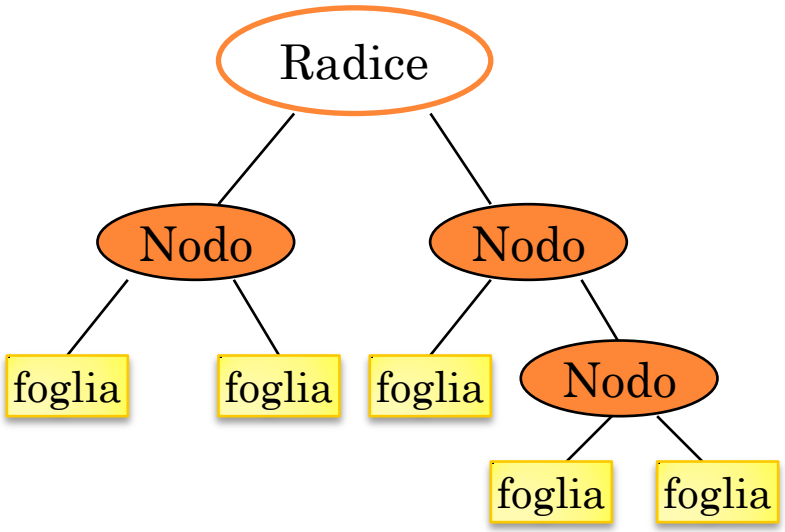
Algoritmo CART in R

Dott.ssa Agnieszka Stawinoga
Dip. Di Scienze Economiche e Statistiche
agnieszka.stawinoga@unina.it

GLI ALBERI DI DECISIONE

- Gli alberi decisionali sono prodotti da procedure di segmentazione
- Sono tipicamente utilizzati per problemi di classificazione e previsione



Il rettangolo superiore è la **radice (R)**

L'albero è costituito da un insieme finito di elementi, **i nodi**

Ogni **nodo** è un gruppo di unità a diversi stadi del processo di classificazione

Il **nodo radice** è un nodo disomogeneo al suo interno rispetto alla variabile obiettivo perché racchiude tutti gli individui considerati

L'insieme dei nodi (ad eccezione della radice) può essere suddiviso

in insiemi distinti: **i sottoalberi** del nodo R

Un nodo viene chiamato

- *padre* rispetto ai nodi che esso genera
- *figlio* rispetto al nodo da cui discende

I **valori di soglia** di una variabile che dividono le unità di un determinato nodo sono chiamati **split**

I rami sono le condizioni che hanno determinato la suddivisione

L'insieme di tutti i nodi terminali di un albero viene indicato con il simbolo $T\sim$

Le **foglie** sono i nodi terminali per i quali non si ritieneutile una ulteriore suddivisione

CART (Breiman, Friedman, Olshen – 1984, Classification And Regression Tree)

- Variabile dipendente QUALITATIVA/QUANTITATIVA
- Variabili esplicative QUALITATIVE/QUANTITATIVE

Obiettivi:

Classification Trees

Individuare il miglior classificatore di N individui appartenenti a J gruppi che siano internamente omogenei ed esternamente eterogenei

(*variabile di risposta in classi*)

Regression Trees:

Predire una variabile dipendente (continua) attraverso M variabili esplicative

(*variabile di risposta continua*)

CART: IL CRITERIO DI SPLIT

Obiettivo:

Generare nodi figli che siano più "puri" del nodo genitore

- **Classificazione ad albero**

Generare nodi figli **omogenei** con una **proporzione minima di individui** di classi **differenti** della variabile di risposta

- **Regressione ad albero**

Generare nodi figli con **varianza** della variabile di risposta **minore** del nodo genitore

Pacchetti in R con l'algoritmo CART:
tree, rprint, party, maptree

Regression Trees

```
install.packages("tree")
install.packages("rpart")
library(tree)
library(rpart)
data(car.test.frame )
```

Installare pacchetti

Caricare pacchetti

car.test.frame → un dataset composto da 60 righe e 8 colonne che rappresenta i dati sulle macchine prese dal Consumer Reports in Aprile del 1990.

Variabili:

- **Price:** variabile quantitqtiva - il prezzo in $ del modello standard
- **Country:** variabile qualitativa con diverse modalità 'France', 'Germany', 'Japan' , 'Japan/USA', 'Korea', 'Mexico', 'Sweden' and 'USA'
- **Reliability:** un vettore numerico con i valori da 1 a 5.
- **Mileage:** variabile quantitativa - il consumo della bensina (miles per US gallon, secondo i testi)
- **Type:** variabile qualitativa con le modalità Compact, Large, Medium, Small, Sporty, Van
- **Weight:** variabile quantitativa peso a vuoto di un'auto (in pounds)
- **Disp.:** variabile quantitativa - la capacità del motore (in litri)
- **HP:** variabile quantitativa - la forza dei cavalli del veicolo

CART con il pacchetto “tree”

Reference: B. Ripley, Package tree: Classification and regression trees, Version 1.0-35 , Date 2014-03-03

Modello 1.1:

In questo modello consideriamo una variabile dipendente (continua) Mileage ed una sola variabile esplicativa Weight.

```
rt1 = tree(Mileage ~ Weight, car.test.frame)

rt1

node), split, n, deviance, yval
      * denotes terminal node
 1) root 60 1355.000 24.58
 2) Weight < 2567.5 15  186.900 30.93
 4) Weight < 2280 6   16.000 34.00 *
 5) Weight > 2280 9   76.890 28.89 *
 3) Weight > 2567.5 45  361.200 22.47
 6) Weight < 3087.5 23  117.700 24.43
 12) Weight < 2747.5 8   39.880 25.62 *
 13) Weight > 2747.5 15   60.400 23.80 *
 7) Weight > 3087.5 22   61.320 20.41
 14) Weight < 3637.5 16   32.940 21.06 *
 15) Weight > 3637.5 6    3.333 18.67 *
```

Function *tree*: a function for fitting a Classification or Regression Tree, a tree is grown by binary recursive partitioning using the response in the specified formula and choosing splits from the terms of the right-hand-side. Numeric variables are divided into X < a and X > a; the levels of an unordered factor are divided into two non-empty groups. The split which maximizes the reduction in impurity is chosen, the data set split and the process repeated. Splitting continues until the terminal nodes are too small or too few to be split.

The columns include var, the variable used at the split (or "<leaf>" for a terminal node), n, the (weighted) number of cases reaching that node, dev the deviance of the node, yval, the fitted value at the node (the mean for regression trees, a majority class for classification trees) and split, a two-column matrix of the labels for the left and right splits at the node.
Classification trees also have yprob, a matrix of fitted probabilities for each response level.

Ogni output della funzione tree dimostra la struttura della partizione: la radice illustra numero totale di osservazioni (60), la devianza della variabile dipendente e la media della variabile dipendente per quel sottoinsieme (per la radice questo valore è una media totale).

```
sum(sapply(car.test.frame$Mileage,function(x)(x-mean( car.test.frame$Mileage))^2))
[1] 1354.583
mean(car.test.frame$Mileage)
[1] 24.58333
```

