**TP n°5 : Les Arbres de Décision**

**Note**

Ce TP est à rendre en fin de séance.

**Objectifs :**

Comprendre l’Apprentissage Supervisé à travers la construction des Arbres de Décisions

1. **Construction d'un Arbre de Décision pour un « Jeu de Tennis» avec R**

**Source : http://www.grappa.univ-lille3.fr/~ppreux/ensg/miashs/fouilleDeDonneesI/tp/arbres-de-decision/**

**Objectif :** Construire un Arbre de Décision à partir de données climatiques, afin de prédire si on pourra jouer au Tennis ou non.

1. **Chargement de la bibliothèque :**

Pour pouvoir construire des arbres de décision, on va utiliser la bibliothèque « **rpart »** de l'environnement R.

Il faut tout d'abord la rendre accessible. Pour cela, on tape la commande suivante :

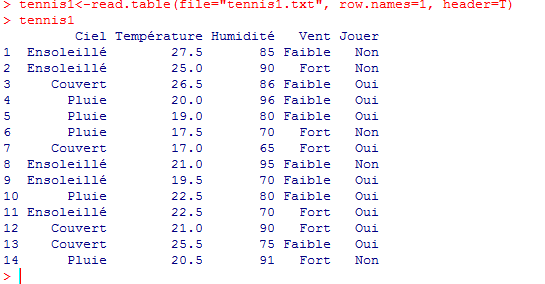
Library(rpart)

1. **Importation de données :**

On commence par charger le jeu de données « Tennis1.txt ». Pour ce faire, placer cet entrepôt de données dans un data frame ‘Tennis’ :

setwd("E:/DataMining/tp5")

tennis1<-read.table(file="tennis1.txt", row.names=1, header=T)



La colonne « jouer » est donc notre variable décisionnelle (y)

1. **Construction et Visualisation de l’arbre de décision :**
2. Les commandes suivantes permettent de construire l'arbre de décision. Tout d'abord, on doit spécifier quelques paramètres qui précisent comment l'arbre de décision doit être construit. On tape la commande suivante :

ad.tennis1.cnt<- rpart.control (minsplit = 1)

*La variable ad.tennis.cnt stocke les paramètres de l'algorithme.*

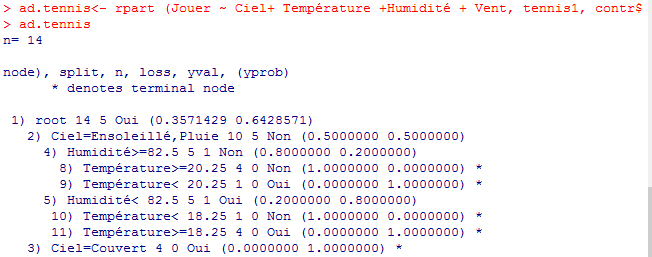
*minsplit = 1 signifie que le nombre minimal d'exemples nécessaires à la création d'un nœud est 1. La valeur par défaut est 20. Comme le jeu de données contient moins de 20 exemples, utiliser la valeur par défaut ne produirait pas d'arbre du tout, juste une racine !*

*Le nom utilisé pour cette variable, ad.tennis.cnt suit la convention R : il indique qu'il s'agît d'un arbre de décision (préfixe ad), pour le jeu de tennis (tennis !) et qu'il s'agît des paramètres de contrôle (cnt) ; des points (.) séparent ces différentes informations.*

On va construire l'arbre de décision en indiquant :

* l'attribut qui représente la variable cible à prédire : ‘**la classe : Jouer’**
* les attributs qui doivent être utilisés pour effectuer cette prédiction (pour l'instant, ce seront les 4 autres attributs : **Ciel, Température, Humidité**et**Vent**)
* l’entrepôt de données avec lequel on construit l'arbre : **Tennis**
* le nom de la variable qui contient les paramètres : **control = ad.tennis.cnt**

ad.tennis<- rpart (Jouer ~ Ciel+ Température +Humidité + Vent, tennis1, control = ad.tennis1.cnt)



