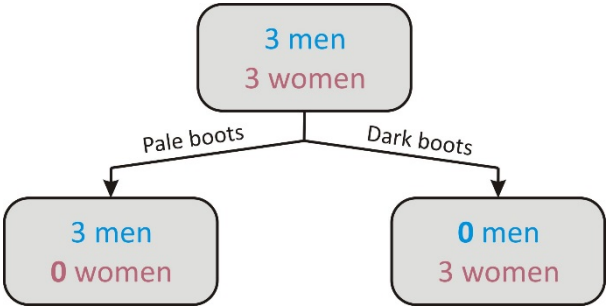
Theme 6

Regression and classification trees

Regression trees and classification trees

We have seen a classification tree before:



- Classification tree: the target is binary or categorical,
- Regression tree: the target is continuous.

Let us consider a toy example and build a tree with more levels.

Toy dataset

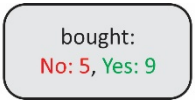
Ham & Kamber, 2012, p.338

age	income	student	credit	bought
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

Target: bought

Features: age, income, student, credit.

Step 1: First node



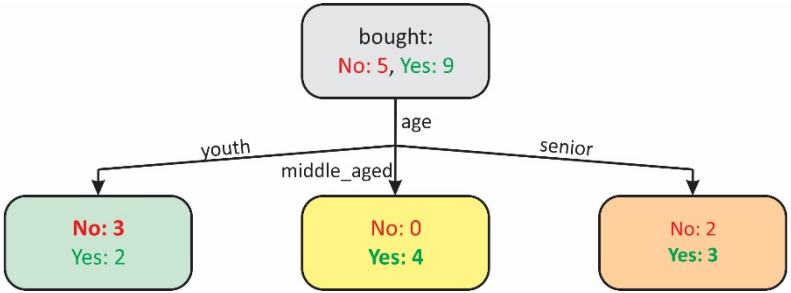
Decision rule:

bought = yes (for all)

Misclassification rate (on training): 5/14 = 36%

Step 2: First split

age	income	student	credit	bought
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no



Decision rule:

If age = youth: No

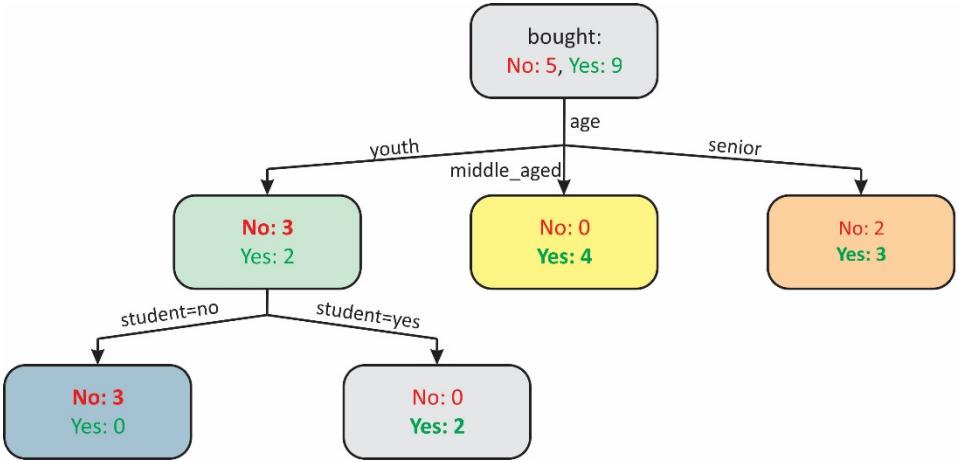
If age = middle_aged: Yes

If age = senior: Yes

Misclassification rate (on training): 4/14 = 29%

Step 3: for age=youth

age	income	student	credit	bought
youth	high	no	fair	no
youth	high	no	excellent	no
youth	medium	no	fair	no
youth	low	yes	fair	yes
youth	medium	yes	excellent	yes



Decision rule:

