DM IntroDataScience

Imad Bendimerad

5/30/2022

Installation des Packages

```
#install.packages("caret")
#install.packages("doParallel")
#install.packages("parallel")
#install.packages("rpart")
#install.packages("rpart.plot")
#install.packages("rmarkdown")
```

Jeu de données

```
rm(list=ls())
cardio <- read.csv("processed.cleveland.data", header = FALSE, na.strings =
'?')
names(cardio) <- c( "age", "sex", "cp", "trestbps", "chol",
                    "fbs", "restecg", "thalach","exang",
                    "oldpeak","slope", "ca", "thal", "status")
```

Lien du site des données de l'étude https://archive.ics.uci.edu/ml/datasets/heart+Disease

1 Appliquer la fonction str au data frame cardio. Les formats des variables du jeu de données sont-ils satisfaisants ?

```
str(cardio)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : num  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex     : num  1 1 1 1 0 1 0 0 1 1 ...
##  $ cp      : num  1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps: num  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol    : num  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs     : num  1 0 0 0 0 0 0 0 0 1 ...
##  $ restecg : num  2 2 2 0 2 0 2 0 2 2 ...
##  $ thalach : num  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang   : num  0 1 1 0 0 0 0 1 0 1 ...
##  $ oldpeak : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope   : num  3 2 2 3 1 1 3 1 2 3 ...
##  $ ca      : num  0 3 2 0 0 0 2 0 1 0 ...
##  $ thal    : num  6 3 7 3 3 3 3 3 7 7 ...
##  $ status  : int  0 2 1 0 0 0 3 0 2 1 ...
```

Les variables “sex”, “cp”, “fbs”, “restecg”, “exang”, “slope”, “thal” et la variable réponse “status” sont codées en tant que variables quantitatives alors qu’elles sont censées être de nature qualitatives (on peut confirmer leurs natures sur le site) et donc les formats des variables des jeux de données ne sont pas satisfaisants.

2 Transformation des variables explicatives qualitatives

Commençons par nous familiariser avec la fonction factor

```
?factor
```

On voit bien l’utilité de cette fonction pour transformer des variables en format “factor”, et on a des exemples de syntaxe.

On cherche à transformer les variables qualitatives au format num en format “factor”:

```
cardio$sex <- factor( cardio$sex,levels = c(0, 1 ),
        labels = c( "Female","Male") )

cardio$cp <- factor( cardio$cp,levels = c(1, 2, 3, 4 ),
        labels = c( "typical angina","atypical angina","non-anginal
pain","asymptomatic") )

cardio$fbs <- factor( cardio$fbs,levels = c(0, 1 ),
                      labels = c( "FBS < 120 mg/dl","FBS > 120 mg/dl") )

cardio$thal <- factor( cardio$thal,levels = c(3,6,7),
                      labels = c( "normal","fixed defect","reversible
defect") )

cardio$slope <- factor( cardio$slope,levels = c(1,2,3),
                       labels = c( "upsloping","flat","downsloping") )

cardio$exang <- factor( cardio$exang,levels = c(0,1),
                        labels = c( "no","yes") )

cardio$restecg <- factor( cardio$restecg,levels = c(0,1,2),
                        labels = c( "normal","ST-T","LVH") )

cardio$status <- factor( cardio$status,levels = c(0, 1,2,3,4 ),
    labels = c("0", "1","2","3","4" ) )
```

On refait un str afin de vérifier la transformation:

```
str(cardio)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : num  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex     : Factor w/ 2 levels "Female","Male": 2 2 2 2 1 2 1 1 2 2 ...
##  $ cp      : Factor w/ 4 levels "typical angina",..: 1 4 4 3 2 2 4 4 4 4
...
##  $ trestbps: num  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol    : num  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs     : Factor w/ 2 levels "FBS < 120 mg/dl",..: 2 1 1 1 1 1 1 1 1 2
...
##  $ restecg : Factor w/ 3 levels "normal","ST-T",..: 3 3 3 1 3 1 3 1 3 3
...
##  $ thalach : num  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang   : Factor w/ 2 levels "no","yes": 1 2 2 1 1 1 1 2 1 2 ...
##  $ oldpeak : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope   : Factor w/ 3 levels "upsloping","flat",..: 3 2 2 3 1 1 3 1 2 3
...
##  $ ca      : num  0 3 2 0 0 0 2 0 1 0 ...
##  $ thal    : Factor w/ 3 levels "normal","fixed defect",..: 2 1 3 1 1 1 1
1 3 3 ...
##  $ status  : Factor w/ 5 levels "0","1","2","3",..: 1 3 2 1 1 1 4 1 3 2
...
```

3 Quelles sont les modalités de la variable réponse ?

```
levels(cardio$status)
```

```
## [1] "0" "1" "2" "3" "4"
```

La variable réponse a 5 modalités.

