UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
School of Professional & Executive Development

Decision Trees

Tomàs Aluja

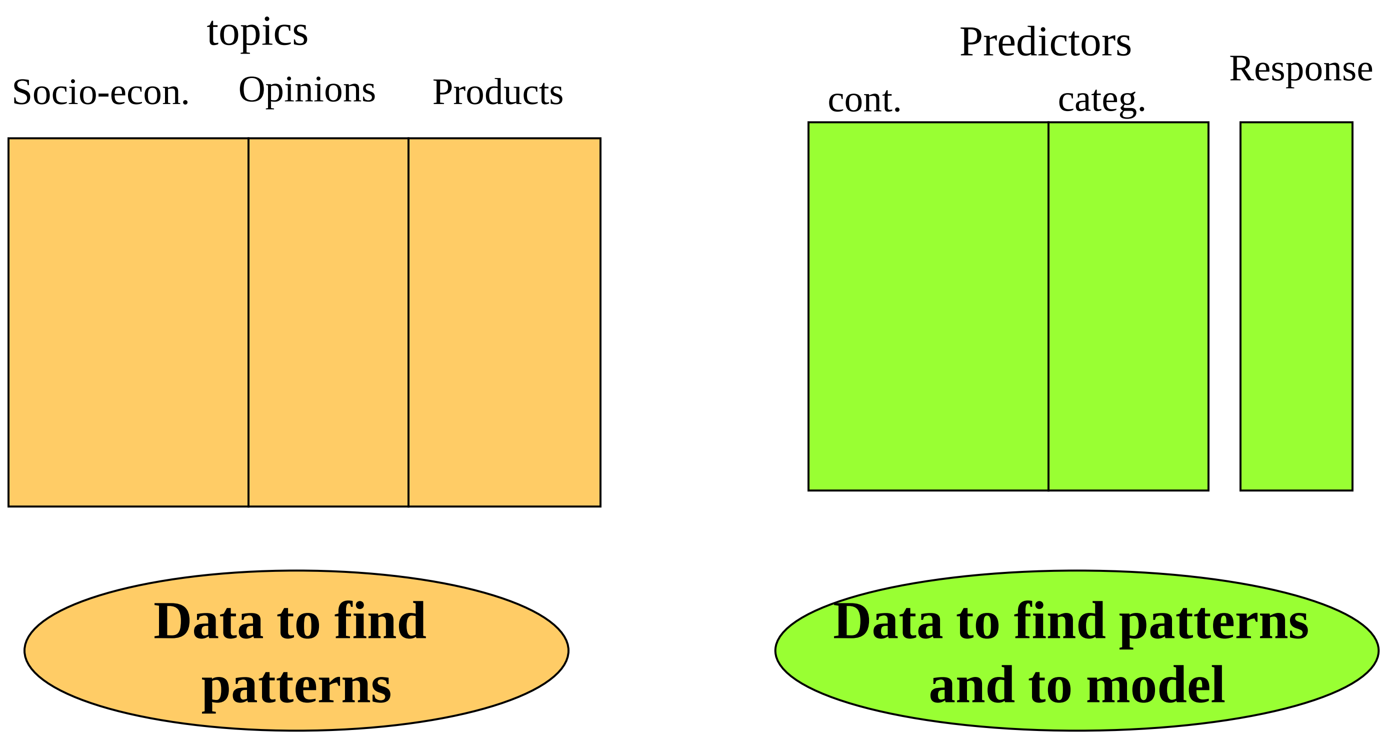
Barcelona; January 13th, 2017

Outline

1. Decision trees
2. CART: Classification and Regression Trees methodology
3. Application to the credit scoring problem
4. Validation of the tree
5. Handling unbalanced classes
6. Improving the precision: Random Forests

Two types of data files

Data in *Big Data*:
massive, not obtained by random sampling, with errors, outliers and missing values



The holy grail: Prediction

- Prediction has always been the key to success or failure.
  "Be aware of the ides of March (Julius Caesar, 44BC)".

- The central task of Data Mining is prediction
  - **Classification** if response is categorical or
  - **Regression** if response is continuous.

- There are plenty of methods (models) to perform prediction: linear regression, logistic regression, discriminant analysis, Naive Bayes, Knn, Neural networks, SVM, ensemble methods, deep learning, … (some come from the Statistics community others from the Machine Learning community and others from both, i. e. the decision trees)

- We start with Decision Trees

DECISION TREES

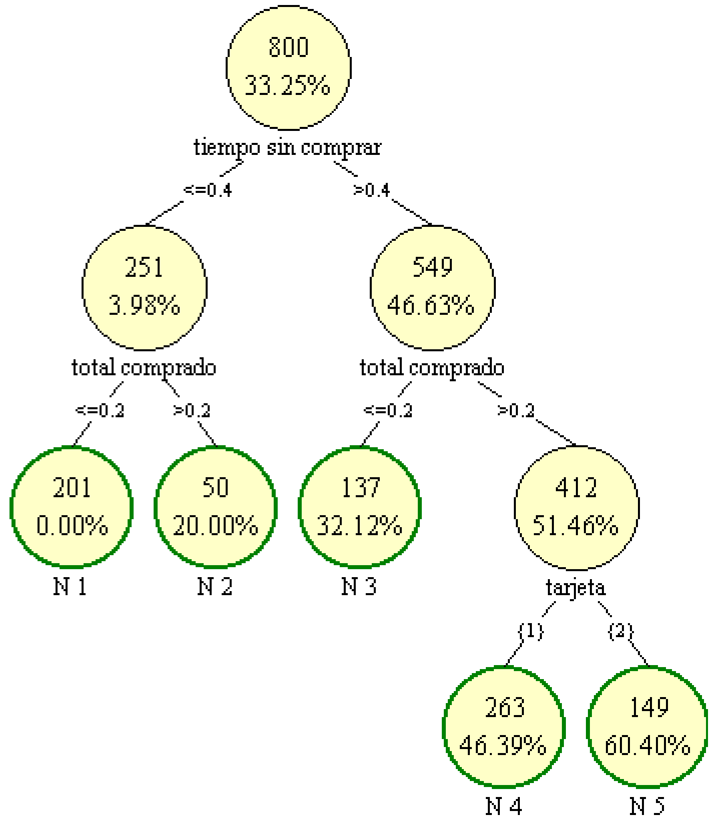
What is a decision tree

- The decision trees are a non parametric method to perform predictions. Its popularity comes from the simplicity of the results (everyone can understand them).
- Visual output, very easy to interpret.
- Outcome directly operational. Every branch of the tree defines a target with a specific evaluated response prediction.
- Minimize pretreatment of data, every variable can be used as it is. Robust respect to the presence of "some" outlying observations.
- But predictions can be less accurate than in other more sophisticated techniques.

Trees may differ:
- Type of the response variable
  - Regression trees (continuous response)
  - Classificaton trees (categorical response)
- Type of explanatory variables (bin, nom, ord, con)

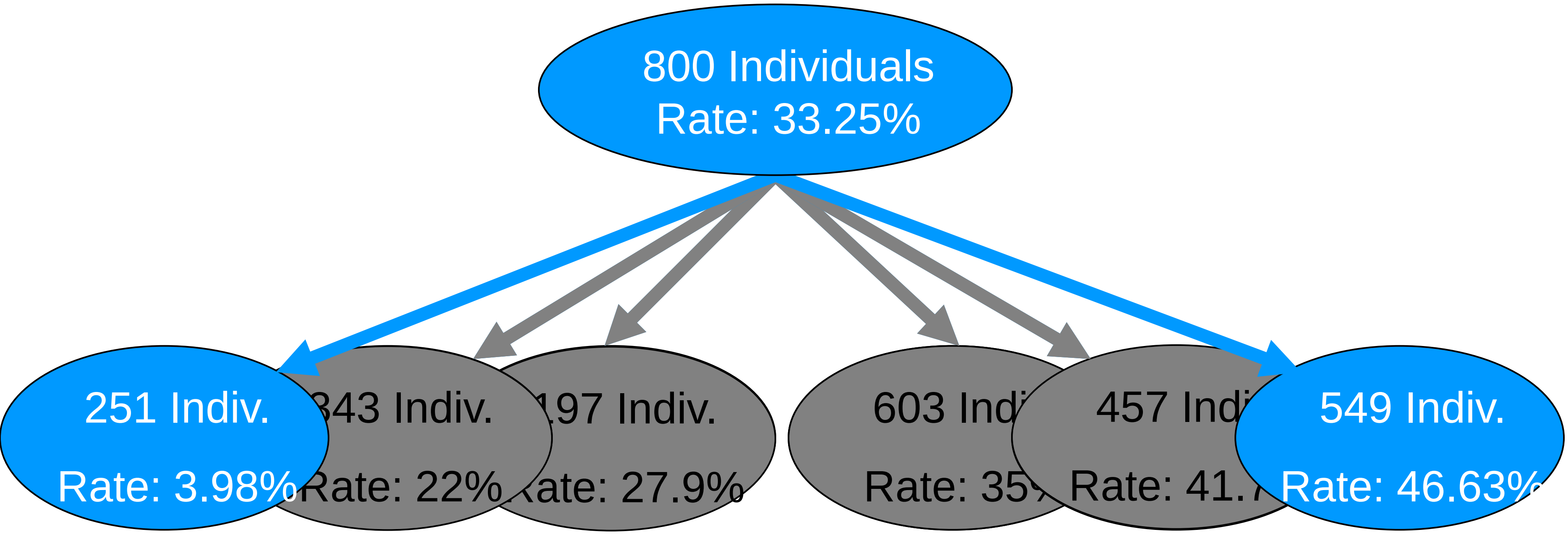
- **Binary** or **multi-way** tree

Árbol: Árbol buenos compradores
Var.Resp: compradores



Building the tree

Purpose: To segment the data in order to find homogeneous groups respect to the response variable.



