Pràctica 2: Com realitzar la neteja i anàlisi de dades?

Autor: Marc López Vila, Eduard López i Fina

Juny, 2023

- 1 Descripció del dataset.
- 2 Integració i selecció
- 3 Neteja de les dades.
- 4 Anàlisi de les dades.
- 5 Resolució del problema i conclusions.


```
if (!require('ggplot2')) install.packages('ggplot2'); library('ggplot2')
if(!require('Rmisc')) install.packages('Rmisc'); library('Rmisc')
if(!require('dplyr')) install.packages('dplyr'); library('dplyr')
if(!require('xfun')) install.packages('xfun'); library('xfun')
if(!require('lubridate')) install.packages('lubridate'); library('lubridate')
if(!require("corrplot")) install.packages("corrplot"); library("corrplot")
if (!require('tidyverse')) install.packages('tidyverse'); library('tidyverse')
if (!require('factoextra')) install.packages('factoextra'); library('factoextra')
if (!require('Stat2Data')) install.packages('Stat2Data'); library('Stat2Data')
if (!require('cluster')) install.packages('cluster'); library('cluster')
if (!require('fpc')) install.packages('fpc'); library('fpc')
if (!require('gridExtra')) install.packages('gridExtra'); library('gridExtra')
if (!require('dbscan')) install.packages('dbscan'); library('dbscan')
if (!require('knitr')) install.packages('knitr'); library('knitr')
if (!require('grid')) install.packages('grid'); library('grid')
if (!require('C50')) install.packages('C50'); library('C50')
if (!require('ggpubr')) install.packages('ggpubr'); library('ggpubr')
if (!require('DescTools')) install.packages('DescTools'); library('DescTools')
if (!require('gmodels')) install.packages('gmodels'); library('gmodels')
if (!require('GGally')) install.packages('GGally'); library('GGally')

if (!require('randomForest')) install.packages('randomForest'); library('randomFores
t')
if (!require('iml')) install.packages('iml'); library('iml')
if (!require('patchwork')) install.packages('patchwork'); library('patchwork')

if (!require('rattle')) install.packages('rattle'); library('rattle')
if (!require('rpart.plot')) install.packages('rpart.plot'); library('rpart.plot')
if (!require('RColorBrewer')) install.packages('RColorBrewer'); library('RColorBrewe
r')

if (!require('caret')) install.packages('caret', repos = "http://cran.us.r-project.or
g"); library('caret')
```

1 Descripció del dataset.

Aquest dataset és interessant perquè ens proporciona dades complexes que ens ajudaran a poder preveure quins pacients són propensos a tenir atacs de cor. Ha estat extret de kaggle, i és el que ens dona d'exemple la pràctica.

El dataset ens proporciona dades de **303 pacients,** com el dolor del pit, pressió sanguínea, colesterol i màxim de pulsacions, entre d'altres. També ens proporciona dades de la saturació d'O2 en un dataset individual.

En la pràctica intentarem unificar els dos datasets, netejar-los i tractarem d'estimar un model que a partir de les dades pugui preveure possibles atacs de cor.

2 Integració i selecció

Primerament carreguem els dos datasets.

```
heart_data <- read.csv("../data/heart.csv");
o2_data <- read.csv("../data/o2Saturation.csv");

nrow(heart_data);
```

```
## [1] 303
```

```
nrow(o2_data);
```

```
## [1] 3585
```

Un cop carregats els fitxers, podem apreciar que la seva mida és diferent, i això sumat a que no tenim un identificador de persona que ens ajudi a relacionar les dades dels dos datasets fa que no tinguem manera de saber com està relacionada la saturació o2 amb el dataset principal. Per tant, hem optat per no integrar les dades del fitxer o2 per l'anàlisi.

En quant al dataset heart_data, ja que no hem trobat cap camp calculat, hem optat per incloure totes les dades actuals i seleccionar més endavant quines utilitzar segons l'anàlisi que volguem fer.

3 Neteja de les dades.

En primer lloc, mirarem la composició del dataset:

```
str(heart_data)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : int  63 37 41 56 57 57 56 44 52 57 ...
##  $ sex     : int  1 1 0 1 0 1 0 1 1 1 ...
##  $ cp      : int  3 2 1 1 0 0 1 1 2 2 ...
##  $ trtbps  : int  145 130 130 120 120 140 140 120 172 150 ...
##  $ chol    : int  233 250 204 236 354 192 294 263 199 168 ...
##  $ fbs     : int  1 0 0 0 0 0 0 0 1 0 ...
##  $ restecg : int  0 1 0 1 1 1 0 1 1 1 ...
##  $ thalachh: int  150 187 172 178 163 148 153 173 162 174 ...
##  $ exng    : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ oldpeak : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
##  $ slp     : int  0 0 2 2 2 1 1 2 2 2 ...
##  $ caa     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ thall   : int  1 2 2 2 2 1 2 3 3 2 ...
##  $ output  : int  1 1 1 1 1 1 1 1 1 1 ...
```

S'obervacom el dataset conté 303 registres amb 14 atributs cadascun d'ells, els quals s'expliquen a continuació:

- **age.** És l'edat de la persona.
- **sex.** És el sexe de la persona. Està codificat de forma factorial amb 0 i 1 segons sexe.
- **cp.** És el nivell de mal de pit. Està codificat de forma factorial de 0-3 segons dolor.
- **trtbps.** És la pressió arterial en repòs.
- **chol.** És el colesterol de la persona.
- **fbs.** És el sucre en sang. Està codificat de forma factorial, 1 si fbs>120 i 0 en cas contrari.
- **restecg.** Són els resultats electrocardiogràfics. Estan codificats de forma factorial de 0-2 segons gravetat.
- **thalacc.** És el màxim de pulsacions.
- **exng.** Ens retorna si la persona té angina induïda per exercici. Està codificat de forma factorial amb 0 en cas de que no tingui angina i 1 en cas de que tingui angina.
- **oldpeak.** És la depressió del ST induïda per l'exercici en relació amb el repòs. En aquest cas el valor pot ser 0.
- **slp.** És el pendent del segment ST de l'exercici màxim. Està codificat de forma factorial de 0-2 segons pendent.
- **caa.** És el nombre de vasos cardíacs. Està codificat de forma factorial de 0-3.
- **thall.** És el grau de talassèmia. Està codificat de forma factorial de 0-3 segons grau.
- **output.** És el diagnòstic final. Està codificat de forma factorial amb 1 si té probabilitats de tenir un atac de cor i 0 si no.

