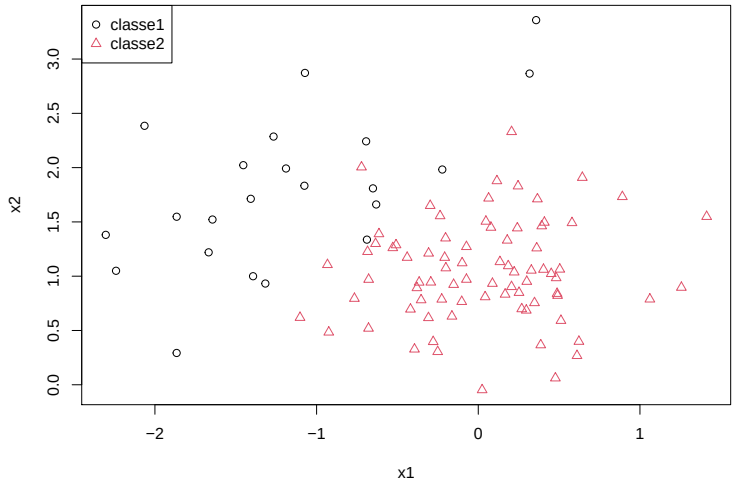
Apprentissage automatique

TP8 : arbres de classification

Exercice 1. On utilise dans cet exercice les données synth_train.txt et synth_test.txt.



1. Lire les aides sur les fonctions `rpart` et `rpart.control` du package `rpart`.

```
library(rpart)
help(rpart)
help(rpart.control)
```

Quelle est par défaut la matrice de coûts ? La fonction d'impureté ? Les critères d'arrêt ? Comment sont gérées les données manquantes ? Quelle est la différente entre une question `competitor` et une question `surrogate` ?

2. Charger les données d'apprentissage.

```
train <- read.table(file="../data/synth_train.txt", header=TRUE)
```

Appliquer la fonction `rpart` aux données d'apprentissage.

```
tree <- rpart(y~., data=train, method="class",
              control=list(maxcompete=0))
```

Expliquer les résultats de la fonction `print`

```
print(tree)
```

Puis visualiser l'arbre de classification.

```
library(rpart.plot)
rpart.plot(tree, extra = 1)
```

Quel est le taux d'erreur d'apprentissage ?

3. Expliquer les résultats de la fonction `summary`. Pourquoi l'algorithme s'arrête ici après une seule division ?

```
summary(tree)
```

4. Charger le jeu de données test.

```
test <- read.table(file="../data/synth_test.txt", header=TRUE)
```

Prédire les données test avec cet abre de classification.

```
pred.test <- predict(tree, newdata=test, type="class")
```

Quel est le taux d'erreur test ?

5. Estimer les probabilités à posteriori des données test.

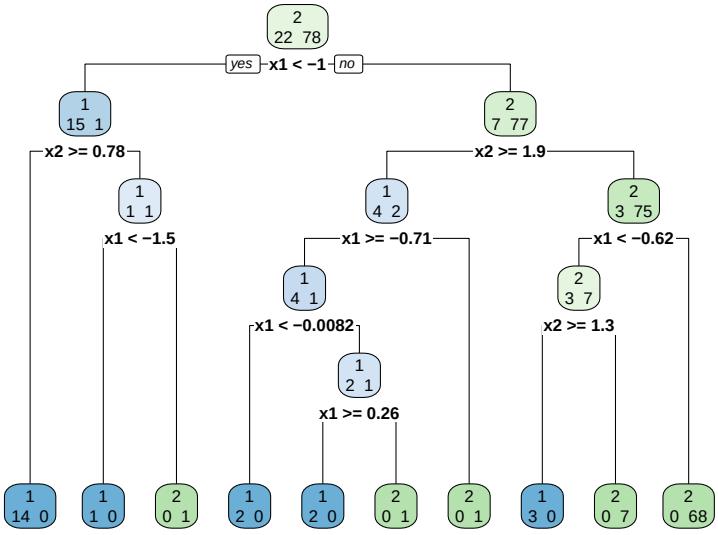
```
prob <- predict(tree, newdata=test, type="prob")
```

Pourquoi les probabilités à posteriori des premières données test sont toutes identiques ?

```
head(prob)
```

6. Construire maintenant l'arbre de longueur maximale avec comme seul critère d'arrêt `minsplit`=2.

```
tree <- rpart(y~., data=train, minsplit=2, cp=0,
              method="class",
             control=list(maxcompete=0))
```