Racine de l’arbre

C’est la décision

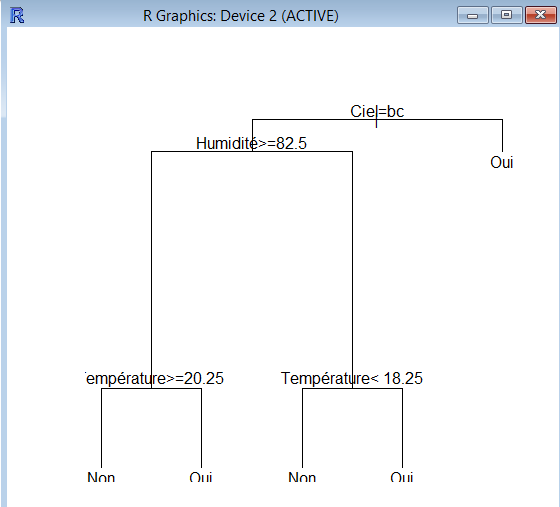
La représentation graphique des Arbres de Décision avec R a deux formes : Représentation Textuelle et Représentation Graphique.

Concernant le premier type de visualisation, l’arbre est donné sous forme de lignes imbriquées dont chacune correspondant à une classe séparatrice. R distingue une variable séparatrice majoritaire (nœud feuille) des classes non majoritaires par le caractère « \* »

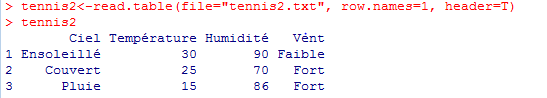
1. Afficher le résultat de la construction sous forme graphique : NB : on utilisera les deux commandes ***plot*** et ***text***

Plot(ad.tennis)

Text(ad.tennis)

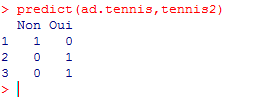


1. **La prédiction de la classe d'une donnée par un arbre de décision**
2. La fonction **predict()** utilise un arbre de décision pour prédire la classe de nouvelles données. Elle prend en paramètres l'arbre et un data frame qui contient les données dont il faut prédire la classe. Pour prédire la classe des données du jeu d'exemples (avec lesquels on a construit l'arbre de décision), on tapera la commande :



1. ***Utilisez l'arbre pour donner une prédiction pour l’entrepôt de données « Tennis2.txt »***

predict(ad.tennis1,tennis2)



1. **Construction d'un Arbre de Décision pour le « Cancer du Sein » avec R**

**Source : http://eric.univ-lyon2.fr/%7Ericco/tanagra/fichiers/breast.txt**

**Objectifs :**

* Construire un Arbre de Décision à partir d’un échantillon d’apprentissage contenant 399 observations concernant des patients portant des tumeurs de seins, dans le but de prédire son type : Bégnine ou Maligne.
* Appliquer l’arbre construit sur un échantillon de test contenant 300 observations pour spécifier le type tumeur.

1. **Préparation des données :**

*[data = breast.app ] - Construction de l’arbre sur l’échantillon « breast.app ».*

*[classe ~.] - Prédire « classe » à partir des autres variables de la base.*

*[method = « class »] - On construit un arbre de décision c.-à-d. apprentissage supervisé.*

rm(list=ls())

#package pour la lecture des fichiers XLS

Library(xlsReadWrite)

#chargement de l’échantillon d'apprentissage

breast.app<-read.xls(file=file.choose(),sheet=1)

#chargement de l’échantillon test

breast.test <-<-read.xls(file=file.choose(),sheet=2)