$$devianza = \sum_{i=1}^{n}(x_i - \mu)^2$$

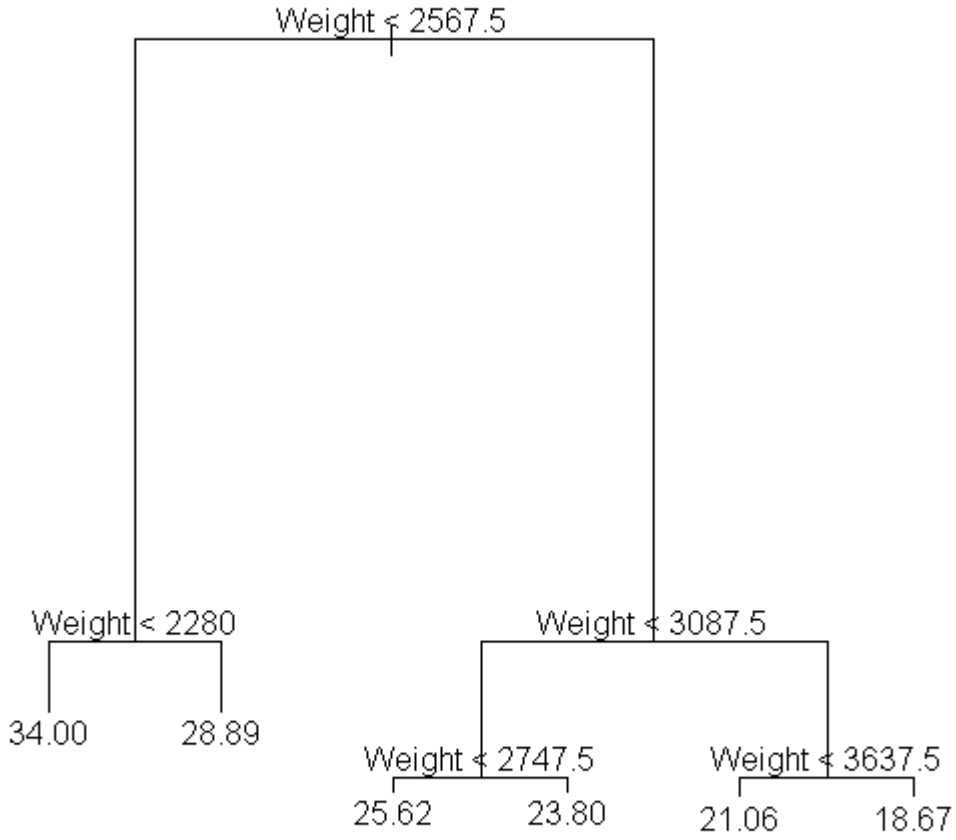
```
summary(rt1)

Regression tree:
tree(formula = Mileage ~ Weight, data car.test.frame)
Number of terminal nodes:  6
Residual mean deviance:  4.249 = 229.4 / 54
Distribution of residuals:
      Min.        1st Qu.     Median       Mean        3rd Qu.      Max.
-3.889e+00 -1.062e+00 -6.250e-02 -3.553e-16  1.233e+00  4.375e+00
```

La funzione *summary* associata alla funzione *tree* elenca il numero dei nodi terminali (foglie), devianza residua media devianza residua totale e (N-numero di nodi terminali), la distribuzione dei residui. La devianza residua totale è uguale alla somma dei quadrati dei residui.

```
sum(sapply(resid(rt1),function(x)(x-mean(resid(rt1)))^2))
 [1] 229.4347
```

Per ottenere il grafico dell'albero usiamo la funzione *plot* e per dare le etichette ai nodi aggiungiamo *text*.

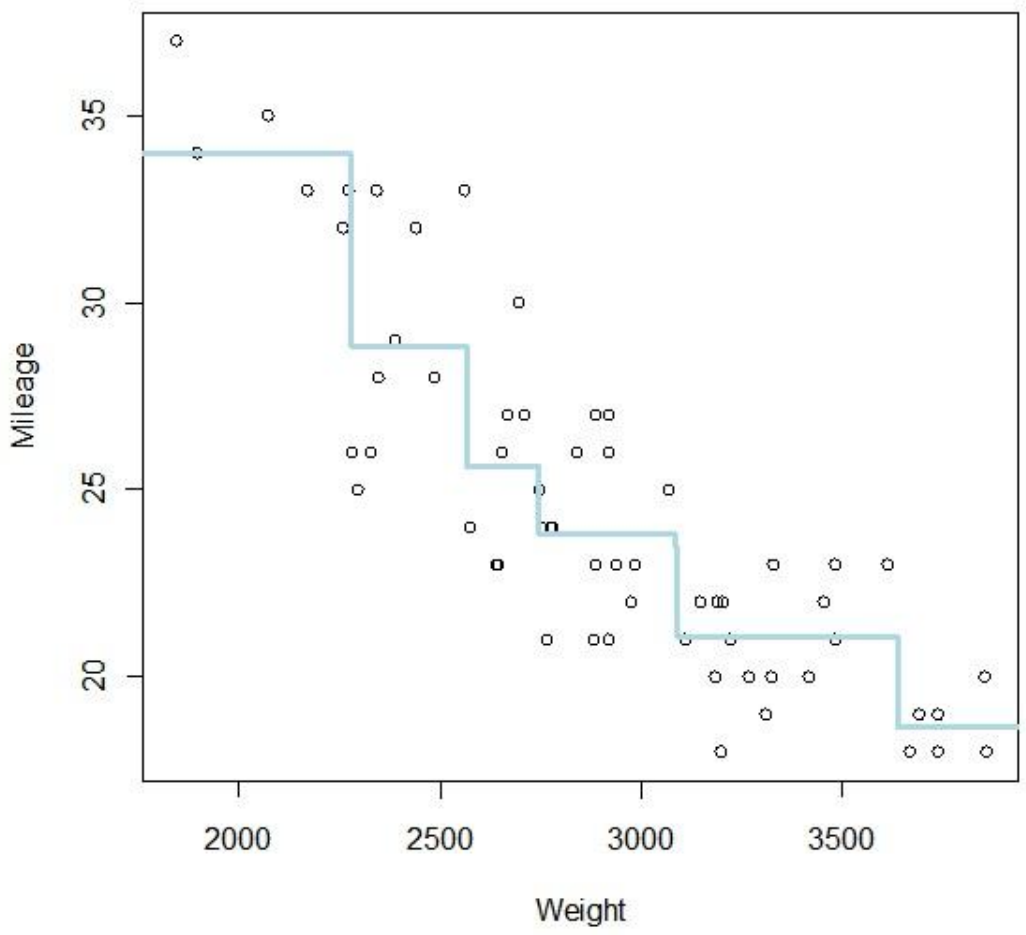
```
plot(rt1)
text(rt1, pretty = 0)
```



Il primo split ha diviso i dati in due insiemi, maggiore e minore di 2567.5 Weigh. Il secondo ha partizionato il secondo gruppo in maggiore e minore di 2280 Weight ed il primo in maggiore e minore di 3087.5 e cosi via. I valori indicati dalle foglie mostrano il valore medio della variabile Mileage di ognuno dei sei sottoinsiemi finali. L'altezza di ogni ramo è proporzionale alla riduzione nella devianza di ogni split.

Per rappresentare graficamente le previsioni del modello e le osservazioni:

```
with(car.test.frame,plot(Weight,Mileage))
x = seq(0,4000)
lines(x,predict(rt1,newdata=list(Weight=x)),col ="lightblue",lwd=3)
```



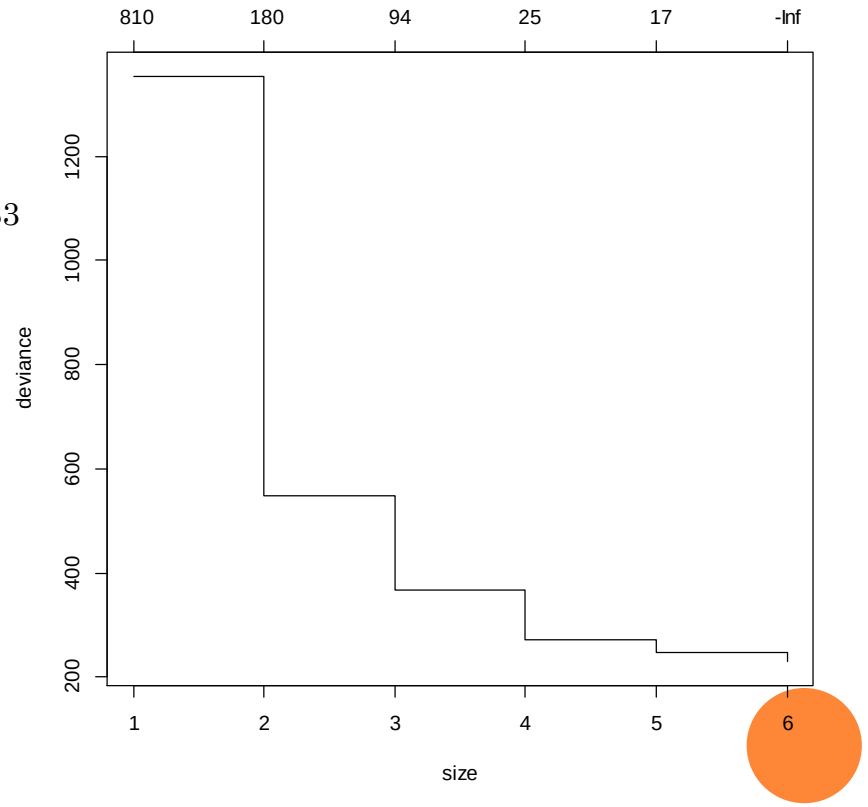
prune.tree : Cost-complexity Pruning of Tree Object Description

Determines a nested sequence of subtrees of the supplied tree by recursively "snipping" off the least important splits.