Tot seguit, analitzem les dades que contenen zeros o elements buits:

```
na_counts <- colSums(is.na(heart_data))
empty_counts <- colSums(heart_data == "")
zero_counts <- colSums(heart_data == 0)
result <- data.frame(NAs = na_counts, BLANK = empty_counts, ZEROS = zero_counts)
kable(result)
```

	NAs	BLANK	ZEROS
age	0	0	0

	NAs	BLANK	ZEROS
sex	0	0	96
cp	0	0	143
trtbps	0	0	0
chol	0	0	0
fbs	0	0	258
restecg	0	0	147
thalachh	0	0	0
exng	0	0	204
oldpeak	0	0	99
slp	0	0	21
caa	0	0	175
thall	0	0	2
output	0	0	138

Podem veure com no hi ha cap atribut en el qual ens falti informació (és a dir, que sigui NAs o estigui en blanc). Pel que fa a la columna de zeros, ens hem de fixar en aquells atributs numèrics no factorials on el zero no sigui una opció, com és el cas de *age*, *trtbps*, *chol*, i *thalachh*, on veiem que en cap cas tenen zeros.

A continuació farem un estudi dels outliers, i per detectar-los creem la funció get_outliers:

```
get_outliers <- function(x) {
  # Calculem la mitjana i la desviació estàndard
  mean = mean(x)
  std = sd(x)

  # Calculem el tmin i el tmax
  Tmin = mean-(3*std)
  Tmax = mean+(3*std)

  # Trobem els outliers
  x[which(x < Tmin | x > Tmax)]
}
```

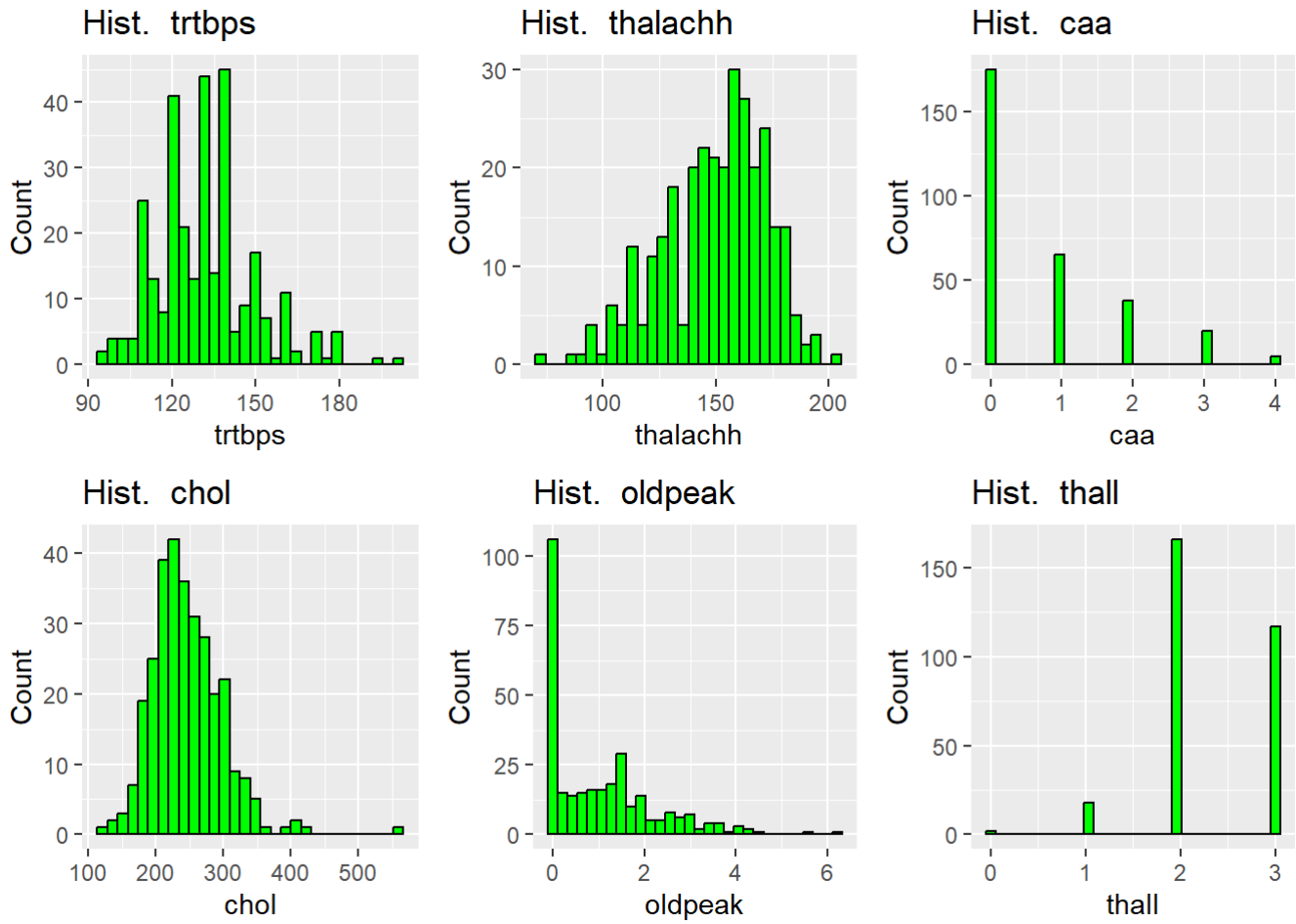
Un cop tenim la funció, la utilitzem per trobar els outliers existents.

```
cp_outliers <- get_outliers(heart_data$cp);
chol_outliers <- get_outliers(heart_data$chol);

for (i in colnames(heart_data)) {
  sprintf("La columna %s té els següents outliers: ", i)
  outliers <- get_outliers(heart_data[[i]])

  if(length(outliers) != 0) {
    cat(sprintf("La columna %s té els següents outliers: ", i), outliers, "\n")
  }
}
```

```
## La columna trtbps té els següents outliers:  200 192
## La columna chol té els següents outliers:  417 564 407 409
## La columna thalachh té els següents outliers:  71
## La columna oldpeak té els següents outliers:  6.2 5.6
## La columna caa té els següents outliers:  4 4 4 4 4
## La columna thall té els següents outliers:  0 0
```

Analitzant les dades detectades com a outliers arribem a la conclusió de que cap és un valor extrem, totes les dades entren dins la normalitat, tal com es pot comprovar a continuació:

```
col = c("trtbps", "chol", "thalachh", "oldpeak", "caa", "thall")

heartDataAux <- heart_data %>% select(all_of(col))
histList <- list()

for (i in 1:ncol(heartDataAux)) {
  col <- names(heartDataAux)[i]

  ggp <- ggplot(heartDataAux, aes_string(x = col)) +
    geom_histogram(bins = 30, fill="green", color = "black") +
    labs(x = col, y = "Count") +
    ggtitle(paste("Hist. ", col))

  histList[[i]] <- ggp  # add each plot to the list
}
multiplot(plotlist = histList, cols = 3)
```