Depuis l'article de référence on peut lire: " The "goal" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4. Experiments with the Cleveland database have concentrated on simply attempting to distinguish presence (values 1,2,3,4) from absence (value 0). "

Ce qui correspond à la valeur 0 pour l'absence de pathologie cardiaque, et aux valeurs 1;2;3 et 4 pour sa présence.

D'où on comprend qu'il est possible de coder la variable réponse qualitative de façon binaire.

4 Transformation de la variable réponse en variable binaire

```
cardio$status <- factor( cardio$status,levels = c("0","1","2","3","4"),
                         labels = c( "0","1","1","1","1") )

summary(cardio)
```

```
##       age            sex                      cp         trestbps
##  Min.   :29.00   Female: 97   typical angina  : 23   Min.   : 94.0
##  1st Qu.:48.00   Male  :206   atypical angina : 50   1st Qu.:120.0
##  Median :56.00                non-anginal pain: 86   Median :130.0
##  Mean   :54.44                asymptomatic    :144   Mean   :131.7
##  3rd Qu.:61.00                                       3rd Qu.:140.0
##  Max.   :77.00                                       Max.   :200.0
##
##       chol                  fbs        restecg       thalach      exang
##  Min.   :126.0   FBS < 120 mg/dl:258   normal:151   Min.   : 71.0   no
:204
##  1st Qu.:211.0   FBS > 120 mg/dl: 45   ST-T  :  4   1st Qu.:133.5   yes:
99
##  Median :241.0                        LVH   :148   Median :153.0
##  Mean   :246.7                                      Mean   :149.6
##  3rd Qu.:275.0                                      3rd Qu.:166.0
##  Max.   :564.0                                      Max.   :202.0
##
##     oldpeak             slope           ca                        thal
##  Min.   :0.00   upsloping  :142   Min.   :0.0000   normal           :166
##  1st Qu.:0.00   flat       :140   1st Qu.:0.0000   fixed defect     : 18
##  Median :0.80   downsloping: 21   Median :0.0000   reversible defect:117
##  Mean   :1.04                     Mean   :0.6722   NA's             :  2
##  3rd Qu.:1.60                     3rd Qu.:1.0000
##  Max.   :6.20                     Max.   :3.0000
##                                   NA's   :4
##  status
##  0:164
##  1:139
```

Avec Summary on a un résumé de chaque variable.

Pour les variables quantitatives on obtient, la moyenne, la médiane et les quartiles.
tandis que pour les variables qualitatives on a les noms et les effectifs de chaque classe.

5 Afficher le nombre de données manquantes du jeu de données. Écarter les données manquantes à l'aide de la fonction na.omit

Nombre de données manquantes:

```
na_count <- length(which(is.na(cardio)))
print(na_count)
```

```
## [1] 6
```

Utilisation de na.omit afin d'écarter les observations pour lesquelles on a des données manquantes .

```
?na.omit
cardio <- na.omit(cardio)
```

Modelisation

6 Fixer la graine du générateur aléatoire à 1 à l'aide de la fonction set.seed et répartir le jeu de données en jeux de données cardio.train et cardio.test de tailles respectives de 70% et 30%.

Fixation de la graine du générateur aléatoire afin d'avoir un aléa contrôlé et identique pour une valeur donnée.

```
set.seed(1)
```

On extrait 70% des données du jeu de données cardio vers un jeu de données d'apprentissage et les 30% restantes vers un jeu de données test.

