TP4

Le principal objectif de ce TP est la manipulation des arbres de décision avec R. Pour cela, on va utiliser les bibliothèques « *party* » et « *rpart* » qui contiennent les fonctions nécessaires.

Exercice 1 (en utilisant "rpart")

1. Ouvrez WEKA et charger le fichier « iris.arff », puis essayer de le convertir en format csv sous le nom « iris.arff.csv ».
2. Ouvrez R et installez le package « *rpart* » avec > install.packages("rpart")
3. Chargez la bibliothèque avec library("rpart")
4. Chargez le jeu de données > iris <- read.csv("iris.arff.csv")
    - getwd() : pour savoir le répertoire courant
    - setwd("//home//… ") : pour definer le repertoire courant
5. Visualisez le jeu de données : > iris
6. Spécifiez le nombre minimal d'exemples nécessaires à la création d'un nœud (par defaut 20) :
   > dt.iris.cnt <- rpart.control (minsplit = 1)
   La variable *dt.iris.cnt* stocke les paramètres de l'algorithme.
7. construire l'arbre de décision en indiquant :
    - l'attribut classe à prédire (class)
    - les attributs qui doivent être utilisés pour effectuer la prédiction (on va utiliser tous les autres attributs)
    - le jeu d'exemples avec lequel on construit l'arbre (iris)
    - le nom de la variable qui contient les paramètres (dt.iris.cnt)

```
> dt.iris <- rpart(class ~ sepallength + sepalwidth + petallength + petalwidth, iris, control = dt.iris.cnt)
```

8. Visualisez l'arbre construit (Représentation textuelle) : > dt.iris
9. Visualisez l'arbre construit (Représentation graphique) :
    - Dessiner l'arbre avec : > plot (dt.iris)
    - Afficher le texte dans les noeurds et sur les branches avec : > text (dt.iris)
10. Vérifiez la compatibilité de ce que vous voyez graphiquement avec ce que vous aviez compris de la représentation textuelle
11. Essayez les paramètres graphiques suivants :
    - plot (dt.iris, uniform=T)
    - text (dt.iris, use.n=T, all=T)
    - plot (dt.iris, branch=0)
    - plot (dt.iris, branch=.7)
    - text (dt.iris, use.n=T)

- plot (dt.iris, branch=.4, uniform=T, compress=T)
- text (dt.iris, all=T,use.n=T)
- plot (dt.iris, branch=.2, uniform=T, compress=T, margin=.1)
- text (dt.iris, all=T, use.n=T, fancy=T)

12. Essayez l'élagage automatique de l'arbre avec (cp est le paramètre de complexité par defaut 0.01) : >dt.iris.optimal <- prune (dt.iris, cp=0.02)
    Réglage de la complexité de l'arbre : plus l'arbre est complexe (beaucoup de noeuds), plus il va bien apprendre l'échantillon d'apprentissage, mais aussi risque le sur-apprentissage.
13. Visualisez et comparez l'arbre avant et après l'élagage
    - > dt.iris
    - > dt.iris.optimal
14. Par défaut, rpart() effectue un élagage de l'arbre et une validation croisée à 10 plis sur chaque arbre élagué. Les mesures effectuées au long de cette procédure sont stockées dans une table dénommée la cptable. Observez les résultats de la commande : > dt.iris$cptable
    xerror mesure le taux d'erreur dans la validation croisée à 10 CV. xstd est l'écart-type de l'erreur de validation croisée. L'arbre qui nous intéresse est celui qui minimise xerror + xstd.

Exercice 2 (en utilisant "party")

1. Ouvrez R et installez le package « *party* » avec > install.packages("party")
2. Chargez la bibliothèque avec library("party")
3. Chargez le jeu de données > iris <- read.csv("iris.arff.csv")
4. Divisez le jeu de donnée iris en 70% apprentissage et 30% test
    - > str(iris)
    - > set.seed(1234)
    - > ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.7, 0.3))
    - > trainData <- iris[ind==1,]
    - > testData <- iris[ind==2,]
5. Construire l'arbre de decision :
    > iris_ctree <- ctree(class ~ sepallength + sepalwidth + petallength + petalwidth, data=trainData)
6. Vérifiez la prédiction de l'arbre construite sur les données d'apprentissage :
    > table(predict(iris_ctree), trainData$class)
7. Visualisez l'arbre construit (Représentation textuelle puis graphique)
    - > print (iris_ctree)
    - > plot (iris_ctree)
    - > plot (iris_ctree, type="simple")
8. Testez l'arbre construit sur les données de test :
    > testPred <- predict(iris_ctree, newdata = testData)
9. Affichez le résultat de la prédiction : > table(testPred, testData$class)