If age = youth and student=no: No

If age = youth and student=yes: Yes

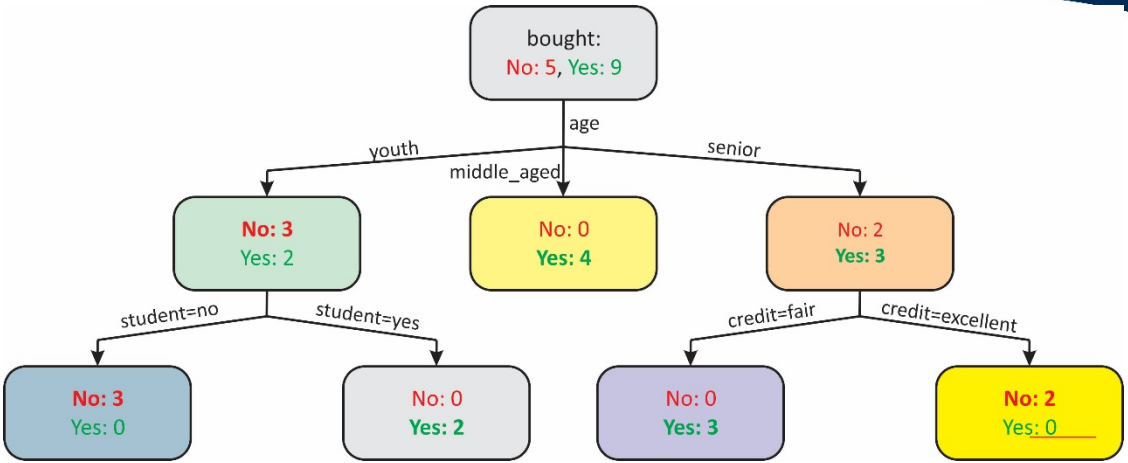
If age = middle_aged: Yes

If age = senior: Yes

Misclassification rate (on training): 2/14 = 14%

Step 4: for age=senior

age	income	student	credit	bought
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
senior	medium	yes	fair	yes
senior	medium	no	excellent	no

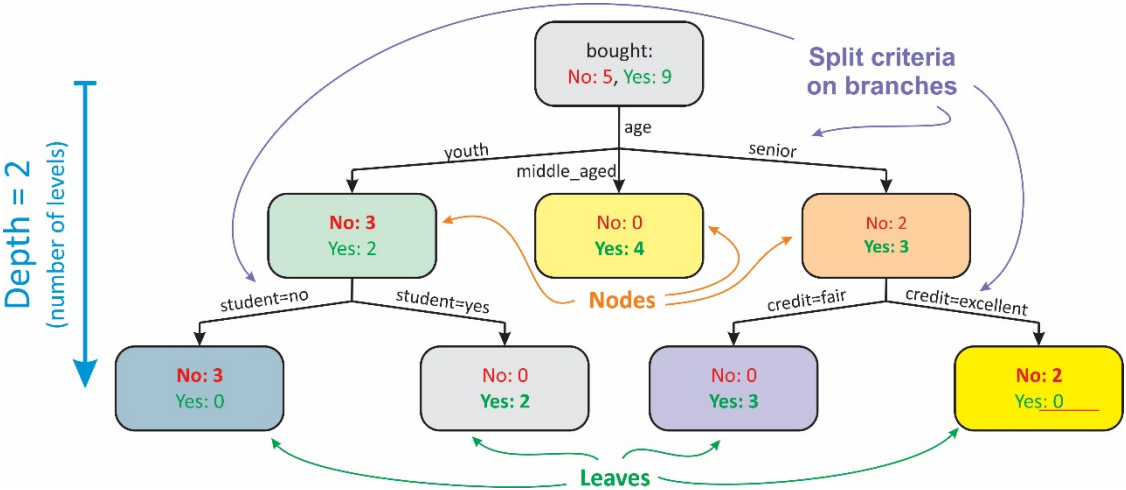


Decision rule:

If age = youth and student=no: No
If age = youth and student=yes: Yes
If age = middle_aged: Yes
If age = senior and credit=fair: Yes
If age = senior and credit=excellent: No

Misclassification rate (on training): 0/14 = 0%

Elements of a tree



The prediction may be the majority class of the leaves.