4 Anàlisi de les dades.

4.0.1 Selecció grups de dades

En primer lloc, i de cara als models que s'aplicaran més endavant, factoritzem aquells atributs que així ho requereixen:

```
# Fem el cast a factorial
heart_data$output <- factor(heart_data$output)
categorical_c = c("sex", "fbs", "exng", "output")
heart_data <- heart_data %>% mutate_at(categorical_c, factor)
```

Tot seguit, mirarem de visualitzar les relacions entre els diferents atributs desglossats segons el seu diagnòstic final (output), per veure si d'aquesta manera podem veure diferències significatives entre els dos grups:

```
ggpairs(heart_data[rowSums(is.na(heart_data)) == 0,], columns = c(1:12),
        aes(color = heart_data$output, alpha = 0.5),
        upper = "blank",
        axisLabels = "none")
```

Tot i que en els núvols de punts es fa difícil diferenciar entre els dos grups, en els histogrames sí que se sembla apreciar diferències significatives entre els dos grups amb relació a diversos atributs, tals com l'edat, el nivell de mal al pit (*cp*), els resultats electrocardiogràfics (*restecg*), o en el nombre de vasos cardíacs (*caa*).

4.0.2 Comprovació normalitat de la variància.

Amb l'objectiu de verificar la suposició de la normalitat utilitzarem el test de Shapiro-Wilk en cadascun dels atributs del dataset. Bàsicament, el que fa és assumir com a hipòtesi nul·la que la població està distribuïda normalment, així que si el p-valor dels atributs és major que el nivell de significació (0,05) aleshores podrem concloure que les dades no compten amb una distribució normal.

```r
# Select only the numeric columns from heart_data
numeric_cols <- heart_data[, sapply(heart_data, is.numeric)]

# Loop through each numeric column
for (col_name in names(numeric_cols)) {
  col <- numeric_cols[[col_name]]

  # Perform Shapiro-Wilk test
  result <- shapiro.test(col)

  # Extract test statistics
  test_stats <- paste("W =", result$statistic, "p-value =", result$p.value)

  # Print the column name and test statistics
  cat("Column:", col_name, "\n")
  cat(test_stats, "\n\n")
}
```

```
## Column: age
## W = 0.986370480625655 p-value = 0.00579835873933943
##
## Column: cp
## W = 0.790158958555487 p-value = 1.85715161395053e-19
##
## Column: trtbps
## W = 0.965917872645145 p-value = 1.45809705294778e-06
##
## Column: chol
## W = 0.946881528098253 p-value = 5.36484787285249e-09
##
## Column: restecg
## W = 0.679316149263391 p-value = 1.37811007492199e-23
##
## Column: thalachh
## W = 0.976315356049362 p-value = 6.62081936641719e-05
##
## Column: oldpeak
## W = 0.844183363226846 p-value = 8.18337828923442e-17
##
## Column: slp
## W = 0.744653977816698 p-value = 2.57477928233927e-21
##
## Column: caa
## W = 0.728115484487627 p-value = 6.27112603799738e-22
##
## Column: thall
## W = 0.750581764380794 p-value = 4.34425302333555e-21
```

Veiem com en tots els casos es rebutja la hipòtesi alternativa, ja que tots els atributs presenten un p-value inferior a 0,05. Encara més, l'únic atribut que s'acosta mínimament a la alternativa és l'edat. Podríem normalitzar les dades abans d'aplicar els diferents models, però s'ha decidit que fer-ho no serà pràctic, ja que dificultarà la comprensió a l'hora de treballar amb les dades i el resultat serà el mateix.

4.0.3 Model de regressió

El primer dels anàlisis que farem una regressió. Com la nostra variable dependent (*output*) és dicotòmica, utilitzarem la regressió logística enlloc de lineal.

Per comprovar l'efectivitat del model resultant dividirem les dades en dos conjunts, un conjunt de training (80% de les dades) i un conjunt de testing (20% de les dades).

```
# Seleccionem una seed per poder reproduir els conjunts
set.seed(15)

# Creem els conjunts de training i de test
train_index <- createDataPartition(y=heart_data$output, p=0.8, list=FALSE)
training <- heart_data[train_index, ]
testing <- heart_data[-train_index, ]
```

Verificarem les dades per veure que la proporció de persones diagnosticades és més o menys la mateixa tan al subset d'entrenament com al de testeig:

```
cat("TRAIN_Y [%]\n")
```

```
## TRAIN_Y [%]
```

```
prop.table(summary(training$output))
```

```
##         0         1
## 0.4567901 0.5432099
```

```
cat("\nTEST_Y [%]\n")
```

```
##
## TEST_Y [%]
```

```
prop.table(summary(testing$output))
```

```
##    0    1
## 0.45 0.55
```

Un cop tenim els conjunts definits i la variable *output* com a factorial, creem el model amb les dades de training. Després d'estudiar les variables, s'ha arribat a la conclusió que les millors variables independents seran *sex*, *cp*, *exng*, *oldpeak*, *slp*, *caa* i *thall*.

```
model <- glm(formula = output~sex+cp+exng+oldpeak+slp+caa+thall,
                 family = binomial,
                 data = heart_data)

summary(model)
```

```
##
## Call:
## glm(formula = output ~ sex + cp + exng + oldpeak + slp + caa +
##     thall, family = binomial, data = heart_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7856  -0.4699   0.1938   0.5768   2.4799
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   2.8468     0.8721   3.264 0.001097 **
## sex1         -1.3652     0.4086  -3.341 0.000835 ***
## cp            0.8811     0.1759   5.010 5.44e-07 ***
## exng1        -1.1795     0.3787  -3.115 0.001842 **
## oldpeak      -0.6294     0.2051  -3.069 0.002145 **
## slp           0.7367     0.3272   2.251 0.024359 *
## caa          -0.8166     0.1784  -4.577 4.72e-06 ***
## thall        -0.8696     0.2694  -3.228 0.001245 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 417.64  on 302  degrees of freedom
## Residual deviance: 225.99  on 295  degrees of freedom
## AIC: 241.99
##
## Number of Fisher Scoring iterations: 5
```

Veiem amb la informació del model com els valors $p(Pr(> |z|))$ són tots inferiors a 0.05, mostrant-nos així que totes són significativament estadístiques i aporten al model.

