DM

Mélissa EVEILLARD

20/05/2020

Exercice 2

1.

On importe le jeu de données `tennis` et la libairie `rpart`.

```
load(file = "tennis.RData")
library(rpart)

# install.packages("rpart.plot")
library(rpart.plot) # Pour la représentation de l'arbre de décision
```

```
arbre = rpart(Jouer ~ ., data = tennis, method = "class", control=rpart.control(minsplit=4,cp=0))
summary(arbre)
```

```
## Call:
## rpart(formula = Jouer ~ ., data = tennis, method = "class", control = rpart.control(minsplit = 4,
##     cp = 0))
##   n= 14
##
##    CP nsplit rel error xerror      xstd
## 1 0.3      0       1.0    1.0 0.3585686
## 2 0.2      2       0.4    2.0 0.3380617
## 3 0.0      4       0.0    1.2 0.3703280
##
## Variable importance
## Temperature    Humidite        Ciel
##          56          22          22
##
## Node number 1: 14 observations,    complexity param=0.3
##   predicted class=Oui  expected loss=0.3571429  P(node) =1
##     class counts:     5     9
##    probabilities: 0.357 0.643
##   left son=2 (10 obs) right son=3 (4 obs)
##   Primary splits:
##       Ciel        splits as  RLL,       improve=1.4285710, (0 missing)
##       Humidite    < 82.5  to the right, improve=1.2857140, (0 missing)
##       Temperature < 27    to the right, improve=0.8901099, (0 missing)
##       Vent        splits as  RL,        improve=0.4285714, (0 missing)
##   Surrogate splits:
##       Temperature < 25.25 to the left,  agree=0.786, adj=0.25, (0 split)
##
## Node number 2: 10 observations,    complexity param=0.3
```

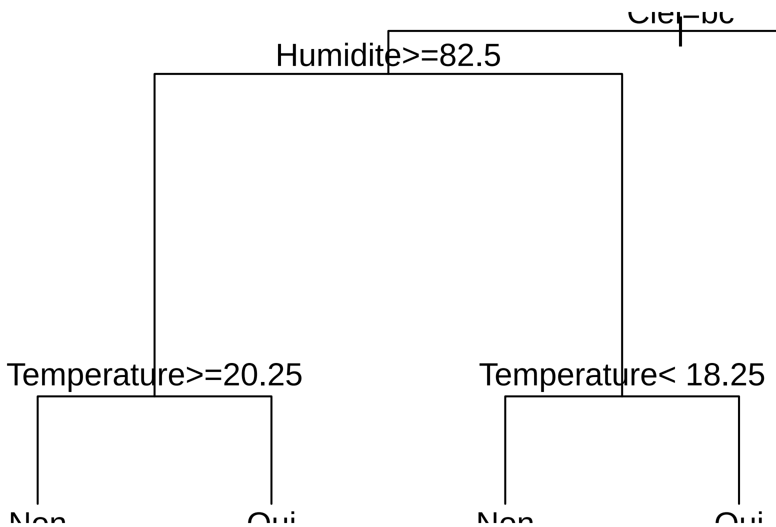
```
##   predicted class=Non  expected loss=0.5  P(node) =0.7142857
##     class counts:     5     5
##    probabilities: 0.500 0.500
##   left son=4 (5 obs) right son=5 (5 obs)
##   Primary splits:
##       Humidite    < 82.5  to the right, improve=1.8000000, (0 missing)
##       Temperature < 23.75 to the right, improve=1.2500000, (0 missing)
##       Vent        splits as  RL,        improve=0.8333333, (0 missing)
##       Ciel        splits as  -LR,       improve=0.2000000, (0 missing)
##   Surrogate splits:
##       Temperature < 19.75 to the right, agree=0.8, adj=0.6, (0 split)
##       Ciel        splits as  -LR,       agree=0.6, adj=0.2, (0 split)
##
## Node number 3: 4 observations
##   predicted class=Oui  expected loss=0  P(node) =0.2857143
##     class counts:     0     4
##    probabilities: 0.000 1.000
##
## Node number 4: 5 observations,    complexity param=0.2
##   predicted class=Non  expected loss=0.2  P(node) =0.3571429
##     class counts:     4     1
##    probabilities: 0.800 0.200
##   left son=8 (4 obs) right son=9 (1 obs)
##   Primary splits:
##       Temperature < 20.25 to the right, improve=1.6000000, (0 missing)
##       Humidite    < 95.5  to the left,  improve=1.6000000, (0 missing)
##       Ciel        splits as  -LR,       improve=0.6000000, (0 missing)
##       Vent        splits as  RL,        improve=0.2666667, (0 missing)
##
## Node number 5: 5 observations,    complexity param=0.2
##   predicted class=Oui  expected loss=0.2  P(node) =0.3571429
##     class counts:     1     4
##    probabilities: 0.200 0.800
##   left son=10 (1 obs) right son=11 (4 obs)
##   Primary splits:
##       Temperature < 18.25 to the left,  improve=1.6000000, (0 missing)
##       Vent        splits as  RL,        improve=0.6000000, (0 missing)
##       Ciel        splits as  -RL,       improve=0.2666667, (0 missing)
##       Humidite    < 75    to the left,  improve=0.2666667, (0 missing)
##
## Node number 8: 4 observations
##   predicted class=Non  expected loss=0  P(node) =0.2857143
##     class counts:     4     0
##    probabilities: 1.000 0.000
##
## Node number 9: 1 observations
##   predicted class=Oui  expected loss=0  P(node) =0.07142857
##     class counts:     0     1
##    probabilities: 0.000 1.000
##
## Node number 10: 1 observations
##   predicted class=Non  expected loss=0  P(node) =0.07142857
##     class counts:     1     0
##    probabilities: 1.000 0.000
```

```
##
## Node number 11: 4 observations
##   predicted class=Oui  expected loss=0  P(node) =0.2857143
##     class counts:     0     4
##    probabilities: 0.000 1.000
```