The proportion of ones(yes) in the leaves may also give a predicted probability.

Constructing the splits

Different algorithms may be used to generate the splits, including:

- CHAID (Chi-Square Automatic Interaction Detection): only for classification trees and categorical features. Tests of independence are used to combine categories of features and to decide which split is best.

  Kass, GV (1980). An Exploratory Technique for Investigating Large Quantities of Categorical Data. Applied Statistics, vol 29, no 2, pp 119-127.
- CART (Classification And Regression Trees): will be treated in more detail.

  Breiman et al., 1984
- ID3, C4.5, and C5.0: refer to different versions of an algorithm. C4.5 was #1 in the top 10 algorithms in data mining in a 2008 paper.

  Quinlan, 1986 et 1993
- GUIDE (Generalized, Unbiased, Interaction Detection, and Estimation): generalization of regression trees.

  Loh, 1997; http://www.stat.wisc.edu/~loh/guide.html
- As well as multiple variants and custom implementations.

Choosing the best split

Splits are selected based on an objective measure of diversity.

Classification trees:

- Gini index
- Entropy

Regression tree:

- Variance

Censored data:

- Log-rank test

Other special data:

- An appropriate measure of diversity…

Diversity indices for classification trees

If the target variable has K classes, and p_k is the proportion of observations in class k for $k = 1, \ldots, K$:

- $\text{Gini} = 1 - \sum_{k=1}^{K} p_k^2$
- $\text{Entropy} = -\sum_{k=1}^{K} p_k \log_2 p_k$

Those diversity indices are:

- maximized when all categories have an equal probability $1/K$
- minimized when one category has probability 1, and all others have 0.

They may be calculated on every leaf or node.

Informational gain

Let N be a node, and $I(N)$ be the diversity index of that node.

A potential split may divide N into B branches that will lead to nodes or leaves.
Let us denote them $b_1, \dots, b_B$.

$$\text{Informational gain} = I(N) - \sum_{i=1}^{B} \frac{\#(b_i)}{\#(N)} I(b_i)$$

Where # denotes the number of data in the corresponding node or leave.

In other words,

- the diversity after a split is a weighted sum of the diversity on all the branches,
- information gain measures the reduction in diversity.

The best split is the one that reduces diversity the most.

Toy data: Step by step

Let us build regression trees on the toy data with:

- binary splits,
- Gini as a measure of diversity.

For each variable, the algorithm considers all possible cuts.

We illustrate the calculations below.

age	income	student	credit	bought
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

We will note I(a,b) the information for a yes and b no.

bought:
No: 5, Yes: 9

Root node: $I(9,5) = 1 - [\ (9/14)^2 + (5/14)^2\ ] = \underline{0.459}$

Possible cuts (binary splits):

youth $I(2,3) = 1 - [(2/5)^2 + (3/5)^2] = 0.480$
middle_aged or senior : $I(7,2) = 1 - [(7/9)^2 + (2/9)^2] = 0.346$
Information gain = $0.459 - [(5/14)(0.480) + (9/14)(0.346)] =$ **0.066**

---

youth or middle_aged: $I(6,3) = 1 - [(6/9)^2 + (3/9)^2] = 0.444$
senior : $I(3,2) = 1 - [(3/5)^2 + (2/5)^2] = 0.480$
Information gain = $0.459 - [(9/14)(0.444) + (5/14)(0.480)] =$ 0.002

---

low income: $I(3,1) = 1 - [(3/4)^2 + (1/4)^2] = 0.375$

medium or high income: $I(6,4) = 1 - [(6/10)^2 + (4/10)^2] = 0.480$

Information gain = $0.459 - [(4/14)(0.375) + (10/14)(0.480)] =$ 0.009

---

low or medium income: $I(7,3) = 1 - [(7/10)^2 + (3/10)^2] = 0.420$
high income: $I(2,2) = 1 - [(2/4)^2 + (2/4)^2] = 0.5$
Information gain = $0.459 - [(10/14)(0.420) + (4/14)(0.5)] =$ **0.016**

