Data\_Mining.R

riserate

Tue May 30 21:24:08 2017

# Task 1  
  
# General Objective  
# To find the best interesting sequential rules  
# Data sets  
# • diab\_trans.data  
# • http://mlr.cs.umass.edu/ml/datasets/Diabetes does not exist  
# https://archive.ics.uci.edu/ml/datasets/diabetes correct location  
  
#Needed libraries  
library(arules)

## Loading required package: Matrix

##   
## Attaching package: 'arules'

## The following objects are masked from 'package:base':  
##   
## abbreviate, write

library(arulesSequences)  
  
#read in the data  
diab.df <- read.csv("diab\_trans.data", header=TRUE, stringsAsFactors = FALSE)  
  
diab.df <- diab.df[complete.cases(diab.df),]  
  
#define the column names  
colnames(diab.df) <- c("ID", "time", "eventID", "value")  
  
#eventID formatting e.g. id\_65 to 65  
eventID\_to\_int <- function(frame) {  
 apply(frame, 1, function(x) strtoi(unlist(strsplit(x[3], "\_"))[2]))  
}  
diab.df$eventID <- eventID\_to\_int(diab.df)  
  
# clean the data for use, throw out data we don't need  
# events where id < 64 we don't want not transactional data items, 65+ we need. We want to create baskets for the events that do have values  
# so that similar values are grouped together during rule mining.  
num\_baskets <- 3  
out.df <- data.frame()  
for (id in unique(diab.df$eventID)) {  
 cat("processing eventID: ", id, "\n")  
 baskets <- list(dim=(num\_baskets+1))  
 if(id >= 65) {  
 next  
 }  
  
 sorted\_values <- sort(diab.df$value[diab.df$eventID == id])  
 len <- length(sorted\_values)  
 step <- len / num\_baskets  
 for (j in 1:(num\_baskets)) {  
 baskets <- append(baskets, sorted\_values[j \* step])  
 }  
 baskets <- append(baskets, len)  
  
 sub <- subset(diab.df, eventID == id)  
 for (row in 1:nrow(sub)) {  
 s <- sub[row,]  
 for (k in 1:(num\_baskets)) {  
 if ((s[4] >= baskets[k]) && (s[4] <= baskets[k+1])) {  
 mod\_sample <- s  
 mod\_sample[3] = 100 \* s[3] + k # change event ID  
 out.df <- rbind(out.df, mod\_sample)  
 break  
 }  
 }  
 }  
}

## processing eventID: 58   
## processing eventID: 33   
## processing eventID: 34   
## processing eventID: 62   
## processing eventID: 48   
## processing eventID: 65   
## processing eventID: 60   
## processing eventID: 35   
## processing eventID: 56   
## processing eventID: 64   
## processing eventID: 61   
## processing eventID: 67   
## processing eventID: 63   
## processing eventID: 57   
## processing eventID: 72   
## processing eventID: 68   
## processing eventID: 69   
## processing eventID: 59   
## processing eventID: 71   
## processing eventID: 66   
## processing eventID: 70   
## processing eventID: 36

unique(out.df$eventID)

## [1] 5801 5803 5802 3303 3302 3301 3402 3403 3401 6202 6203 6201 4801 4803  
## [15] 4802 6003 6001 6002 3501 3502 3503 5601 5602 5603 6402 6401 6403 6101  
## [29] 6102 6103 6302 6301 6303 5702 5701 5703 5903 5901 5902 3601

data.df <- out.df  
data.df <- rbind(data.df, subset(diab.df, eventID > 64))  
  
head(subset(data.df, eventID==5803))

## ID time eventID value  
## 7 1 96852901 5803 216  
## 13 1 96938701 5803 257  
## 18 1 97026721 5803 239  
## 39 1 97293781 5803 259  
## 84 1 97804081 5803 305  
## 118 1 98235781 5803 251

data.df.sorted <- data.df[order(data.df["ID"], data.df["time"]),]  
  
write.table(data.df.sorted, "diab\_baskets.data", sep = ",", row.names = FALSE, col.names = FALSE)  
write.table(data.df.sorted[,-c(4)], "diab\_baskets\_novalues.data", sep = ",", row.names = FALSE, col.names = FALSE)  
  
diabSeq <- read\_baskets(con = "diab\_baskets\_novalues.data", sep =",", info = c("sequenceID","eventID"))  
  
seqParam = new ("SPparameter",support = 0.5, maxsize = 4, mingap=600, maxgap =172800, maxlen = 3 )  
patSeq= cspade(diabSeq,seqParam, control = list(verbose = TRUE, tidLists = FALSE, summary= TRUE))

##   
## parameter specification:  
## support : 0.5  
## maxsize : 4  
## maxlen : 3  
## mingap : 600  
## maxgap : 172800  
##   
## algorithmic control:  
## bfstype : FALSE  
## verbose : TRUE  
## summary : TRUE  
## tidLists : FALSE  
##   
## preprocessing ... 1 partition(s), 0.48 MB [0.057s]  
## mining transactions ... 0.03 MB [0.038s]  
## reading sequences ... [0.089s]  
##   
## total elapsed time: 0.184s

#set the confidence at 80% but can be increased to 90%  
seqRules = ruleInduction(patSeq,confidence = 0.8)  
  
length(seqRules)

## [1] 531

#Summary of the sequence rules  
summary(seqRules)

## set of 531 sequencerules with  
##   
## rule size distribution (lhs + rhs)  
## sizes  
## 2 3   
## 99 432   
##   
## rule length distribution (lhs + rhs)  
## lengths  
## 2 3   
## 137 394   
##   
## summary of quality measures:  
## support confidence lift   
## Min. :0.5000 Min. :0.8000 Min. :0.8949   
## 1st Qu.:0.5758 1st Qu.:0.8265 1st Qu.:0.9483   
## Median :0.6364 Median :0.8545 Median :0.9859   
## Mean :0.6368 Mean :0.8623 Mean :0.9941   
## 3rd Qu.:0.6970 3rd Qu.:0.8936 3rd Qu.:1.0342   
## Max. :0.8333 Max. :0.9821 Max. :1.5178   
##   
## mining info:  
## data ntransactions nsequences support confidence  
## diabSeq 22491 66 0.5 0.8