2.

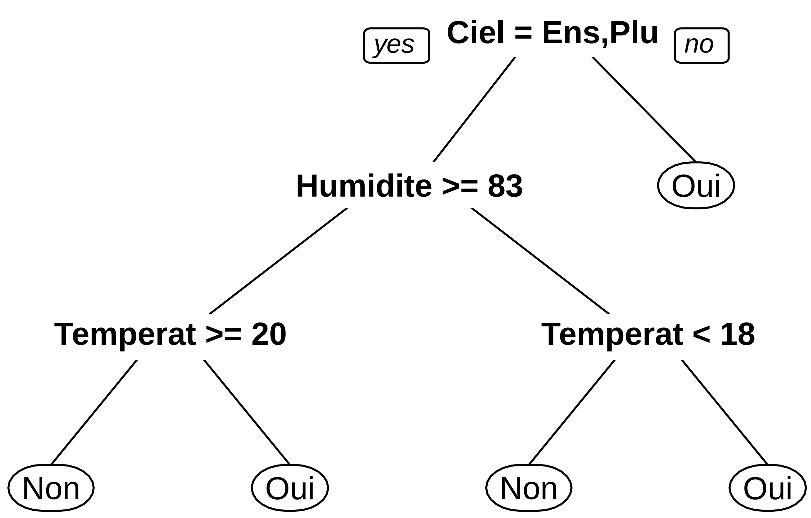
Pour visualier le résultat renvoyé par la fonction `rpart` on peut utiliser les fonctions suivantes :

```r
plot(arbre)
text(arbre)
```



Egalement, nous pouvons utiliser la commande `prp` du package `rpart.plot`

```r
prp(arbre)
```



3.

On utilise le classifieur que l'on vient de construire pour faire de la prédiction.

```
tennis_predict = predict(arbre, newdata = tennis, type = "class")
```

On affiche ci-dessous la matrice de confusion :

```
tab = table(tennis$Jouer, tennis_predict)
tab
```

```
##      tennis_predict
##       Non Oui
##   Non   5   0
##   Oui   0   9
```

L'erreur d'apprentissage est donc :

```
erreur_app = (tab[1,2]+tab[2,1])/sum(tab)
erreur_app
```

```
## [1] 0
```

On obtient un taux d'erreur d'apprentissage nul.