---

student: $I(6,1) = 1 - [(6/7)^2 + (1/7)^2] = 0.245$
not a student: $I(3,4) = 1 - [(3/7)^2 + (4/7)^2] = 0.490$ **FIRST SPLIT on student**
Information gain = $0.459 - [(7/14)(0.245) + (7/14)(0.490)] =$ **0.092**

---

fair credit $I(6,2) = 1 - [(6/8)^2 + (2/8)^2] = 0.375$
excellent credit : $I(3,3) = 1 - [(3/6)^2 + (3/6)^2] = 0.5$
Information gain = $0.459 - [(8/14)(0.375) + (6/14)(0.5)] =$ **0.031**

Possible cuts (if <u>not a binary tree</u>):

youth : $I(2,3) = 1 - [(2/5)^2 + (3/5)^2] = 0.48$
middle_aged: $I(4,0) = 1 - [(4/4)^2 + (0/4)^2] = 0$ **FIRST SPLIT on age**
senior: $I(3,2) = 1 - [(3/5)^2 + (2/5)^2] = 0.48$
Information gain = $0.459 - [(5/14)(0.48) + (4/14)(0) + (5/14)(0.48)] =$ **0.116**

---

high income: $I(2,2) = 1 - [(2/4)^2 + (2/4)^2] = 0.5$
medium income: $I(4,2) = 1 - [(4/6)^2 + (2/6)^2] = 0.444$
low income: $I(3,1) = 1 - [(3/4)^2 + (1/4)^2] = 0.375$
Information gain = $0.459 - [(4/14)(0.5) + (6/14)(0.444) + (4/14)(0.375)] =$ **0.019**

---

student: $I(6,1) = 1 - [(6/7)^2 + (1/7)^2] = 0.245$
not a student: $I(3,4) = 1 - [(3/7)^2 + (4/7)^2] = 0.490$
Information gain = $0.459 - [(7/14)(0.245) + (7/14)(0.490)] =$ **0.092**

---

fair credit $I(6,2) = 1 - [(6/8)^2 + (2/8)^2] = 0.375$
excellent credit : $I(3,3) = 1 - [(3/6)^2 + (3/6)^2] = 0.5$
Information gain = $0.459 - [(8/14)(0.375) + (6/14)(0.5)] =$ **0.031**

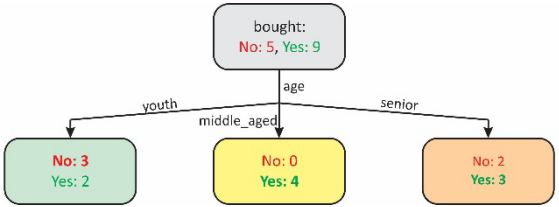
Choices of parameters and criterion will make the resulting tree change!

Our initial example had the best first split.

Best split for subsequent nodes

Consider the node "youth":

$I(2,3) = 1 - [\ (2/5)^2 + (3/5)^2\ ] = 0.48$



high income: $I(0.2) = 1 - [\ (0/2)^2 + (2/2)^2\ ] = 0$
medium income: $I(1,1) = 1 - [\ (1/2)^2 + (1/2)^2\ ] = 0.5$
low income: $I(1,0) = 1 - [\ (1/1)^2 + (0/1)^2\ ] = 0$
Information gain = $0.48 - [(2/5)(0) + (2/5)(0.5) + (1/5)(0)] =$ **0.28**

---

student: $I(2,0) = 1 - [\ (2/2)^2 + (0/2)^2\ ] = 0$
not a student : $I(0.3) = 1 - [\ (0/3)^2 + (3/3)^2\ ] = 0$
Information gain = $0.48 - [(2/5)(0) + (3/5)(0)] =$ **0.48**

SECOND SPLIT on student

---

fair credit: $I(1,2) = 1 - [\ (1/3)^2 + (2/3)^2\ ] = 0.444$
excellent credit: $I(1,1) = 1 - [\ (1/2)^2 + (1/2)^2\ ] = 0.5$
Information gain = $0.48 - [(3/5)(0.444) + (2/5)(0.5)] =$ **0.013**