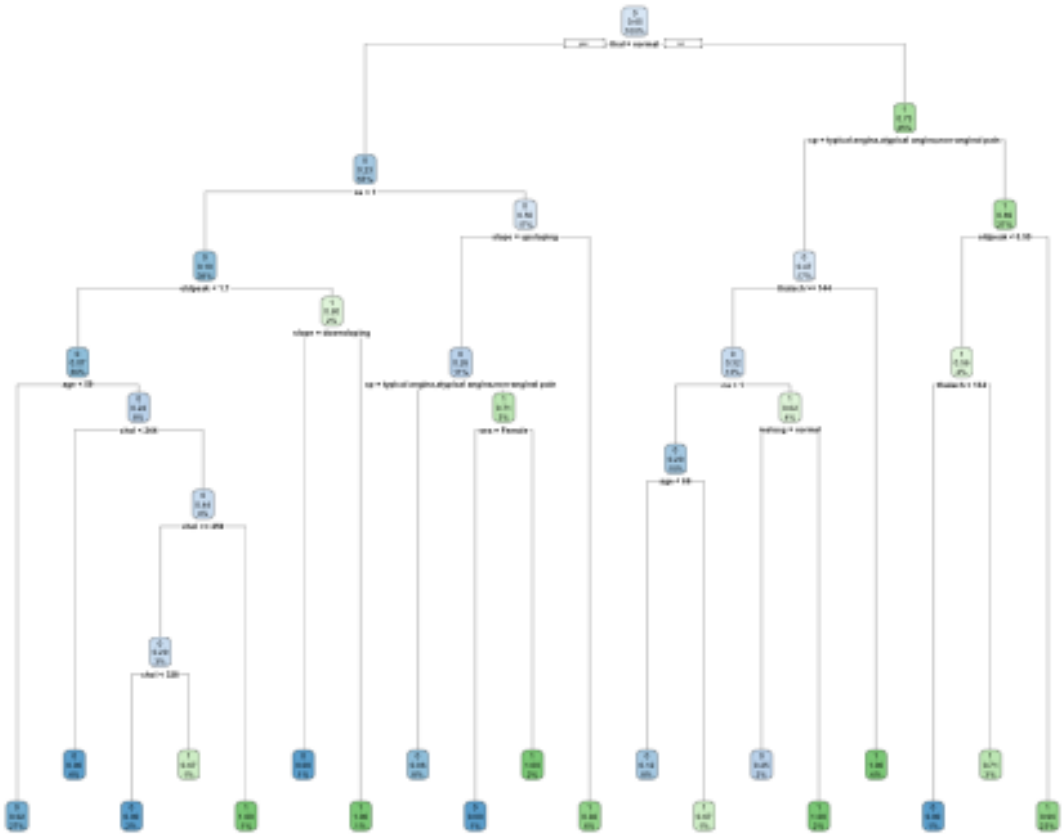
```
train = sample(1:nrow(cardio), round(0.70*nrow(cardio)))
cardio.train = cardio[train,]
cardio.test = cardio[-train,]
```

7 Ajuster, tracer et comparer en terme d'erreur de test les arbres de décisions obtenus respectivement sans et avec élagage.

CART sans élagage

On demande à l'algorithme CART de réaliser au sein de notre arbre de décision, des partitions binaires récursives jusqu'à obtenir un arbre profond, contenant des feuilles avec un nombre minimal d'individus qu'on a fixé à 5:

```
require(rpart)

## Loading required package: rpart

require(rpart.plot)

## Loading required package: rpart.plot

set.seed(1)
cart.0 <- rpart(status~.,
                data=cardio.train,
                control=rpart.control(minsplit=5,cp=0, xval=5))

rpart.plot(cart.0)
```



```
pred.0 <- predict(cart.0, cardio.test, type ="class")
TMC_sansE <- mean(pred.0!=cardio.test$status)
```

Taux de mauvais classements:

