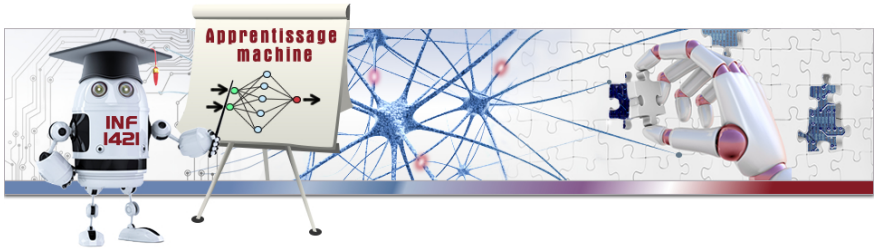


Module 7 - Arbres de décisions

Exercices - Corrigé

Exercice 2

L'entropie peut être calculée selon :

Individu	Taille	Couleur	*Class*
1	P	U	Pure
2	P	U	Pure
3	G	U	Pure
4	G	M	Pure
5	G	M	NPure
6	G	M	NPure
7	P	U	NPure
8	P	M	NPure

$$E(P) = -\sum_{i=1}^{n} p_i \log p_i \qquad (1)$$

avec log désigne la fonction de logarithme en base 2.

1. L'entropie de la variable couleur est donnée par :

$$E(Couleur) = \frac{4}{8} \times I(3,1) + \frac{4}{8} \times I(1,3)$$

avec

$$I(3,1) = -\frac{3}{4} \times \log(\frac{3}{4}) - \frac{1}{4} \times \log(\frac{1}{4}) = 0.8113$$

$$I(1,3) = -\frac{1}{4} \times \log(\frac{1}{4}) - \frac{3}{4} \times \log(\frac{3}{4})$$

d'où :

$$E(Couleur) = 2 \times (\frac{4}{8}) \times I(3,1) = 0.8113$$

2. L'entropie de la variable taille est donnée par :

$$E(Taille) = \frac{4}{8} \times I(2,2) + \frac{4}{8} \times I(2,2)$$

avec

$$I(2,2) = -\frac{2}{4} \times \log(\frac{2}{4}) - \frac{2}{4} \times \log(\frac{2}{4}) = 1$$

d'où :

$$E(Taille) = \frac{4}{8} \times 1 + \frac{4}{8} \times 1 = 1$$

3. Soit un ensemble de données T, le gain d'informations de T par rapport à une partition T_j donnée est la variation d'entropie causée par la partition de T selon T_j.

$$\text{Gain}(X,T) = E(T) - E(X,T) = E(T) - \sum_{j=1}^{m} \frac{|T_j|}{|T|} E(T_j) \qquad (2)$$

Nous pouvons, donc, calculer le gain de chacune des variables :

$$E(T) = E(Race) = 1$$

$$Gain(Race, Couleur) = 1 - E(Couleur) = 1 - 0.8113 = 0,19$$

$$Gain(Race, Taille) = 0$$

Le gain de taille étant nul, cette variable ne constitue pas un bon prédicteur de la race.

Exercice 3

1. Calcul de l'entropie de la variable `Couleur` :

```
install.packages("DescTools")
library("DescTools")
Data<- matrix(c(1,1, 1, 1,2, 1, 1, 1,3, 0, 1, 1,4, 0, 0,
    1,5, 0, 0, 0, 6, 0, 0, 0, 6, 1, 1, 0, 6, 1, 0, 0),
    nrow=8, ncol=4, byrow=T)
# Entropie de la variable Couleur
EntrCouleur=Entropy(Data[,3],Data[,4],base=2)-1
View(EntrCouleur)
# Entropie de la variable classe
EntrClasse=Entropy(Data[,4],base=2)-1
View(EntrClasse)
# Calcul du Gain de la variable Couleur
GainCouleur=EntrClasse-EntrCouleur
View(GainCouleur)
```

L'entropie de la variable `Couleur` est 0,8113

2. Calcul de l'entropie de la variable `Taille` :

```
# Entropie de la variable classe
EntrClasse=Entropy(Data[,4],base=2)-1
View(EntrClasse)
# Calcul du Gain de la variable Taille
GainTaille=EntrClasse-EntrTaille
View(GainTaille)
```