Split variable :

Decision tree Algorithm

Top-down, recursive, greedy algorithm:

1. Set all individuals in the root node
2. Evaluate all possible splits and find the optimal partition in children nodes.
3. For every child node: Decide if we stop the process or we go back to step 2.

We need:

- a split criterion
- a stop criterion (if prepruning)

How many splits can we make in a node

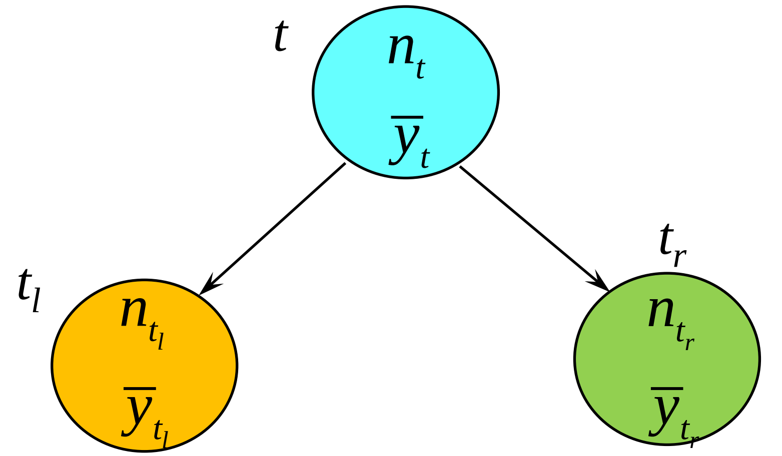
Splits are defined by the number and the type of the explanatory variables and the type of the tree

	Multi way	*Binary tree*	
Binary	*1*	*1*	
Nominal	*1*	$2^{q-1}-1$	★ Attention (q levels)
Ordinal	*1*	$q-1$	
Continuous	n_t-1	n_t-1	★ Attention (n_t distinct values)

Algorithms for decision trees

- AID (Automatic Detection Interaction, Sonquist and Morgan 1964)
  - Response: Continuous
  - Binary tree, but can be extended to multiway
  - Split criterion: Fisher's F
  - Stop criterion: threshold upon the p.value of the split
- CHAID (Kaas, 1980)
  - Response: Categorical
  - Binary tree, but can be extended to multiway
  - Split criterion: $\chi 2$
  - Stop criterion: threshold upon the p.value of the split
- Information based trees (ID3, C4.5, C5.0, …) (Ross Quinlan, 1975)
  - Response: Any
  - Multiway tree
  - Split criterion: *Entropy gain*
  - Pruning according missclassification probability
- CART family (1984)
  - Response: Any
  - Binary tree
  - Split criterion: *Impurity gain*
  - Pruning according a complexity parameter

AID split criterion



Response variable: **continuous** y

AID split criterion is based in the wellknown decomposition of variance, *between* the nodes and *whithin* the nodes.
The interest is to maximize the between part.

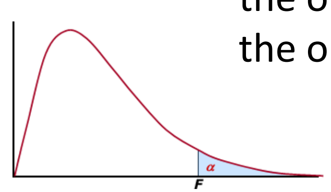
Total variation of y in node t

$$\sum_{i=1}^{n_t}(y_i-\overline{y}_t)^2 = \underbrace{n_{t_l}(\overline{y}_{t_l}-\overline{y}_t)^2 + n_{t_r}(\overline{y}_{t_r}-\overline{y}_t)^2}_{SSB} + \underbrace{\sum_{i\in t_l}^{n_{t_l}}(y_i-\overline{y}_{t_l})^2 + \sum_{i\in t_r}^{n_{t_r}}(y_i-\overline{y}_{t_r})^2}_{SSW} = SSB + SSW$$

If splits are at random:

$$F = \frac{SSB/q-1}{SSW/n-q} \sim F_{q-1,n_t-q}$$

in binary splits:
$q=2$

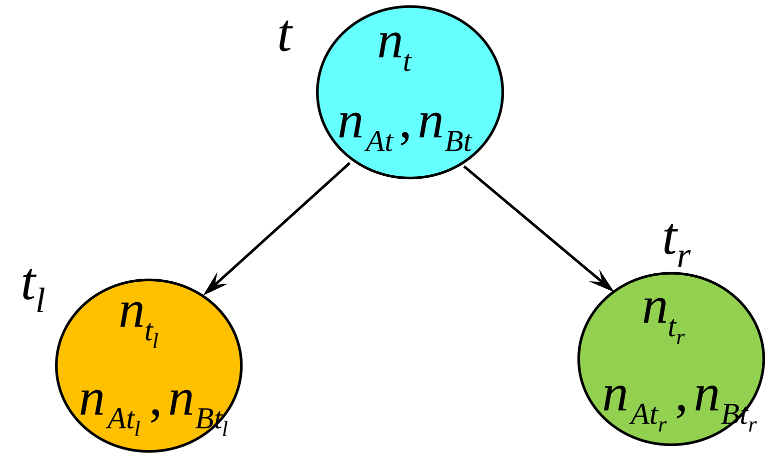


Then, among all possible splits in a node, the one giving the most significant F defines the optimal split

CHAID split criterion

Response variable: **Categorical** (suppose with two modalities : A, B)

CHAID split criterion is based in the *independence test* between two categorical variables (the *response* and the *tentative split*)



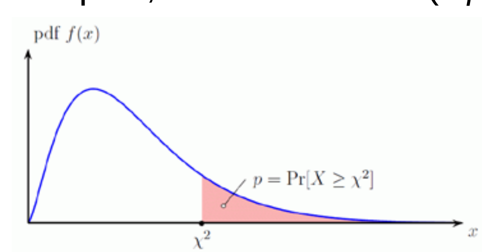
tentative split

response			
	n_{At_l}	n_{At_r}	n_{At}
	n_{Bt_l}	n_{Bt_r}	n_{Bt}
	n_{t_l}	n_{t_r}	n_t

$$\chi^2 = \sum_{j=A}^{B}\sum_{k=l}^{r} \frac{\left(n_{jt_k} - n_{t_k}\dfrac{n_{jt}}{n_t}\right)^2}{n_{t_k}\dfrac{n_{jt}}{n_t}} \sim \chi^2_{df=(r-1)(c-1)}$$

r is the number of classes of the response variable, in our case 2 (A and B).
c is the number of child splits, in our case two (*left* and *right*) since it is binary

The split giving the most significant χ^2 defines the optimal one



Business applications of decision trees

- Customer Segmentation
- Direct Marketing
- Customer Retention (Attrition)
- Cross selling
- Basket Analysis
- Fraud Detection
- . . .

CART: CLASSIFICATION AND REGRESSION TREES

Classification and Regression Trees - CART

Leo Breiman,
American, 1928-2005

Jerome Friedman,
American

Developed 1974-1984 by 4 statistics professors: Leo Breiman (Berkeley), Jerry Friedman (Stanford), Charles Stone (Berkeley), Richard Olshen (Stanford). Distributed by Salford Systems http://salford-systems.com/

- Just performs Binary trees
- Unifies the categorical and continuous response under the same framework.
  - **Classification tree**
  - **Regression tree**
- Any kind of explanatory variable
- Split criterion: Impurity of the node
- Post pruning (without stop criterion)
- Delivers honest estimates of the quality of a tree

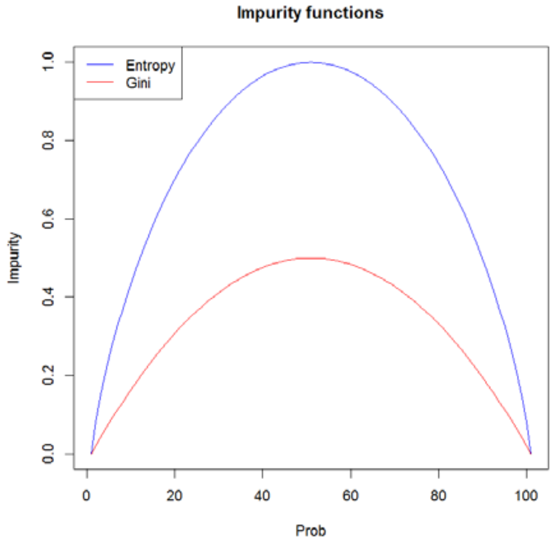
Split criterion in CART

Impurity of a node

For categorical responses (classification tree):

j indicates a response class
t indicates a node

- Gini (≈variance): $$i(t)=\sum_{j\neq j'} p(C_j|t)\,p(C_{j'}|t)$$
- Entropy (information): $$i(t)=-\sum_{j} p(C_j|t)\log_2 p(C_j|t)$$



Gini impurity index in case of 2 classes
$$i(t)=2\times p(C_1/t)\times p(C_2/t)$$

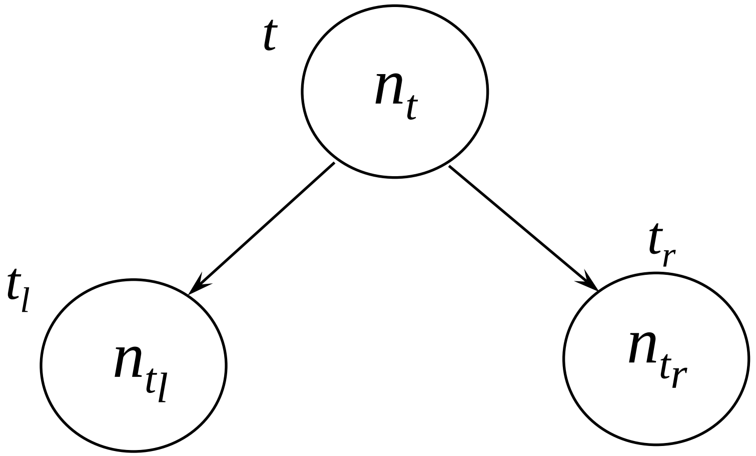
For continuous responses (regression tree):