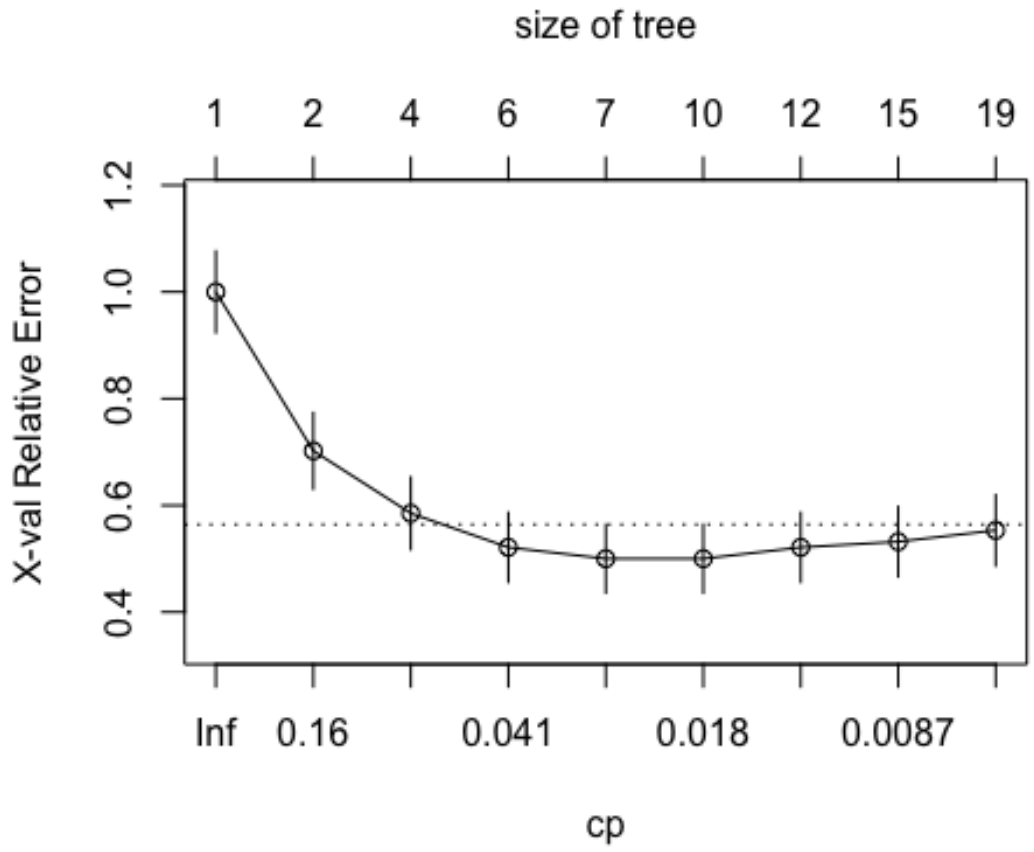
```
print(TMC_sansE)
```

```
## [1] 0.2808989
```

CART avec élagage

Pour faire l'élagage il faut trouver le "alpha" qui minimise l'erreur:

```
plotcp(cart.0)
```




```
cart.0$cptable
```

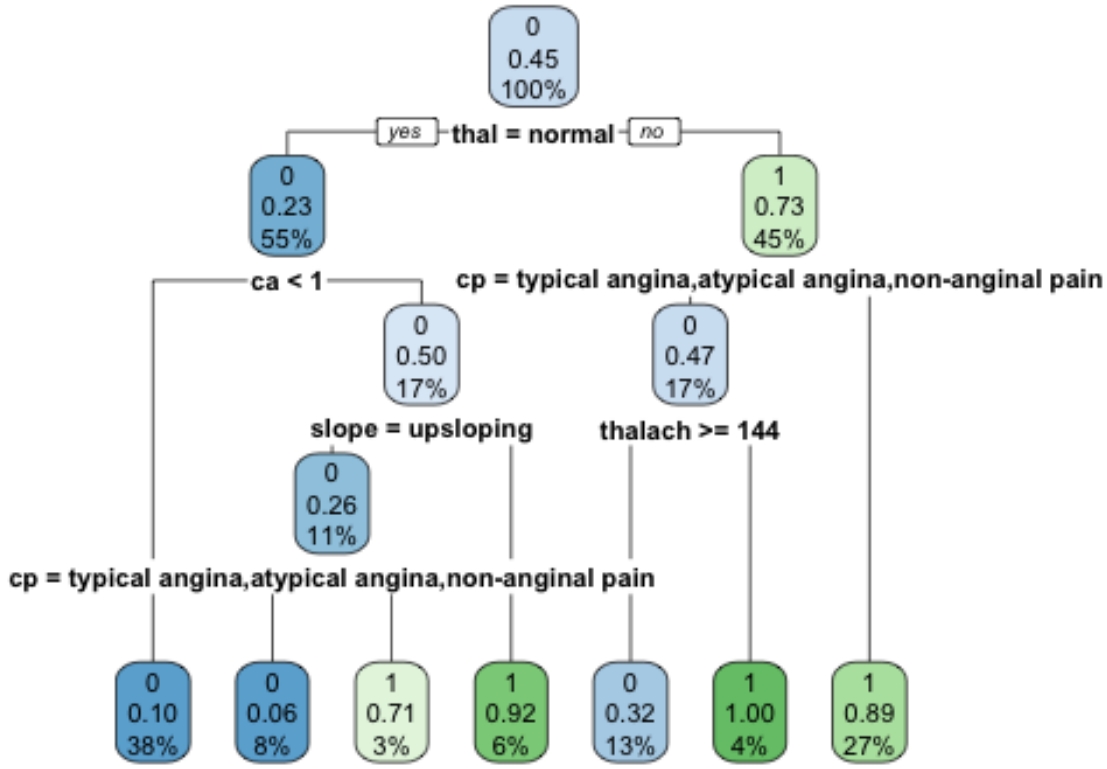
```
##            CP nsplit rel error    xerror       xstd
## 1 0.457446809      0 1.0000000 1.0000000 0.07635840
## 2 0.058510638      1 0.5425532 0.7021277 0.07140960
## 3 0.053191489      3 0.4255319 0.5851064 0.06766554
## 4 0.031914894      5 0.3191489 0.5212766 0.06510841
## 5 0.021276596      6 0.2872340 0.5000000 0.06416561
## 6 0.015957447      9 0.2234043 0.5000000 0.06416561
## 7 0.010638298     11 0.1914894 0.5212766 0.06510841
## 8 0.007092199     14 0.1595745 0.5319149 0.06556228
## 9 0.000000000     18 0.1276596 0.5531915 0.06643615
```

```
which.min(cart.0$cptable[,"xerror"])
```

```
## 5
## 5
```