L'entropie de la variable `Taille` est 1

3. Le gain qui est defini par :

```
# Calcul du gain de la variable "Taille"
I<-function(p,n)
{
  return(((-p/(n+p))*log(p/(p+n)))-(n/(n+p))*log(n/(p+n))
      )
}
Entropie<- -(4/8)*log(4/8)-(4/8)*log(4/8)
GTaille<-Entropie - EntropieTaille
```

On peut aussi calculer le gain en utilisant la bibliotheque "FSelector" :

```
# Installer la bibliotheque "FSelector"
install.packages("FSelector")

# Charger la bibliotheque
library(FSelector)

weights <- information.gain(Data$Class~Data$Taille + Data
    $Couleur , Data, unit = "log2")
```

Donc, puisque la fonction Gain égale à zéro, alors, la variable Taille ne constitue pas un bon prédicteur de la race.

Exercice 4

1. Installer et charger les bibliothèques `rpart` et `FSelector` :

```
install.packages("rpart", dep=TRUE)
library(rpart)
install.packages("FSelector")
library(FSelector)
```

2. Importation des deux base de données : `iris` et `kyphosis`

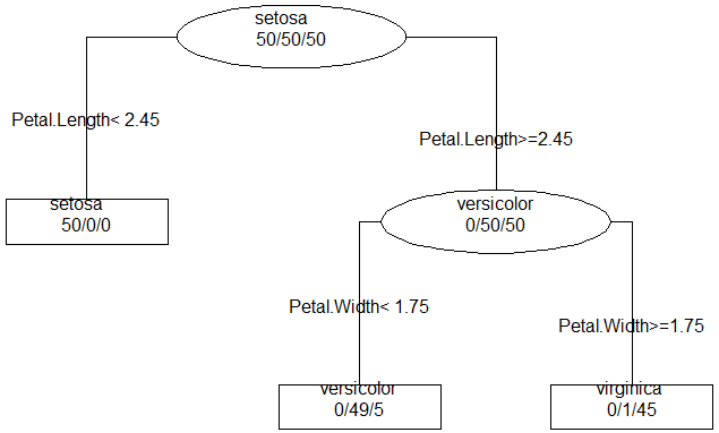
```
# Importer la base de donnees "iris"
data(iris)

# Importer la base de donnees "kyphosis"
data(kyphosis)
```

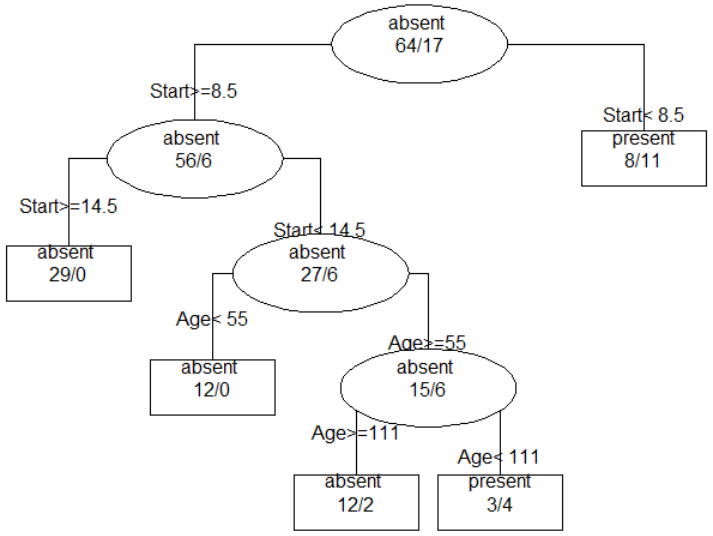
3. Les arbres de décisions des ensembles de données sont :

```
# Arbre de decision de la base de donnees "iris"
dev.new()
arbreIris = rpart(iris$Species ~ ., method="class",
    minsplit=20, xval=81, data=iris)
plot(arbreIris, uniform=TRUE, margin=0.1, main="Arbre de
    decision 'iris'")
text(arbreIris, fancy=TRUE, use.n=TRUE, pretty=0, all=
    TRUE)
# Arbre de decision de la base de donnees "Kyphosis"
dev.new()
arbreKyphosis = rpart(Kyphosis ~ ., method="class",
    minsplit=20, xval=81, data= kyphosis)
plot(arbreKyphosis, uniform=TRUE, margin=0.1, main="Arbre
     de decision 'kyphosis'")
text(arbreKyphosis, fancy=TRUE, use.n=TRUE, pretty=0, all
    =TRUE)
```