Exercice 3

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

On travaille à présent avec le jeu de données *iris*.

```
data("iris")
```

1.

On sépare nos données en un jeu de données d'apprentissage et un autre de test.

```
n = nrow(iris)
data_train = sample(1:n, floor(n*0.75)) #Nombre de lignes de l'échantillon d'apprentissage : 75% du dat

train = iris[data_train,] #Echantillon d'apprentissage
test = iris[-data_train,] #Echantillon de test
```

2.

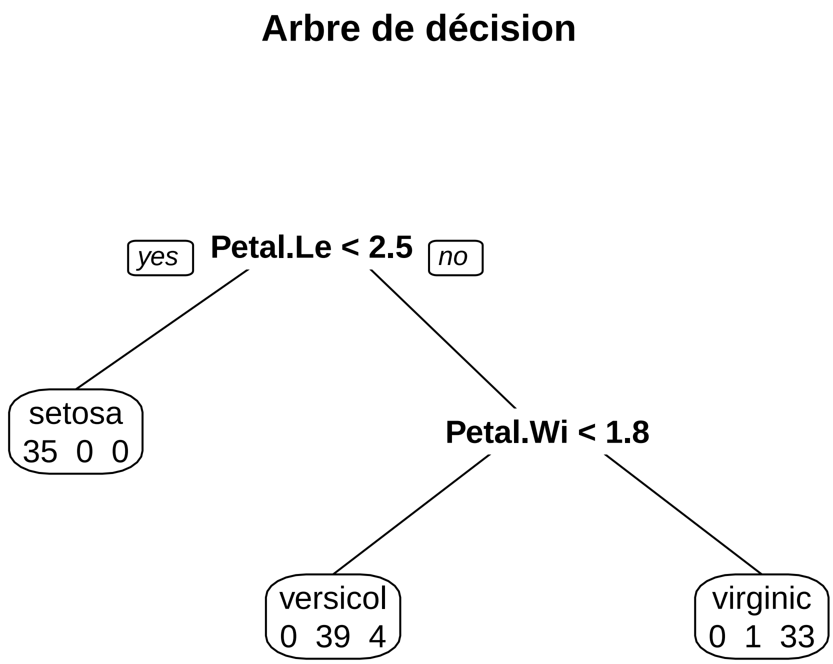
On applique maintenant la méthode CART sur le jeu de données d'apprentissage

```
m1 = rpart(formula = Species ~. , data = train, method = "class")
summary(m1)
```

```
## Call:
## rpart(formula = Species ~ ., data = train, method = "class")
##   n= 112
##
##          CP nsplit  rel error    xerror       xstd
## 1 0.4861111      0 1.00000000 1.1111111 0.06640159
## 2 0.4444444      1 0.51388889 0.6250000 0.07206131
```

```
## 3 0.0100000      2 0.06944444 0.1388889 0.04191395
##
## Variable importance
##  Petal.Width Petal.Length Sepal.Length  Sepal.Width
##           35           32           21           12
##
## Node number 1: 112 observations,    complexity param=0.4861111
##   predicted class=versicolor  expected loss=0.6428571  P(node) =1
##     class counts:    35    40    37
##    probabilities: 0.312 0.357 0.330
##   left son=2 (35 obs) right son=3 (77 obs)
##   Primary splits:
##       Petal.Length < 2.45 to the left,  improve=36.11201, (0 missing)
##       Petal.Width  < 0.8  to the left,  improve=36.11201, (0 missing)
##       Sepal.Length < 5.45 to the left,  improve=25.73236, (0 missing)
##       Sepal.Width  < 3.35 to the right, improve=12.09693, (0 missing)
##   Surrogate splits:
##       Petal.Width  < 0.8  to the left,  agree=1.000, adj=1.000, (0 split)
##       Sepal.Length < 5.45 to the left,  agree=0.929, adj=0.771, (0 split)
##       Sepal.Width  < 3.35 to the right, agree=0.821, adj=0.429, (0 split)
##
## Node number 2: 35 observations
##   predicted class=setosa      expected loss=0  P(node) =0.3125
##     class counts:    35     0     0
##    probabilities: 1.000 0.000 0.000
##
## Node number 3: 77 observations,    complexity param=0.4444444
##   predicted class=versicolor  expected loss=0.4805195  P(node) =0.6875
##     class counts:     0    40    37
##    probabilities: 0.000 0.519 0.481
##   left son=6 (43 obs) right son=7 (34 obs)
##   Primary splits:
##       Petal.Width  < 1.75 to the left,  improve=29.244570, (0 missing)
##       Petal.Length < 4.75 to the left,  improve=27.716630, (0 missing)
##       Sepal.Length < 6.15 to the left,  improve=10.098700, (0 missing)
##       Sepal.Width  < 2.95 to the left,  improve= 2.306057, (0 missing)
##   Surrogate splits:
##       Petal.Length < 4.75 to the left,  agree=0.909, adj=0.794, (0 split)
##       Sepal.Length < 6.15 to the left,  agree=0.740, adj=0.412, (0 split)
##       Sepal.Width  < 2.95 to the left,  agree=0.662, adj=0.235, (0 split)
##
## Node number 6: 43 observations
##   predicted class=versicolor  expected loss=0.09302326  P(node) =0.3839286
##     class counts:     0    39     4
##    probabilities: 0.000 0.907 0.093
##
## Node number 7: 34 observations
##   predicted class=virginica   expected loss=0.02941176  P(node) =0.3035714
##     class counts:     0     1    33
##    probabilities: 0.000 0.029 0.971
```

```r
prp(m1, extra = 1, main = "Arbre de décision")
```