On utilise cet alpha, correspondant au CP de la 5eme ligne, pour pénaliser notre modèle et construire un modèle d'arbre de décision avec élagage:

```
cpOptim = cart.0$cptable[which.min(cart.0$cptable[,"xerror"]),"CP"]
cart.pruned <- prune(cart.0, cpOptim)
rpart.plot(cart.pruned)
```




```
pred.pruned <- predict(cart.pruned, cardio.test, type ="class")
TMC_avecE <- mean(pred.pruned!=cardio.test$status)
```

Taux de mauvais classements:

```
print(TMC_avecE)
```

```
## [1] 0.2134831
```

Comparaisons

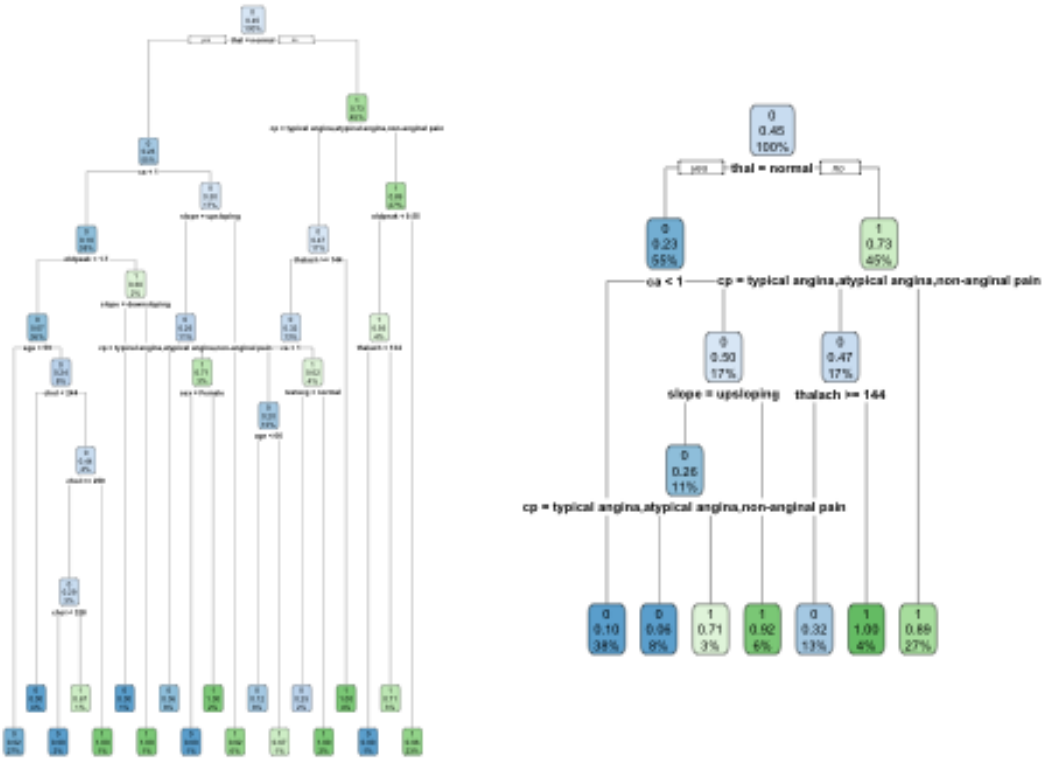
Comparaison des erreurs de test des 2 modèles

```
print(TMC_sansE)

## [1] 0.2808989

print(TMC_avecE)

## [1] 0.2134831
```