A continuació, fem una matriu de confusió amb el conjunt de testing.

```
# Obtenim els valors predits
predicted_values <- ifelse(predict(model, newdata = testing, type = "response") >= 0.
5, 1, 0)
predicted_values <- factor(predicted_values, levels = c("0", "1"))

# Recolectem els valors reals
real_values <- factor(testing$output, levels = c("0", "1"))

# Calculem la matriu de confusió
confusion_matrix <- confusionMatrix(predicted_values, real_values)
confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 21  4
##          1  6 29
##
##                Accuracy : 0.8333
##                  95% CI : (0.7148, 0.9171)
##     No Information Rate : 0.55
##     P-Value [Acc > NIR] : 3.483e-06
##
##                   Kappa : 0.661
##
##  Mcnemar's Test P-Value : 0.7518
##
##             Sensitivity : 0.7778
##             Specificity : 0.8788
##          Pos Pred Value : 0.8400
##          Neg Pred Value : 0.8286
##              Prevalence : 0.4500
##          Detection Rate : 0.3500
##    Detection Prevalence : 0.4167
##       Balanced Accuracy : 0.8283
##
##        'Positive' Class : 0
##
```

El model ha encertat un 83,33% de les dades de testing, i podem dir amb un 95% de certesa que la precisió d'aquest model estarà entre un 71,48% i 91.71%.

També tenim altres dades com la sensibilitat que ens mostra que s'ha encertat un 77,78% dels positius, o la especificitat que ens mostra que s'ha encertat un 87,88% dels negatius (entenent negatiu com a output 0 i positiu com a output 1).

4.0.4 Arbre de decisió.

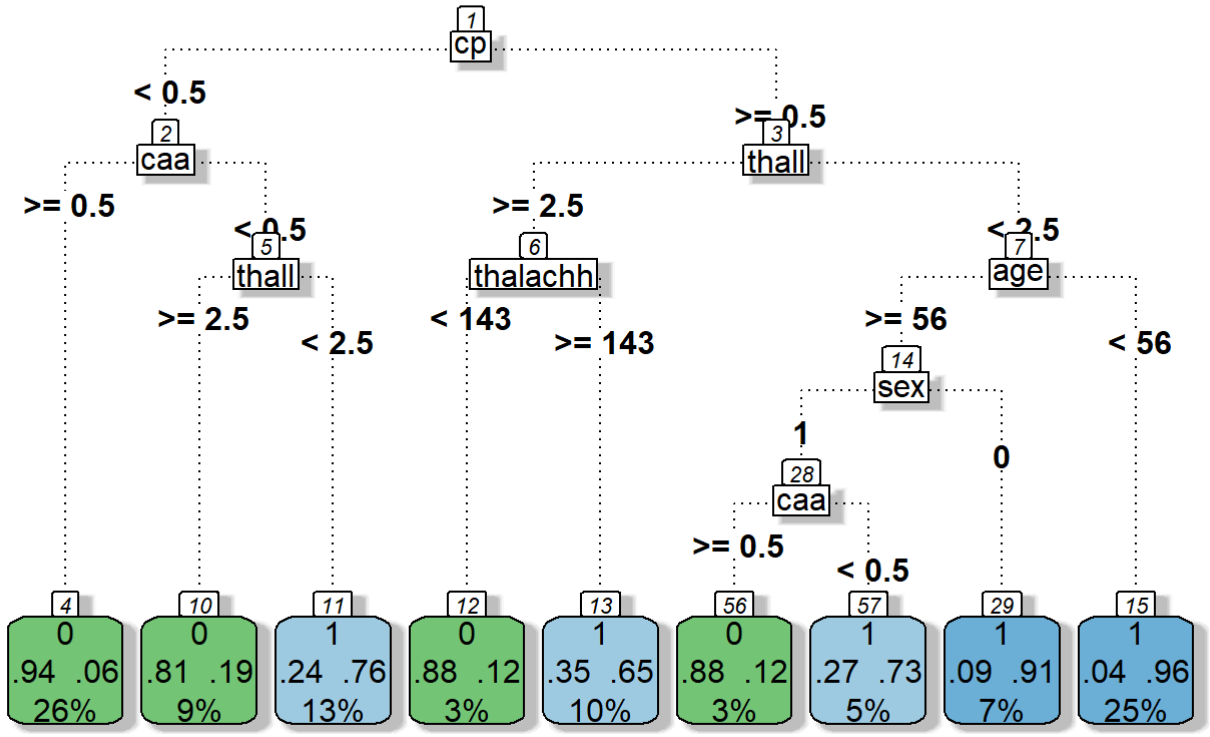
Un cop creats els subsets de dades, podem generar un arbre de decisió que ens permetrà predir si el pacient serà diagnosticat o no gràcies al package *rpart*:

```
model_r <- rpart(output ~ ., # default vs all other columns
                 data = heart_data) # ds

rules <- rpart.rules(model_r)
print(rules)
```

```
##  output
##    0.06 when cp <  1              & caa >= 1
##    0.12 when cp >= 1 & thall >= 3                                  & thalachh <  1
43
##    0.12 when cp >= 1 & thall <  3 & caa >= 1 & age >= 56 & sex is 1
##    0.19 when cp <  1 & thall >= 3 & caa <  1
##    0.65 when cp >= 1 & thall >= 3                                  & thalachh >= 1
43
##    0.73 when cp >= 1 & thall <  3 & caa <  1 & age >= 56 & sex is 1
##    0.76 when cp <  1 & thall <  3 & caa <  1
##    0.91 when cp >= 1 & thall <  3            & age >= 56 & sex is 0
##    0.96 when cp >= 1 & thall <  3            & age <  56
```

A continuació podem visualitzar les regles, el que facilita molt la comprensió del model:

```
fancyRpartPlot(model_r, type = 5, space = 0, caption = NULL)
```



La forma correcta de llegir cada regla és, mirant la primera de totes, que

- si el pacient presenta un *cp* (nivell de mal al pit) inferior a 0,5
- i té un *caa* (nombre de vasos cardíacs) major o igual a 0,5

aleshores, podem afirmar amb un 94% de confiança que el pacient no serà diagnosticat (*output*=0). El 26% és el percentatge de les dades de training van a parar a aquest node.

El tercer node és una regla que ens indica una predicció positiva, però és llegiria de forma similar:

- si el pacient presenta un *cp* (nivell de mal al pit) inferior a 0,5
- té un *caa* (nombre de vasos cardíacs) menor a 0,5
- i té un *thall* (grau de talassèmia) menor a 2,5

aleshores, podem afirmar amb un 76% de confiança que el pacient sí serà diagnosticat (*output*=1). El 13% de les dades de training van a parar a aquest node.