- Variance: $$i(t)=\frac{\sum_{i\in t}(y_i-\bar{y}_t)^2}{n_t}\equiv \mathrm{var}(y)_t$$

The impurity measures how similar is a node regarding the response variable

Optimal split criterion in CART

Maximize the decrement of impurity between the parent and its children



$$\Delta i(t) = i(t) - \frac{n_{t_l}}{n_t} i(t_l) - \frac{n_{t_r}}{n_t} i(t_r)$$

How good is a node
in classification trees:

The quality of a node depends on how well separated is the response variable in that node. Compare:

t_1: 100; 90,10 — $i(t_1)=0.18$

t_2: 100; 51,49 — $i(t_2)=0.4998$

The quality is measured from the respective conditional probabilities $p(j/t)$

Every node is assigned to the maximum of $p(j/t)$

Hence, the missclassification is (=cost):

$$r(t) = 1 - Max\left(p(j/t)\right)$$

$r(t_1)=0.10$ $r(t_2)=0.49$

For the sake of interpretability, the goodness of a node is measured from the missclassification probability (not from the impurity)

node t: n_t, n_{tj}

Computing $p(j/t)$

j indicates a response class

t indicates a node

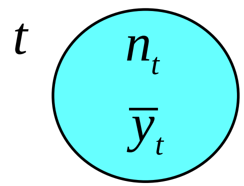
If the sample is representative:

$$if\ \pi_j = \frac{n_j}{n} \rightarrow p(j/t) = \frac{n_{tj}}{n_t}$$

In general: $$p(j/t) = \frac{p(j,t)}{p(t)} = \frac{p(j)p(t/j)}{p(t)} = \frac{\pi_j \, n_{tj}/n_j}{p(t)}$$

How good is a node in regression trees

- In a regression tree, every individuals is assigned to the mean value of the leave.



- Cost of the tree = residual variance

$$\boxed{r(t) = \frac{1}{n_t} \sum_{i \in t}^{n_t} (y_{it} - \bar{y}_t)^2}$$

Overall goodness of a tree

Cost of a single node:

$$r(t) = 1 - \max_j \, p(j/t)$$

$$r(t) = \frac{1}{n_t} \sum_{i \in t}^{n_t} (y_{it} - \overline{y}_t)^2$$

If there are missclassification costs

$$r(t) = \min_i \sum_j c(i/j)\, p(j/t)$$

c(i/j) cost of assigning to *i* an element belonging to *j*

The quality of the tree is the averaged cost of its leaves nodes

Cost of the tree:

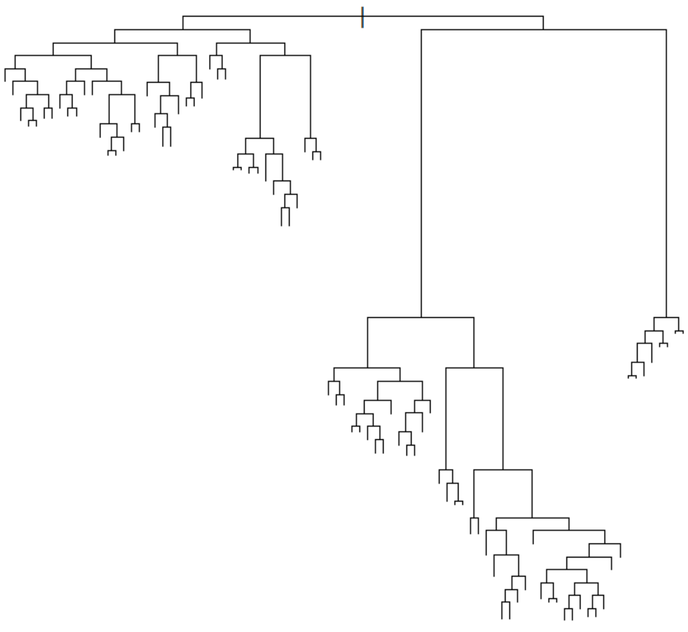
$$R(T) = \frac{\sum_{t \in \tilde{T}} p(t) r(t)}{r(root)} \times 100$$

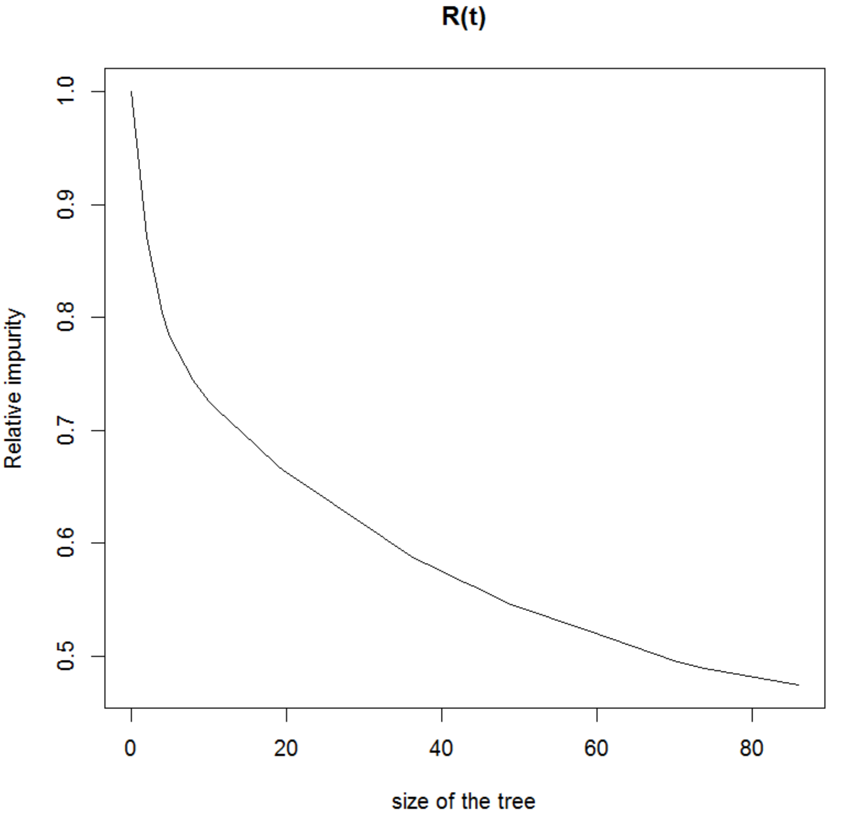
$p(t)$ Proportion of individuals in leaf t

Building "a good" tree

Criterion to optimize: $Min\ R(T)$

- Solution: build a very large tree





$R(T) = 0$ when all leaves are pure

but, is it reliable?