On constate que le taux de mauvais classements de l'arbre de décision avec élagage est inférieur à celui de l'arbre de décision sans élagage.

les 2 arbres avec et sans élagage côte à côte

```
par(mfrow = c(1, 2))
rpart.plot(cart.0)
rpart.plot(cart.pruned)
```



On peut facilement voir que l'arbre avec élagage contient moins de feuilles que l'arbre sans élagage.

Afficher l'importance relative de chaque variable explicative du modèle dans la prédiction de la variable réponse :

R nous affiche par ordre décroissant l'importance de chaque variable explicative dans la prédiction de la variable réponse pour le modèle d'arbres de décision sans élagage:

```
cart.0$variable.importance

##   thalach      thal   oldpeak     slope        cp     exang        ca
sex
## 27.324008 26.952113 21.217215 18.147456 15.532239 10.639821 10.488346
10.422207
##       age      chol  trestbps   restecg       fbs
##  7.452678  7.182359  3.638883  2.323045  0.627451

cart.0$variable.importance/sum(cart.0$variable.importance)

##     thalach        thal     oldpeak       slope          cp       exang
## 0.168721058 0.166424674 0.131012659 0.112057428 0.095908909 0.065699067
##          ca         sex         age        chol    trestbps     restecg
## 0.064763739 0.064355341 0.046019006 0.044349832 0.022469481 0.014344404
##         fbs
## 0.003874402
```

On peut dire que dans cet arbre sans élagage, environs 16.9% de la prédiction de la présence d'une pathologie cardiaque est expliquée par la variable thalach "maximum heart rate achieved", et que la somme des diminutions du taux de mauvais classement apportés par les partitions de cette variable est de 27.324008, ainsi de suite pour les autres variables, pour lesquels on remarque une baisse progressive de leurs pourcentages et donc de leurs importance.

Même chose pour le modèle d'arbres de décision avec élagage:

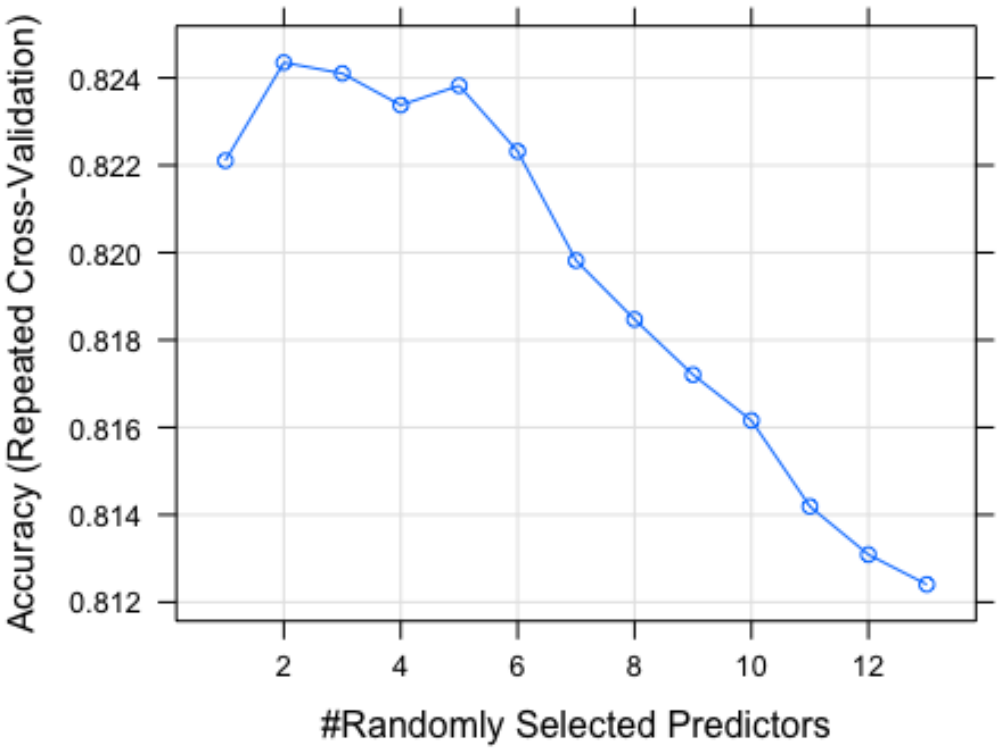
```
cart.pruned$variable.importance

##      thal   thalach     slope   oldpeak        cp     exang        ca
sex
## 26.952113 21.929149 15.747456 14.350277 14.133137 10.639821  8.424061
7.052646
##       age  trestbps
##  2.622421  2.276267

cart.pruned$variable.importance/sum(cart.pruned$variable.importance)

##       thal    thalach      slope    oldpeak         cp      exang
ca
## 0.21713276 0.17666654 0.12686532 0.11560931 0.11385998 0.08571697
0.06786627
##        sex        age   trestbps
## 0.05681783 0.02112686 0.01833816
```