#inspect the firs 100 rules  
#inspect(head(seqRules,100))  
#inspect all rules 531  
#inspect(seqRules,531)  
  
#top 10 rules  
inspect(head(sort(seqRules, by=c("confidence", "support")),10))

## lhs rhs support confidence lift   
## 1 <{5803}> => <{6203}> 0.8333333 0.9821429 1.117611   
## 2 <{5803}> => <{5803}> 0.8333333 0.9821429 1.157526   
## 3 <{5803}> => <{5802}> 0.8333333 0.9821429 1.098668   
## 4 <{5803},   
## {6001}> => <{5802}> 0.6969697 0.9787234 1.094843   
## 5 <{3302},   
## {6001}> => <{5802}> 0.6060606 0.9756098 1.091360   
## 6 <{3303},   
## {3303}> => <{3302}> 0.5151515 0.9714286 1.144898   
## 7 <{5801},   
## {3303}> => <{3302}> 0.5000000 0.9705882 1.143908   
## 8 <{5803},   
## {3303}> => <{3302}> 0.5000000 0.9705882 1.143908   
## 9 <{5803}> => <{5801}> 0.8181818 0.9642857 1.078692   
## 10 <{6001}> => <{5802}> 0.8030303 0.9636364 1.077966   
##

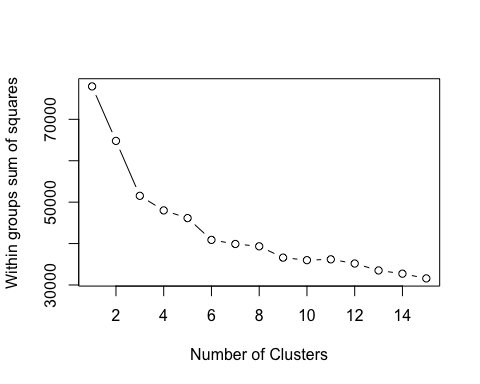
inspect(head(sort(seqRules, by=c("support", "confidence")),10))

## lhs rhs support confidence lift   
## 1 <{5803}> => <{6203}> 0.8333333 0.9821429 1.117611   
## 2 <{5803}> => <{5803}> 0.8333333 0.9821429 1.157526   
## 3 <{5803}> => <{5802}> 0.8333333 0.9821429 1.098668   
## 4 <{6203}> => <{5802}> 0.8333333 0.9482759 1.060783   
## 5 <{6201}> => <{6203}> 0.8333333 0.9322034 1.060783   
## 6 <{5802}> => <{6201}> 0.8333333 0.9322034 1.042804   
## 7 <{6201}> => <{5803}> 0.8333333 0.9322034 1.098668   
## 8 <{5802}> => <{5802}> 0.8333333 0.9322034 1.042804   
## 9 <{5802}> => <{5801}> 0.8333333 0.9322034 1.042804   
## 10 <{6201}> => <{5801}> 0.8333333 0.9322034 1.042804   
##

# Task 2  
# To find the best classifier for a selected dataset.  
# Datasets: • Wine Quality (two sets) - http://archive.ics.uci.edu/ml/datasets/Wine+Quality  
  
# Algorithm used k-means  
  
wine <- read.csv('winequality-white.csv', sep=';')  
wine <- rbind(wine, read.csv('winequality-red.csv', sep=';'))  
summary(wine)

## fixed.acidity volatile.acidity citric.acid residual.sugar   
## Min. : 3.800 Min. :0.0800 Min. :0.0000 Min. : 0.600   
## 1st Qu.: 6.400 1st Qu.:0.2300 1st Qu.:0.2500 1st Qu.: 1.800   
## Median : 7.000 Median :0.2900 Median :0.3100 Median : 3.000   
## Mean : 7.215 Mean :0.3397 Mean :0.3186 Mean : 5.443   
## 3rd Qu.: 7.700 3rd Qu.:0.4000 3rd Qu.:0.3900 3rd Qu.: 8.100   
## Max. :15.900 Max. :1.5800 Max. :1.6600 Max. :65.800   
## chlorides free.sulfur.dioxide total.sulfur.dioxide  
## Min. :0.00900 Min. : 1.00 Min. : 6.0   
## 1st Qu.:0.03800 1st Qu.: 17.00 1st Qu.: 77.0   
## Median :0.04700 Median : 29.00 Median :118.0   
## Mean :0.05603 Mean : 30.53 Mean :115.7   
## 3rd Qu.:0.06500 3rd Qu.: 41.00 3rd Qu.:156.0   
## Max. :0.61100 Max. :289.00 Max. :440.0   
## density pH sulphates alcohol   
## Min. :0.9871 Min. :2.720 Min. :0.2200 Min. : 8.00   
## 1st Qu.:0.9923 1st Qu.:3.110 1st Qu.:0.4300 1st Qu.: 9.50   
## Median :0.9949 Median :3.210 Median :0.5100 Median :10.30   
## Mean :0.9947 Mean :3.219 Mean :0.5313 Mean :10.49   
## 3rd Qu.:0.9970 3rd Qu.:3.320 3rd Qu.:0.6000 3rd Qu.:11.30   
## Max. :1.0390 Max. :4.010 Max. :2.0000 Max. :14.90   
## quality   
## Min. :3.000   
## 1st Qu.:5.000   
## Median :6.000   
## Mean :5.818   
## 3rd Qu.:6.000   
## Max. :9.000