- size: number of terminal nodes in each tree in the cost-complexity pruning sequence.
- deviance: total deviance of each tree in the cost-complexity pruning sequence.
- k: the value of the cost-complexity pruning parameter of each tree in the sequence.

```
prune.tree(rt1)

$size
[1] 6 5 4 3 2 1

$dev
[1]  229.4347  246.8119  271.8592  365.9037  548.1333 1354.5833

$k
[1]      -Inf  17.37717  25.04735  94.04444 182.22964 806.45000

$method
[1] "deviance"

attr(,"class")
[1] "prune"         "tree.sequence"
```

```
plot(prune.tree(rt1))
```

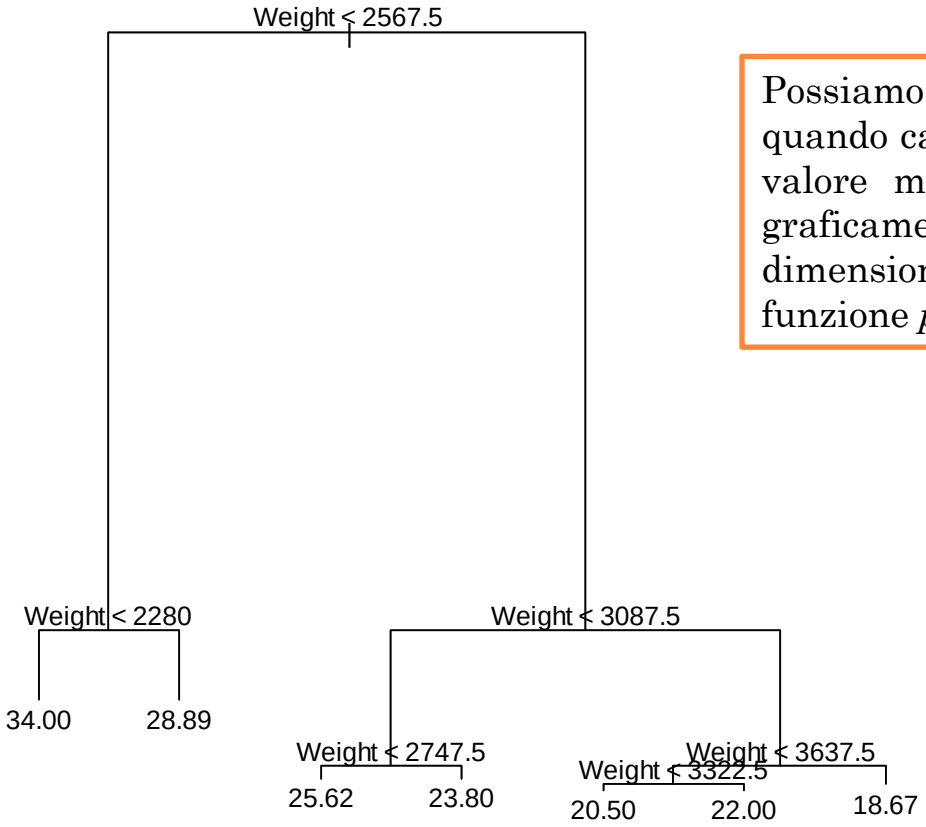


Nella funzione *tree* ci sono i valori default che determinano la regola stop dell'albero (funzione *tree.control*):

- il numero delle osservazioni in un training set (deve essere uguale al numoeri di osservazioni)
- il numero minimo delle osservazioni da inserire in ogni nodo figlio (default 5),
- la dimensione minima del nodo (default 10)
- il valore soglia (default 0.01, il valore aggiunto della devianza spiegata ottenuto con un altro split dovrebbe essere almeno uguale 1% della devianza totale.

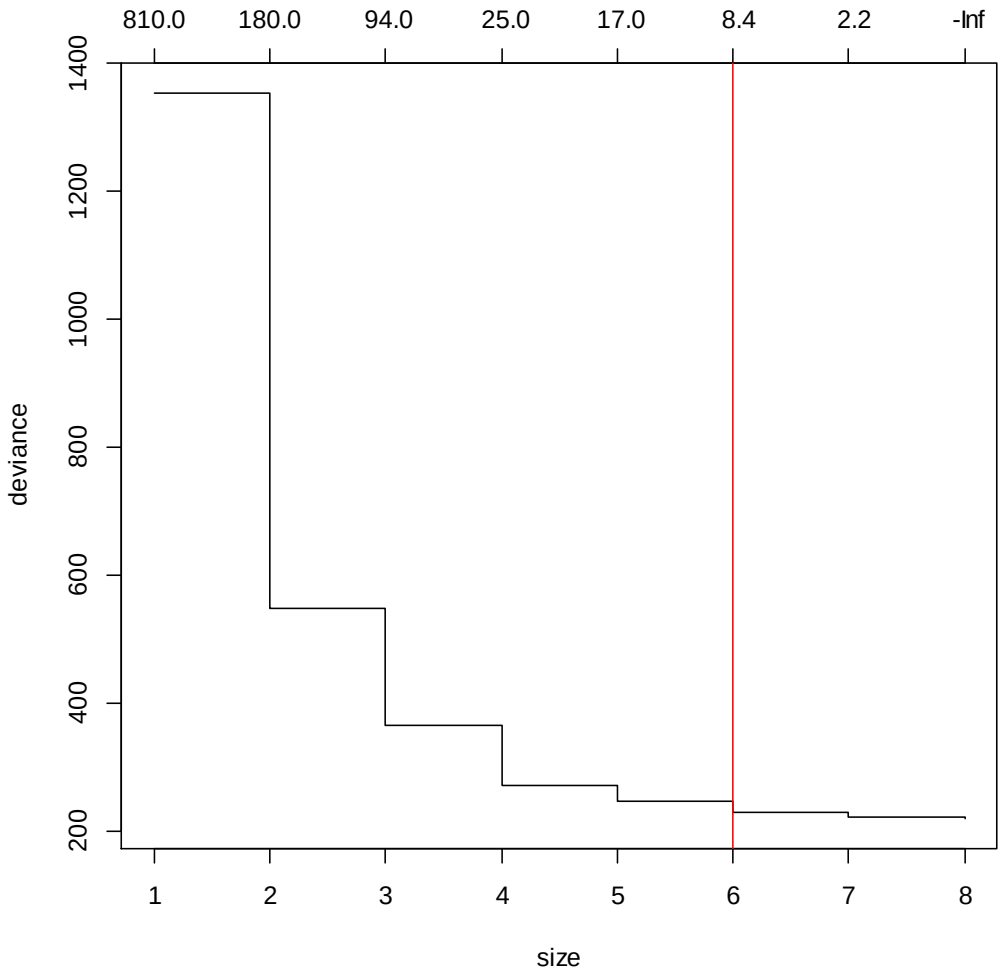
Modello 1.2:

```
rt2 = tree(Mileage~Weight, data=car.test.frame, control=tree.control(60,mindev=.001))
rt2
plot(rt2)
text(rt2, pretty = 0)
```

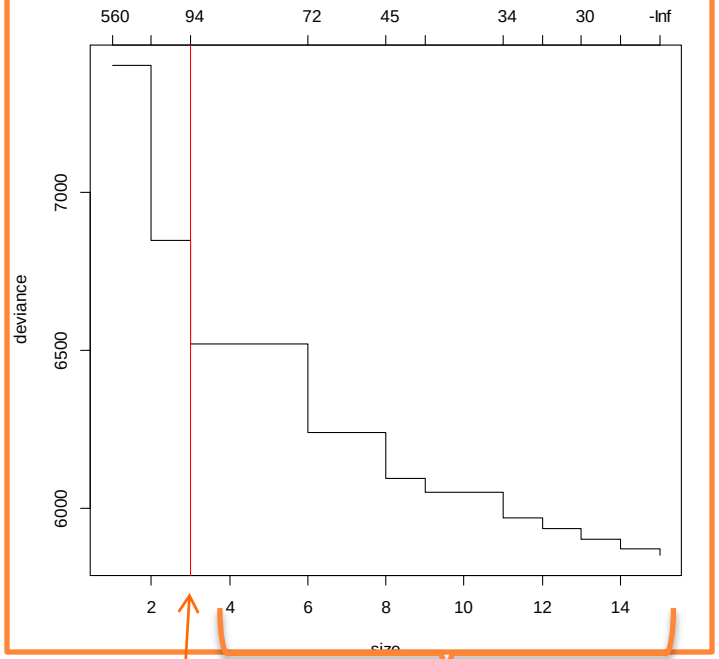


Possiamo controllare che cosa succede quando cambiamo il valore soglia a un valore minore e dopo rappresentare graficamente la varianza residua e la dimensione dell'albero usando la funzione *prune.tree*:

```
plot(prune.tree(rt2))
abline(v=6,col="red")
```