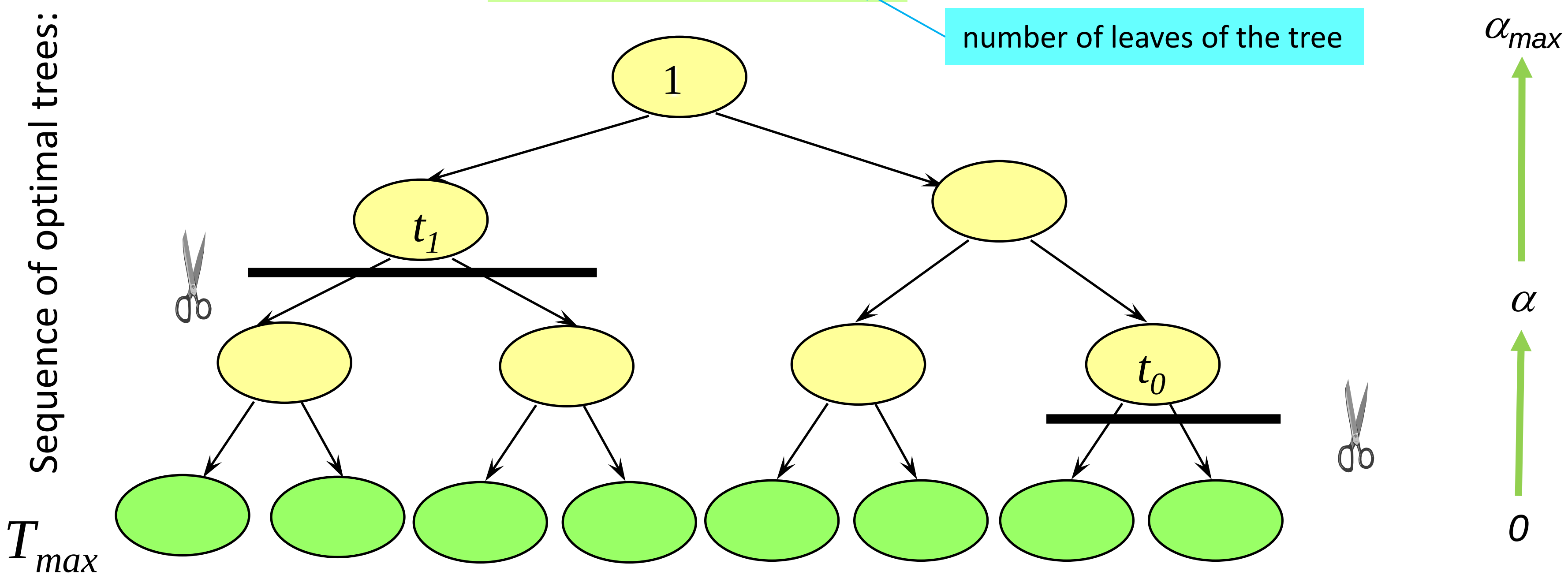
The CART solution: Prune the tree

Prune non interesting branches

Define a penalization parameter α due to the complexity (=size) of the tree

$$Min\left(R(T)+\alpha\left|T\right|\right)$$

number of leaves of the tree



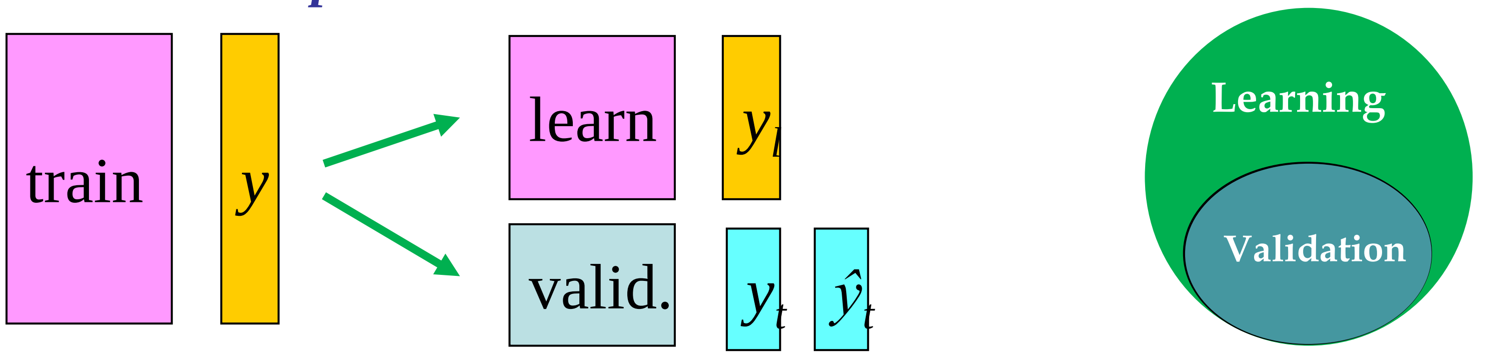
$$T_{\max}, T_{\max}-T_{t_0}, T_{\max}-T_{t_0}-T_{t_1}, \ldots, 1$$

but, which subtree to choose?

Each subtree is optimum *(min R(T)),* within the subtrees of size $|T|$

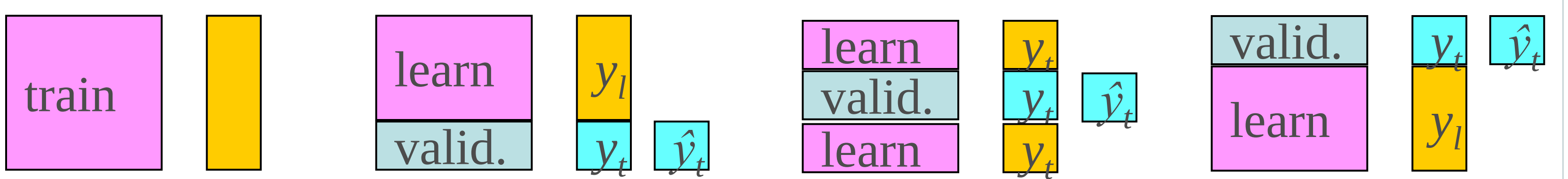
Empirical techniques for tree selection

Validation sample



With very large data sets, we just divide at random the training data in two parts, one for estimating the model, the other to validate it.

Crossvalidation



With not so large data set, we divide at random the training data in *k* (usually 10) parts at random *(ten-fold crossvalidation)*.
With small data set (not usual in a Big Data context), we divide the training data in *n* parts (each individual is a part *(Leave One Out LOO crossvalidation)*

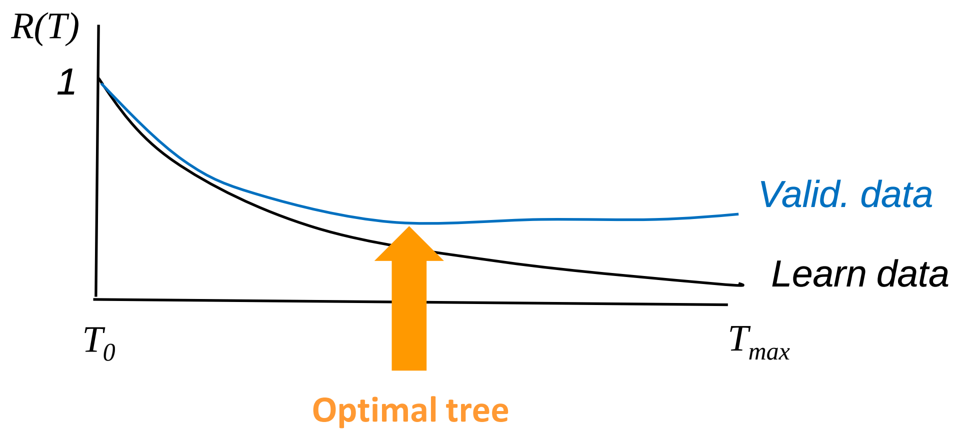
Optimal tree selection

We compute the Cost of the tree for every tree of the sequence: $T_{\max}, T_{\max} - T_{t_0}, T_{\max} - T_{t_0} - T_{t_1}, \ldots, 1$
Using the validation data (or by cross validation)

$$R^{valid}(T) = \frac{\sum_{t \in \tilde{T}} p(t) r^{valid}(t)}{r^{valid}(root)} \times 100$$

The optimal tree is the one with

$$Min\left(R^{valid}(T)\right)$$



Selection by crossvalidation

- We divide at random the data matrix in k parts (usually 10, ten-fold cross validation). Lets call n_1, n_2 , … the groups formed.
- We compute the maximal tree for the following data sets formed with n-n_1, n-n_2, … individuals.
- For a sequence of complexity parameters (α values), we compute the cost of the corresponding trees. Then, the *cv* cost is:

$$R^{cv}(T_\alpha) = \sum_{l=1}^{k} \frac{\sum_{t\in\tilde{T}_\alpha} p_l(t) r_l^{cv}(t)}{r_l^{vc}(root)} \times \frac{100}{k}$$

- **1 se rule**. To have a more prudent tree, is possible to take, instead the minimum of $R^{cv}(T)$, the tree corresponding to the minimum value of $R^{cv}(T)+1se$

$$Min\left(R^{cv}(T_\alpha) + 1 \times se(R^{cv}(T_\alpha))\right)$$

APPLICATION TO THE CREDIT SCORING PROBLEM

Application to the credit scoring problem

One of the first succesful applications of decision trees.
The goal is to produce a set of rules for automatic granting or rejecting of credits, from a database of experts decisions.

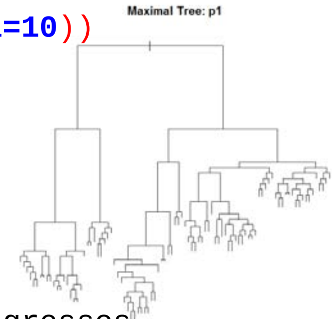
Dictamen	Positiu	Negatiu	Total	Prob_pos
Training data	2155	829	2984	72.22
Test data	1045	425	1470	71.09
Total	3200	1254	4454	

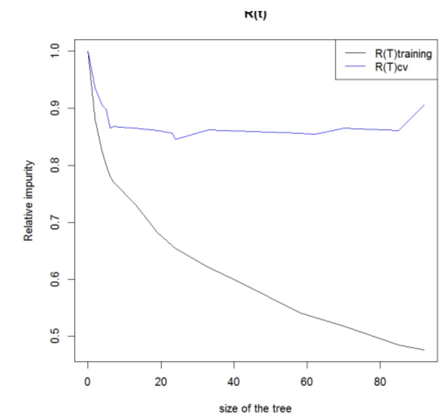
```
> names(dd)
 [1] "Dictamen"    "Antig_feina" "Vivenda"     "Plaç"        "Edat"
 [6] "Estat_civil" "Registres"   "Tipus_feina" "Despeses"    "Ingressos"
[11] "Patrimoni"   "Carrecs_pat" "Import_sol"  "Preu_finan"  "Rati_fin"
[16] "Estalvi"
```

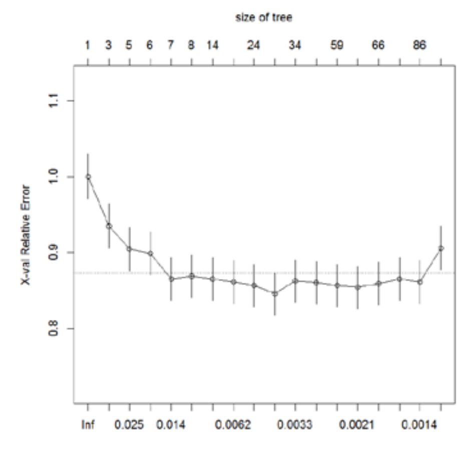
The complexity parameter table

```
> p2 = rpart(Dictamen ~ ., data=dd[learn,],control=rpart.control(cp=0.001, xval=10))

> printcp(p2)
Classification tree:
rpart(formula = Dictamen ~ ., data = dd[learn, ],
    control = rpart.control(cp = 0.001, xval = 10))

Variables actually used in tree construction:
 [1] Antig_feina Carrecs_pat Despeses    Edat        Estalvi     Import_sol  Ingressos
 [8] Patrimoni   Plaç        Preu_finan  Rati_fin    Registres   Tipus_feina Vivenda
Root node error: 829/2984 = 0.27782
n= 2984
          CP nsplit rel error  xerror     xstd
1  0.0603136      0   1.00000 1.00000 0.029515
2  0.0283474      2   0.87937 0.93486 0.028893
3  0.0229192      4   0.82268 0.90470 0.028584
4  0.0193004      5   0.79976 0.89867 0.028520
5  0.0096502      6   0.78046 0.86490 0.028153
6  0.0066345      7   0.77081 0.86852 0.028194
7  0.0064335     13   0.73100 0.86490 0.028153
8  0.0060314     19   0.68275 0.86128 0.028113
9  0.0048251     23   0.65862 0.85645 0.028059
10 0.0036188     24   0.65380 0.84560 0.027936
11 0.0030157     33   0.62123 0.86248 0.028127
12 0.0024125     41   0.59710 0.86007 0.028100
13 0.0021110     58   0.54162 0.85645 0.028059
14 0.0020105     62   0.53317 0.85404 0.028032
15 0.0019300     65   0.52714 0.85887 0.028086
16 0.0015509     70   0.51749 0.86490 0.028153
17 0.0012063     85   0.48492 0.86128 0.028113
18 0.0010000     92   0.47648 0.90591 0.028596

> plotcp(p2)
```