Arbre de décision 'iris'



Arbre de décision 'kyphosis'



4. Rapport de gain et gain pour chaque attribut :

```r
# Calcul Gain ration et Gain pour les attributs
print("Iris Database")
information.gain(Species~., data=iris, unit = "log2")
gain.ratio(Species ~. , data=iris, unit = "log2")

print("kyphosis Database")
information.gain(Kyphosis~., data=kyphosis, unit = "log2"
    )
gain.ratio(Kyphosis ~. , data=kyphosis, unit = "log2")
```

```
> print("Iris Database")
[1] "Iris Database"
> information.gain(Species~., data=iris, unit = "log2")
             attr_importance
Sepal.Length       0.6522837
Sepal.Width        0.3855963
Petal.Length       1.3565450
Petal.Width        1.3784027
> gain.ratio(Species ~. , data=iris, unit = "log2")
             attr_importance
Sepal.Length       0.4196464
Sepal.Width        0.2472972
Petal.Length       0.8584937
Petal.Width        0.8713692
> print("kyphosis Database")
[1] "kyphosis Database"
> information.gain(Kyphosis~., data=kyphosis, unit = "log2")
       attr_importance
Age          0.0000000
Number       0.0000000
Start        0.1690017
> gain.ratio(Kyphosis ~. , data=kyphosis, unit = "log2")
       attr_importance
Age          0.0000000
Number       0.0000000
Start        0.1712875
>
```

5. Les attributs qui ont plus d'importance pour la base de données '**iris**' sont :
   - Petal.Length
   - Petal.With

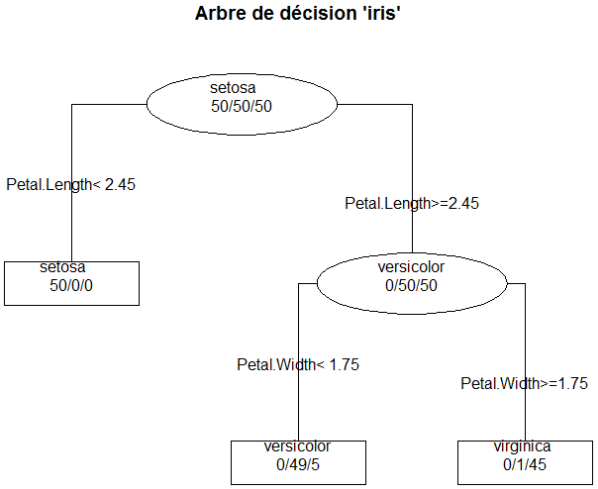
   L'attribut qui a plus d'importance pour la base de données '**kyphosis**' est :

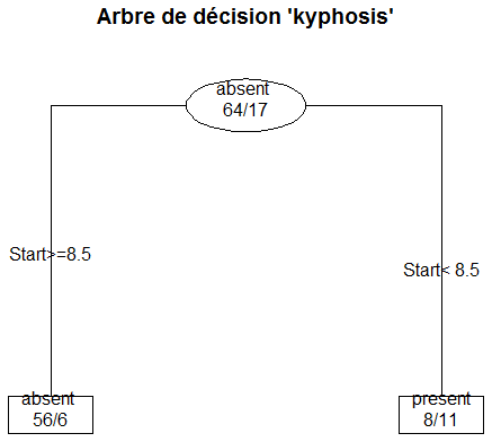
— Start

6. Après élimination des attributs non importants, nous obtenons les arbres suivants :

```
# Arbre de decision de la base de donnees "iris"
dev.new()
arbre = rpart(iris$Species ~ Petal.Length +Petal.Width,
    method="class", minsplit=20, xval=81, data=iris)
plot(arbre, uniform=TRUE, margin=0.1, main="Arbre de
    decision 'iris'")
text(arbre, fancy=TRUE, use.n=TRUE, pretty=0, all=TRUE)

# Arbre de decision de la base de donnees "Kyphosis"
dev.new()
arbre = rpart(Kyphosis ~ Start, method="class", minsplit
    =20, xval=81, data= kyphosis)
plot(arbre, uniform=TRUE, margin=0.1, main="Arbre de
    decision 'kyphosis'")
text(arbre, fancy=TRUE, use.n=TRUE, pretty=0, all=TRUE)
```