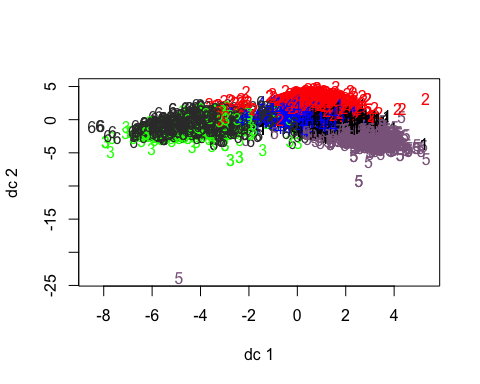
wine<-scale(wine)  
wss<-(nrow(wine)-1)\*sum(apply(wine,2,var))  
for(i in 1:15) wss[i]<-sum(kmeans(wine,centers=i)$withinss)  
plot(1:15,wss,type='b',xlab="Number of Clusters",ylab='Within groups sum of squares')



fit1 <- kmeans(wine,6)  
fit2 <- kmeans(wine,8)  
  
table(fit1$cluster)

##   
## 1 2 3 4 5 6   
## 1201 1431 611 1151 1120 983

library(fpc)  
plotcluster(wine, fit1$cluster)



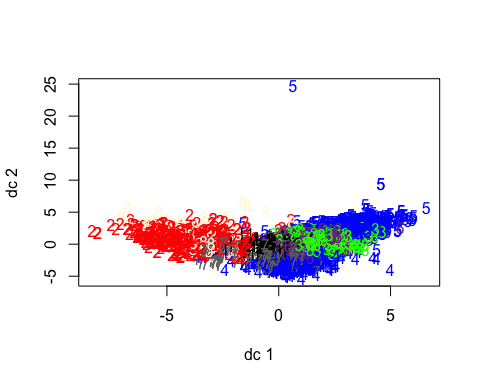
aggregate(wine,by=list(fit1$cluster),FUN=mean)

## Group.1 fixed.acidity volatile.acidity citric.acid residual.sugar  
## 1 1 -0.45598976 -0.4458078 -0.01213849 0.1093561  
## 2 2 -0.53328814 -0.3652544 0.01604170 -0.4171189  
## 3 3 2.04103581 0.4522594 1.02337998 -0.5689049  
## 4 4 0.04542406 -0.4219367 0.12575069 -0.4446513  
## 5 5 -0.08694450 -0.3447580 0.38733316 1.7316431  
## 6 6 0.11068250 1.6821383 -1.23317873 -0.6251122  
## chlorides free.sulfur.dioxide total.sulfur.dioxide density  
## 1 -0.09232645 0.77011629 0.86903917 0.02170063  
## 2 -0.56386829 0.03664529 -0.06179296 -1.19341934  
## 3 1.28295398 -0.89857769 -1.28913057 0.97762651  
## 4 -0.30356794 -0.49862273 -0.08274334 -0.56448819  
## 5 -0.17725139 0.82407929 0.92969849 1.10134434  
## 6 0.69361441 -0.79081602 -1.13291582 0.50926865  
## pH sulphates alcohol quality  
## 1 0.28285317 -0.2039170 -0.46146411 -0.21441267  
## 2 0.13440724 -0.2244981 1.23475866 1.00181910  
## 3 -0.06345638 1.4537452 0.06995071 0.08615958  
## 4 -0.62001897 -0.4102149 0.02123665 -0.45661633  
## 5 -0.62227849 -0.2269884 -0.91660394 -0.16623252  
## 6 0.93318635 0.4112993 -0.25768881 -0.52593205

mydata <- data.frame(wine, fit1$cluster)  
  
  
table(fit2$cluster)

##   
## 1 2 3 4 5 6 7 8   
## 902 964 663 1017 855 728 795 573

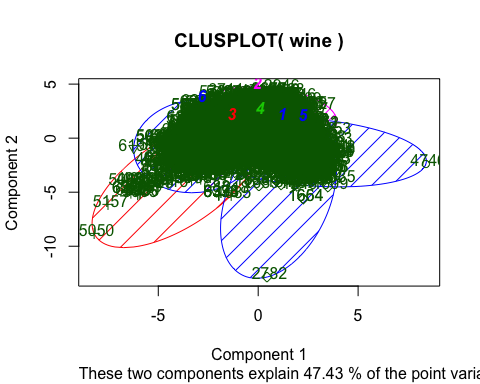
library(fpc)  
plotcluster(wine, fit2$cluster)



aggregate(wine,by=list(fit2$cluster),FUN=mean)

## Group.1 fixed.acidity volatile.acidity citric.acid residual.sugar  
## 1 1 0.03514077 -0.3534208 0.17892453 -0.5827393  
## 2 2 0.13878160 1.7009182 -1.22197724 -0.6254801  
## 3 3 -0.28695564 -0.4573682 -0.09272669 0.9009838  
## 4 4 -0.31996654 -0.3048361 0.08458060 -0.4065815  
## 5 5 -0.02660248 -0.3575088 0.49070570 1.8531411  
## 6 6 -0.43138580 -0.3588646 0.04475807 0.2128010  
## 7 7 -0.67763941 -0.5162849 -0.07533610 -0.5922140  
## 8 8 2.13907930 0.4707222 1.04678954 -0.5651132  
## chlorides free.sulfur.dioxide total.sulfur.dioxide density  
## 1 -0.1867242321 -0.534563411 -0.021038624 -0.4887800  
## 2 0.7115385087 -0.794890294 -1.144487533 0.5376254  
## 3 -0.3525359478 0.116012102 0.260269784 0.1960253  
## 4 -0.5909606061 0.005221288 -0.054194232 -1.3214028  
## 5 -0.1375438697 0.861035751 1.012978538 1.2505641  
## 6 -0.0004568368 1.287659043 1.230887540 0.1358937  
## 7 -0.4046401053 0.024900911 -0.007476079 -0.7691930  
## 8 1.3208758714 -0.920023049 -1.311378376 1.0119653  
## pH sulphates alcohol quality  
## 1 -0.52952999 -0.3798457 -0.22547421 -0.7187937  
## 2 0.91739501 0.4190161 -0.29116965 -0.5487125  
## 3 -0.26440483 -0.5604280 -0.12650229 0.2442545  
## 4 -0.41141830 -0.4977008 1.41352492 0.8892122  
## 5 -0.67590134 -0.1570740 -1.04941105 -0.2098928  
## 6 0.00189131 -0.2109588 -0.64601495 -0.4479560  
## 7 1.03454702 0.3368681 0.56962140 0.6991691  
## 8 -0.10290758 1.4598271 0.07866976 0.1060595

mydata <- data.frame(wine, fit2$cluster)  
  
library(cluster)  
clusplot(wine, fit1$cluster, color=TRUE, shade=TRUE,labels=2, lines=0)