Example: dati ecologici



By changing the default value of mindev to a lower threshold splits are added and the deviance continues to decrease

Number of leaves obtained from the tree with default values of mindev

Sources: http://plantecology.syr.edu/fridley/bio793/cart.html

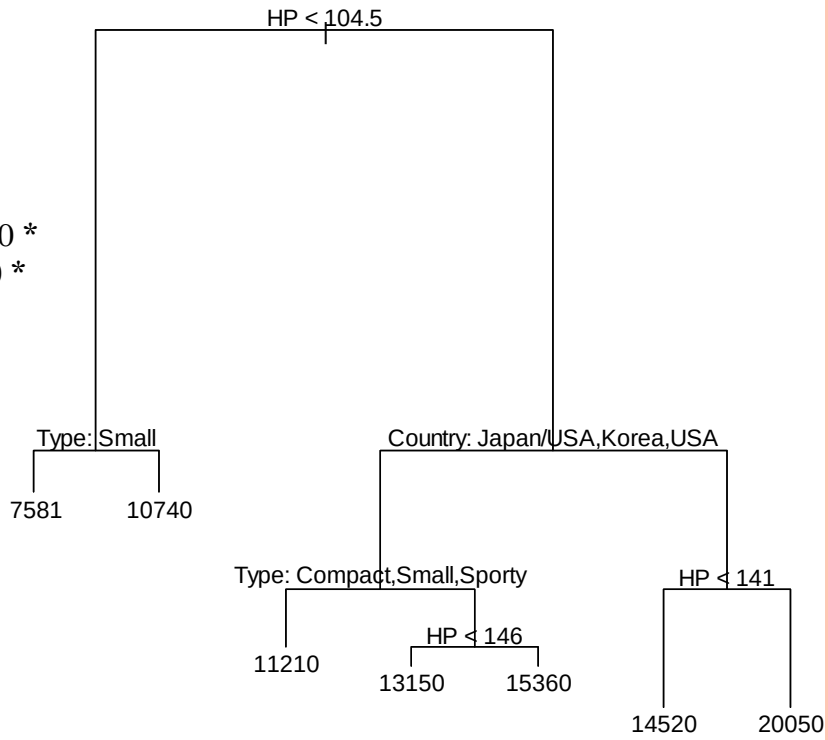
Modello 1.3:

```
rt3 = tree(Price ~ Country+Type+HP, car.test.frame)
rt3
```

node), split, n, deviance, yval
* denotes terminal node

```
 1) root 60 983600000 12620
   2) Weight < 2980 36 283700000 10440
     4) Type: Small 13  21800000  7682
       8) Mileage < 30.5 5   4461000  8854 *
       9) Mileage > 30.5 8   6198000  6950 *
     5) Type: Compact,Medium,Sporty 23 106900000 12000
      10) Country: Japan/USA,Korea,USA 15  21270000 11260 *
      11) Country: France,Germany,Japan 8  61870000 13390 *
   3) Weight > 2980 24 274900000 15880
     6) Country: USA 14  47840000 14190
      12) Type: Compact,Sporty,Van 6  13830000 12720 *
      13) Type: Large,Medium 8  11410000 15290 *
     7) Country: Japan,Sweden 10 131100000 18240
      14) HP < 146 5  16330000 16000 *
      15) HP > 146 5  64620000 20480 *
```

```
plot(rt3)
text(rt3, pretty = 0)
```




```
summary(rt3)
```

```
Regression tree:
tree(formula = Price ~ Country + Mileage + Type + Disp. + HP +
    Weight, data = car.test.frame)
Variables actually used in tree construction:
[1] "Weight"  "Type"    "Mileage" "Country" "HP"
Number of terminal nodes:  8
Residual mean deviance:  3846000 = 2e+08 / 52
Distribution of residuals:
      Min.       1st Qu.      Median        Mean       3rd Qu.       Max.
-5.536e+03 -1.107e+03 -2.439e+02 -3.335e-13  1.077e+03  5.507e+03
```

predict.tree: Predictions from a Fitted Tree Object

It returns a vector of predicted responses from a fitted tree object.

predict(rt3,type="vector")

for Classification Tree type="class"

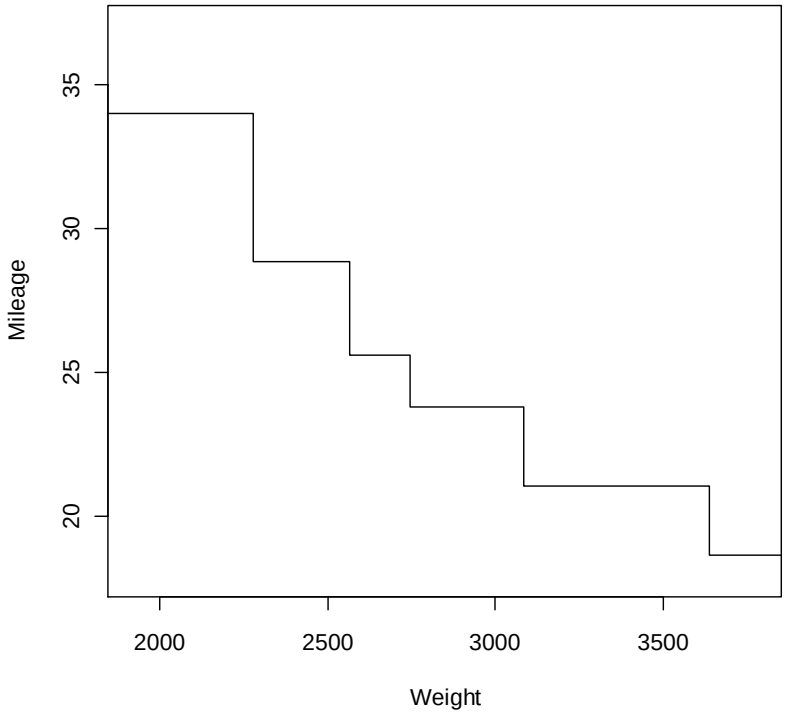
```
> predict(rt3,type="vector")
          Eagle Summit 4           Ford Escort   4
               11206.364                7581.333
          Ford Festiva 4          Honda Civic 4
                7581.333                7581.333
         Mazda Protege 4       Mercury Tracer 4
                7581.333                7581.333
         Nissan Sentra 4      Pontiac LeMans 4
                7581.333                7581.333
         Subaru Loyale 4       Subaru Justy 3
                7581.333                7581.333
        Toyota Corolla 4       Toyota Tercel 4
                7581.333                7581.333
     Volkswagen Jetta 4     Chevrolet Camaro V8
                7581.333               11206.364
           Dodge Daytona      Ford Mustang V8
               10743.833               11206.364
              Ford Probe  Honda Civic CRX Si 4
               11206.364               14521.100
 Honda Prelude Si 4WS 4       Nissan 240SX 4
               14521.100               14521.100
        Plymouth Laser          Subaru XT 4
               10743.833               10743.833
           Audi 80 4          Buick Skylark 4
               14521.100               11206.364
```

