TP_Arbre.RMD

Nadjib BENAMROUCHE

07/11/2022

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

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

1 -Partie apprentissage :

Création des données d'apprentissage :

```
#Creer la base de donneés d'apprentissage
Meteo <- c('soleil','soleil','soleil','soleil','nuages','nuages','nuages','nuages',
           'soleil')
Amis <- c('présents','absents','présents','absents','absents','présents','absents',
          'présents','absents')
Vent <- c('faible','fort','fort','faible','faible','fort','fort','faible','faible')
Jour <- c('week-end','semaine','semaine','semaine','week-end','week-end','semaine',
          'week-end','week-end')
Decision <- c('oui','non','non','oui','non','non','non','oui','non')
data_app <- data.frame(Meteo, Amis, Vent, Jour, Decision)
glimpse(data_app)
```

```
## Rows: 9
## Columns: 5
## $ Meteo    <fct> soleil, soleil, soleil, soleil, nuages, nuages, nuages, nuage~
## $ Amis     <fct> présents, absents, présents, absents, absents, présents, abse~
## $ Vent     <fct> faible, fort, fort, faible, faible, fort, fort, faible, faible
## $ Jour     <fct> week-end, semaine, semaine, semaine, week-end, week-end, sema~
## $ Decision <fct> oui, non, non, oui, non, non, non, oui, non
```

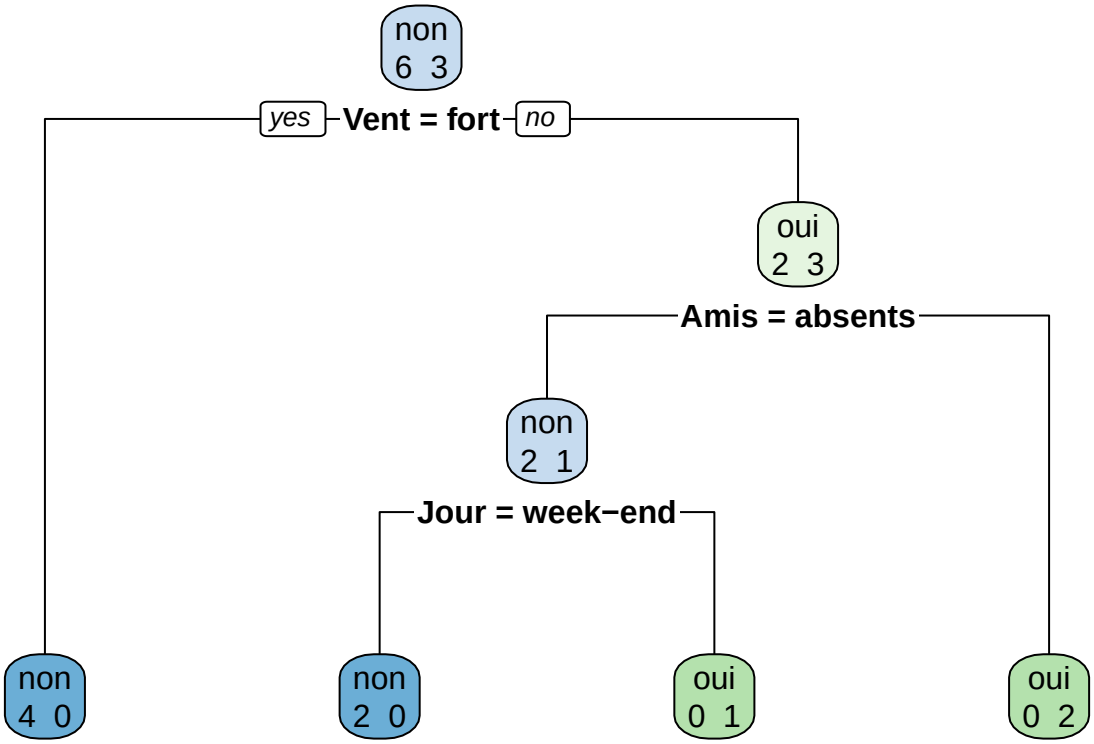
Construction de l'arbre de décision à partir des données d'apprentissage :

```
arbre <- rpart( Decision ~., data = data_app,
                method = "class",minsplit=2,cp=0)
arbre
```

```
## n= 9
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 9 3 non (0.6666667 0.3333333)
##    2) Vent=fort 4 0 non (1.0000000 0.0000000) *
##    3) Vent=faible 5 2 oui (0.4000000 0.6000000)
##      6) Amis=absents 3 1 non (0.6666667 0.3333333)
##       12) Jour=week-end 2 0 non (1.0000000 0.0000000) *
##       13) Jour=semaine 1 0 oui (0.0000000 1.0000000) *
##      7) Amis=présents 2 0 oui (0.0000000 1.0000000) *
```