Retracer à la main cet arbre en utilisant la numérotation des noeuds de la fonction `summary`, sans les questions binaires mais en indiquant à chaque noeud la valeur du paramètre de complexité, le nombre d'observations et leur répartition dans chaque classe (en souligant le nombre de mals classés).

```
summary(tree)
```

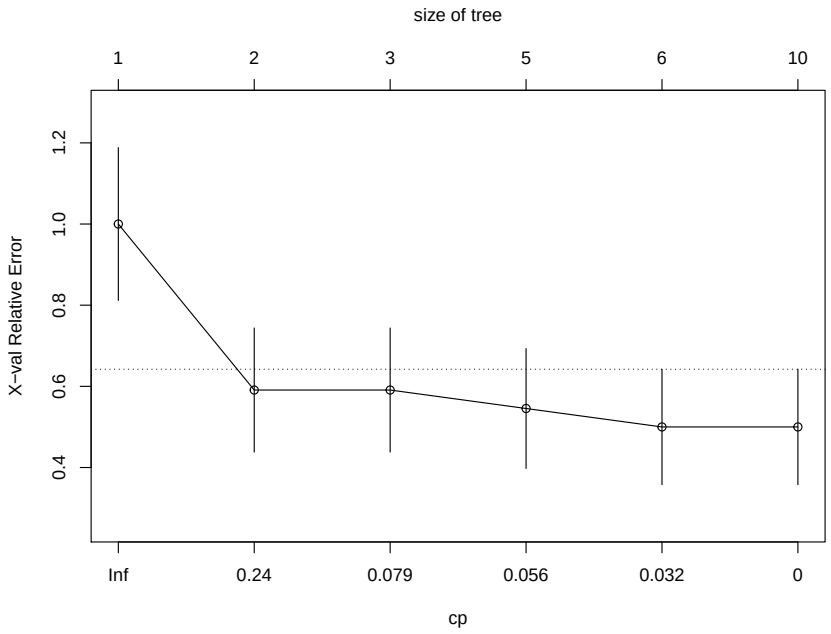
7. Expliquer les résultats de la fonction `printcp`.

```
printcp(tree)
#tree$cptable
```

Quels sont les sous-arbres élagués associés ?

8. Executer le code ci-dessous :

```
plotcp(tree)
```



Retrouver les valeur en abscisse de ce graphique à partir des paramètres de complexités des noeuds de l'arbre.

Expliquer le lien entre la taille de l'arbre (axe du haut) et la valeur du paramètre de complexité (axe du bas).

Expliquer les valeurs représentées en ordonnées. Expliquer la signification de la ligne pointillée.

9. Choisir une valeur du paramètre de complexité et élaguer l'arbre avec la fonction `prune`.

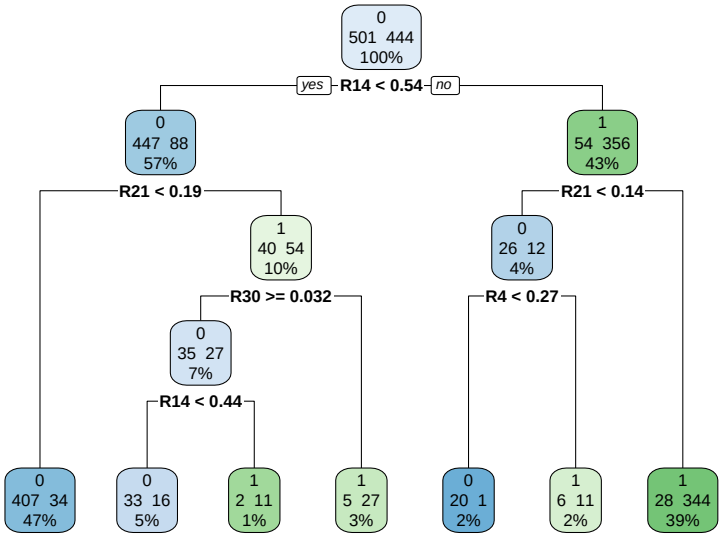
Exercice 2. On reprend le jeu de données où 1260 exploitations agricoles saines ou défaillantes sont décrites par $p = 22$ critères économiques et financiers.

```
load("../data/Desbois_complet.rda")
```