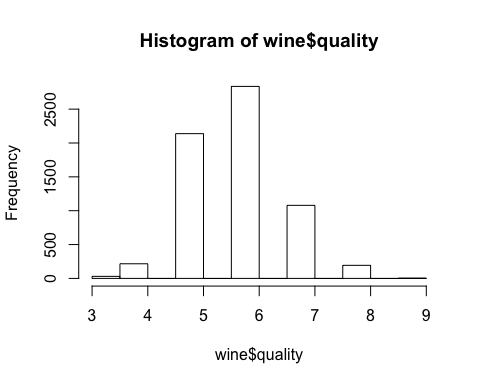
cluster.stats(fit1$cluster, fit2$cluster)

## Warning in as.dist.default(d): non-square matrix

## Warning in as.matrix.dist(d): number of items to replace is not a multiple  
## of replacement length

## $n  
## [1] 6497  
##   
## $cluster.number  
## [1] 8  
##   
## $cluster.size  
## [1] 902 964 663 1017 855 728 795 573  
##   
## $min.cluster.size  
## [1] 573  
##   
## $noisen  
## [1] 0  
##   
## $diameter  
## [1] 6 6 6 6 6 6 6 6  
##   
## $average.distance  
## [1] 3.378670 3.375786 3.404424 3.405727 3.388405 3.374193 3.383813 3.404209  
##   
## $median.distance  
## [1] 4 4 4 4 4 4 4 4  
##   
## $separation  
## [1] 1 1 1 1 1 1 1 1  
##   
## $average.toother  
## [1] 3.380231 3.388016 3.390936 3.386425 3.381273 3.385318 3.383641 3.386469  
##   
## $separation.matrix  
## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]  
## [1,] 0 1 1 1 1 1 1 1  
## [2,] 1 0 1 1 1 1 1 1  
## [3,] 1 1 0 1 1 1 1 1  
## [4,] 1 1 1 0 1 1 1 1  
## [5,] 1 1 1 1 0 1 1 1  
## [6,] 1 1 1 1 1 0 1 1  
## [7,] 1 1 1 1 1 1 0 1  
## [8,] 1 1 1 1 1 1 1 0  
##   
## $ave.between.matrix  
## [,1] [,2] [,3] [,4] [,5] [,6] [,7]  
## [1,] 0.000000 3.382235 3.387517 3.379801 3.380345 3.375618 3.379042  
## [2,] 3.382235 0.000000 3.386811 3.386684 3.378852 3.401853 3.398753  
## [3,] 3.387517 3.386811 0.000000 3.404717 3.390116 3.387694 3.384892  
## [4,] 3.379801 3.386684 3.404717 0.000000 3.387500 3.383150 3.382216  
## [5,] 3.380345 3.378852 3.390116 3.387500 0.000000 3.374124 3.375247  
## [6,] 3.375618 3.401853 3.387694 3.383150 3.374124 0.000000 3.375598  
## [7,] 3.379042 3.398753 3.384892 3.382216 3.375247 3.375598 0.000000  
## [8,] 3.376536 3.382076 3.392525 3.383647 3.382971 3.404055 3.389287  
## [,8]  
## [1,] 3.376536  
## [2,] 3.382076  
## [3,] 3.392525  
## [4,] 3.383647  
## [5,] 3.382971  
## [6,] 3.404055  
## [7,] 3.389287  
## [8,] 0.000000  
##   
## $average.between  
## [1] 3.385139  
##   
## $average.within  
## [1] 3.388416  
##   
## $n.between  
## [1] 18386444  
##   
## $n.within  
## [1] 2715812  
##   
## $max.diameter  
## [1] 6  
##   
## $min.separation  
## [1] 1  
##   
## $within.cluster.ss  
## [1] 47117.17  
##   
## $clus.avg.silwidths  
## 1 2 3 4 5 6   
## -0.09862469 -0.03711766 -0.09831683 -0.09583828 -0.09954665 -0.09524632   
## 7 8   
## -0.09301316 -0.05059970   
##   
## $avg.silwidth  
## [1] -0.08385151  
##   
## $g2  
## NULL  
##   
## $g3  
## NULL  
##   
## $pearsongamma  
## [1] -0.0006295562  
##   
## $dunn  
## [1] 0.1666667  
##   
## $dunn2  
## [1] 0.9907208  
##   
## $entropy  
## [1] 2.063772  
##   
## $wb.ratio  
## [1] 1.000968  
##   
## $ch  
## [1] -926.7147  
##   
## $cwidegap  
## [1] 1 1 1 1 1 1 1 1  
##   
## $widestgap  
## [1] 1  
##   
## $sindex  
## [1] 1  
##   
## $corrected.rand  
## NULL  
##   
## $vi  
## NULL

# Task 2 Alterative Regression Trees and Models  
wine <- read.csv('winequality-white.csv', sep=';', header = TRUE)  
wine <- rbind(wine, read.csv('winequality-red.csv', sep=';', header = TRUE))  
hist(wine$quality)



# The wine quality values appear to follow a fairly normal, bell-shaped distribution, centered around a value of six.  
summary(wine)

