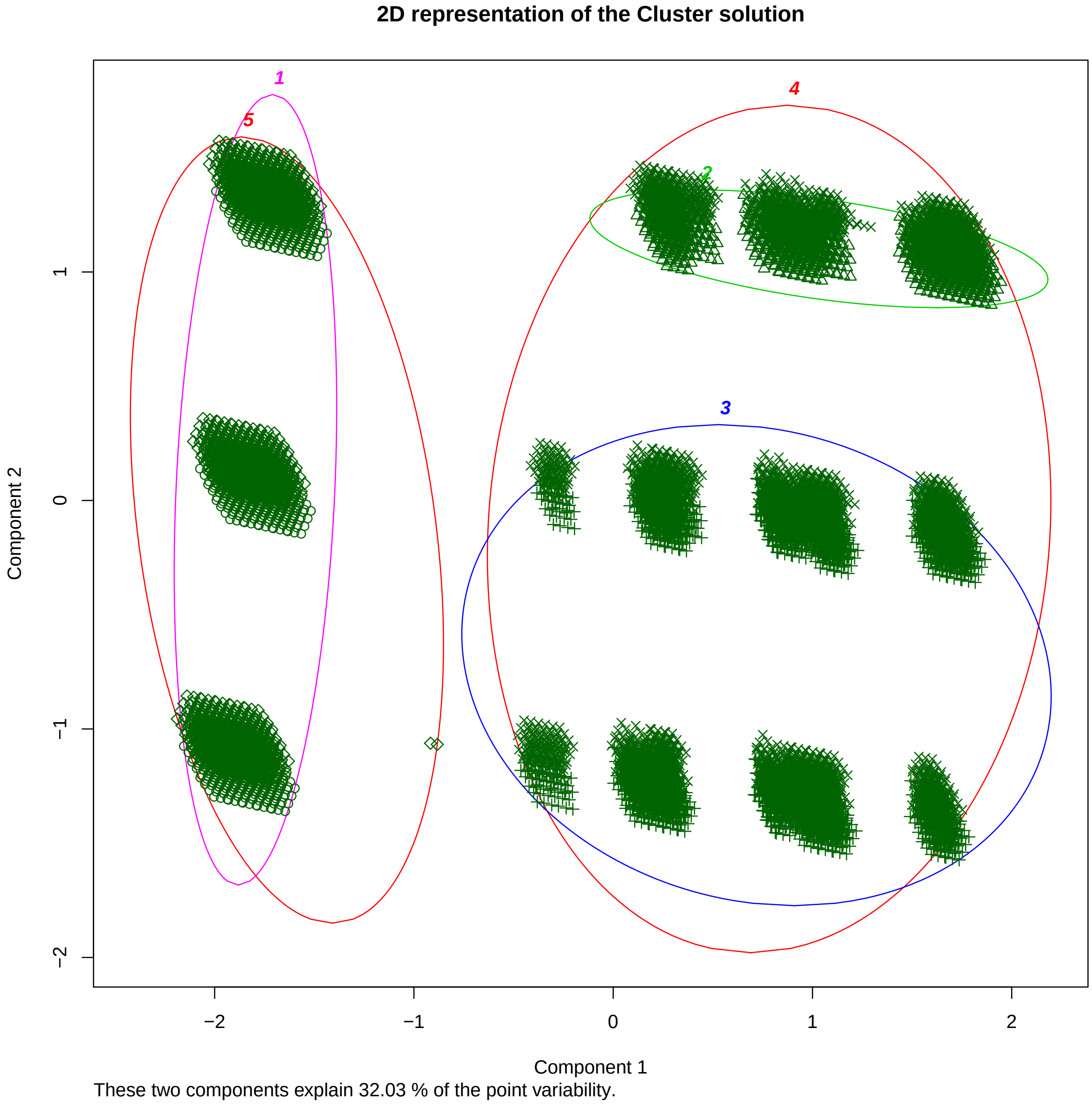
```r
set.seed(420)
clusters_num =5
k.means.fit <- kmeans(scaled, clusters_num,iter.max = 1000)
# attributes(k.means.fit)
k.means.fit$centers
```

```
##     parents  has_nurs      form  children   housing finance    social
## 1 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000       1 0.5000000
## 2 1.0000000 0.5000000 0.5000000 0.5000000 0.5000000       1 0.5000000
## 3 0.2500000 0.5000000 0.5000000 0.5000000 0.5000000       1 0.5000000
## 4 0.5002316 0.5002316 0.5002316 0.5002316 0.5002316       0 0.5001158
## 5 0.4995375 0.4995375 0.4995375 0.4995375 0.4995375       0 0.4997687
##         health        class
## 1 0.0000000000 0.0000000000
## 2 0.7500000000 0.9399305556
## 3 0.7500000000 0.8354166667
## 4 0.7501157943 0.8448355720
## 5 0.0004625347 0.0002312673
```

```r
# plot(k.means.fit$centers[,c("RS","ALC")])
# k.means.fit$cluster
k.means.fit$size
```

```
## [1] 2160 1440 2880 4318 2162
```

```r
library(cluster)
clusplot(scaled, k.means.fit$cluster, main='2D representation of the Cluster solution',
         color=TRUE, shade=FALSE,
         labels=clusters_num, lines=0)
```



Explain clusters

Explain by 'class'

Let's try to explain clusters by the 'class'. Code below builds a matrix whe columns are cluster numbers and rows are target classes.

```
table(nursery_data$class,k.means.fit$cluster)
```

```
##
##                1    2    3    4    5
##   not_recom  2160    0    0    0 2160
##   recommend     0    0    0    0    2
##   very_recom    0    0  110  218    0
##   priority      0  346 1676 2244    0
##   spec_prior    0 1094 1094 1856    0
```

