Arbres de décision

La base de données étudiée pour ce TP est un ensemble d'informations sur les clients d'une banque, qui leur a (ou non) attribué un prêt. Le but est de prédire la "créditabilité" du client.

1 Importation des données

Deux jeux de données sont disponibles en ligne : un jeu de donnée d'apprentissage, que vous utiliserez pour construire un classifieur, et un jeu de données de test, que vous utiliserez pour tester ce dernier. Ce jeu de données de test correspond à des nouveaux clients pour lesquels vous devez faire des prédictions.

Importer les données à l'aide de la commande suivante.

```
train <-read.csv("german_credit_train.csv")
train <- train[2:dim(train)[2]]
test <-read.csv("german_credit_test.csv")
test <- test[2:dim(test)[2]]
```

Toutes les variables explicatives qualitatives sont codées comme des entiers, ce qui fait sens pour des variables ordinales (par exemple `Account.Balance`, qui se réfère à une quantité d'argent sur un compte), mais pas pour des variables nominales (par exemple Sex and Marital Status). Vous pourrez donc transformer en facteur (à l'aide de la commande `as.factor`) les variables qui vous paraissent nominales.

Vous trouverez des détails sur les différentes variables et le jeu de données ici :

`https://onlinecourses.science.psu.edu/stat857/node/216`.

2 Arbres de décisions

Deux packages existent sous R pour les arbres de décision : `tree` et `rpart`. Dans ce TP nous verrons l'utilisation du premier, dont l'aide complète peut être trouvée ici : `https://cran.r-project.org/web/packages/tree/tree.pdf`.

2.1 Construction de l'arbre complet

La commande

```
cart.pred <- tree(factor(Creditability) ~ . ,data=smalltrain,
    control=tree.control(nTrain, mincut = 15, minsize = 30,
    mindev =0),split="deviance")
```

construit un objet de type "tree", qui correspond à l'arbre de classification complet, que l'on peut visualiser de différentes manières à l'aide des commandes

```
summary(cart.pred)
cart.pred
plot(cart.pred)
text(cart.pred, pretty=0,cex=0.4)
```

On peut ensuite s'en servir pour la prédiction sur l'ensemble de test à l'aide de la fonction predict.

```
predicted <- predict(cart.pred, newdata=test, type="class")
table(predicted,test$Creditability)
```

A l'aide du manuel, et d'essais, bien comprendre le rôle de chacun des paramètre et les différentes options possibles (ex. choix du critère d'impureté, nombre minimal d'observations par feuille, etc.).

2.2 Elagage de l'arbre

L'élagage s'effectue avec la commande `prune.tree`, et peut se faire soit en fixant un paramètre de pénalité `k`, soit en fixant un ordre de grandeur pour le nombre total de feuilles, `best`. Et bien sûr, en précisant le critère d'impureté utilisé pour l'élagage.

```
pruned <- prune.tree(cart.pred,k=2.3,method = "deviance")
pruned <- prune.tree(cart.pred,best=8,method = "deviance")
```

2.3 Choix optimal du paramètre d'élagage

L'algorithme CART dépend du choix du paramètre d'élagage (k ou best selon l'approche adoptée), qui ne peut pas être choisi sur l'ensemble d'apprentissage. On séparera donc la base de données `train` en un ensemble d'apprentissage `smalltrain` et un ensemble de validation `valid`.

Construire un arbre de décision complet en se basant sur les données de `smalltrain`, et effectuer la sélection du paramètre optimal sur l'ensemble de validation.

2.4 Validation croisée

La procédure de choix de paramètre effectuée est basée sur une seule division de l'ensemble d'apprentissage. La technique de *validation croisée* V-fold est plus robuste, car elle moyenne les résultats sur plusieurs choix d'ensembles de validation. Voici un exemple d'implémentation.

```
n <- dim(train)[1]; V <- 5;
I <- array(0, dim=c(V, n/V))

# Indices des ensembles de validation successifs
reste <- 1:n
for (i in 1:V){
  I[i,] <- sample(reste, n/V)
  reste <- setdiff(reste, I[i,])}
# Valeurs du parametre "best"
Ibest <- c(2:25); N <- length(Ibest);
scor <- array(0, N);

for (i in 1:V){
  cv_valid <- train[I[i,], ];cv_train <- train[-I[i,], ];
  cart <- tree(factor(Creditability) ~ .,data=cv_train)
  for (k in 1:N){b <- Ibest[k];
    pruned <- prune.tree(cart, best = b, method = "misclass")
    predictions <- predict(pruned, newdata=cv_valid, type="
    class")
    scor[k] <- scor[k] + mean(predictions != cv_valid$
    Creditability)}}
```

3 Evaluation du classifieur

On peut tout d'abord mesurer l'erreur de test des trois arbres construits : arbre complet construit à l'aide de `train`, arbre élagué "optimalement" à l'aide d'un ensemble de validation ou par validation croisée.

Du point de vue de la banque attribuer un crédit à quelqu'un qui ne pourra pas le rembourser (prédire une Créditabilité de 1 alors qu'elle est de 0) est plus grave que ne pas attribuer un crédit à quelqu'un qui pourra le rembourser (prédire 0 au lieu de 1). Cela peut se traduire par l'introduction d'un coût de nos décisions, par exemple :

— chaque "vrai positif" rapporte +0.5 à la banque

— mais chaque "faux positif" rapporte -1 à la banque

On pourra ainsi calculer, sur l'ensemble de test, ce que notre classifieur a fait gagner à la banque.

De manière générale pour la classification binaire, il est important d'aller au-delà de l'erreur de test. La courbe ROC (package `ROCR`) affiche ainsi le taux de vrai positifs en fonction du taux de faux positifs.

```
predictProba <- predict(tree1,newdata=test,type="vector")[,2]
pr <- prediction(predictProba, test$Creditability)
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf)
```

4 Autres méthodes

Tester les autres approches déjà vues : régression logistique, k-plus proches voisins. Essayer les svm, contenus dans le package `e1071` : `https://cran.r-project.org/web/packages/e1071/e1071.pdf`.

```
svm.fit <- svm(factor(Creditability) ~ ., data=train)
```