A continuació tenim la matriu de confusió de l'arbre:

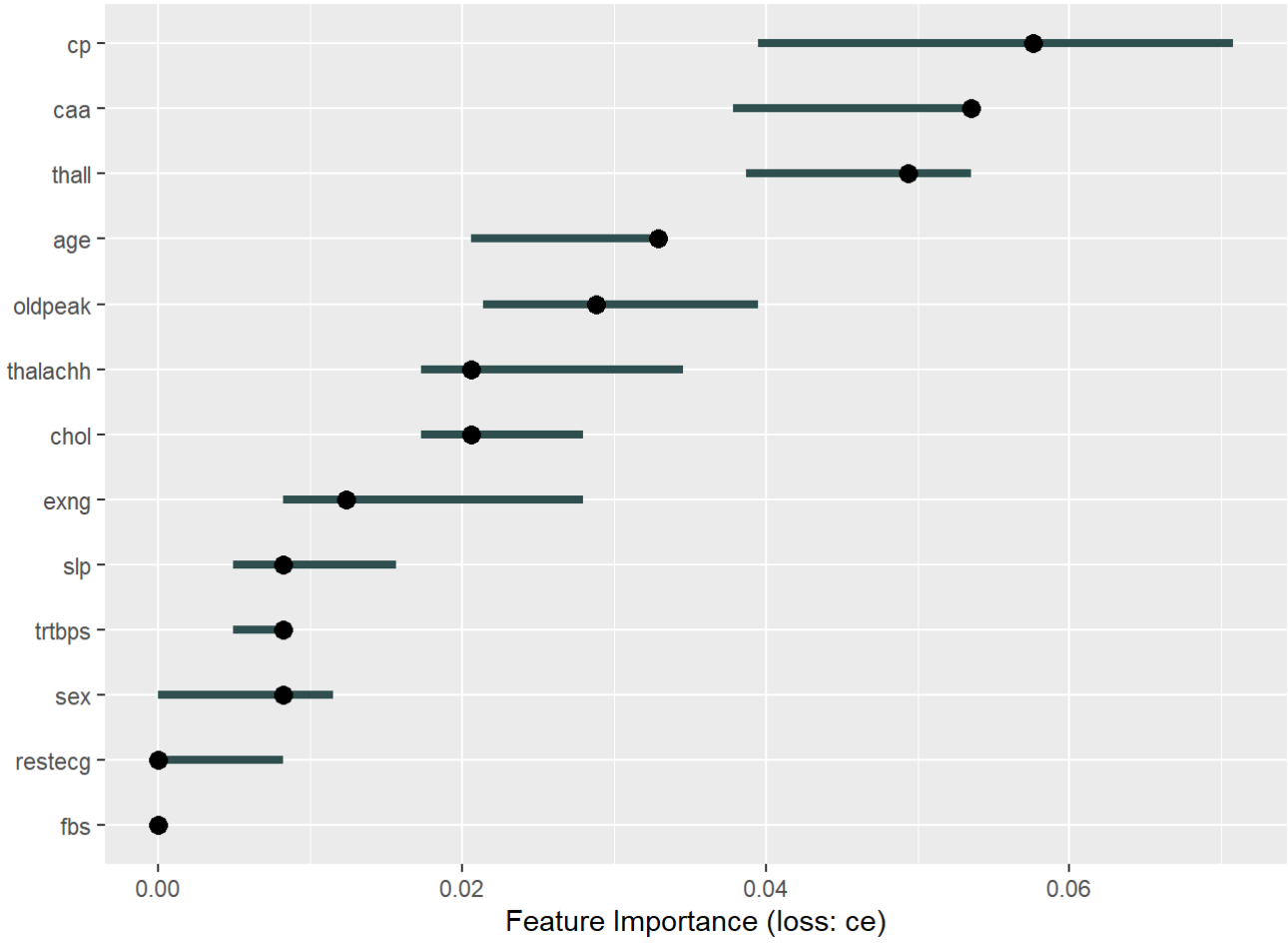
```
# Obtenim els valors predits
predicted_values <- predict(model_r, testing, type = "class")
predicted_values <- factor(predicted_values, levels = c("0", "1"))

# Recolectem els valors reals
real_values <- factor(testing$output, levels = c("0", "1"))

# Calculem la matriu de confusió
confusion_matrix <- confusionMatrix(predicted_values, real_values)
confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 20  2
##          1  7 31
##
##                Accuracy : 0.85
##                  95% CI : (0.7343, 0.929)
##     No Information Rate : 0.55
##     P-Value [Acc > NIR] : 8.068e-07
##
##                   Kappa : 0.6918
##
##  Mcnemar's Test P-Value : 0.1824
##
##             Sensitivity : 0.7407
##             Specificity : 0.9394
##          Pos Pred Value : 0.9091
##          Neg Pred Value : 0.8158
##              Prevalence : 0.4500
##          Detection Rate : 0.3333
##    Detection Prevalence : 0.3667
##       Balanced Accuracy : 0.8401
##
##        'Positive' Class : 0
##
```

Veiem com el model ha encertat el 85% dels registres, el mateix percentatge que el model de regressió de l'apartat anterior. Ara bé, en aquest cas la sensibilitat ens indica que hem encertat el 74,07% dels *outputs* positius (és a dir, de diagnòstics positius), el que representa un empitjorament respecte al 77,78% del model anterior.

Finalment, podem utilitzar randomForest per veure quines són les variables amb més pes dins de l'arbre de decisió, i per tant, les dades més rellevants pel problema que volem resoldre.

```
rf <-  randomForest(output ~ ., data = training, ntree = 50)
X <- training[which(names(training) != "output")]
predictor <- Predictor$new(rf, data = X, y = training$output)
imp <- FeatureImp$new(predictor, loss = "ce") # ce = obj. de classific.
plot(imp)
```