Optimal tree



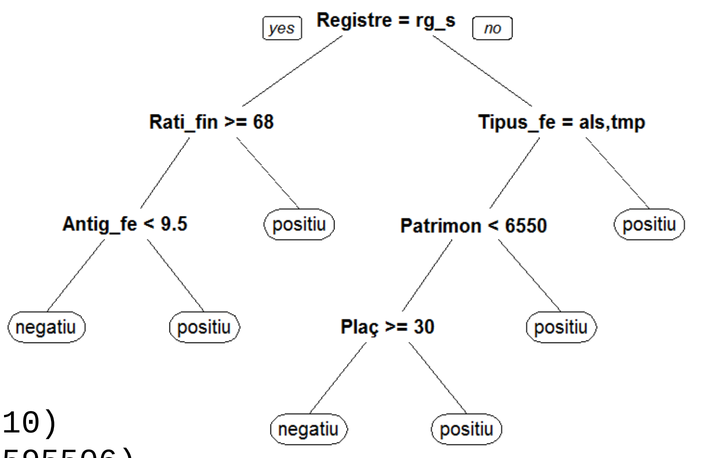




Pruning the tree

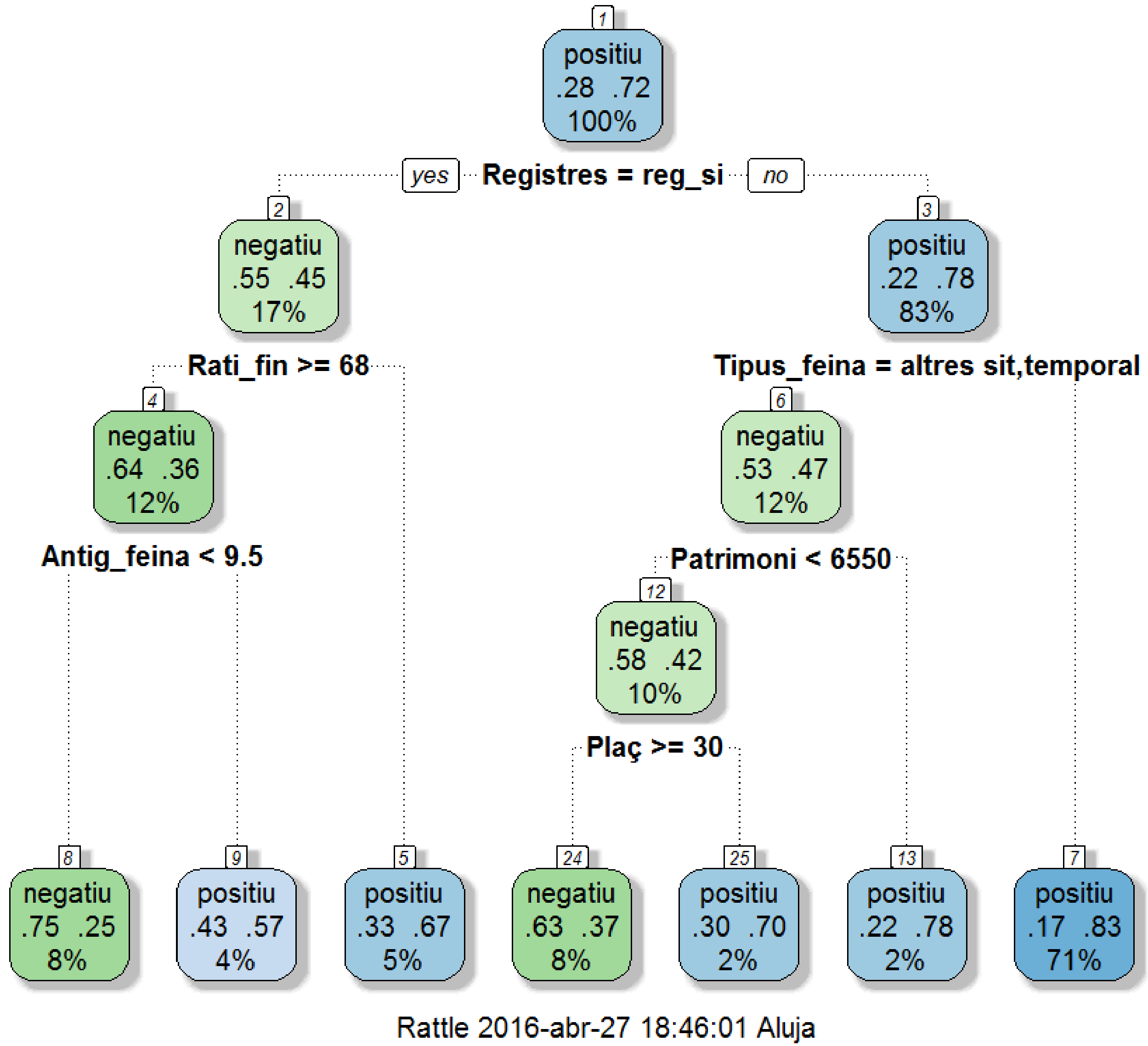
We obtain the optimal tree by pruning the maximal one up to the minimal crossvalidation error (+- 1 s.e.)

Optimal tree p2



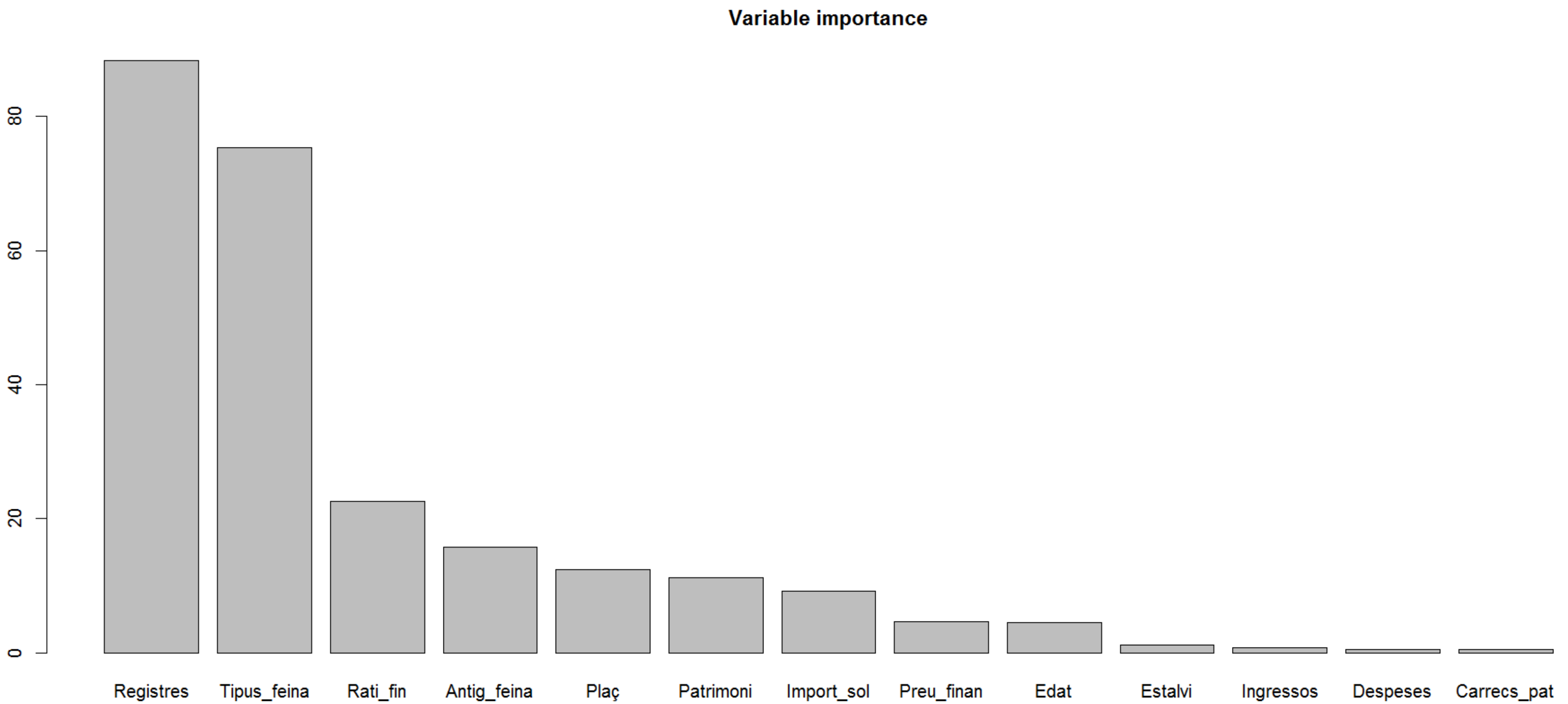

```
p2$cptable = as.data.frame(p2$cptable)
ind = which.min(p2$cptable$xerror)
xerr <- p2$cptable$xerror[ind]
xstd <- p2$cptable$xstd[ind]
i = 1
while (p2$cptable$xerror[i] > xerr+xstd) i = i+1
alfa = p2$cptable$CP[i]
# AND PRUNE THE TREE ACCORDINGLY
p1 <- prune(p2,cp=alfa)
n= 2984
node), split, n, loss, yval, (yprob)
      * denotes terminal node
 1) root 2984 829 positiu (0.2778150 0.7221850)
   2) Registres=reg_si 509 231 negatiu (0.5461690 0.4538310)
     4) Rati_fin>=68.12767 356 128 negatiu (0.6404494 0.3595506)
       8) Antig_feina< 9.5 234  59 negatiu (0.7478632 0.2521368) *
       9) Antig_feina>=9.5 122  53 positiu (0.4344262 0.5655738) *
     5) Rati_fin< 68.12767 153  50 positiu (0.3267974 0.6732026) *
   3) Registres=reg_no 2475 551 positiu (0.2226263 0.7773737)
     6) Tipus_feina=altres sit,temporal 349 165 negatiu (0.5272206 0.4727794)
      12) Patrimoni< 6550 299 126 negatiu (0.5785953 0.4214047)
        24) Plaç>=30 252  93 negatiu (0.6309524 0.3690476) *
        25) Plaç< 30 47  14 positiu (0.2978723 0.7021277) *
      13) Patrimoni>=6550 50  11 positiu (0.2200000 0.7800000) *
     7) Tipus_feina=autonom,fixe 2126 367 positiu (0.1726246 0.8273754) *

> rpart.plot(p1)
```

