assignement2

Lilach Herzog & Leslie Cohen

13 5 2022

Table of Contents

[preproccessing 1](#_Toc106271105)

[Upload data and libraries 1](#_Toc106271106)

[split into training and test sets 2](#_Toc106271107)

[initial look at the data 2](#_Toc106271108)

[Splitting 4](#_Toc106271109)

Our whole project and code can be found on github at <https://github.com/LeslieLebon/assignment-2.git>

Our objective in this project was to analyze the “wheat-glaucoma” dataset (where the ‘Class’ of the seed is the column used for labeling and classifying) using two classifier models that we built. We decided on using the KNN algorithm and the Decision Tree (DT) algorithm.

# preproccessing

## Upload data and libraries

library(gmodels)  
library(C50)  
library(class)   
library(tidymodels) # for the rsample package, along with the rest of tidymodels  
library(ROCR)  
library(reshape2)  
library(ggpmisc)  
library(ggplot2)  
library(gridExtra)  
library(pROC)  
  
glaucoma <- read.csv("GlaucomaM.csv")# read data  
set.seed(1234)#to initialize a pseudo random number generator.

## split into training and test sets

### initial look at the data

We know the class ‘Class’ is the column (label) to classify according to. We took a quick look at the data as a whole, and specifically the ‘Class’ column. In order to further understand the data and be able to work with it we also used the “table” function. this function builds a contingency table of the counts at each combination of factor levels- we used it for the ‘Class’.