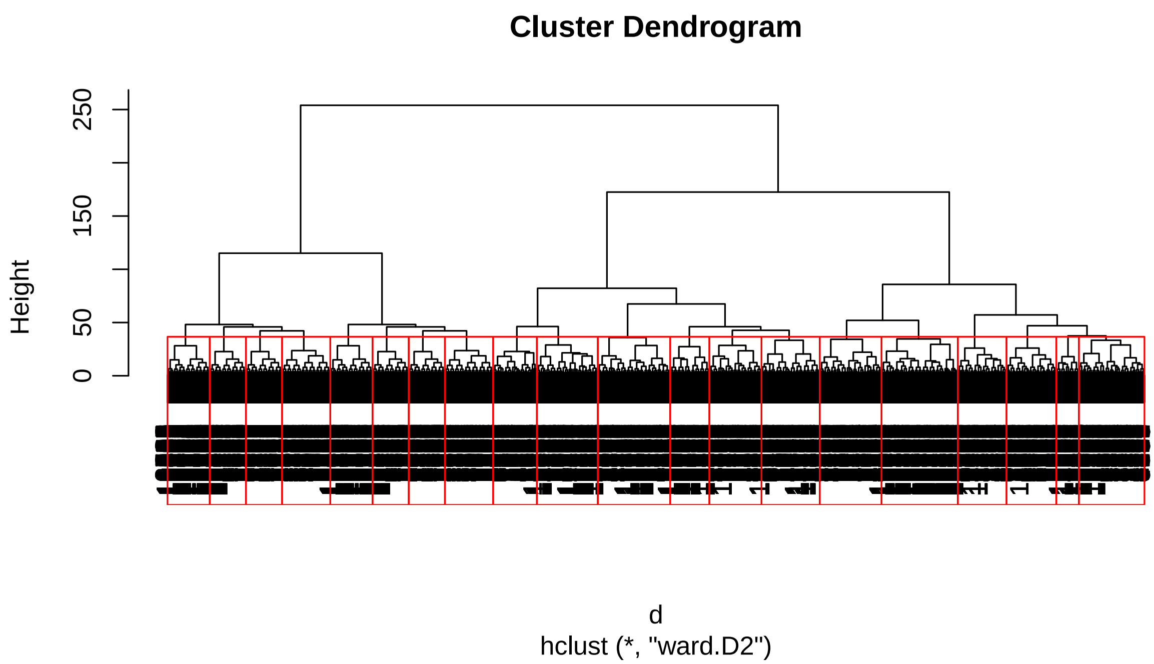
Hierarchical Clustering

Hierarchical methods uses a distance matrix as an input for the clustering algorithm. The choice of an appropriate metric will influence the shape of the clusters, as some element may be close to one another according to one distance and farther away according to another. We use the Euclidean distance as an input for the clustering algorithm ward.2D minimum variance criterion minimizes the total within-cluster variance:

```
d <- dist(scaled, method = "manhattan")
H.fit <- hclust(d, method="ward.D2")
```

The clustering output can be displayed in a dendrogram

```
clusters_num = 20
plot(H.fit)
groups <- cutree(H.fit, k=clusters_num)
rect.hclust(H.fit, k=clusters_num, border="red")
```



The clustering performance can be evaluated with the aid of a confusion matrix as follows. Let's look at the groups that have mixed valued of 'class'. Group 1 contains class 'recommend' and also 'very_recom' and 'priority'. Since our idea is relable rows to 'recommend' to increase it's presense, let's check if there is any justification to do this.

```
table(nursery_data$class,groups)
```

```
##             groups
##                 1    2    3    4    5    6    7    8    9   10   11   12
##   not_recom     0  640    0  480    0  640    0  480    0  480    0  480
##   recommend     2    0    0    0    0    0    0    0    0    0    0    0
##   very_recom  208    0    0    0  100    0    0    0   10    0   10    0
##   priority    650    0  636    0  416    0  400    0  650    0  562    0
##   spec_prior   10    0   10    0  176    0  560    0    0    0    8    0
##             groups
##                13   14   15   16   17   18   19   20
##   not_recom     0    0    0    0    0  560    0  560
##   recommend     0    0    0    0    0    0    0    0
##   very_recom    0    0    0    0    0    0    0    0
##   priority    404    0    8  300    0    0  240    0
##   spec_prior  368  820  800    0 1012    0  280    0
```

Let's find what are the most significant factors that separate group 1 from also mixed group 5. It looks that group a has less vavorable financial situation. We could arbitrary say that it is justified for relable group 1 down grading it to 'recommend' even thogh most of therows were previouslbeen labeled higher.

```
dif <- colMeans(scaled[groups == 1,]) - colMeans(scaled[groups == 5,])
dif <- dif[order(abs(dif), decreasing = T)]
print(dif)
```

```
##     finance        form     housing       class     parents    children
## -1.00000000  0.22487985 -0.12126437 -0.08550262  0.06343100 -0.05783004
##      health    has_nurs      social
```

```
## -0.03638629 -0.02445020  0.01551724
```

Group 5 has significant amount of 'very_recommend' values in addition to 'priority'. Let's find what are the most significant factors that separate group 5 from also mixed group 8. It looks that group a has more vavorable financial situation. We could arbitrary say that it is justified for relable group 1 down grading it to 'recommend' even thogh most of therows were previouslbeen labeled higher.

```
dif <- colMeans(scaled[groups == 5,]) - colMeans(scaled[groups == 9,])
dif <- dif[order(abs(dif), decreasing = T)]
print(dif)
```

```
##     finance     housing        form    has_nurs    children      health
##  1.00000000 -0.56515152 -0.30687219  0.22782011 -0.09110761  0.03878963
##       class     parents      social
##  0.03124453  0.01452093  0.00000000
```