#appel du package 'rpart' pour l'induction des arbres

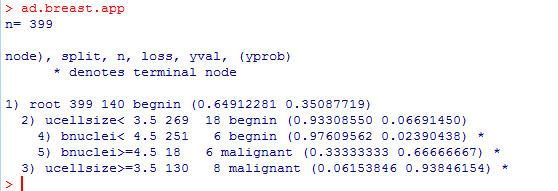
library(rpart)

1. **Construction de l’arbre de décision sur l’échantillon breast.app**,

#arbre de décision avec paramètres par défaut

ad.breast.app.cnt<- rpart.control (minsplit = 50)

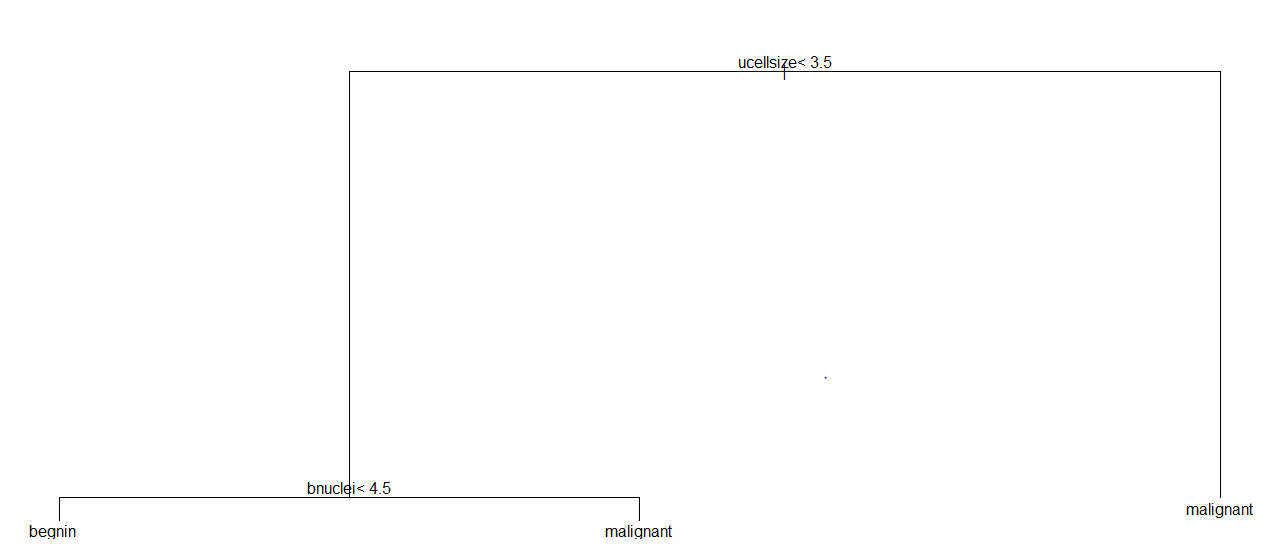
ad.breast.app<- rpart (classe ~.,breast.app, control = ad.breast.app.cnt)



Donner ci-dessous sa description graphique :

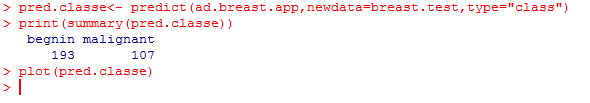
plot(ad.breast.app)

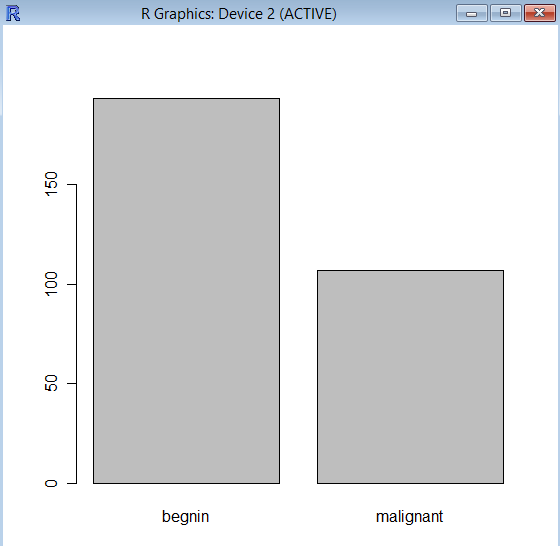
> text(ad.breast.app)