On remarque qu'avec l'élagage on a des variables exclues, et l'importance des variables change.

8 Peut-on améliorer les performances de prédiction à l'aide d'une forêt aléatoire à 500 arbres ?

On réalise une forêt aléatoire grâce à la l'algorithme "caret" avec sa fonction trainControl qui fait une validation croisée répétée, tout en parallélisant les calculs sur les différents coeurs du CPU avec le package "doParallel":

```
require(caret)

require(parallel)

require(doParallel)

cl <- makePSOCKcluster(detectCores()-1)
registerDoParallel(cl)
control <- trainControl(method="repeatedcv", number=5, repeats=100)
rfGrid <-  expand.grid(mtry = 1:13)
RFmodel <- train(x = cardio.train[,-14],
                 y = cardio.train$status,
                 method="rf",
                 trControl=control,
                 n.trees=500,
                 tuneGrid = rfGrid)
stopCluster(cl)
plot(RFmodel)
```




```
print(RFmodel)
```

```
## Random Forest
##
## 208 samples
##  13 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 100 times)
## Summary of sample sizes: 167, 166, 166, 166, 167, 166, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    1    0.8221052  0.6350117
##    2    0.8243512  0.6418255
##    3    0.8241012  0.6421732
##    4    0.8233765  0.6412675
##    5    0.8238213  0.6423594
##    6    0.8223184  0.6395555
##    7    0.8198187  0.6345541
##    8    0.8184715  0.6320842
##    9    0.8172052  0.6296769
##   10    0.8161575  0.6276191
```

```
##   11    0.8141901  0.6236494
##   12    0.8130890  0.6215111
##   13    0.8124045  0.6202321
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Après stabilisation du résultat, on voit que **l'hyperparamètre mtry=2** est celui qui a la meilleure “accuracy” et donc celui qui minimise le taux de mauvais classement. On utilisera cette valeur de l'hyperparamètre pour calculer le taux de mauvais classement du modèle de Forêt Aléatoire.

```
pred.rf.caret <- predict(RFmodel, cardio.test)
TMC_RF <- mean(pred.rf.caret!=cardio.test$status)
```

Taux de mauvais classements:

```
print(TMC_RF)
```

```
## [1] 0.1797753
```

On compare les taux de mauvais classements entre nos 3 modèles:

```
TMC_avecE<TMC_sansE
```

```
## [1] TRUE
```

```
TMC_RF<TMC_avecE
```

```
## [1] TRUE
```

Malgré le fait qu’on ne peut visualiser le RandomForest par un plot comme pour les arbres décisionnels, ce modèle reste très intéressant pour la minimisation de la variance en faisant **un vote majoritaire** des arbres non ou faiblement corrélés, générés en fonction d’un nombre de variables explicatives choisies au hasard parmis les p variables explicatives (p=13 ici) en utilisant la fonction mtry.

On peut voir que le RandomForest de 500 arbres **peut améliorer les performances de prédiction face aux arbres de décision avec ou sans élagage**, car il a un taux de mauvais classement encore plus réduit.