str(glaucoma) #quick look at the data as a whole

## 'data.frame': 196 obs. of 63 variables:  
## $ ag : num 2.22 2.68 1.98 1.75 2.99 ...  
## $ at : num 0.354 0.475 0.343 0.269 0.599 0.483 0.355 0.312 0.365 0.53 ...  
## $ as : num 0.58 0.672 0.508 0.476 0.686 0.763 0.601 0.463 0.57 0.834 ...  
## $ an : num 0.686 0.868 0.624 0.525 1.039 ...  
## $ ai : num 0.601 0.667 0.504 0.476 0.667 0.77 0.586 0.483 0.572 0.814 ...  
## $ eag : num 1.267 2.053 1.2 0.612 2.513 ...  
## $ eat : num 0.336 0.44 0.299 0.147 0.543 0.462 0.298 0.226 0.253 0.472 ...  
## $ eas : num 0.346 0.52 0.396 0.017 0.607 0.637 0.243 0.247 0.353 0.534 ...  
## $ ean : num 0.255 0.639 0.259 0.044 0.871 0.504 0.153 0.023 0.038 0.517 ...  
## $ eai : num 0.331 0.454 0.246 0.405 0.492 0.597 0.387 0.192 0.221 0.576 ...  
## $ abrg : num 0.479 1.09 0.465 0.17 1.8 ...  
## $ abrt : num 0.26 0.377 0.209 0.062 0.431 0.394 0.188 0.126 0.073 0.162 ...  
## $ abrs : num 0.107 0.257 0.112 0 0.494 0.365 0.102 0.048 0.033 0.163 ...  
## $ abrn : num 0.014 0.212 0.041 0 0.601 0.251 0.033 0 0 0.197 ...  
## $ abri : num 0.098 0.245 0.103 0.108 0.274 0.301 0.159 0.04 0.034 0.186 ...  
## $ hic : num 0.214 0.382 0.195 -0.03 0.383 0.442 0.18 -0.023 -0.005 0.181 ...  
## $ mhcg : num 0.111 0.14 0.062 -0.015 0.089 0.128 0.097 -0.042 -0.002 0.037 ...  
## $ mhct : num 0.412 0.338 0.356 0.074 0.233 0.375 0.329 0.168 0.197 0.283 ...  
## $ mhcs : num 0.036 0.104 0.045 -0.084 0.145 0.049 0.037 -0.073 -0.004 -0.035 ...  
## $ mhcn : num 0.105 0.08 -0.009 -0.05 0.023 0.111 0.019 -0.109 -0.038 0.019 ...  
## $ mhci : num -0.022 0.109 -0.048 0.035 0.007 0.052 0.093 -0.077 -0.098 -0.049 ...  
## $ phcg : num -0.139 -0.015 -0.149 -0.182 -0.131 -0.088 -0.125 -0.183 -0.22 -0.15 ...  
## $ phct : num 0.242 0.296 0.206 -0.097 0.163 0.281 0.238 0.084 0.103 0.227 ...  
## $ phcs : num -0.053 -0.015 -0.092 -0.125 0.055 -0.067 -0.029 -0.183 -0.088 -0.107 ...  
## $ phcn : num 0.01 -0.015 -0.081 -0.138 -0.131 -0.062 -0.125 -0.151 -0.214 -0.1 ...  
## $ phci : num -0.139 0.036 -0.149 -0.182 -0.115 -0.088 -0.121 -0.138 -0.22 -0.15 ...  
## $ hvc : num 0.613 0.382 0.557 0.373 0.405 0.507 0.486 0.394 0.484 0.455 ...  
## $ vbsg : num 0.303 0.676 0.3 0.048 0.889 0.972 0.252 0.107 0.116 0.497 ...  
## $ vbst : num 0.103 0.181 0.084 0.011 0.151 0.213 0.073 0.043 0.042 0.083 ...  
## $ vbss : num 0.088 0.186 0.088 0 0.253 0.316 0.058 0.038 0.042 0.122 ...  
## $ vbsn : num 0.022 0.141 0.046 0 0.33 0.197 0.024 0.001 0.002 0.138 ...  
## $ vbsi : num 0.09 0.169 0.082 0.036 0.155 0.246 0.096 0.026 0.03 0.155 ...  
## $ vasg : num 0.062 0.029 0.036 0.07 0.02 0.043 0.057 0.072 0.116 0.037 ...  
## $ vast : num 0 0.001 0.002 0.005 0.001 0.001 0.002 0.003 0.004 0.001 ...  
## $ vass : num 0.011 0.007 0.004 0.03 0.004 0.005 0.016 0.011 0.014 0.009 ...  
## $ vasn : num 0.032 0.011 0.016 0.033 0.008 0.028 0.03 0.045 0.075 0.02 ...  
## $ vasi : num 0.018 0.01 0.013 0.002 0.007 0.009 0.01 0.012 0.023 0.007 ...  
## $ vbrg : num 0.075 0.37 0.081 0.005 0.532 0.467 0.085 0.032 0.012 0.146 ...  
## $ vbrt : num 0.039 0.127 0.034 0.001 0.103 0.136 0.03 0.017 0.005 0.03 ...  
## $ vbrs : num 0.021 0.099 0.019 0 0.173 0.148 0.018 0.011 0.002 0.029 ...  
## $ vbrn : num 0.002 0.05 0.007 0 0.181 0.078 0.006 0 0 0.04 ...  
## $ vbri : num 0.014 0.093 0.021 0.004 0.075 0.104 0.031 0.004 0.005 0.047 ...  
## $ varg : num 0.756 0.41 0.565 0.38 0.228 0.54 0.587 0.479 0.704 0.696 ...  
## $ vart : num 0.009 0.006 0.014 0.032 0.011 0.008 0.013 0.015 0.035 0.029 ...  
## $ vars : num 0.209 0.105 0.132 0.147 0.026 0.133 0.185 0.115 0.154 0.219 ...  
## $ varn : num 0.298 0.181 0.243 0.151 0.105 0.232 0.263 0.223 0.302 0.246 ...  
## $ vari : num 0.24 0.117 0.177 0.05 0.087 0.167 0.126 0.127 0.214 0.202 ...  
## $ mdg : num 0.705 0.898 0.687 0.207 0.721 0.927 0.573 0.559 0.476 0.648 ...  
## $ mdt : num 0.637 0.85 0.643 0.171 0.638 0.842 0.525 0.444 0.439 0.577 ...  
## $ mds : num 0.738 0.907 0.689 0.022 0.73 0.953 0.58 0.672 0.403 0.627 ...  
## $ mdn : num 0.596 0.771 0.684 0.046 0.73 0.906 0.581 0.124 0.217 0.62 ...  
## $ mdi : num 0.691 0.94 0.7 0.221 0.64 0.898 0.575 0.411 0.581 0.666 ...  
## $ tmg : num -0.236 -0.211 -0.185 -0.148 -0.052 -0.04 -0.149 -0.298 -0.278 -0.19 ...  
## $ tmt : num -0.018 -0.014 -0.097 -0.035 -0.105 0.087 -0.036 -0.14 -0.144 -0.288 ...  
## $ tms : num -0.23 -0.165 -0.235 -0.449 0.084 0.018 -0.18 -0.378 -0.273 -0.193 ...  
## $ tmn : num -0.51 -0.317 -0.337 -0.217 -0.012 -0.094 -0.394 -0.457 -0.194 -0.084 ...  
## $ tmi : num -0.158 -0.192 -0.02 -0.091 -0.054 -0.051 -0.07 -0.25 -0.371 -0.096 ...  
## $ mr : num 0.841 0.924 0.795 0.746 0.977 ...  
## $ rnf : num 0.41 0.256 0.378 0.2 0.193 0.339 0.308 0.263 0.316 0.319 ...  
## $ mdic : num 0.137 0.252 0.152 0.027 0.297 0.333 0.113 0.058 0.053 0.158 ...  
## $ emd : num 0.239 0.329 0.25 0.078 0.354 0.442 0.233 0.155 0.134 0.237 ...  
## $ mv : num 0.035 0.022 0.029 0.023 0.034 0.028 0.012 0.023 0.029 0.022 ...  
## $ Class: chr "normal" "normal" "normal" "normal" ...

table(glaucoma$Class) # quick look specifically at the 'Class' column

##   
## glaucoma normal   
## 98 98

The column we want to classify is ‘Class’. There are 3 Classs of glaucoma found in in the column ‘Class’: 1, 2 and 3 (We will sometimes refer to them as Class\_1, Class\_2 and Class\_3, to make it easier to understand and differentiate them from numbers that represent other things). We can see the Class has 66 times Class\_1, 68 times Class\_2, and 65 times Class\_3, about a third of each Class.

Before starting on teaching the algorithm, we converted the ‘Class’ column to a factor, as that is what is required by the C50 package.

glaucoma$Class<-as.factor(glaucoma$Class)

Since the whole table is sorted according to the ‘Class’ column, and we want to work randomly, we will shuffle the whole table

### Splitting

length\_of\_training\_set <-round(0.8\*(nrow(glaucoma)),0)  
train\_set <- glaucoma[1:length\_of\_training\_set, ]  
test\_set <- glaucoma[(length\_of\_training\_set+1):nrow(glaucoma), ]  
train\_set\_Classs <-train\_set[, 8]  
test\_set\_Classs <-test\_set[, 8]  
prop.table(table(train\_set$Class))

##   
## glaucoma normal   
## 0.566879 0.433121

Since we know that there are about a third of each Class in the data set, we wanted to be sure that the distribution is as we defined (about 1/3 of each Class in of both training and test sets)