1. **Evaluation de l’arbre sur un échantillon test :**

#prédiction sur l'échantillon test



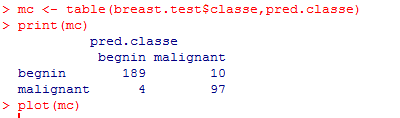


1. **Matrice de de Confusion (Table de Contingence)**

*Cette matrice construit un tableau croisé entre la cible observée (classe) et la prédiction du modèle (pred.classe)*

*La table mc se comporte comme une matrice à 2 dimensions, on en déduit le taux d’erreur*

#matrice de confusion





En déduire les pourcentages suivants :

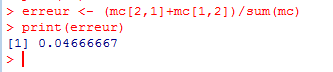
|  |  |  |
| --- | --- | --- |
| Les Classes de Prédiction | | |
|  | Bénigne | Maligne |
| Bégnine | **189** | **10** |
| Maligne | **4** | **97** |

1. **Calcul du taux de l’erreur :**

*Calculer le taux d’erreur en appliquant la formule suivante :*

***erreur = Somme des éléments hors diagonale principale / Nombre total des observations.***

#taux d'erreur



**III. Construction d'un Arbre de Décision pour la décision d’accorder un crédit**

**Objectif :** Construire un Arbre de Décision à partir de données sur des clients d’une banque afin de prendre une décision concernant l’acceptation ou le refus d’une demande de crédit.

La décision est décrite par une variable qualitative Decision, ayant trois modalités : accepted, tolerated, ou refused*.*

1. Chargement de la bibliothèque : Pour pouvoir construire des arbres de décision, on va utiliser la bibliothèque « **rpart »** de l'environnement R. Il faut tout d'abord la rendre accessible. Pour cela, on tape la commande suivante :

#importation de library

Library(rpart)

1. Importation de données : On commence par charger le jeu de données « ScoreData.xls». Pour ce faire, placer cet entrepôt de données dans un data frame ‘score’ :

#chargement des données : échantillon d’apprentissage

Library(xlsReadWrite)

score.app<-read.xls(file=file.choose(),sheet=1)

1. Construction et Visualisation de l’arbre de décision :

Les commandes suivantes permettent de construire l'arbre de décision. Tout d'abord, on doit spécifier quelques paramètres qui précisent comment l'arbre de décision doit être construit. On tape la commande suivante :

ad.score.cnt<- rpart.control (minsplit = 10)

*La variable ad.score.cnt stocke les paramètres de l'algorithme. minsplit = 10 signifie que le nombre minimal d'exemples nécessaires à la création d'un nœud est 10. La valeur par défaut est 20. Comme le jeu de données contient moins de 20 exemples, utiliser la valeur par défaut ne produirait pas d'arbre du tout, juste une racine !*

*Le nom utilisé pour cette variable, ad.score.cnt suit la convention R : il indique qu'il s'agît d'un arbre de décision (préfixe ad), pour le jeu de données des scores (score) et qu'il s'agît des paramètres de contrôle (cnt) ; des points (.) séparent ces différentes informations.*