partition.tree: Plot the Partitions of a simple Tree Model

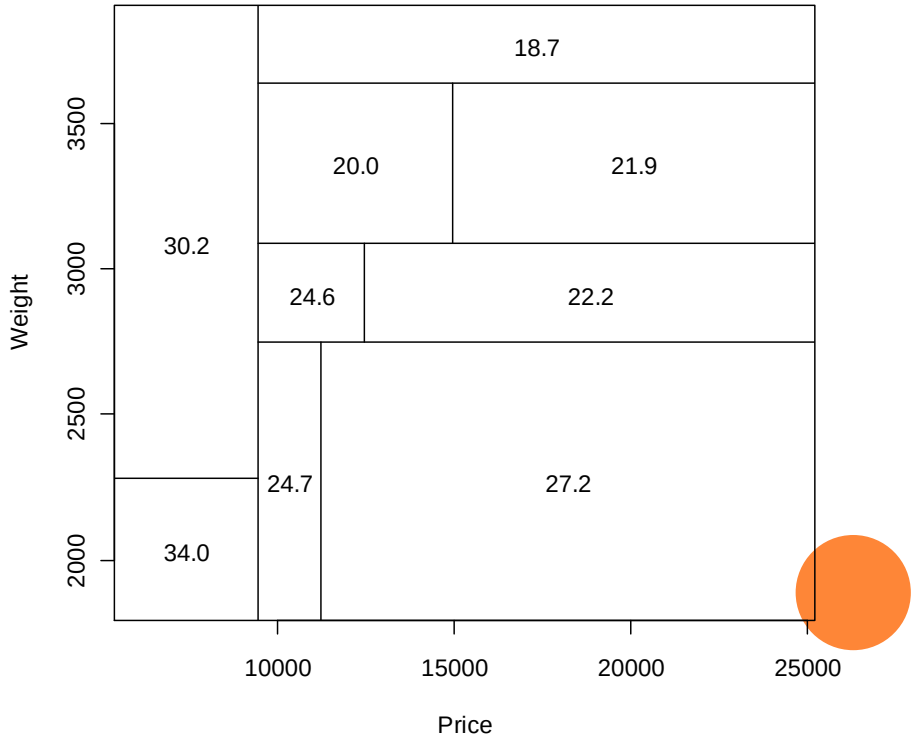
Plot the partitions of a tree involving one or two variables.

This can be used with a regression or classification tree containing one or two continuous predictors (only).
If the tree contains one predictor, the predicted value (a regression tree) or the probability of the first class (a classification tree) is plotted against the predictor over its range in the training set.
If the tree contains two predictors, a plot is made of the space covered by those two predictors and the partition made by the tree is superimposed.

```
rt2 = tree(Mileage~Weight, data=car.test.frame)
partition(rt2)
```




```
rt4 = tree(Mileage~Weight+Price, data=car.test.frame)
partition(rt4)
```



Classification Trees

```
install.packages("tree")
library(tree)
dat = read.csv("Laureati2002.csv", sep=";" , header=TRUE)
```

- **Voto di laurea VOTOLAU1 <-** variabile categorica

1. Basso (VOT1) 2. Alto (VOT2)

- **Genere GENERE**

1. Maschio 2. Femmina

- **Residenza RESID**

1. Napoli 2. PrNAP 3. Italia 4. Campania

- **Età attuale ETATT 1**

1. minore di 25 anni (ETA1) 2. tra 26 e 30 anni (ETA2) 3. tra 31 e 35 (ETA3)
4. oltre 35 anni (ETA4)

- **Diploma DIPLOMA**

1. diploma tecnico 2. m.Class/Ling/Mag 3. m scient 4. D.professionale

- **Anni impiegati per la laurea**

1. 4 anni (ANN1) 2. 5-6 anni (ANN2)3. 7-10 anni (ANN3) 4. >10 anni(ANN4)

- **Tesi di laurea**

1. Giuridica 2. Quanti 3. Economica 4. Aziendale 5. Geografica

ID	GENERE	RESID	VOTOLAU	VOTOLAU1	ETALAU	ETALAU1	ETAATT1	ETATT	DIPLOMA	CORSOLAU	TESI	ANNILAU	ANNILAU1	
1	Maschio	Napoli	91	VOT1	30	ETA2	ETA3	32	D.tecnico	Ec Com	Giuridica	11	ANN4	
2	Maschio	PrNAP	100	VOT1	31	ETA3	ETA2	32	m.Class/Ling/ Mag	Ec Com	Quanti	11	ANN4	
3	Maschio	Italia	84	VOT1	39	ETA4	ETA2	41	D.tecnico	Ec Com	Giuridica	20	ANN4	
4	Maschio	Campania	92	VOT1	29	ETA2	ETA2	30	D.tecnico	Ec Com	Economica	10	ANN3	
5	Femmina	PrNAP	92	VOT1	33	ETA3	ETA2	37	m scie	Ec Com	Economica	12	ANN4	
6	Maschio	Napoli	110	VOT2	29	ETA2	ETA3	31	D.tecnico	Ec Com	Quanti	10	ANN3	
7	Femmina	PrNAP	85	VOT1	30	ETA2	ETA3	32	D.tecnico	Ec Com	Quanti	10	ANN3	
8	Maschio	Napoli	84	VOT1	33	ETA3	ETA3	35	D.tecnico	Ec Com	Economica	13	ANN4	
9	Maschio	Napoli	90	VOT1	38	ETA4	ETA2	41	m scie	Ec Com	Quanti	18	ANN4	
10	Maschio	PrNAP	86	VOT1	39	ETA4	ETA2	43	D.tecnico	Ec Com	Economica	18	ANN4	
11	Femmina	Campania	92	VOT1	29	ETA2	ETA2	30	D.tecnico	Ec Com	Economica	10	ANN3	
12	Femmina	Napoli	97	VOT1	38	ETA4	ETA2	39	D.tecnico	Ec Com	Economica	19	ANN4	
13	Femmina	Napoli	92	VOT1	29	ETA2	ETA3	31	m scie	Ec Com	Giuridica	10	ANN3	
14	Maschio	PrNAP	92	VOT1	31	ETA3	ETA3	34	D.professional e	Ec Com	Economica	10	ANN3	
15	Maschio	PrNAP	92	VOT1	28	ETA2	ETA2	30	D.tecnico	Ec Com	Giuridica	10	ANN3	
16	Maschio	PrNAP	99	VOT1	29	ETA2	ETA2	30	m scie	Ec Com	Giuridica	10	ANN3	
17	Femmina	Napoli	99	VOT1	31	ETA3	ETA3	33	D.tecnico	Ec Com	Giuridica	12	ANN4	
18	Femmina	PrNAP	102	VOT2	30	ETA2	ETA3	31	m.Class/Ling/ Mag	Ec Com	Economica	12	ANN4	
19	Maschio	PrNAP	99	VOT1	33	ETA3	ETA3	35	m scie	Ec Com	Economica	12	ANN4	
20	Femmina	Napoli	103	VOT2	31	ETA3	ETA3	33	D.professional e	Ec Com	Aziendale	13	ANN4	
21	Maschio	Napoli	95	VOT1	29	ETA2	ETA3	32	m scie	Ec Com	Giuridica	10	ANN3	
22	Femmina	PrNAP	101	VOT2	29	ETA2	ETA3	32	m scie	Ec Com	Quanti	10	ANN3	
23	Maschio	PrNAP	98	VOT1	44	ETA4	ETA2	45	m scie	Ec Com	Aziendale	16	ANN4	
24	Maschio	PrNAP	92	VOT1	28	ETA2	ETA2	29	D.tecnico	Ec Com	Giuridica	10	ANN3	
25	Maschio	PrNAP	102	VOT2	28	ETA2	ETA2	29	m scie	Ec Com	Giuridica	10	ANN3	
26	Femmina	PrNAP	108	VOT2	28	ETA2	ETA2	29	m.Class/Ling/ Mag	Ec Com	Giuridica	11	ANN4	
27	Femmina	Napoli	89	VOT1	30	ETA2	ETA3	33	D.tecnico	Ec Com	Aziendale	10	ANN3	

Modello 2.1:

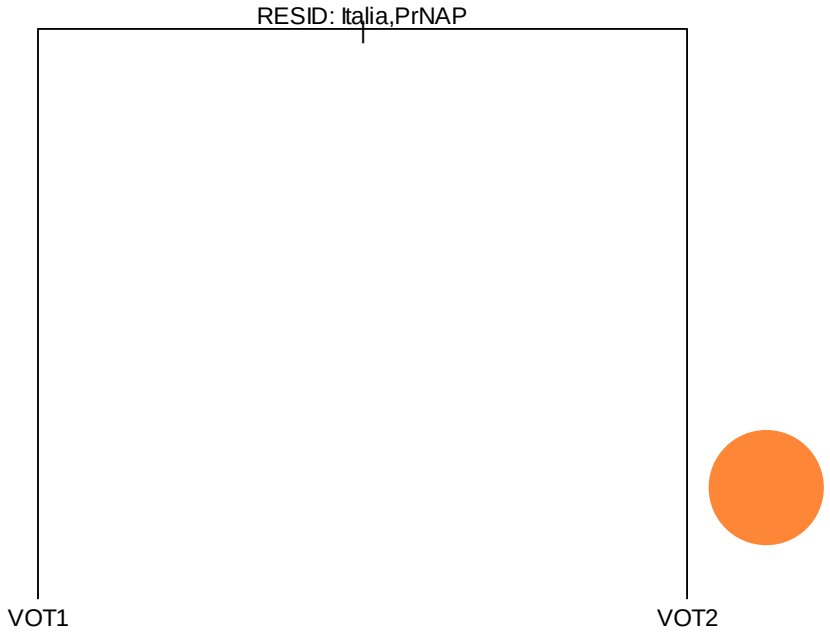
In questo modello consideriamo una variabile dipendente (qualitativa) VOTOLAU1 ed una sola variabile esplicativa RESID.