Représentation de l'arbre de décision non élagué :

```
rpart.plot(arbre,extra = 1)
```



Délégation de l'arbre de décision :

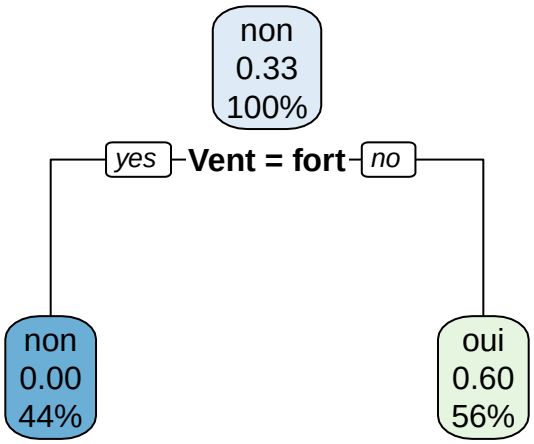
```
arbre_elague <- rpart(formula = Decision ~ ., data = data_app,
                      minsplit = 2, minbucket=3)
arbre_elague
```

```
## n= 9
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
```

```
## 1) root 9 3 non (0.6666667 0.3333333)
##   2) Vent=fort 4 0 non (1.0000000 0.0000000) *
##   3) Vent=faible 5 2 oui (0.4000000 0.6000000) *
```

Représentation de l'arbre élagué :

```
rpart.plot(arbre_elague)
```



2 -Partie test :

Création des données de test :

```
Meteo <- c("soleil","nuages","soleil","soleil","nuages")
Amis <- c("présents","absents","absents","absents","présents")
Vent <- c("fort","faible","faible","faible","faible")
Jour <- c("week-end","semaine","week-end","semaine","semaine")
Decision <- c("non","non","oui","non","oui")

data_test <- data.frame(Meteo,Amis,Vent,Jour,Decision)
glimpse(data_test)
```

```
## Rows: 5
## Columns: 5
## $ Meteo    <fct> soleil, nuages, soleil, soleil, nuages
## $ Amis     <fct> présents, absents, absents, absents, présents
## $ Vent     <fct> fort, faible, faible, faible, faible
## $ Jour     <fct> week-end, semaine, week-end, semaine, semaine
## $ Decision <fct> non, non, oui, non, oui
```

Prédiction du modèle sur les données de test :

```
data_test_predict <- predict(arbre_elague, newdata = data_test,
                             type = "class")
data_test_predict
```

```
##   1   2   3   4   5
## non oui oui oui oui
## Levels: non oui
```

```
data_test$Decision
```

```
## [1] non non oui non oui
## Levels: non oui
```

Matrice de confusion :

```
mc<-table(data_test$Decision,data_test_predict)
mc
```

```
##      data_test_predict
##       non oui
##   non   1   2
##   oui   0   2
```

Calculer l'erreur de classement et le taux de prédiction :

```
erreur.classement<-1.0-(mc[1,1]+mc[2,2])/sum(mc)
print( " Erreur = ")
```

```
## [1] " Erreur = "
```

```
print(erreur.classement)
```

```
## [1] 0.4
```

```
prediction=mc[2,2]/sum(mc[2,])
print (" Taux de prédiction = ")
```

```
## [1] " Taux de prédiction = "
```

```
print(prediction)
```

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