## fixed.acidity volatile.acidity citric.acid residual.sugar   
## Min. : 3.800 Min. :0.0800 Min. :0.0000 Min. : 0.600   
## 1st Qu.: 6.400 1st Qu.:0.2300 1st Qu.:0.2500 1st Qu.: 1.800   
## Median : 7.000 Median :0.2900 Median :0.3100 Median : 3.000   
## Mean : 7.215 Mean :0.3397 Mean :0.3186 Mean : 5.443   
## 3rd Qu.: 7.700 3rd Qu.:0.4000 3rd Qu.:0.3900 3rd Qu.: 8.100   
## Max. :15.900 Max. :1.5800 Max. :1.6600 Max. :65.800   
## chlorides free.sulfur.dioxide total.sulfur.dioxide  
## Min. :0.00900 Min. : 1.00 Min. : 6.0   
## 1st Qu.:0.03800 1st Qu.: 17.00 1st Qu.: 77.0   
## Median :0.04700 Median : 29.00 Median :118.0   
## Mean :0.05603 Mean : 30.53 Mean :115.7   
## 3rd Qu.:0.06500 3rd Qu.: 41.00 3rd Qu.:156.0   
## Max. :0.61100 Max. :289.00 Max. :440.0   
## density pH sulphates alcohol   
## Min. :0.9871 Min. :2.720 Min. :0.2200 Min. : 8.00   
## 1st Qu.:0.9923 1st Qu.:3.110 1st Qu.:0.4300 1st Qu.: 9.50   
## Median :0.9949 Median :3.210 Median :0.5100 Median :10.30   
## Mean :0.9947 Mean :3.219 Mean :0.5313 Mean :10.49   
## 3rd Qu.:0.9970 3rd Qu.:3.320 3rd Qu.:0.6000 3rd Qu.:11.30   
## Max. :1.0390 Max. :4.010 Max. :2.0000 Max. :14.90   
## quality   
## Min. :3.000   
## 1st Qu.:5.000   
## Median :6.000   
## Mean :5.818   
## 3rd Qu.:6.000   
## Max. :9.000

#partition the data  
wine\_train <- wine[1:3248, ]  
wine\_test <- wine[3249:6497, ]  
  
m.rpart <- rpart::rpart(quality ~ ., data = wine\_train)  
m.rpart

## n= 3248   
##   
## node), split, n, deviance, yval  
## \* denotes terminal node  
##   
## 1) root 3248 2697.47300 5.848522   
## 2) alcohol< 10.85 2247 1387.27400 5.595461   
## 4) volatile.acidity>=0.2525 1172 604.76450 5.368601   
## 8) volatile.acidity>=0.4225 172 84.94767 4.982558 \*  
## 9) volatile.acidity< 0.4225 1000 489.77500 5.435000 \*  
## 5) volatile.acidity< 0.2525 1075 656.43160 5.842791 \*  
## 3) alcohol>=10.85 1001 843.28470 6.416583   
## 6) free.sulfur.dioxide< 11.5 77 80.70130 5.415584 \*  
## 7) free.sulfur.dioxide>=11.5 924 679.00000 6.500000   
## 14) alcohol< 11.85 495 362.46460 6.313131 \*  
## 15) alcohol>=11.85 429 279.30540 6.715618 \*

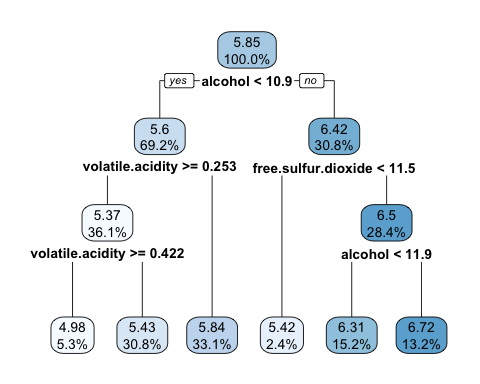
summary(m.rpart)