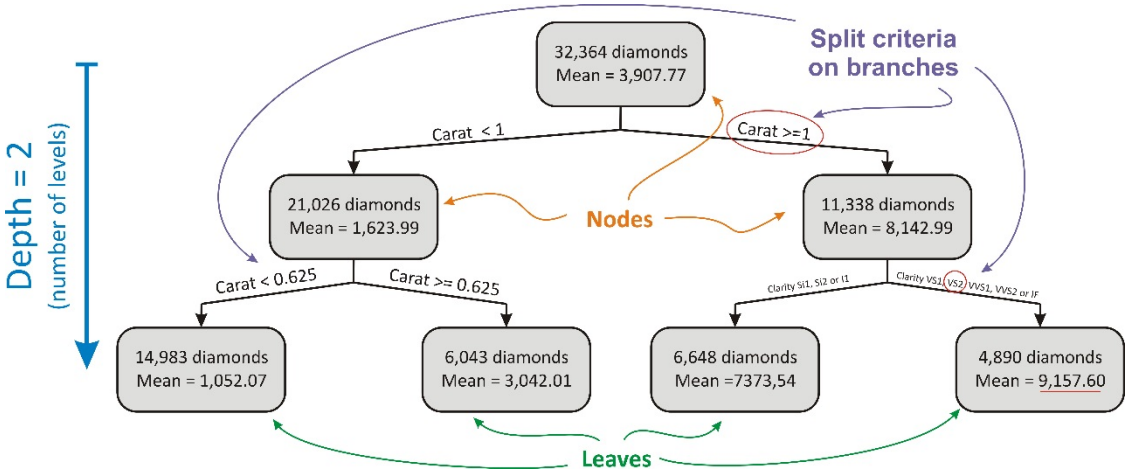
When do we stop splitting?

We can split the data until:

- The node is pure (all data belong to the same class),
- There are no more features X available to split further,
- There are too few observations in the node – a parameter is the minimum number of data that a leaf can contain,
- We reached the maximum allowed depth.

Diamond example revisited

Let us consider the diamond example again, but with a bigger dataset of 53,940 diamonds whose color and clarity may vary. A regression tree will be quite similar to a classification tree in appearance:

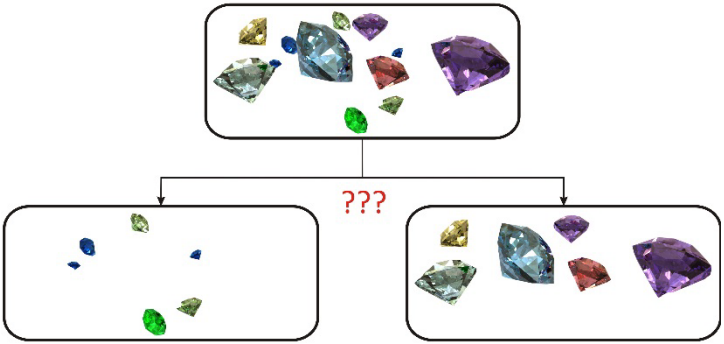


The diamond we were looking at as a weight > 1 carat and a VS2 clarity.
The predicted value is$9,157.60, the average price in his leaf.

To find the split criterion "**???**":

All variables and all
possible splits are considered:

- Carat <=.2 vs >.2
- Carat <=.21 vs >.21
- Carat <=.22 vs >.22
- Etc.



Then:

- Color I vs. the rest
- Color J vs. the rest
- Color I or J vs. the rest
- Etc.

And among all those possible cuts, the **best one** is chosen. The information gain is based on the variance of a node vs. the weighted sum of the variance of the branches considered.

Automatic treatment of missing values

Missing values are treated in the construction of a tree.

Some implementations:

- Missing values are assigned to one of the branches. All options are considered, and the "best" branch wins.

Most implementations:

- When fitting the tree, the top x best rules are recorded (instead of keeping only one)
- If a splitting rule cannot be evaluated due to a missing value, the next best rule prevails.
- If too many missing to continue, the node at which we are is used (as if it were a leaf).