1. Construction de l'arbre de décision :

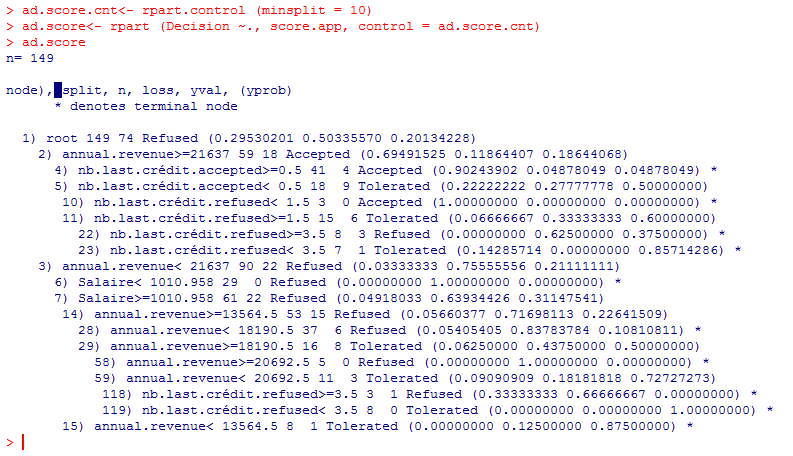
* l'attribut qui représente la variable cible à prédire : ‘**Decision’**
* les attributs qui doivent être utilisés pour effectuer cette prédiction (pour l'instant, ce seront tous les autres attributs : **‘.’**)
* l’entrepôt de données avec lequel on construit l'arbre : **score**
* le nom de la variable qui contient les paramètres : **control = ad.score.cnt**

ad.score<- rpart (Decision ~., score.app, control = ad.score.cnt)

La représentation graphique des Arbres de Décision avec R a deux formes : Représentation Textuelle et Représentation Graphique.

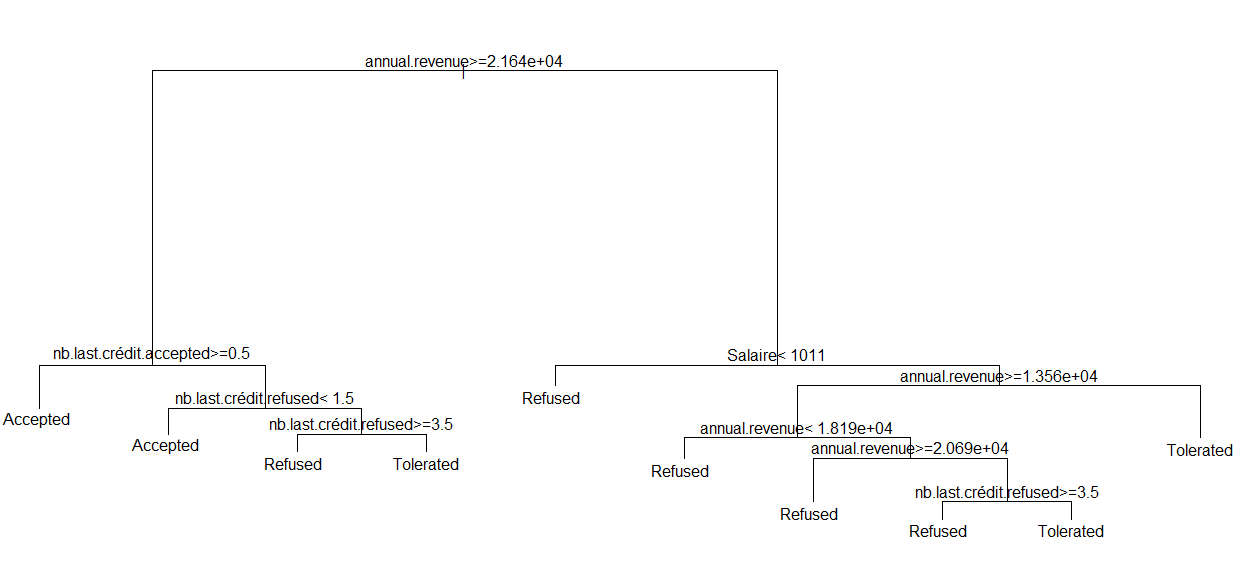
Concernant le premier type de visualisation, l’arbre est donné sous forme de lignes imbriquées dont chacune correspondant à une classe séparatrice. R distingue une variable séparatrice majoritaire (nœud feuille) des classes non majoritaires par le caractère « \* »