## Call:  
## rpart::rpart(formula = quality ~ ., data = wine\_train)  
## n= 3248   
##   
## CP nsplit rel error xerror xstd  
## 1 0.17309330 0 1.0000000 1.0009756 0.02583728  
## 2 0.04673914 1 0.8269067 0.8284360 0.02410107  
## 3 0.03098582 2 0.7801676 0.7908229 0.02375595  
## 4 0.01380180 3 0.7491817 0.7603411 0.02229241  
## 5 0.01113703 4 0.7353799 0.7515058 0.02204882  
## 6 0.01000000 5 0.7242429 0.7434067 0.02183699  
##   
## Variable importance  
## alcohol density volatile.acidity   
## 39 22 13   
## chlorides free.sulfur.dioxide total.sulfur.dioxide   
## 11 6 5   
## citric.acid pH sulphates   
## 1 1 1   
## fixed.acidity   
## 1   
##   
## Node number 1: 3248 observations, complexity param=0.1730933  
## mean=5.848522, MSE=0.8305027   
## left son=2 (2247 obs) right son=3 (1001 obs)  
## Primary splits:  
## alcohol < 10.85 to the left, improve=0.17309330, (0 missing)  
## density < 0.992265 to the right, improve=0.11069620, (0 missing)  
## chlorides < 0.0395 to the right, improve=0.07529120, (0 missing)  
## free.sulfur.dioxide < 11.75 to the left, improve=0.03771371, (0 missing)  
## total.sulfur.dioxide < 153.5 to the right, improve=0.03310689, (0 missing)  
## Surrogate splits:  
## density < 0.992065 to the right, agree=0.870, adj=0.579, (0 split)  
## chlorides < 0.0355 to the right, agree=0.782, adj=0.292, (0 split)  
## total.sulfur.dioxide < 102.5 to the right, agree=0.716, adj=0.077, (0 split)  
## sulphates < 0.335 to the right, agree=0.698, adj=0.020, (0 split)  
## fixed.acidity < 5.25 to the right, agree=0.695, adj=0.010, (0 split)  
##   
## Node number 2: 2247 observations, complexity param=0.04673914  
## mean=5.595461, MSE=0.6173893   
## left son=4 (1172 obs) right son=5 (1075 obs)  
## Primary splits:  
## volatile.acidity < 0.2525 to the right, improve=0.09088154, (0 missing)  
## free.sulfur.dioxide < 13.5 to the left, improve=0.04071408, (0 missing)  
## alcohol < 10.15 to the left, improve=0.03109034, (0 missing)  
## citric.acid < 0.205 to the left, improve=0.02987939, (0 missing)  
## pH < 3.325 to the left, improve=0.02087396, (0 missing)  
## Surrogate splits:  
## total.sulfur.dioxide < 153.5 to the right, agree=0.587, adj=0.137, (0 split)  
## citric.acid < 0.265 to the left, agree=0.587, adj=0.136, (0 split)  
## pH < 3.275 to the left, agree=0.580, adj=0.123, (0 split)  
## alcohol < 10.05 to the left, agree=0.575, adj=0.111, (0 split)  
## density < 0.994235 to the right, agree=0.550, adj=0.060, (0 split)  
##   
## Node number 3: 1001 observations, complexity param=0.03098582  
## mean=6.416583, MSE=0.8424423   
## left son=6 (77 obs) right son=7 (924 obs)  
## Primary splits:  
## free.sulfur.dioxide < 11.5 to the left, improve=0.09911648, (0 missing)  
## alcohol < 11.85 to the left, improve=0.05615603, (0 missing)  
## fixed.acidity < 8.05 to the right, improve=0.04155984, (0 missing)  
## pH < 3.245 to the left, improve=0.03745208, (0 missing)  
## total.sulfur.dioxide < 78.5 to the left, improve=0.03096356, (0 missing)  
## Surrogate splits:  
## total.sulfur.dioxide < 48.5 to the left, agree=0.932, adj=0.117, (0 split)  
##   
## Node number 4: 1172 observations, complexity param=0.01113703  
## mean=5.368601, MSE=0.5160107   
## left son=8 (172 obs) right son=9 (1000 obs)  
## Primary splits:  
## volatile.acidity < 0.4225 to the right, improve=0.04967525, (0 missing)  
## free.sulfur.dioxide < 22.5 to the left, improve=0.04669360, (0 missing)  
## alcohol < 10.25 to the left, improve=0.02522470, (0 missing)  
## chlorides < 0.0495 to the right, improve=0.02450074, (0 missing)  
## total.sulfur.dioxide < 86.5 to the left, improve=0.02248320, (0 missing)  
## Surrogate splits:  
## density < 0.9912 to the left, agree=0.858, adj=0.029, (0 split)  
## fixed.acidity < 9.85 to the right, agree=0.857, adj=0.023, (0 split)  
## citric.acid < 0.11 to the left, agree=0.856, adj=0.017, (0 split)  
## chlorides < 0.206 to the right, agree=0.854, adj=0.006, (0 split)  
## alcohol < 8.55 to the left, agree=0.854, adj=0.006, (0 split)  
##   
## Node number 5: 1075 observations  
## mean=5.842791, MSE=0.6106341   
##   
## Node number 6: 77 observations  
## mean=5.415584, MSE=1.048069   
##   
## Node number 7: 924 observations, complexity param=0.0138018  
## mean=6.5, MSE=0.7348485   
## left son=14 (495 obs) right son=15 (429 obs)  
## Primary splits:  
## alcohol < 11.85 to the left, improve=0.05483062, (0 missing)  
## fixed.acidity < 8.05 to the right, improve=0.04720628, (0 missing)  
## pH < 3.245 to the left, improve=0.04398018, (0 missing)  
## residual.sugar < 1.175 to the left, improve=0.02526233, (0 missing)  
## density < 0.990005 to the right, improve=0.02499349, (0 missing)  
## Surrogate splits:  
## density < 0.99111 to the right, agree=0.713, adj=0.382, (0 split)  
## volatile.acidity < 0.305 to the left, agree=0.661, adj=0.270, (0 split)  
## chlorides < 0.0355 to the right, agree=0.623, adj=0.189, (0 split)  
## residual.sugar < 1.85 to the left, agree=0.562, adj=0.056, (0 split)  
## fixed.acidity < 6.175 to the right, agree=0.558, adj=0.049, (0 split)  
##   
## Node number 8: 172 observations  
## mean=4.982558, MSE=0.4938818   
##   
## Node number 9: 1000 observations  
## mean=5.435, MSE=0.489775   
##   
## Node number 14: 495 observations  
## mean=6.313131, MSE=0.7322518   
##   
## Node number 15: 429 observations  
## mean=6.715618, MSE=0.6510614

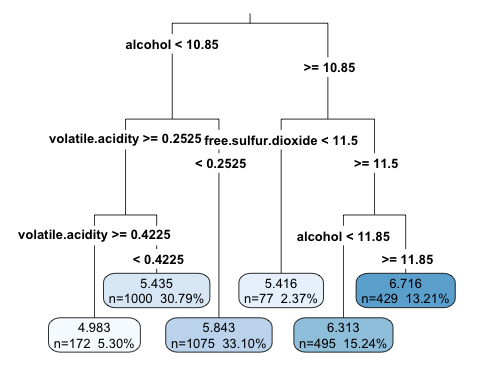
library(rpart.plot)

## Loading required package: rpart

rpart.plot(m.rpart, digits = 3)



###  
# Because alcohol was used first in the tree, it is the single most important predictor of wine quality.  
###  
#adjustments to the diagram  
rpart.plot(m.rpart, digits = 4, fallen.leaves = TRUE, type = 3, extra = 101)



p.rpart <- predict(m.rpart, wine\_test)  
  
#predicted values vs. actual values  
summary(p.rpart)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 4.983 4.983 5.435 5.711 6.313 6.716