Il risultato dell'albero di classificazione rispetto alla regressione è leggermente diverso, in quanto oltre alle informazioni relative alla soglia di ogni divisione, la dimensione del campione nodo, la devianza residua associata, e il valore previsto, si ottiene anche la frazione di presenze e assenze (per esempio, la frequenza di ciascuna classe di risposta) per i campioni di ciascun nodo.

```
ct1 = tree(VOTOLAU1 ~ RESID, dat)
ct1
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

1) root 241 333.2 VOT2 ( 0.4689 0.5311 )
  2) RESID: Italia,PrNAP 105 143.9 VOT1 ( 0.5619 0.4381 ) *
  3) RESID: Campania,Napoli 136 182.7 VOT2 ( 0.3971 0.6029 ) *
```

```
summary(ct1)

Classification tree:
tree(formula = VOTOLAU1 ~ RESID, data = dat)
Number of terminal nodes:  2
Residual mean deviance:  1.367 = 326.7 / 239
Misclassification error rate: 0.4149 = 100 / 241
```

```
plot(ct1)
text(ct1,pretty=0)
```



L'INDICE DI ETEROGENEITÀ DI GINI

L'indice di Gini I è una misura della eterogeneità (omogeneità) di una distribuzione statistica a partire dai valori delle frequenze relative associate alle k modalità di una generica variabile X

Se i dati sono distribuiti in modo eterogeneo su tutte le k modalità di X (cioè, se le modalità hanno numerosità simili), l'indice di Gini è elevato

Il suo massimo (tutte le numerosità sono uguali) è pari a $k/(k\text{-}1)$

Viceversa, in caso di distribuzione di frequenza omogenea (tutti gli individui appartengono ad una sola classe) l'indice sarà pari a 0.

$$I = 1 - \sum_{i=1}^{k} f_i^2$$

ESEMPI DI USO DELL'INDICE DI GINI

Nella scelta dello split, una volta che un nodo t è stato suddiviso in k nodi figlio, si calcola il cosiddetto I_{split} :

$$I_{split} = \sum_{i=1}^{k} \frac{n_i}{n} I(i)$$

dove n_i è il numero di elementi nel nodo figlio i

e n è il numero di elementi nel nodo t

n_i/n è il peso dei vari indici di I(i)

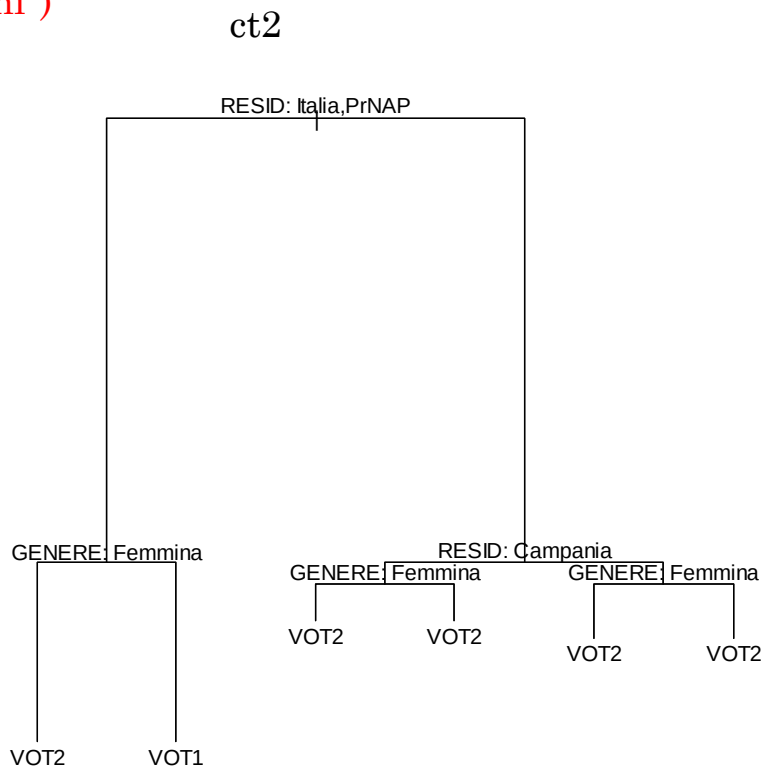
Si sceglie lo split corrispondente all' I_{split} più piccolo

Per scegliere quindi il miglior *split* occorre enumerare tutte le possibili partizioni

Modello 2.2a:

```
ct2 = tree(VOTOLAU1 ~ RESID+GENERE, dat,split="gini")
ct2
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 241 333.20 VOT2 ( 0.4689 0.5311 )
   2) RESID: Italia,PrNAP 105 143.90 VOT1 ( 0.5619 0.4381 )
     4) GENERE: Femmina 50  69.23 VOT2 ( 0.4800 0.5200 ) *
     5) GENERE: Maschio 55  72.10 VOT1 ( 0.6364 0.3636 ) *
   3) RESID: Campania,Napoli 136 182.70 VOT2 ( 0.3971 0.6029 )
     6) RESID: Campania 31  40.32 VOT2 ( 0.3548 0.6452 )
      12) GENERE: Femmina 17  23.03 VOT2 ( 0.4118 0.5882 ) *
      13) GENERE: Maschio 14  16.75 VOT2 ( 0.2857 0.7143 ) *
     7) RESID: Napoli 105 142.10 VOT2 ( 0.4095 0.5905 )
      14) GENERE: Femmina 42  54.75 VOT2 ( 0.3571 0.6429 ) *
      15) GENERE: Maschio 63  86.56 VOT2 ( 0.4444 0.5556 ) *
```

```
plot(ct2)
text(ct2, pretty=0)
```



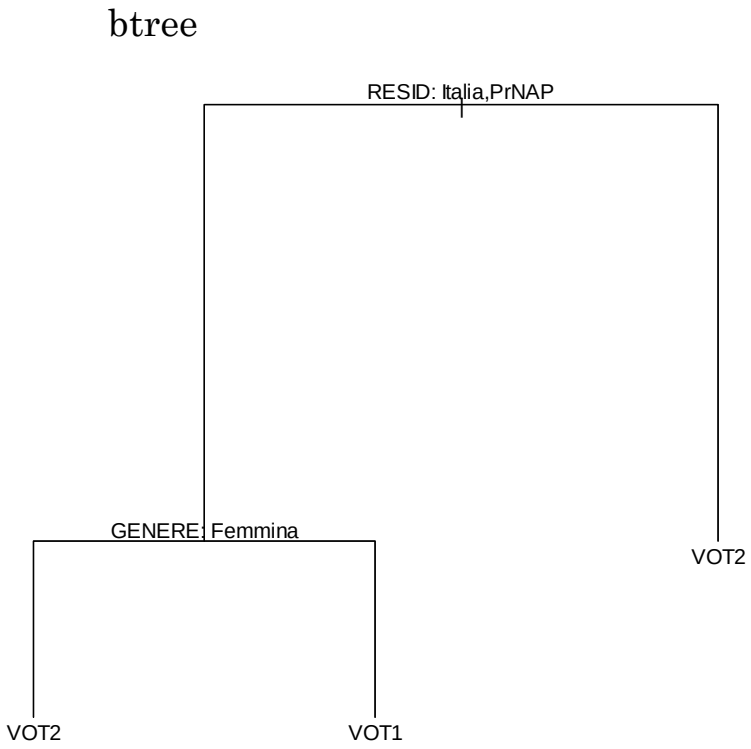

```
summary(ct2)

Classification tree:
tree(formula = VOTOLAU1 ~ RESID + GENERE, data = dat, split = "gini")
Number of terminal nodes:  6
Residual mean deviance:  1.372 = 322.4 / 235
Misclassification error rate: 0.4066 = 98 / 241
```