A fancy plot of the granting credits tree



Variable importance in defining the tree

```
> p1$variable.importance
```



the positive rules:

```
Rule number: 7 [Dictamen=positiu cover=2126 (71%) prob=0.83]
   Registres=reg_no
   Tipus_feina=autonom,fixe

 Rule number: 13 [Dictamen=positiu cover=50 (2%) prob=0.78]
   Registres=reg_no
   Tipus_feina=altres sit,temporal
   Patrimoni>=6550

 Rule number: 25 [Dictamen=positiu cover=47 (2%) prob=0.70]
   Registres=reg_no
   Tipus_feina=altres sit,temporal
   Patrimoni< 6550
   Plaç< 30

 Rule number: 5 [Dictamen=positiu cover=153 (5%) prob=0.67]
   Registres=reg_si
   Rati_fin< 68.13

 Rule number: 9 [Dictamen=positiu cover=122 (4%) prob=0.57]
   Registres=reg_si
   Rati_fin>=68.13
   Antig_feina>=9.5
```

and the negative rules:

```
Rule number: 24 [Dictamen=negatiu cover=252 (8%) prob=0.37]
   Registres=reg_no
   Tipus_feina=altres sit,temporal
   Patrimoni< 6550
   Plaç>=30

 Rule number: 8 [Dictamen=negatiu cover=234 (8%) prob=0.25]
   Registres=reg_si
   Rati_fin>=68.13
   Antig_feina< 9.5
```

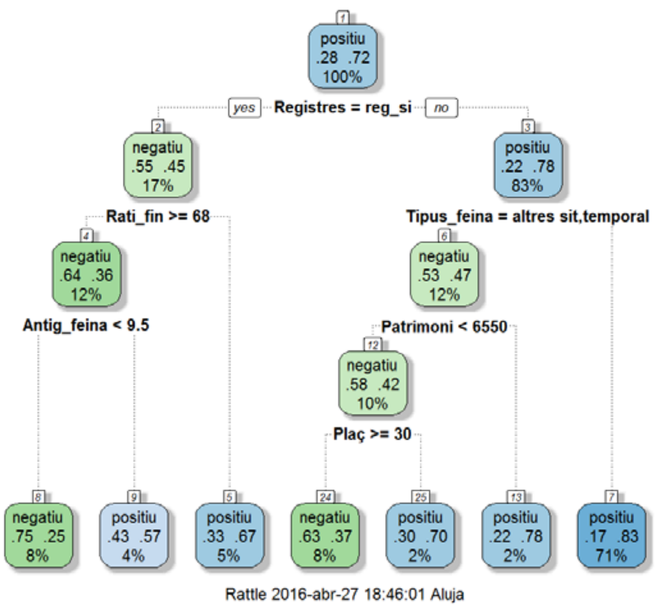
Table of Results

	n_train	class_train	n1_train	n2_train	p1_train	p2_train	probnode_train
7	2126	2	367	1759	0.1726246	0.8273754	0.71246649
13	50	2	11	39	0.2200000	0.7800000	0.01675603
25	47	2	14	33	0.2978723	0.7021277	0.01575067
5	153	2	50	103	0.3267974	0.6732026	0.05127346
9	122	2	53	69	0.4344262	0.5655738	0.04088472
24	252	1	159	93	0.6309524	0.3690476	0.08445040
8	234	1	175	59	0.7478632	0.2521368	0.07841823

VALIDATION OF THE TREE

Dropping the test individuals through the tree

Leaf number	n_train	n1_train	n2_train	p2_train	n_test	n1_test	n2_test	p2_test
7	2126	367	1759	0.8274	1022	189	833	0.8151
13	50	11	39	0.78	24	4	20	0.8333
25	47	14	33	0.7021	25	8	17	0.68
5	153	50	103	0.6732	78	35	43	0.5513
9	122	53	69	0.5656	59	21	38	0.6441
24	252	159	93	0.369	135	73	62	0.4593
8	234	175	59	0.2521	127	95	32	0.252



What threshold for prediction?

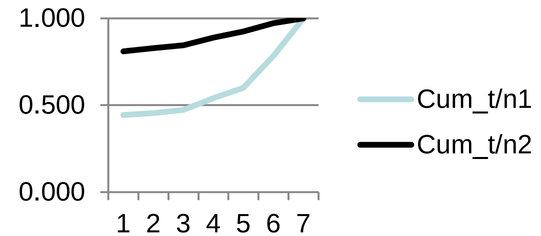
We assume that we want to predict the positives (n2)

Leaf_number	n	Prob_t	Cum_t	n1	Prob_t/n1	Cum_t/n1	n2	Prob_t/n2	Cum_t/n2	p2	Dif_cum
7	**3148**	0.707	0.707	**556**	0.443	0.443	**2592**	0.810	0.810	**0.823**	0.367
13	**74**	0.017	0.723	**15**	0.012	0.455	**59**	0.018	0.828	**0.797**	0.373
25	**72**	0.016	0.740	**22**	0.018	0.473	**50**	0.016	0.844	**0.694**	0.371
5	**231**	0.052	0.791	**85**	0.068	0.541	**146**	0.046	0.890	**0.632**	0.349
9	**181**	0.041	0.832	**74**	0.059	0.600	**107**	0.033	0.923	**0.591**	0.323
24	**387**	0.087	0.919	**232**	0.185	0.785	**155**	0.048	0.972	**0.401**	0.187
8	**361**	0.081	1.000	**270**	0.215	1.000	**91**	0.028	1.000	**0.252**	0.000
Total	**4454**			**1254**			**3200**			**0.718**	

Establishing the threshold for the decision (pos. or neg.) is not just a statistical matter:

Usually we predict as positive an outcome ≥ threshold (= 0.5) and negative otherwise

- Default: $P_{success}$ > 50%
- $P_{population}$ optimal = 79.7%
  When the difference between positives and negatives is maximal (37.3%)
- ……
- Decided by economical reasons (ROI)



Measuring the prediction error y_{test} - $\hat{y}_{test}$

In Classification trees

training sample y_l

Independent dataset

test sample y_t $\hat{y}_t$

Confusion matrix

In Test data	Predicted class YES	Predicted class NO
Real class YES	n_{TP}	n_{FN}
Real class NO	n_{FP}	n_{TN}

$$Error\ rate = \frac{n_{FN} + n_{FP}}{n}$$

$$Accuracy = \frac{n_{TP} + n_{TN}}{n}$$

$$Precision_P = \frac{n_{TP}}{n_{TP} + n_{FP}}$$

$$Recall = \frac{n_{TP}}{n_{TP} + n_{FN}}$$

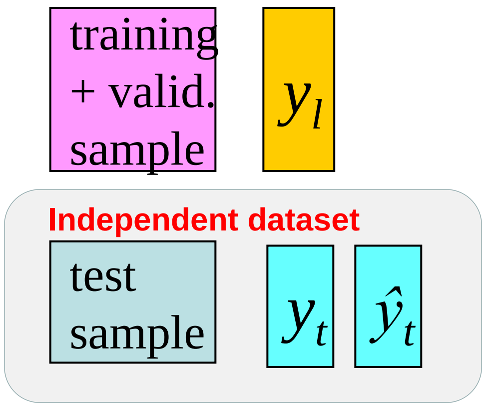
$$Precision_N = \frac{n_{TN}}{n_{FN} + n_{TN}}$$

$$F = \frac{2P_pR}{P_p + R} = \frac{2}{\frac{1}{R_p} + \frac{1}{P}}$$

$$\overline{Precision} = \frac{Prec_P + Prec_N}{2}$$

Depending on the threshold, error rate, precision and recall will vary

Measuring the prediction error y_{test} - $\hat{y}_{test}$



In Regression trees

Test sample	Actual value	Predicted value
1	y_1	$\hat{y}_1$
2	y_2	$\hat{y}_2$
:		
n_{test}	y_{ntest}	$\hat{y}_{ntest}$

$$PRESS_{test}(\hat{y}) = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} (y_i^{test} - \hat{y}_i^{test})^2$$

$$R^2_{test} = 1 - \frac{PRESS_{test}(\hat{y})}{\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} (y_i^{test} - \overline{y}^{test})^2}$$

Measuring the error in the credit scoring application

threshold: pred_pos if ≥ 0.50

Confusion table	Pred_pos	Pred_neg	*totals*
True_pos	2954	246	*3200*
True_neg	752	502	*1254*
totals	*3706*	*748*	*4454*