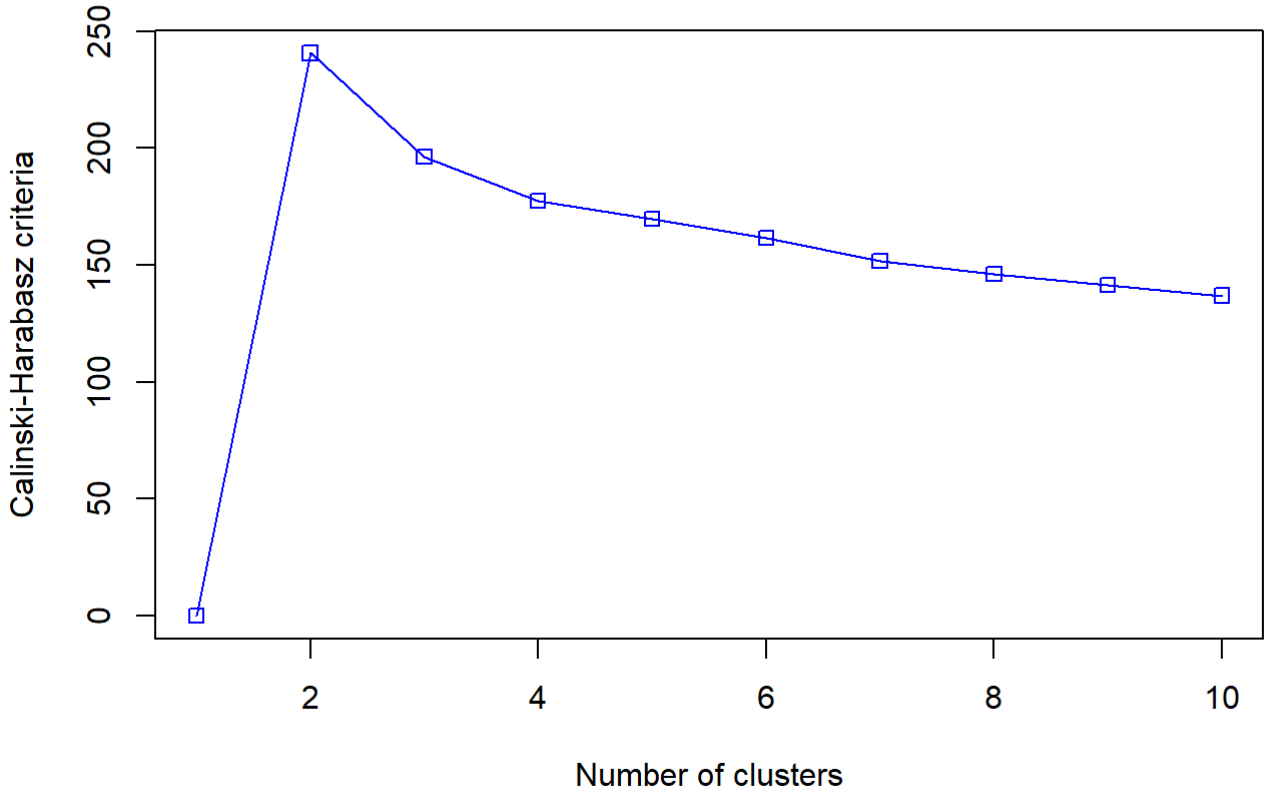
4.0.5 Model de clustering.

El primer que farem abans d'aplicar el model de clustering és excloure la columna output del dataset, ja que al tractar-se d'un model no supervisat representa que d'entrada no sabem quants grups hi ha ni com estan repartits:

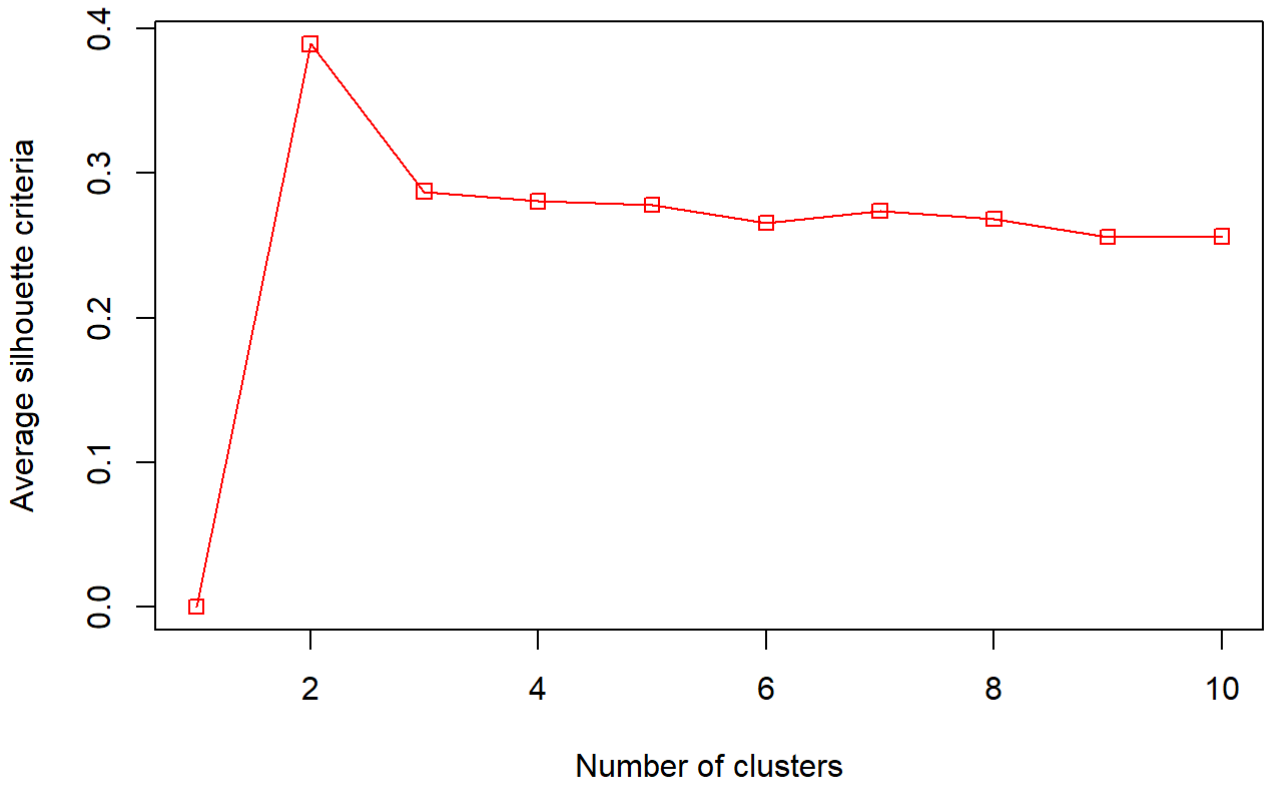
```
heartDataAux <- heart_data[1:13] # We exclude the output column
heartDataAux <- as.data.frame(lapply(heartDataAux, as.numeric))
```

Per determinar el nombre de clústers utilitzarem tan el criteri de *Calinski-Harabasz* com el de Silhouette, per veure si marquen el mateix nombre de clústers:

```
fit_ch  <- kmeansruns(heartDataAux, krange = 1:10, criterion = "ch")
fit_asw <- kmeansruns(heartDataAux, krange = 1:10, criterion = "asw")

plot(1:10,fit_ch$crit,type="o",col="blue",pch=0,xlab="Number of clusters",
     ylab="Calinski-Harabasz criteria")
```




```
plot(1:10,fit_asw$crit,type="o",col="red",pch=0,xlab="Number of clusters",
     ylab="Average silhouette criteria")
```