Afficher le résultat de la construction sous forme de texte:



Afficher le résultat de la construction sous forme graphique : NB : on utilisera les deux commandes ***plot*** et ***text***



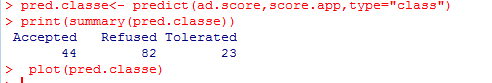


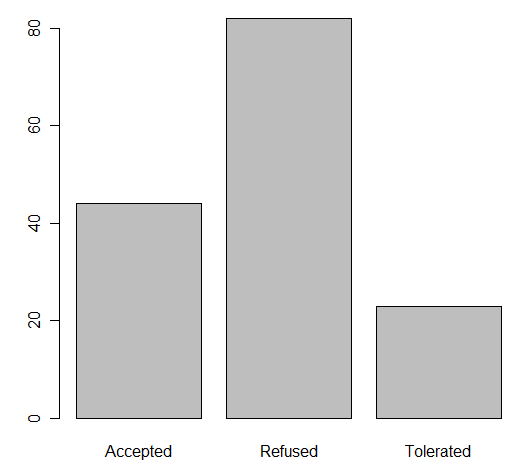
1. Evaluation de l’arbre de décision
2. Prédiction sur le même échantillon

La fonction **predict()** utilise un arbre de décision pour prédire la classe de nouvelles données. Elle prend en paramètres l'arbre et un data frame qui contient les données dont il faut prédire la classe. Pour prédire la classe des données du jeu d'exemples (avec lesquels on a construit l'arbre de décision), on tapera la commande :

#prédiction sur le même échatillon

pred.classe<- predict(ad.score,score.app,type="class")





1. Matrice de de Confusion (Table de Contingence)

*Cette matrice construit un tableau croisé entre la cible observée (Decision) et la prédiction du modèle (pred.classe)*

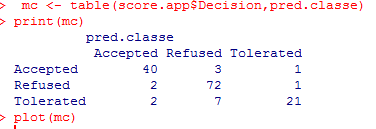
*La table mc se comporte comme une matrice à 2 dimensions, on en déduit le taux d’erreur*

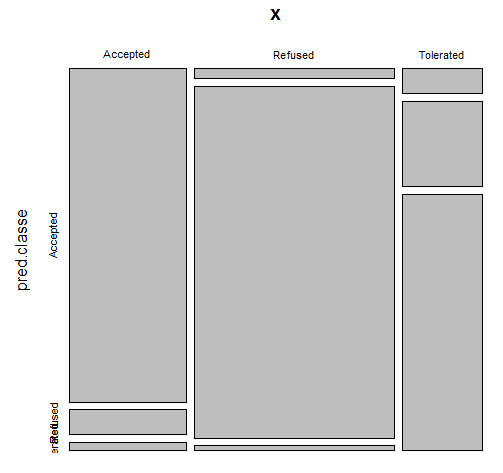
#matrice de confusion

mc <- table(score.app$Decision,pred.classe)

print(mc)

plot(mc)





En déduire les pourcentages suivants :

|  |  |  |  |
| --- | --- | --- | --- |
| Les Classes de Prédiction | | | |
|  | **Accepted** | **Refused** | **Tolerated** |
| Accepted | 40 | 3 | 1 |
| Refused | 2 | 72 | 1 |
| Tolerated | 2 | 7 | 21 |

1. Calcul du taux de l’erreur :

*Calculer le taux d’erreur en appliquant la formule suivante :*

***erreur = Somme des éléments hors diagonale principale / Nombre total des observations.***

#taux d'erreur

Erreur<- (mc[2,1]+mc[1,2])/sum(mc)

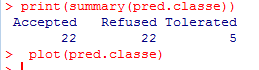


1. ***Utilisez l'arbre pour donner une prédiction pour l’échantillon du test***
2. La prédiction de la classe d'une donnée par un arbre de décision