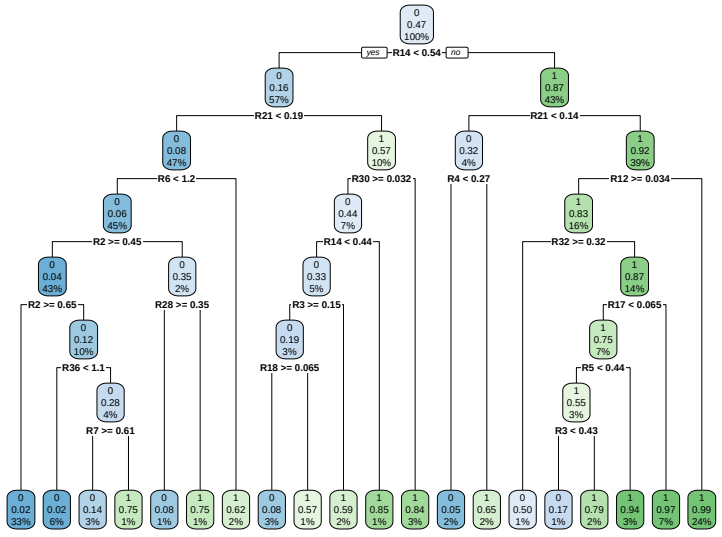
1. Découper aléatoirement les $n = 1260$ exploitations agricoles en 945 exploitations pour les données d'apprentissage et 360 exploitations pour les données test.

```
set.seed(10)
tr <- sample(1:nrow(data), 945)
train <- data[tr,]
test <- data[-tr,]
```

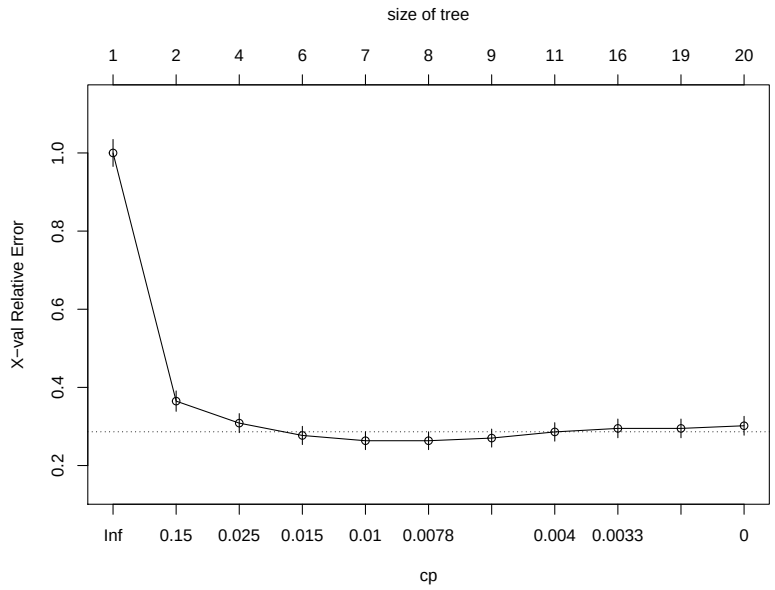
2. Constuire l'arbre de classification en laissant les valeurs par défaut. Visualisez l'arbre avec la fonction `rpart.plot`.



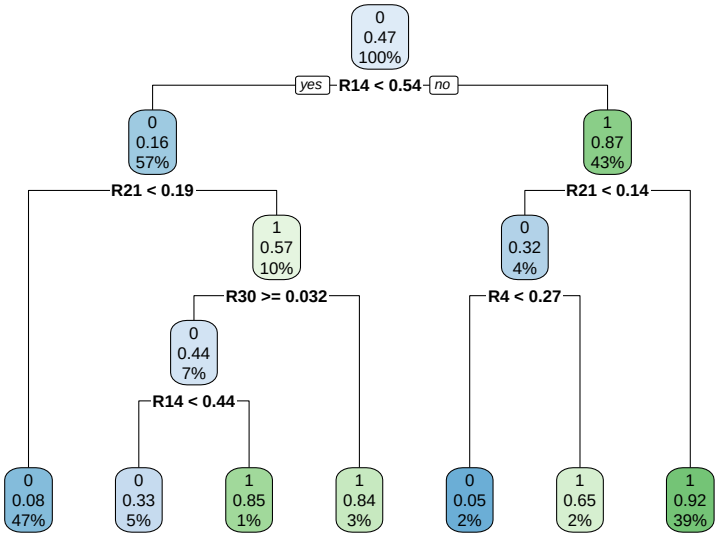
3. Constuire maintenant l'arbre de longeur maximale.