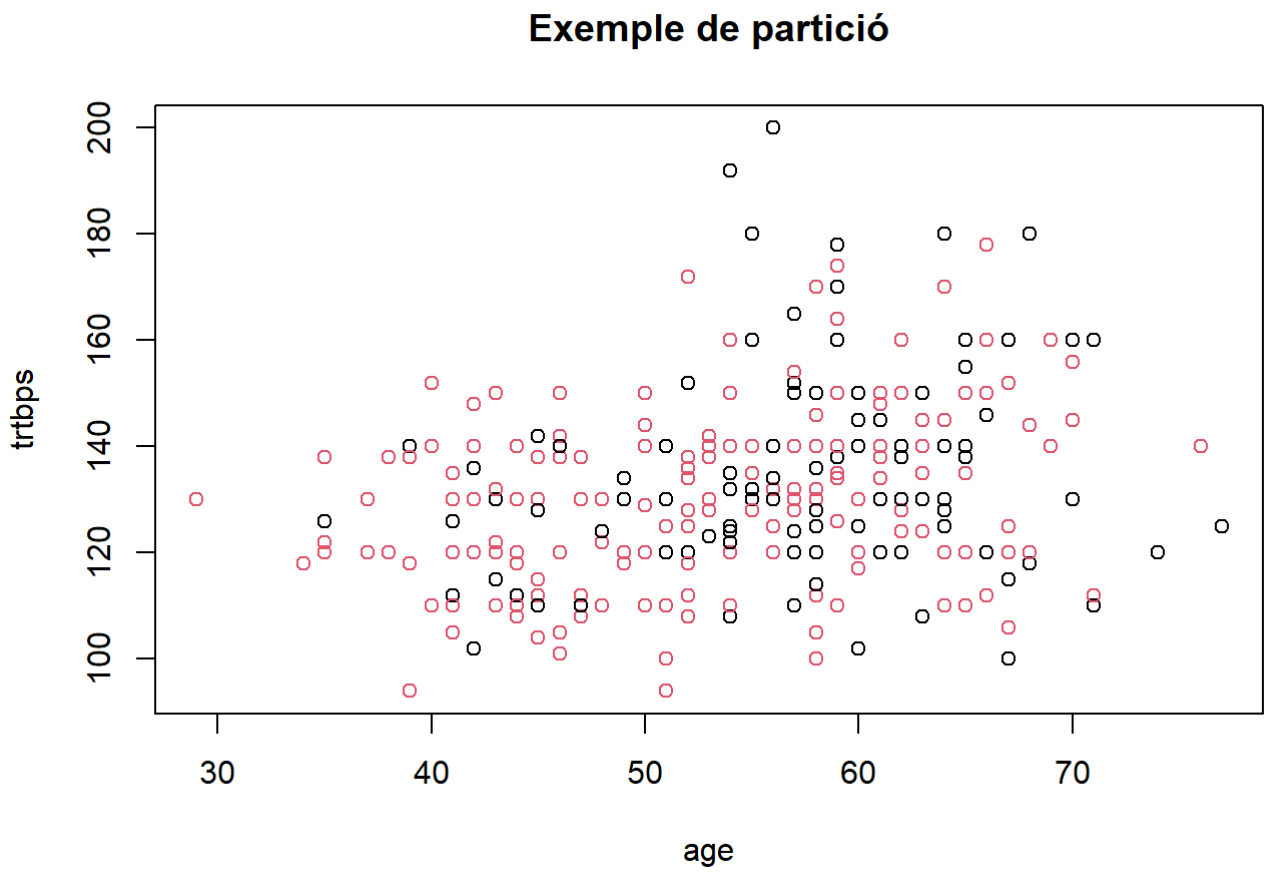


Veiem com en ambdós casos tenim que el nombre òptim de clústers és 2, el que a més encaixa amb la informació que nosaltres sabem (el pacient és diagnosticat o no és diagnosticat). Per tant, a continuació aplicarem el model kmeans amb 2 clústers en total:

```
heart2clusters <- kmeans(heartDataAux, 2)
```

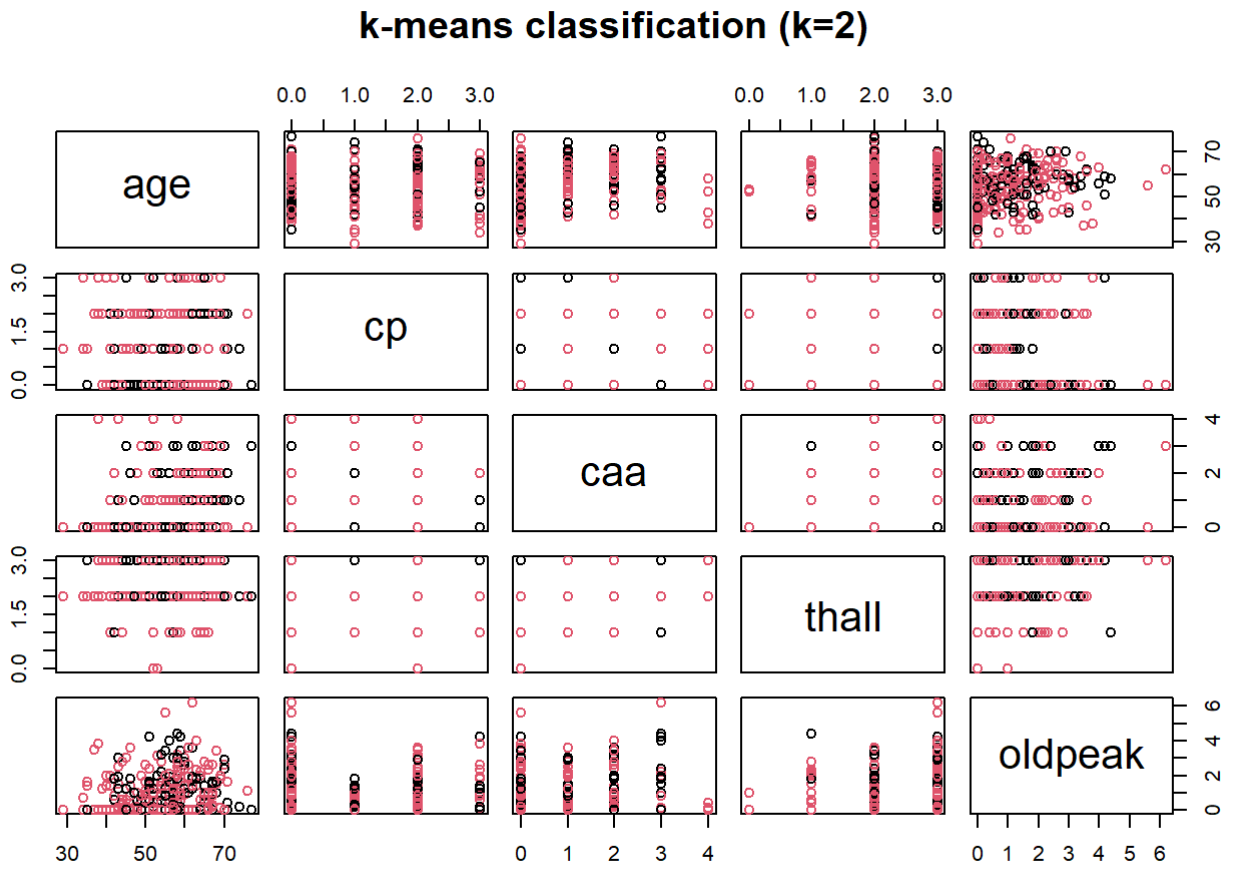
Un cop aplicat el model veiem com el dataset auxiliar s'ha dividit en dos grups, els quals s'haurien de correspondre amb la columna *output* eliminada inicialment.

```
plot(heartDataAux[, c(1, 4)], col=as.factor(heart2clusters$cluster), main="Exemple de
partició")
```