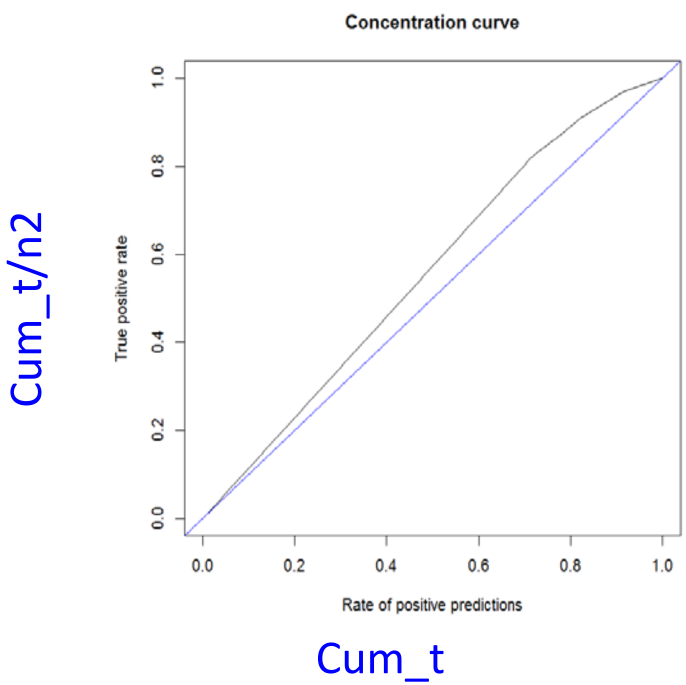
Error Rate	22.41%

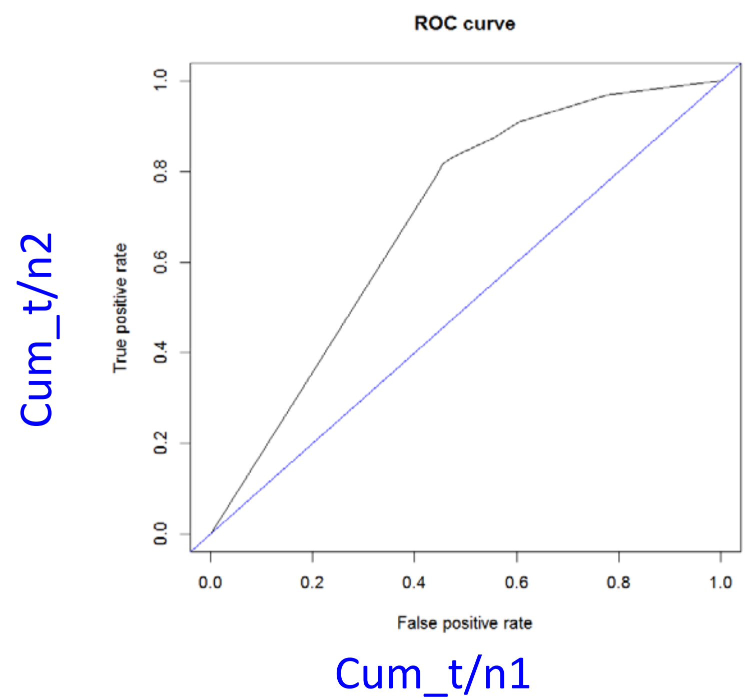
```
Baseline error.rate = 1254/4454 = 0.28
```

Precision	Pred_pos	Pred_neg	
True_pos	79.71%	32.89%	
True_neg	20.29%	67.11%	
Average			*73.41%*

Recall	Pred_pos	Pred_neg
True_pos	92.31%	
True_neg		40.03%

Visual display of the quality of a decision tree






```
library(ROCR)
pred_test = as.data.frame(predict(p1, newdata=dd[-learn,],type="prob"))
pred <- prediction(pred_test$positiu, dd$Dictamen[-learn])
con <- performance(pred,measure="tpr",x.measure="rpp")
plot(con, main="Concentration curve")
abline(0,1,col="blue")
```

```
roc <- performance(pred,"tpr","fpr")
plot(roc, main="ROC curve")
abline(0,1,col="blue")
auc = performance(pred,"auc")
auc = as.numeric(auc@y.values)
auc
[1] 0.6982347
```

HANDLING NON REPRESENTATIVES SAMPLES

(CLASSIFICATION TREES)

Handling unbalanced response classes

- In the application, the response is the actual “Dictamen” given by an expert, having 2142 “positives” and 842 “negatives” (28.22% of credit’s denial). So, the decision tree generates rules to imitate the expert.

- However, very often, response classes have very unequal frequencies
  - Credit granting: defaulting payers are 14%.
  - Attrition prediction: 97% stay, 3% attrite (in a month)
  - Medical diagnosis: 90% are healthy, 10% have disease
  - eCommerce: 99% don’t buy, 1% buy
- In those situations, decision trees tend to classify better the majority class (usually the less important for business purposes).

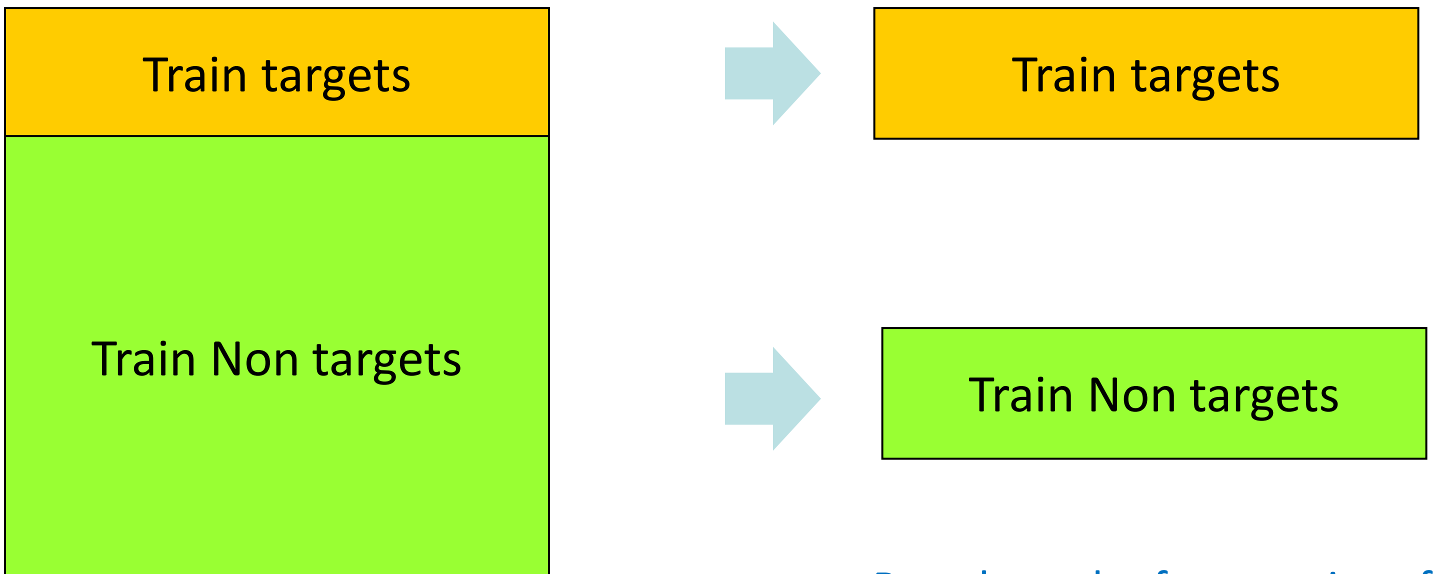
CART cost function is intended for balanced classes (it is hard to a minority class to become the majority in a node).

- → Need to balance response classes in the training data

Balancing response classes

Subsamplig the majority class to equilibrate the “yes” and the “noes” in the response variable.

You can do it several times, taking different subsamples for the “non targets” to find more stable rules



But, then, the frequencies of classes and the probabilities found in nodes, do not reflect the real ones

Taking into account the a priori knowledge

What if the response *were the defaulting rate for a credit* (which at that time was 7%), and not the experts' decision and we have managed to form a fairly balanced training sample with 2142 "good payers" and 842 "defaults" $\rightarrow$ hence the sample is not representative.

Thomas Bayes works:

"Divine Benevolence, or an Attempt to Prove That the Principal End of the Divine Providence and Government is the Happiness of His creatures" (1731)

"An Essay towards solving a Problem in the Doctrine of Chances". 1763

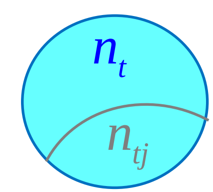
Thomas Bayes, 1701-1761

Bayes' theorem

Computes the probability of response class C_j knowing the sample composition of a node t and our a priori belief about C_j