Modello 2.2b:

```
btree = prune.tree(ct2,best=3)
plot(btree)
text(btree, pretty=0)
btree
```

node), split, n, deviance, yval, (yprob)
* denotes terminal node

1) root 241 333.20 VOT2 (0.4689 0.5311)
  2) RESID: Italia,PrNAP 105 143.90 VOT1 (0.5619 0.4381)
    4) GENERE: Femmina 50 69.23 VOT2 (0.4800 0.5200) *
    5) GENERE: Maschio 55 72.10 VOT1 (0.6364 0.3636) *
  3) RESID: Campania,Napoli 136 182.70 VOT2 (0.3971 0.6029) *

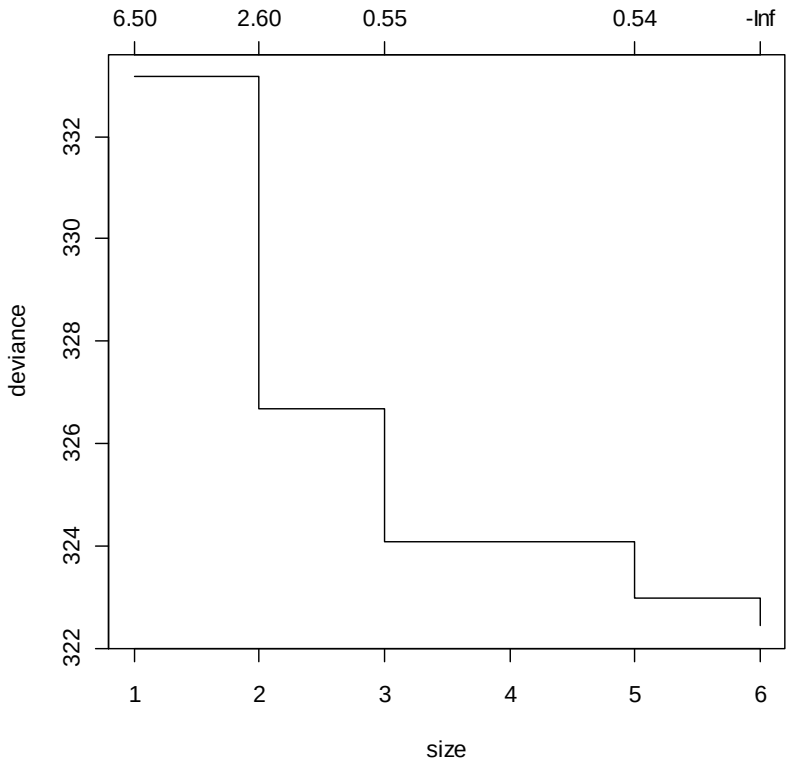



```
summary(btree)
```

Classification tree:
snip.tree(tree = ct2, nodes = 3)
Number of terminal nodes: 3
Residual mean deviance: 1.362 = 324.1 / 238
Misclassification error rate: 0.4066 = 98 / 241

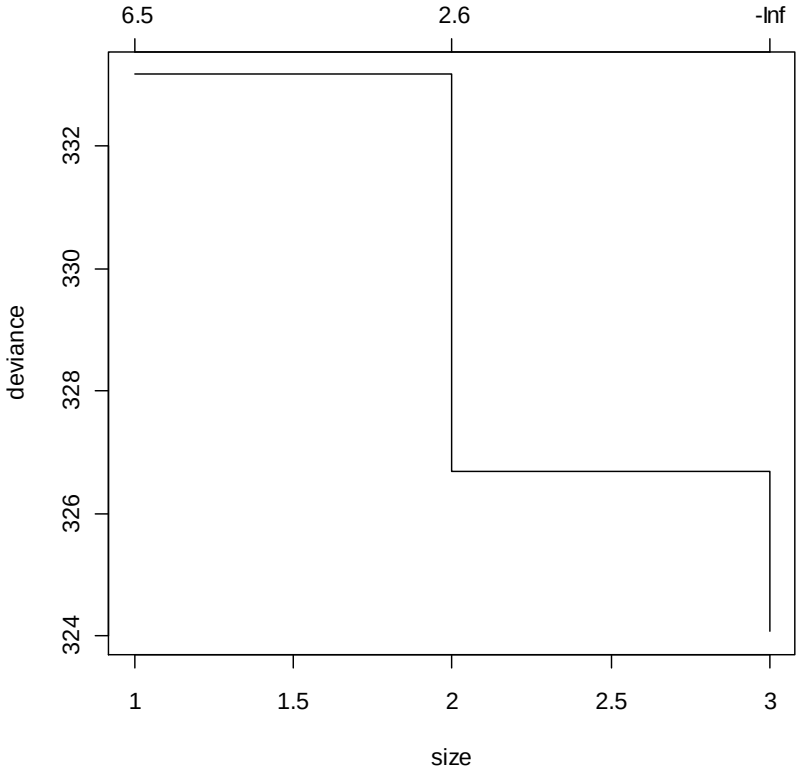
prune.tree(ct2)

```
$size
[1] 6 5 3 2 1
$dev
[1] 322.4288 322.9666 324.0676 326.6771 333.1627
$k
[1]      -Inf 0.5378615 0.5504829 2.6095548 6.4855853
$method
[1] "deviance“
attr(,"class")
[1] "prune"         "tree.sequence"
```



prune.tree(btree)

```
$size
[1] 3 2 1
$dev
[1] 324.0676 326.6771 333.1627
$k
[1]     -Inf 2.609555 6.485585
$method
[1] "deviance“
attr(,"class")
[1] "prune"         "tree.sequence"
```



predict(btree,type="class")

```
> predict.tree(btree,type="class")
  [1] VOT2 VOT1 VOT1 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT1 VOT2 VOT2 VOT2 VOT1 VOT1
 [16] VOT1 VOT2 VOT2 VOT1 VOT2 VOT2 VOT2 VOT1 VOT1 VOT1 VOT2 VOT2 VOT1 VOT2 VOT2
 [31] VOT2 VOT2 VOT2 VOT2 VOT2 VOT1 VOT2 VOT2 VOT2 VOT2 VOT1 VOT2 VOT1 VOT1 VOT1
 [46] VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT1 VOT2 VOT1 VOT2 VOT2
 [61] VOT2 VOT1 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT1 VOT1 VOT2
 [76] VOT2 VOT2 VOT2 VOT2 VOT2 VOT1 VOT1 VOT2 VOT2 VOT1 VOT1 VOT2 VOT2 VOT2 VOT1
 [91] VOT2 VOT2 VOT2 VOT2 VOT1 VOT2 VOT2 VOT2 VOT2 VOT2 VOT1 VOT2 VOT2 VOT2 VOT2
[106] VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT1 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT1 VOT2
[121] VOT2 VOT2 VOT1 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT1 VOT2 VOT2 VOT2
[136] VOT2 VOT1 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2
[151] VOT2 VOT1 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT1 VOT2 VOT1 VOT2 VOT2 VOT2 VOT2
[166] VOT2 VOT2 VOT2 VOT2 VOT1 VOT2 VOT1 VOT2 VOT2 VOT1 VOT2 VOT2 VOT1 VOT2 VOT2
[181] VOT2 VOT2 VOT2 VOT1 VOT1 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT1 VOT2
[196] VOT2 VOT1 VOT2 VOT2 VOT2 VOT2 VOT1 VOT1 VOT1 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2
[211] VOT1 VOT2 VOT1 VOT2 VOT1 VOT2 VOT2 VOT1 VOT2 VOT2 VOT2 VOT2 VOT1 VOT1 VOT1
[226] VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT2 VOT1 VOT2 VOT2 VOT2
[241] VOT2
Levels: VOT1 VOT2
>
```