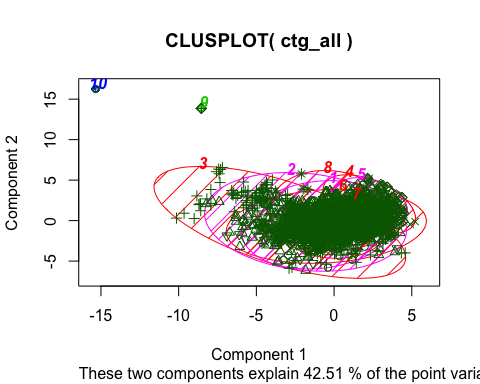
summary(wine\_test$quality)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 3.000 5.000 6.000 5.788 6.000 8.000

#Task 3  
# Aim: Determine the best grouping according to the given evaluation method.  
# • Data Description: http://archive.ics.uci.edu/ml/datasets/Cardiotocography  
# • Additional constraints:  
# – In the analysis the fields: Class and NSP should not be taken into consideration  
# – Maximum number of groups 15 (the references grouping has 10 groups).  
# – The reference grouping is defined by the Class attribute.  
# – Minimum 10 tests are required.  
  
library(readxl)  
CTG <- read\_excel("~/Git-Repos/r-code/Data Mining/CTG-raw.xls")  
  
ctg\_noClass <- read.csv("cardioto\_noClass\_corr.csv",row.names = 1)  
ctg\_all <- read.csv("cardioto\_all\_corr.csv",row.names = 1)  
  
distC = dist(ctg\_noClass)  
#10 Tests  
card.kmeans = kmeans(distC,10)  
res3 = table(ctg\_all$CLASS,card.kmeans$cluster)  
res3

##   
## 1 2 3 4 5 6 7 8 9 10  
## 0 0 0 0 0 0 3 0 0 0 0  
## 1 38 11 25 3 0 0 133 23 94 57  
## 2 129 80 0 27 2 0 29 43 80 189  
## 3 18 5 0 0 0 0 2 0 21 7  
## 4 6 20 3 22 4 0 0 7 0 19  
## 5 7 6 1 0 0 0 8 31 0 19  
## 6 105 119 0 78 9 0 0 0 12 9  
## 7 91 73 0 54 5 0 1 0 23 5  
## 8 2 8 0 60 29 7 0 0 1 0  
## 9 4 2 44 1 0 0 4 11 3 0  
## 10 25 8 49 2 0 0 62 42 3 6

# The Vector Matrix correlates with  
# 1= A calm sleep  
# 2= B REM sleep  
# 3= C calm vigilance  
# 4= D active vigilance  
# 5= SH shift pattern (A or Susp with shifts)  
# 6= AD accelerative/decelerative pattern (stress situation)  
# 7= DE decelerative pattern (vagal stimulation)  
# 8= LD largely decelerative pattern  
# 9= FS flat-sinusoidal pattern (pathological state)  
# 10= SUSP suspect pattern  
  
library(fpc)  
library(cluster)  
distance <- dist(ctg\_noClass, method = "binary")  
  
fit <- kmeans(distance, centers=10)  
res = table(ctg\_all$CLASS, fit$cluster )  
  
fit <- kmeans(distance, centers=12)  
res = table(ctg\_all$CLASS, fit$cluster )  
  
fit <- kmeans(distance, centers=8)  
res = table(ctg\_all$CLASS, fit$cluster )  
  
fit <- kmeans(distance, centers=14)  
res = table(ctg\_all$CLASS, fit$cluster )  
  
fit <- kmeans(distance, centers=14, iter.max=50)  
res = table(ctg\_all$CLASS, fit$cluster )  
  
fit <- kmeans(distance, centers=14, iter.max=500)  
res = table(ctg\_all$CLASS, fit$cluster )  
  
fit <- kmeans(distance, centers=14, iter.max=500, nstart=3)  
res = table(ctg\_all$CLASS, fit$cluster )  
  
fit <- kmeans(distance, centers=14, iter.max=500, nstart=5)  
res = table(ctg\_all$CLASS, fit$cluster )  
  
clusterTree <- hclust(distance)  
clusters <- cutree(hclust(distance), 10)  
res = table(ctg\_all$CLASS, clusters)  
library(cluster)  
clusplot(ctg\_all, clusters, color=TRUE, shade=TRUE,labels=5, lines=0)



cluster.stats(distance, ctg\_all$CLASS, fit$cluster)

## Warning in cluster.stats(distance, ctg\_all$CLASS, fit$cluster): clustering  
## renumbered because maximum != number of clusters