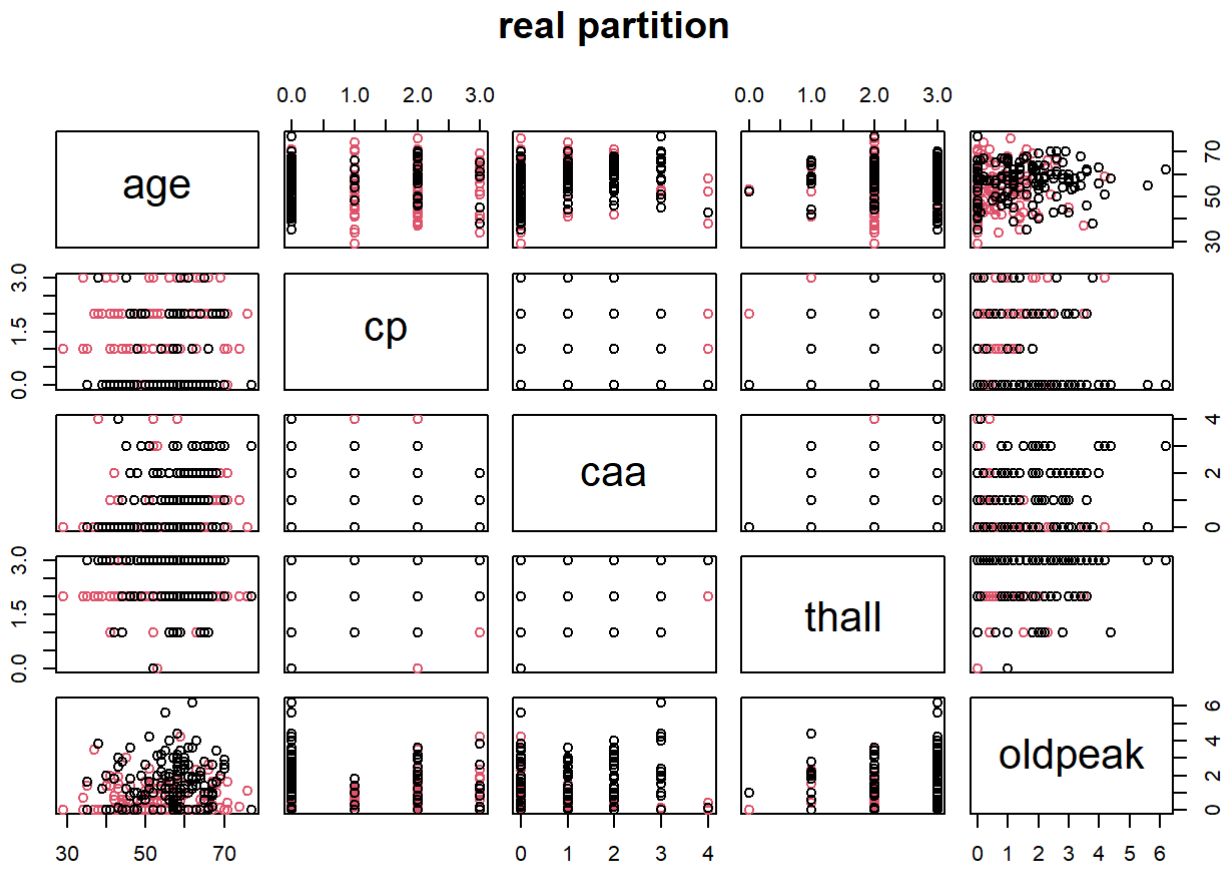
Podem veure la partició en relació a alguns dels atributs més importants:

```
plot(heartDataAux[, c(1, 3, 12:13, 10)], col=heart2clusters$cluster, main="k-means cla
ssification (k=2)")
polygon(heartDataAux[, c(1, 3, 12:13, 10)])
```



I comparar-la amb la partició real gràcies a que coneixem els valors reals de *output*:

```
plot(heart_data[, c(1, 3, 12:13, 10)], col=heart_data$output, main="real partition")
polygon(heart_data[, c(1, 3, 12:13, 10)])
```



En aquest cas, no té gaire sentit calcular la precisió del model, ja que els models de clustering estan pensats per agrupar les dades gràcies a una sèrie de tècniques, però sense que nosaltres sapiguem a priori quants n'hi haurà. En aquest cas, nosaltres sabíem que teníem dos grups diferenciats (els que són diagnosticats i els que no), i per tant els grups trobats per k-means s'hi haurien de semblar, però això no vol dir ni molt menys que aquest model tingui cap validesa per diagnosticar.

5 Resolució del problema i conclusions.

El nostre objectiu era poder predir el diagnòstic final d'un pacient a partir de la resta de variables. Per aquesta raó s'han fet dos models predictius, un de regressió logística i un arbre de decisió, els quals han donat una precisió molt similar.

Tot i això, el model de regressió ha presentat una major sensibilitat que l'arbre (un 77,78% en contra d'un 74,07%), i per això podem afirmar que és millor model. S'ha de tenir present que en aquest cas ens interessa predir el màxim de positius possibles, per així poder tractar al pacient en conseqüència, tot i l'increment que això suposi de falsos positius, els quals en aquest cas tampoc "no farien mal".

Pel que fa a la precisió, segurament s'hauria pogut millorar provant altres algorismes similars per veure quin s'adapta millor al problema, però tal com està ens sembla prou encertada. Segurament també es podria millorar provant altres percentatges en la partició de dades d'entrenament, o inclús jugant amb mètodes de validació creuada com el k-fold.

Finalment, gràcies a la llibreria *randomForest*, hem pogut determinar que els atributs amb més pes a l'hora de diagnosticar al pacient són el *cp* (nivell de mal de pit), el caa (nombre de vasos cardíacs), el *thall* (grau de talassèmia), l'*age* i l'*oldpeak* (la depressió del ST induïda per l'exercici en relació amb el repòs).