- $\Pr(C_j)$ a priori probability of class j
- $\Pr(t/C_j)$ probability of node t knowing that the individuals belongs to C_j
- $\Pr(C_j/t)$ probability of Class j knowing that the individual has fallen in node t

node t



$$\boxed{\Pr(C_j / t) = \frac{\Pr(C_j)\Pr(t / C_j)}{\Pr(t)} = \frac{\pi_j \, {n_{tj}}/{n_j}}{\Pr(t)}}$$

$$\Pr(t / C_j) = {n_{tj}}/{n_j}$$

$$\Pr(t) = \sum_{j=1}^{J} \Pr(C_j)\Pr(t / C_j)$$

Table of results with Prob. a priori 0.93 and 0.07

threshold: pred_pos.weighted if ≥ 0.80

Leaf_number	Prob_t/n1	Prob_t/n2	Prob.w_t	n.w	Prob.w_n1/t	Prob.w_n2/t	n1.w	n2.w	p2.w	Cum_n.w	Cum_n2.w	Tot_p2.w_<t
7	0.4434	0.8100	0.7843	3493	0.0396	0.9604	138	3355	**0.9604**	3493	3355	0.9604
13	0.0120	0.0184	0.0180	80	0.0466	0.9534	4	76	**0.9534**	3574	3432	0.9603
25	0.0175	0.0156	0.0158	70	0.0779	0.9221	5	65	**0.9221**	3644	3496	0.9595
5	0.0678	0.0456	0.0472	210	0.1006	0.8994	21	189	**0.8994**	3854	3685	0.9563
9	0.0590	0.0334	0.0352	157	0.1173	0.8827	18	139	**0.8827**	4011	3824	0.9534
24	0.1850	0.0484	0.0580	258	0.2233	0.7767	58	201	**0.7767**	4269	4024	0.9427
8	0.2153	0.0284	0.0415	185	0.3630	0.6370	67	118	**0.6370**	4454	4142	0.93
Total				4454			312	4142	**0.93**			
Prob. A priori	π_{neg}= 0.07	π_{pos}= 0.93										
Weights	0.248628389	1.29444375										

$$\text{Prob.w\_t} = \pi_{neg} \times \text{Prob\_t/n1} + \pi_{pos} \times \text{Prob\_t/n2}$$
$$\text{n.w} = \text{Prob.w\_t} \times n$$

$$\text{n1.w} = \text{Prob.w\_n1} / \text{t} \times \text{n.w} = \text{weight\_1} \times \text{n1}$$
$$\text{n2.w} = \text{Prob.w\_n2} / \text{t} \times \text{n.w} = \text{weight\_2} \times \text{n2}$$
$$\text{p2.w} = \text{n2.w} / \text{n.w}$$

$$\text{weight\_1} = \frac{\pi_{neg} \times \text{n}}{\text{n1}}$$
$$\text{weight\_2} = \frac{\pi_{pos} \times \text{n}}{\text{n2}}$$

$$\text{Prob.w\_n1/t} = \frac{\pi_{neg} \times \text{Prob\_t/n1}}{\text{Prob\_t.w}}$$
$$\text{Prob.w\_n2/t} = \frac{\pi_{pos} \times \text{Prob\_t/n2}}{\text{Prob\_t.w}}$$

The problem of false negatives

Since our a priori is very strong
$$\begin{cases} \Pr(C_{pos}) = 0.93 \\ \Pr(C_{neg}) = 0.07 \end{cases}$$

Taking the threshold for positive prediction p2.w > 80%

Confusion matrix

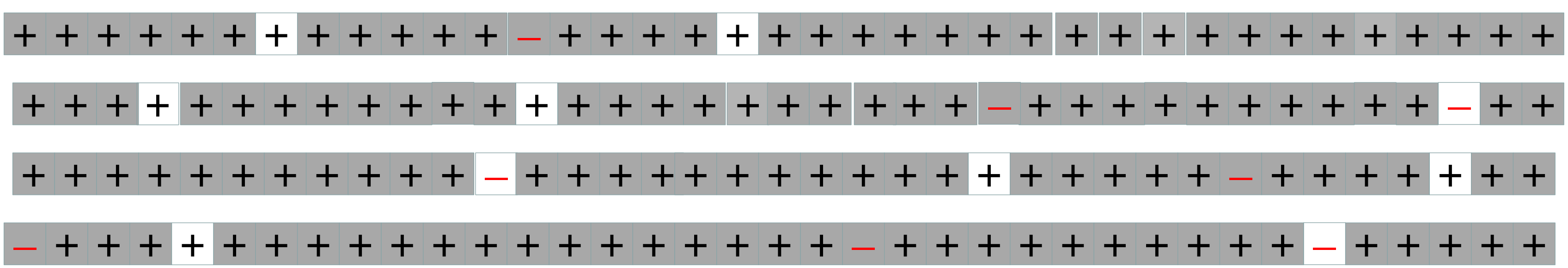
Confusion table	Pred_pos	Pred_neg	*totals*
True_pos	3824	318	*4142*
True_neg	187	125	*312*
totals	*4011*	*443*	*4454*
	90.05%	*9.95%*	

Error Rate	11.35%	Baseline error rate 7%

Precision	Pred_pos	Pred_neg	
True_pos	95.34%	71.84%	
True_neg	4.66%	28.16%	
Average			*61.75%*

Recall	Pred_pos	Pred_neg
True_pos	92.31%	
True_neg		40.03%

\+ Positive case — Negative case Positive prediction Negative prediction



A negative prediction in that case, is not that telling !

Advantages of the decision trees

- Very easy to interpret .
- The branches define the assignment rules. Results are directly operational. Automatically define "targets" for a product.
- The branches simulate fairly well the human process for taking decisions, but with improved depth.
- Define appropriate communication strategies for each "target“. Maximizing the ROI of marketing campaigns.

- Minimizing the preprocess, they can work up to certain level of error and missing data.
- Detect automatically complex interactions among variables (no need to specify them).
- Computationally efficient (scalability guaranteed for …)

- But rough approximation to the response, since it approximates the response by leaps (discontinuities) (worse prediction error than other methods)

Decision trees in R – CART

package:rpart Recursive Partitioning and Regression Trees

rpart(formula, data, weights na.action = na.rpart, method, parms, control, cost, ...)

formula: a formula, as in the 'lm' function.
data: an optional data frame in which to interpret the variables named in the formula
weights: optional case weights.
na.action: The default action deletes all observations for which 'y' is missing, but keeps those in which one or more predictors are missing.
method: one of '"anova"', '"poisson"', '"class"' or '"exp"'. If 'method' is missing then the routine tries to make an intellegent guess.
parms=list(prior=c(.85,.15), split='information')
control=rpart.control(minsplit=20, minbucket=round(minsplit/3), cp=0.01,maxcompete=4, maxsurrogate=5, usesurrogate=2, xval=10, maxdepth=30, ...)

minsplit: the minimum number of observations that must exist in a node, in order for a split to be attempted.
minbucket: the minimum number of observations in any terminal '<leaf>' node.
cp: complexity parameter.
maxcompete: the number of competitor splits retained in the output.
maxsurrogate: the number of surrogate splits retained in the output.
usesurrogate: 1= use surrogates, in order, to split subjects missing the primary variable; if all surrogates are missing the observation is not split. 2= if all surrogates are missing, then send the observation in the majority direction.
xval: number of cross-validations
maxdepth: Set the maximum depth of any node of the final tree, with the root node counted as depth 0

IMPROVING THE PRECISION: RANDOM FORESTS

Improving the precision

Performing predictions from consensus bootstrap resampling.

Bagging:

- Extract *M* bootstrap samples.
- Obtain the optimal tree for each bootstrap resample
- Predict every individual by the mean of the *M* trees if continuous response or by the majority vote if categorical response

Boosting

- Is equal to the previous, but with bootstrap resampling with probabilities not uniform, but proportional to the previous misclassification error (to force the classifier to focus on the difficult cases.

Random Forest

- The same idea as Bagging, but taking a different set of variables chosen at random in every node to decide the best split. The idea is gaining "independence" in the decision trees outcomes.

Random Forest

Breiman, 2001

Let the number of training cases be n, and the number of explanatory variables p.

Let q the number of variables to be used in a node (defaults: $q=sqrt(p)$ or $q=p/3$).

Build many (simple) decision trees (e.g., 500)

For each one, choose a bootstrap sample from the training data set. Use the rest of the cases as validation sample (out of bag OOB cases, on average 0.368).

Build a tree

In each node of the tree, randomly choose q variables to derive the best split. The tree is fully grown (not pruned).

Use the OOB cases to compute the error rate.

The final error rate is the mean of the OOB errors rates.

For prediction a new sample is pushed down the trees. It is assigned the label of the training sample in the terminal node it ends up in. This procedure is iterated over all trees in the ensemble, and the average vote of all trees is reported as random forest prediction.

```
randomForest library
p1.rf <- randomForest(Dictamen ~ ., data=dd[learn,], mtry=3, importance=TRUE,
xtest=dd[-learn,-1], ytest=dd[-learn,1], nodesize=50, maxnodes=14 )
```

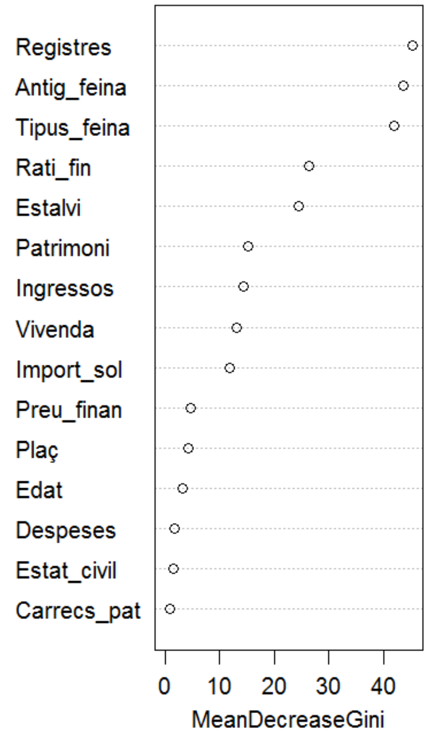
Random Forest in the credit scoring application

```
> print(p1.rf)

Call:
randomForest(formula = Dictamen ~ ., data = dd[learn, ], mtry = 3, importance = TRUE,
xtest = dd[-learn, -1], ytest = dd[-learn,1], nodesize = 50, maxnodes = 14)

               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 23.22%
Confusion matrix:
        negatiu positiu class.error
negatiu     211     618  0.74547648
positiu      75    2080  0.03480278
                Test set error rate: 22.99%
Confusion matrix:
        negatiu positiu class.error
negatiu     113     312  0.73411765
positiu      26    1019  0.02488038

> varImpPlot(p1.rf)
```

Test data:

Precision_{pos} = 76.56%

Precision_{neg} = 81.29%



References

- Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management. Gordon S. Linoff, Michael J. A. Berry. Wiley 2011.