On calcule notre taux d'erreur d'apprentissage :

```
#Prédiction du modèle sur les données de test
iris_predict<-predict(m1,newdata=test, type= "class")

#Matrice de confusion
mc<-table(test$Species, iris_predict)
mc
```

```
##             iris_predict
##              setosa versicolor virginica
##   setosa         15          0         0
##   versicolor      0         10         0
##   virginica       0          1        12
```

```
erreur_classement  = 1 - (mc[1,1]+ mc[2,2]+mc[3,3]) / sum(mc)
erreur_classement
```

```
## [1] 0.02631579
```

Le taux d'erreur obtenu sur les données test est de 5.3%.

3.

On utilise à présent la fonction `randomForest` afin de construire la forét aléatoire.

```
iris_foret = randomForest(Species ~., data = train)
iris_foret
```

```
##
## Call:
##  randomForest(formula = Species ~ ., data = train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
```

```
##         OOB estimate of  error rate: 6.25%
## Confusion matrix:
##            setosa versicolor virginica class.error
## setosa         35          0         0   0.0000000
## versicolor      0         37         3   0.0750000
## virginica       0          4        33   0.1081081
```

Ci-dessus sont affichées les quelques caractéristiques de l'objet produit. On voit que la forêt est composée de 500 arbres. A chaque noeud l'algorithme fait un essai sur 2 variables. Le taux d'erreur d'apprentissage nous est donné; il vaut 6.25%.

On cherche maintenant le taux d'erreur de généralisation:

```r
iris_foret_predict = predict(iris_foret, newdata = test)
```

On affiche la matrice de confusion :

```r
iris_foret$confusion
```

```
##            setosa versicolor virginica class.error
## setosa         35          0         0   0.0000000
## versicolor      0         37         3   0.0750000
## virginica       0          4        33   0.1081081
```

```r
tx_erreur = 1-sum(diag(iris_foret$confusion))/sum(iris_foret$confusion)
tx_erreur
```

```
## [1] 0.06403021
```

Le taux d'erreur obtenu est de 6.4%. Il y a une différence de 1.1 point de pourcentage de plus par rapport à celui obtenu avec la méthode CART.