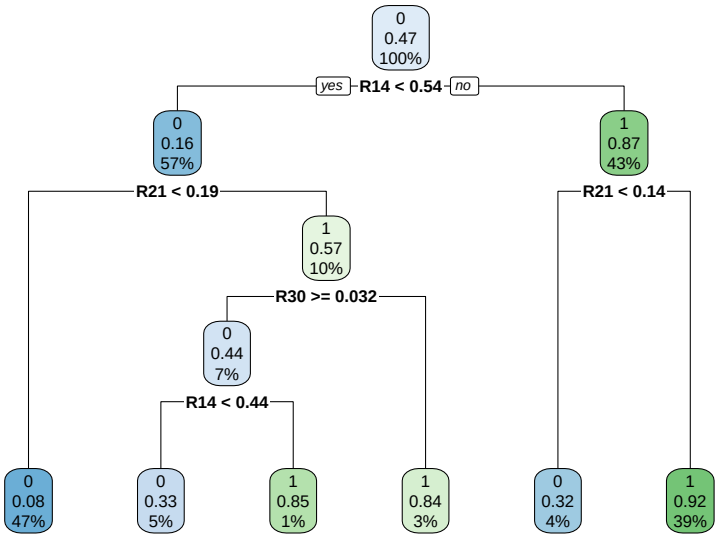
4. Faire le graphique des erreurs de validation de la suite des sous-arbres optimaux. Recommencez plusieurs fois. Que constatez-vous ? Quelle valeur du paramètre de complexité pourriez-vous choisir pour élaguer cet arbre ?



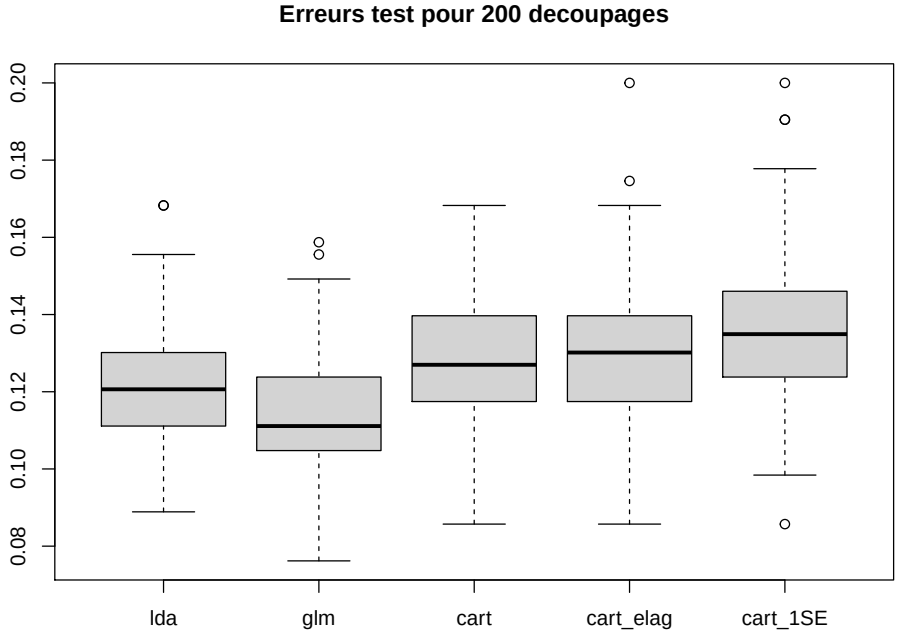
5. Elaguer cet arbre en choisissant **automatiquement** la valeur `cp.min` du paramètre de complexité.



6. Elaguer cet arbre en choisissant automatiquement la valeur `cp.1se` du paramètre de complexité.



7. Prédire les données test. Quel est le taux d'erreur ?
8. Tracer la courbe ROC et calculer le AUC.
9. Comparer à partir de $B = 200$ découpages aléatoires des données (945 observations d'apprentissage et 315 observations test à chaque découpage), le taux d'erreur des méthodes `lda`, `glm` (pour la régression logistique, CART (avec le paramétrage par défaut), CART avec élagage (avec `cp.min`) CART avec élagage et `cp.1se`.



Conclure.