You may control how many surrogate splitting rules are recorded in the model.

Tree pruning

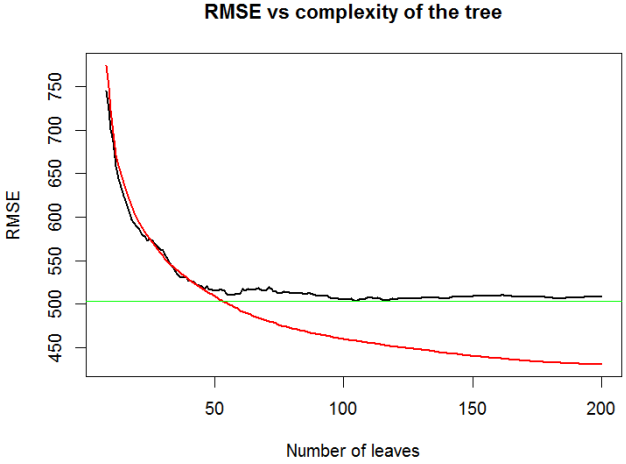
Number of leaves ≈ number of parameters in a regression

Pruning consists of determining the optimal complexity of the tree. Validation data may be used for this.

We predict the price of diamonds between 1.00 and 1.05 carats.
In the figure, we have:

- RMSE training set
- RMSE validation set
- Minimum RMSE

The ideal model has 105 leaves.
Performance levels after 50 leaves.



Pruning with penalized risk

For a given tree T, let $R(T)$ be a measure of risk (e.g., misclassification, cost, variance) and $|T|$ the number of leaves (terminal nodes). Then the penalized risk of T is

$$R_\alpha(T) = R(T) + \alpha|T|$$

Where α is a penalization parameter.

When considering all possible pruning of a (complete) tree, we denote by T_α the tree that minimizes $R_\alpha(T)$.

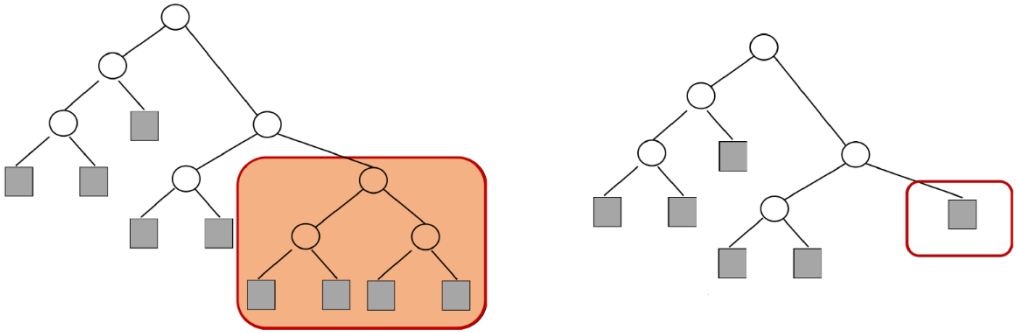
- If $\alpha = 0$, there is no penalty for the size, so the complete tree T_0 is optimal
- If $\alpha = \infty$, then no branch sprouts from the root node.
- In between, mathematical results guarantee that there is only one optimal tree for any given α.

The questions are then:

- For a given α, how can we find that optimal tree?
- How to choose α?

Pruning

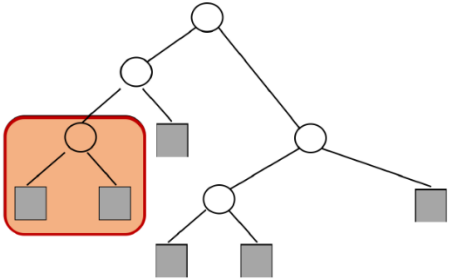
- Start with the complete tree T_0.
- Find α_1, the smallest value of α such that T_0 is not optimal anymore.
- Then there exists a tree T_1 such that $R_{\alpha_1}(T_1) < R_{\alpha_1}(T_0)$.