Exercice 4

1. Chargement da la base de données "CreditData.txt" en utilisant R :

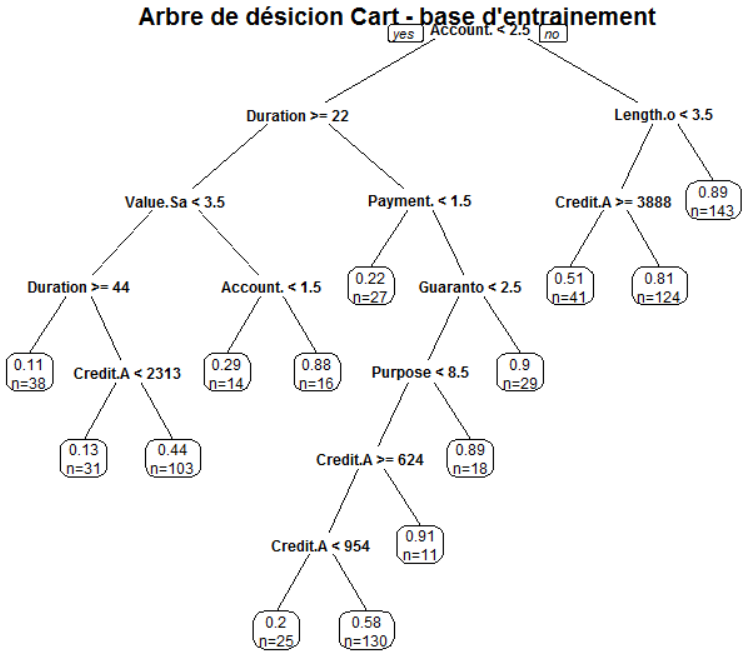
```
# Charger la base de donnees
Data<-read.csv("C:/Users/youssef/Dropbox/INF1421-Module7-
    ArbresDecision/R/Exercice4/CreditData.txt", header =
    TRUE, sep = ",")
```

2. Divisez la base de données en base d'apprentissage et en base de test :

```
#Base d'entrainenemt
Train<-Data[251 :1000,]
#Base de Test
Test<-Data[1 :250,]
```

3. Affichage de l'arbre de décision complète de la base d'apprentissage.
   — Algorithmes CART :

```
# Arbre de decision CART
library(rpart)
install.packages("rpart.plot")
library(rpart.plot)
dev.new()
Arbre1= rpart(Train$Creditability~ .,data= Train)
prp(Arbre1,extra=1)
title("Arbre de decision Cart - base d'entrainement"
    )
```



— Algorithmes C5.0 :

```r
install.packages("C50")
library(C50)
Train<-Train[1:750,]
#Arbre de decision C5
dev.new()
Arbre2<- C5.0((Train$Creditability) ~ ., data =
    Train[,-1])
plot.(Arbre2)
# Ou
dev.new()
Arbre2 <- C5.0(x = (iris[,-5]), y = iris$Species)
summary(Arbre2)
plot(Arbre2)

Arbre2 <- C5.0(x = (kyphosis[,-1]), y = kyphosis$
    Kyphosis)
summary(Arbre2)
plot(Arbre2)

dev.new()
data(churn)
treeModel <- C5.0(x = Train[, -1], y = Train$
    Creditability)

plot(treeModel) #to compare outputi

dev.new()
data(churn)
treeModel <- C5.0(x = churnTrain[, -20], y =
    churnTrain$churn)

plot(treeModel) #to compare outputi
```

4. Prédiction de la variable `Creditability` sur la base de test :

— Algorithmes CART :

```r
# Predire les donnees de la base test en utilisant l
    'arbre CART
result<-predict(Arbre1,Test[,-1],type="class")
result

# Matrice de confusion
table(result,Test[,1])

# Taux de prediction
mean(result==Test[,1])
```

— Algorithmes C5.0 :

```r
# Predire les donnees de la base test en utilisant l
    'arbre C5.0
result2<-predict(Arbre2,Test[,-1],type="class")
result2

# Matrice de confusion
table(result2,Test[,1])

# Taux de prediction
mean(result2==Test[,1])*100
```