## $n  
## [1] 2129  
##   
## $cluster.number  
## [1] 11  
##   
## $cluster.size  
## [1] 3 384 579 53 81 72 332 252 107 69 197  
##   
## $min.cluster.size  
## [1] 3  
##   
## $noisen  
## [1] 0  
##   
## $diameter  
## [1] 1.0000000 0.4736842 0.4210526 0.4210526 0.3529412 0.4210526 0.3684211  
## [8] 0.4000000 0.4210526 0.4375000 0.4444444  
##   
## $average.distance  
## [1] 0.6666667 0.1712228 0.1336844 0.1711275 0.1329437 0.1585710 0.1202390  
## [8] 0.1543760 0.1620661 0.1651052 0.1723524  
##   
## $median.distance  
## [1] 1.0000000 0.1764706 0.1250000 0.1875000 0.1250000 0.1666667 0.1176471  
## [8] 0.1666667 0.1666667 0.1538462 0.1875000  
##   
## $separation  
## [1] 0.5789474 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000  
## [8] 0.0000000 0.0000000 0.0000000 0.0000000  
##   
## $average.toother  
## [1] 0.9112418 0.2162964 0.2068112 0.2123065 0.2050293 0.2048939 0.2113812  
## [8] 0.2195964 0.2709547 0.2712032 0.2489187  
##   
## $separation.matrix  
## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 0.0000000 0.6470588 0.6470588 0.6666667 0.7000000 0.63157895  
## [2,] 0.6470588 0.0000000 0.0000000 0.0000000 0.0000000 0.00000000  
## [3,] 0.6470588 0.0000000 0.0000000 0.0000000 0.0000000 0.00000000  
## [4,] 0.6666667 0.0000000 0.0000000 0.0000000 0.0000000 0.00000000  
## [5,] 0.7000000 0.0000000 0.0000000 0.0000000 0.0000000 0.00000000  
## [6,] 0.6315789 0.0000000 0.0000000 0.0000000 0.0000000 0.00000000  
## [7,] 0.6315789 0.0000000 0.0000000 0.0000000 0.0000000 0.00000000  
## [8,] 0.5789474 0.0000000 0.0000000 0.0000000 0.0000000 0.00000000  
## [9,] 0.5789474 0.0000000 0.0000000 0.0000000 0.0000000 0.05555556  
## [10,] 0.6250000 0.0000000 0.0000000 0.0000000 0.1176471 0.00000000  
## [11,] 0.6111111 0.0000000 0.0000000 0.0000000 0.0625000 0.00000000  
## [,7] [,8] [,9] [,10] [,11]  
## [1,] 0.63157895 0.5789474 0.57894737 0.6250000 0.61111111  
## [2,] 0.00000000 0.0000000 0.00000000 0.0000000 0.00000000  
## [3,] 0.00000000 0.0000000 0.00000000 0.0000000 0.00000000  
## [4,] 0.00000000 0.0000000 0.00000000 0.0000000 0.00000000  
## [5,] 0.00000000 0.0000000 0.00000000 0.1176471 0.06250000  
## [6,] 0.00000000 0.0000000 0.05555556 0.0000000 0.00000000  
## [7,] 0.00000000 0.0000000 0.00000000 0.1111111 0.05555556  
## [8,] 0.00000000 0.0000000 0.00000000 0.0625000 0.00000000  
## [9,] 0.00000000 0.0000000 0.00000000 0.1111111 0.05555556  
## [10,] 0.11111111 0.0625000 0.11111111 0.0000000 0.00000000  
## [11,] 0.05555556 0.0000000 0.05555556 0.0000000 0.00000000  
##   
## $ave.between.matrix  
## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 0.0000000 0.9112417 0.9187450 0.9249050 0.9272719 0.9092257  
## [2,] 0.9112417 0.0000000 0.1946986 0.1938490 0.2249491 0.1675161  
## [3,] 0.9187450 0.1946986 0.0000000 0.1964951 0.1503606 0.1891834  
## [4,] 0.9249050 0.1938490 0.1964951 0.0000000 0.2025016 0.1959221  
## [5,] 0.9272719 0.2249491 0.1503606 0.2025016 0.0000000 0.2178950  
## [6,] 0.9092257 0.1675161 0.1891834 0.1959221 0.2178950 0.0000000  
## [7,] 0.9064351 0.2396541 0.1645183 0.2274340 0.1715703 0.2335900  
## [8,] 0.9018742 0.2253771 0.2170811 0.2108599 0.2267432 0.2228238  
## [9,] 0.8996406 0.2998673 0.2727437 0.2745262 0.2589175 0.2997272  
## [10,] 0.9104279 0.2140751 0.2790506 0.2439543 0.3061759 0.2094279  
## [11,] 0.9063284 0.1939078 0.2490334 0.2254007 0.2757722 0.1858756  
## [,7] [,8] [,9] [,10] [,11]  
## [1,] 0.9064351 0.9018742 0.8996406 0.9104279 0.9063284  
## [2,] 0.2396541 0.2253771 0.2998673 0.2140751 0.1939078  
## [3,] 0.1645183 0.2170811 0.2727437 0.2790506 0.2490334  
## [4,] 0.2274340 0.2108599 0.2745262 0.2439543 0.2254007  
## [5,] 0.1715703 0.2267432 0.2589175 0.3061759 0.2757722  
## [6,] 0.2335900 0.2228238 0.2997272 0.2094279 0.1858756  
## [7,] 0.0000000 0.1725114 0.2125105 0.3325923 0.2940075  
## [8,] 0.1725114 0.0000000 0.2078479 0.2978656 0.2618821  
## [9,] 0.2125105 0.2078479 0.0000000 0.3684280 0.3383178  
## [10,] 0.3325923 0.2978656 0.3684280 0.0000000 0.1749187  
## [11,] 0.2940075 0.2618821 0.3383178 0.1749187 0.0000000  
##   
## $average.between  
## [1] 0.2222328  
##   
## $average.within  
## [1] 0.1441051  
##   
## $n.between  
## [1] 1903317  
##   
## $n.within  
## [1] 361939  
##   
## $max.diameter  
## [1] 1  
##   
## $min.separation  
## [1] 0  
##   
## $within.cluster.ss  
## [1] 29.74128  
##   
## $clus.avg.silwidths  
## 1 2 3 4 5 6   
## 0.23297392 -0.14817087 -0.01746550 -0.00778913 0.06204519 -0.07808736   
## 7 8 9 10 11   
## 0.21751217 0.03415034 0.14493337 0.02928809 -0.10940536   
##   
## $avg.silwidth  
## [1] 0.004450517  
##   
## $g2  
## NULL  
##   
## $g3  
## NULL  
##   
## $pearsongamma  
## [1] 0.3034907  
##   
## $dunn  
## [1] 0  
##   
## $dunn2  
## [1] 0.2255408  
##   
## $entropy  
## [1] 2.027198  
##   
## $wb.ratio  
## [1] 0.6484419  
##   
## $ch  
## [1] 188.9702  
##   
## $cwidegap  
## [1] 1.00000000 0.14285714 0.07142857 0.13333333 0.12500000 0.13333333  
## [7] 0.11111111 0.06666667 0.11764706 0.12500000 0.12500000  
##   
## $widestgap  
## [1] 1  
##   
## $sindex  
## [1] 0  
##   
## $corrected.rand  
## [1] 0.1940502  
##   
## $vi  
## [1] 2.898478

#Best grouping sleeping classes