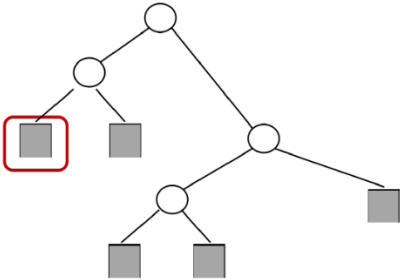


A schematic T_0 is on the left, and the pruned T_1 on the right.

We then repeat the pruning process:

- Starting from T_1 this time.
- Find α_2, the smallest value of α such that T_1 is not optimal.
- Then there exists a tree T_2 such that $R_{\alpha_2}(T_2) < R_{\alpha_2}(T_1)$.





Repeating the process yields a sequence of m imbedded trees of decreasing complexity

$$T_0 > T_1 > \cdots > T_m = T_\infty$$

As well as a corresponding sequence of penalization parameters

$$0 = \alpha_0 < \alpha_1 < \cdots < \alpha_m$$

Finding the best α

The best α may be found by evaluating each of the m trees found on a validation set.

Finding α by cross-validation is possible, and in fact, done by default in the R package `rpart`. If we have a sequence of values for α, we can just apply what we know. The challenge is to first determine a good range for these values.

- The whole training set is used to find $\alpha_1 < \cdots < \alpha_m$.
- Those values are used to determine a sequence of potential penalization values based on the geometric mean of the α:
    - $\beta_0 = 0$
    - $\beta_1 = \sqrt{\alpha_1 \alpha_2}$
    - $\beta_2 = \sqrt{\alpha_2 \alpha_3}$
    - $\vdots$
    - $\beta_m = \infty$
- We now have $m + 1$ penalization values that can be evaluated by cross-validation
- Each value β yields a measure of performance, which allows us to determine the winning penalization parameter.

Variable importance

The relative importance of each variable may also be seen. The unscaled value corresponds to the total information gain provided by each variable.

	[,1]	[,2]
clarte	9864107879	1.00000000
couleur	2715418125	0.27528269
coupe	356059523	0.03609647
y	330152329	0.03347006
profondeur	252743236	0.02562251
x	196539665	0.01992473
table	178398977	0.01808567
z	153505820	0.01556206

- The importance (variance reduction here) gets rescaled.
- The best variable always gets 1.
- Subsequent variables have a relative importance with respect to the most important variable: Color explains 27.53% of the amount of variance that clarity explains.

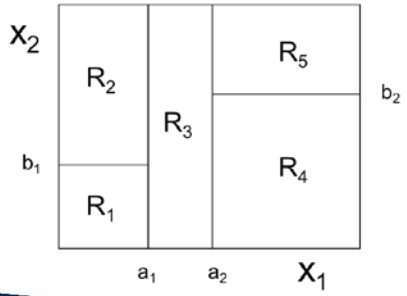
Variable importance is useful to interpret the results, but it is not an inference tool. If you need to prove a significant link between some feature and your target variable, use the appropriate inference tools.

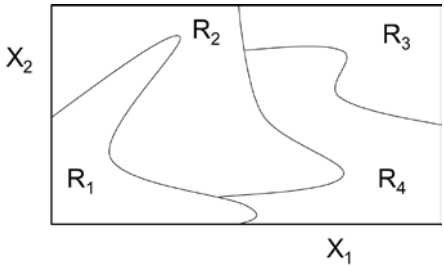
Strengths of trees:

- Clear and easy interpretation; easy to explain to untrained people,
- Seamless treatment of missing values,
- No need to transform the variables,
- Takes account of interactions implicitly,
- Robust to extreme values.

Weaknesses of tress:

- Do not provide the best performances
- Better when the target Y is similar on "hypercubes" in the space of the predictors X.





Creating new variables

To break from the hypercube constraint, one option is to add new variables.

For instance, with features X_1 and X_2 available, we could create:

- $X_1 - X_2$
- X_1/X_2
- X_1/X_2^2
- etc.

and the criteria that would be considered to split on those new variables yields shapes that are not hypercubes.

Note that creating variables for a tree does not seem to be a very common practice.