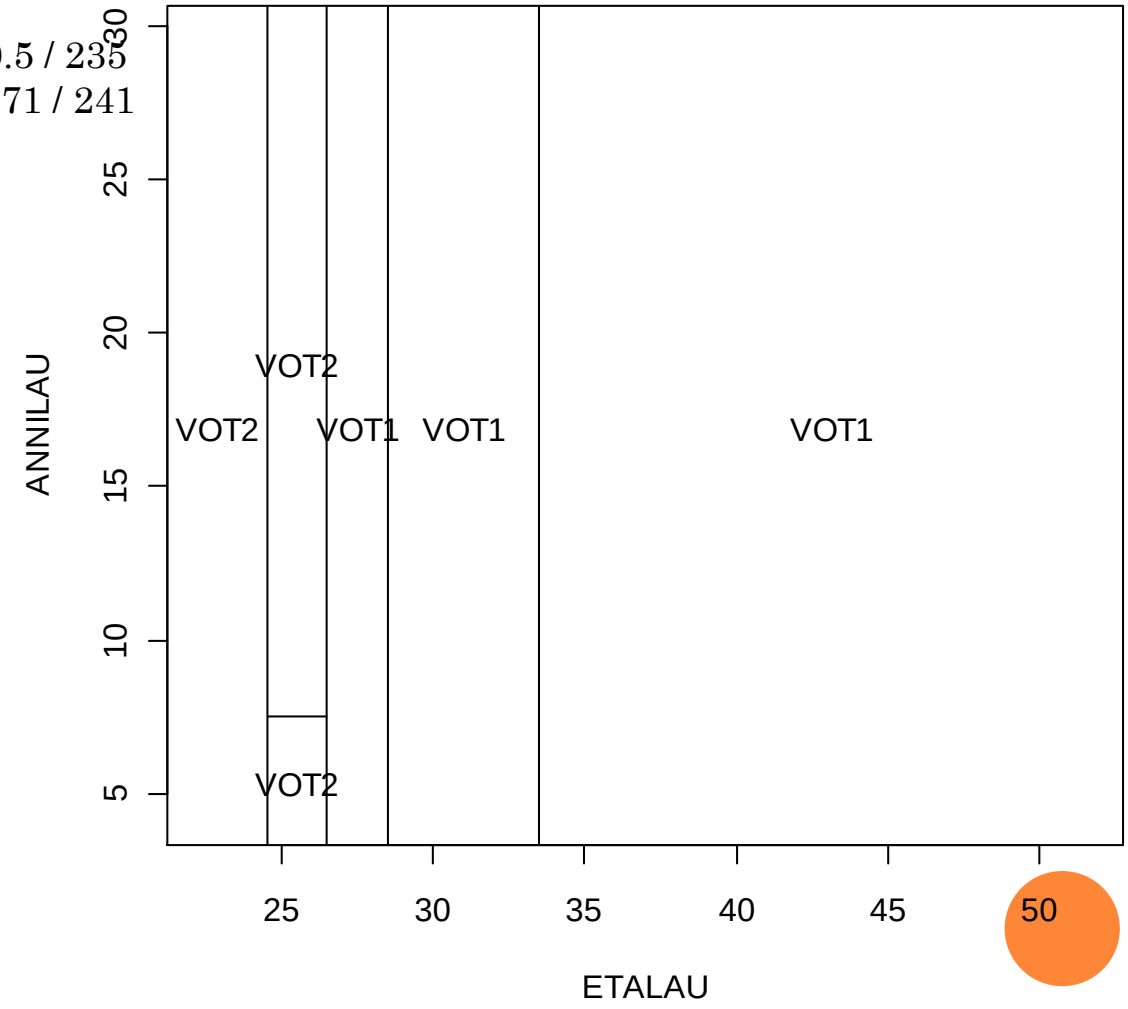
misclass.tree: Misclassifications by a Classification Tree

Report the number of mis-classifications made by a classification tree, either overall or at each node.
Either the overall number of misclassifications or the number for each node.

misclass.tree(btree)
[1] 98

```
ct3 = tree(VOTOLAU1~ETALAU+ANNILAU, data=dat)
partition(ct3)
```

```
summary(ct3)
Classification tree:
tree(formula = VOTOLAU1 ~ ETALAU + ANNILAU,
data = dat)
Number of terminal nodes:  6
Residual mean deviance:  1.109 = 260.5 / 235
Misclassification error rate: 0.2946 = 71 / 241
```

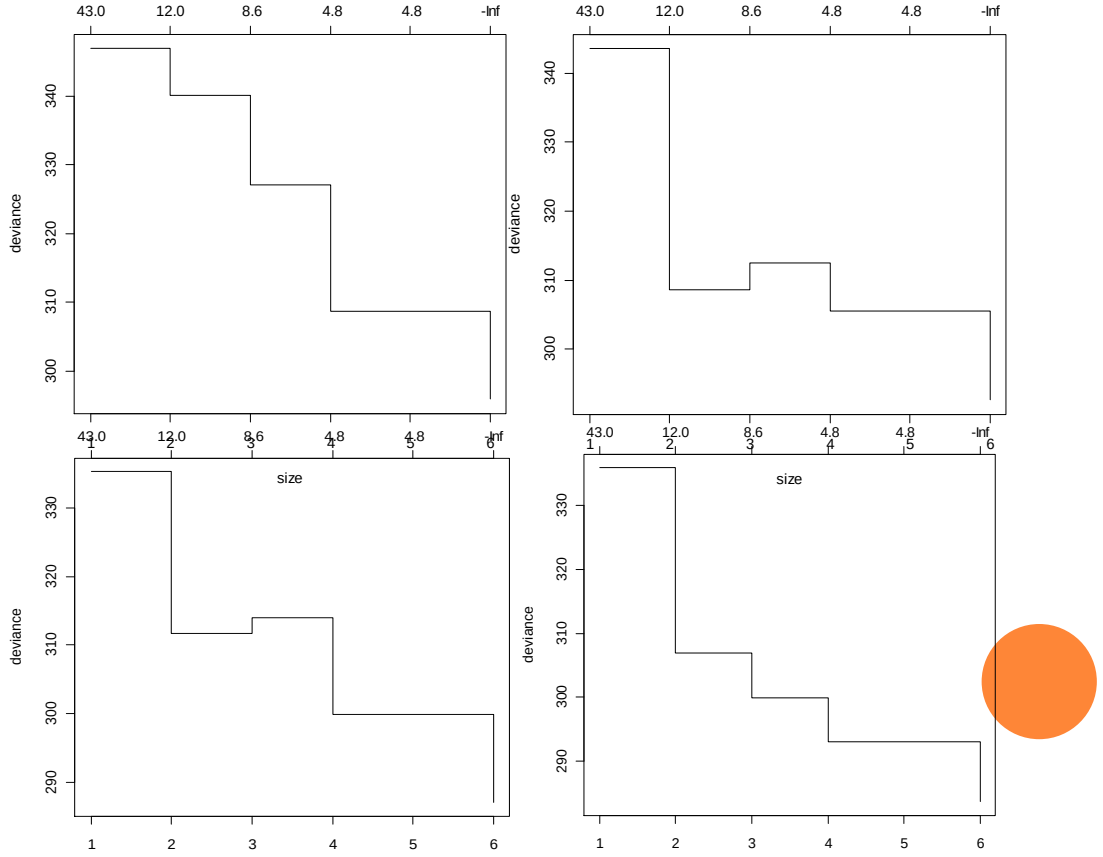


Considerando che i modelli di alberi tendono a sovrastimare dati, come si fa a sapere quando abbiamo un buon modello? Per utilizzare un criterio stop basato su dati di quando smettere di tagliare è stata proposta una tecnica di validazione incrociata (cross-validation) che suddivide i dati in un training set per la creazione del modello e un validation set per valutare la bontà di adattamento. Nella funzione *cv.tree* l'impostazione predefinita è un '10-fold 'cross-validazione, il che significa che il 10% dei dati è lasciato fuori di ogni training set.

cv.tree: Cross-validation for Choosing Tree Complexity

Runs a K-fold cross-validation experiment to find the deviance or number of misclassifications as a function of the cost-complexity parameter k.

cv.tree(ct3)



Una volta verificata una scara stabilità e bontà del modello si applica la funzione *prune.tree* per stabilire il miglior taglio dell'albero

Compiti:

1). Provate creare un modello più complesso dell'albero di classificazione per i dati "Laureati2002".

2) E' possibile usare questi dati anche per creare gli alberi di regressione?

3) Caricate il dataset "iris", analizzate le variabili e create almeno un albero di regressione ed un albero di classificazione

```
data(iris)
?iris
```

Iris is a data frame with 150 cases (rows) and 5 variables (columns) named Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, and Species.