Prédiction sur un échantillon de test (2ème feuille du fichier ScoreData.xls)

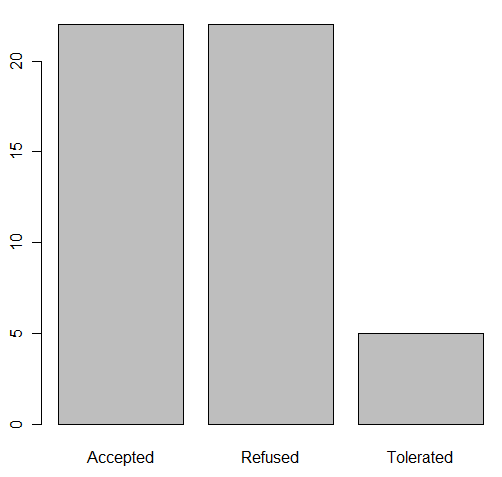
score<-read.xls(file=file.choose(),sheet=2)

pred.classe<- predict(ad.score,score,type="class")



***Utilisez l'arbre pour donner une prédiction pour l’échantillon de test***

**Résultat et interprétation**



1. Modification des paramètres de construction de l’arbre :
2. Décrire l’utilité de la fonction rpart.control :

*Avec son paramètre « minsplit » qui prend par défaut 20 on décide du nombre minimal d'exemples nécessaires à la*  *création d'un nœud. Il est conseillé de le changer à 1 si l’échantillon compte moins de 20 lignes comme l’exemple de Tennis de la 1ére partie.*

1. En fixant les paramètres suivants, donner le nouvel arbre de décision :

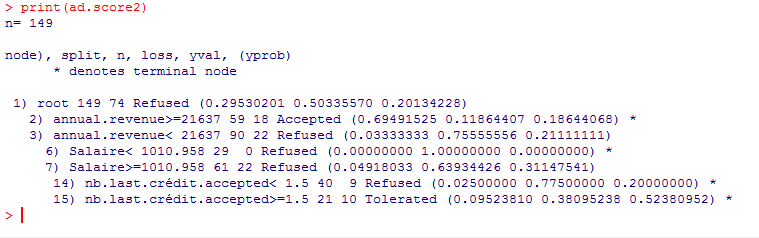
*Minsplit = taille minimale pour segmenter = 50*

*Minbucket = effectif d’admissibilité = 20*

Ad.control<- rpart.control( minsplit = 50 , minbucket = 20 )

> ad.score2<- rpart (Decision ~., score.app, control = Ad.control)

> print(ad.score2)



1. Utilisation du package « tree » :

*Paramètres d’apprentissage :*

*nobs = nombre d’obs. dans l’éch. d’apprentissage ;*

*mincut = effectif d’admissibilité,*

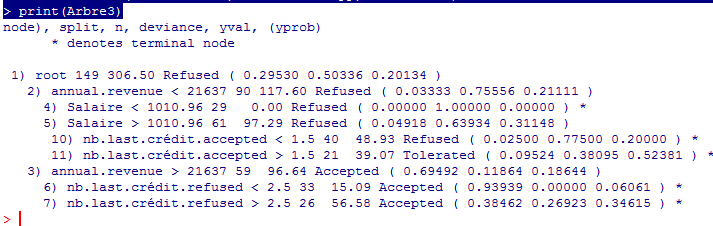
*minsize = taille min. pour segmenter*

*Apprentissage : classe vs. toutes les autres variables du data.frame*

cnt<- tree.control(nobs=nrow(score.app), mincut=20, minsize=50)

> Arbre3<- tree(Decision ~ ., data=score.app,control=cnt)

> print(Arbre3)



1. Utilisation du package « party » :

> param.party <- ctree\_control(minsplit=20,minbucket=10)

> arbre.party<- ctree(Decision ~ ., score.app, controls= param.party)

> plot(arbre.party)

