# Machine Learning
# Trees, Bagging and Random forests

Audrey Poterie
*Univ. Bretagne Sud – LMBA Vannes .*

October 15, 2021

# Outlines

**Objectives:**

- Know and understand the principle of decision trees (specially CART) and model averaging (especially bagging and random forests).
- Be able both to apply properly CART, bagging and random forests on a dataset and to understand the results.

**Course content**



**Organization**

- Volume: 6h, 2 sessions ($2 \times 1.5$h lessons $+ 2 \times 1.5$h practicals).
- Assessment: a written test (common for the entire Machine Learning module) and lab sessions.

# Content

# General context

- More and more complex (available) data
  - Complicated and poorly understood phenomena that we want to explain.
  - Various fields: Medicine, Biology, Physics, Finance, etc... all fields with available data!

↪ Use of statistical learning approaches.

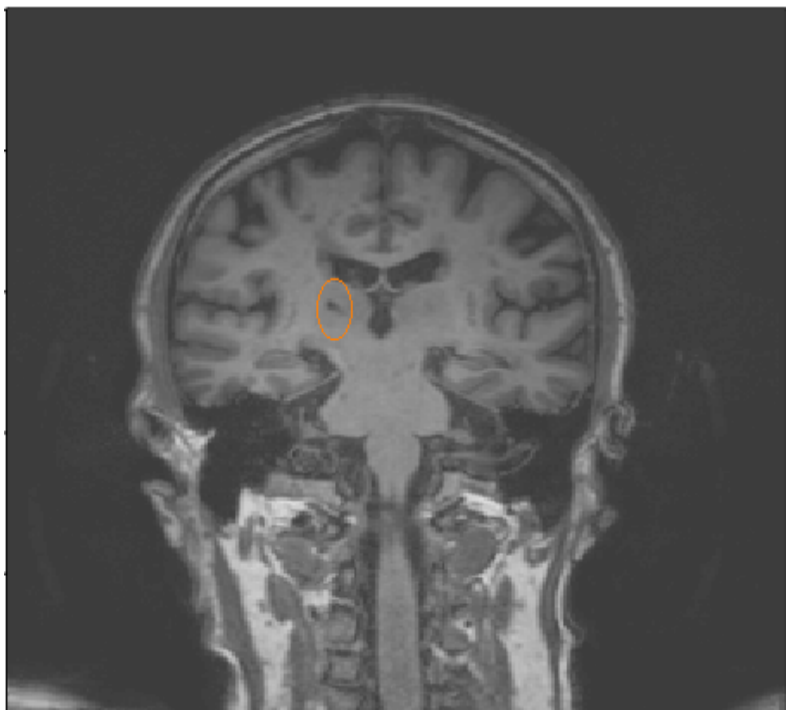
**What is statistical learning ?**

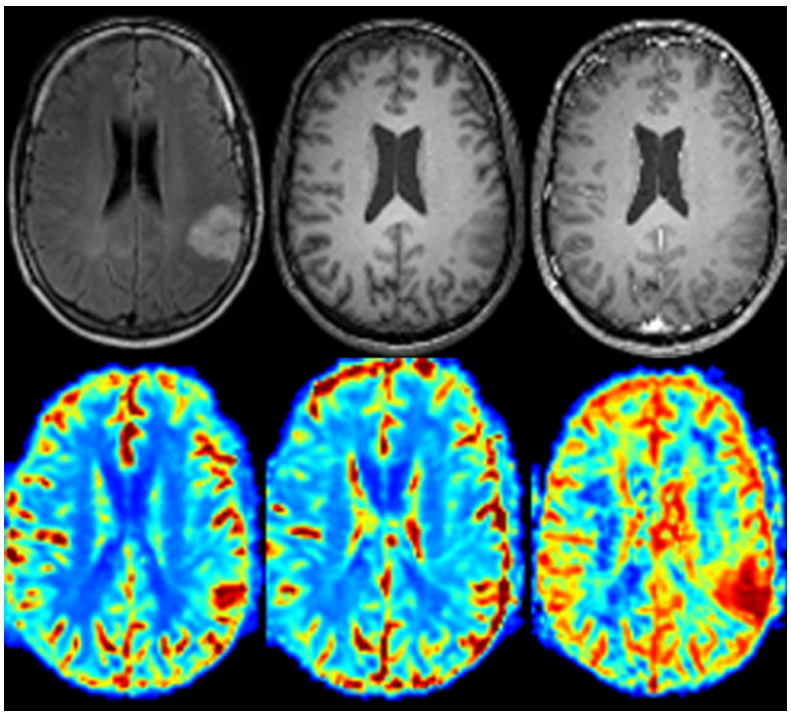
➢ **To characterize and to predict** phenomena described by many variables based on observed data.

➢ "*...to extract important patterns and trends and to* **understand what the data says**". [Friedman et al., 2008]

# Example 1: Automated detection of brain lacunes

**How to identify automatically brain lacunes based on MRI data?**





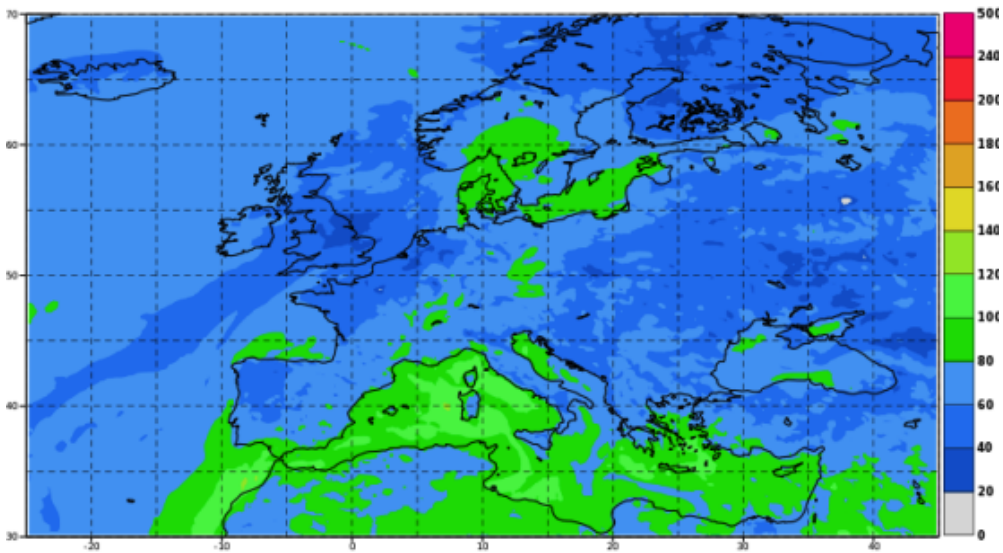
*Project lead by the Centre for Healthy Brain Ageing (CHeBA), Sydney.*

# Example 2 : Improving MetoFrance's forcast of the ozone level

## How to forecast the ozone level for the following days?

An available dataset:

- 832 measures from 2002 to 2005 in France.
- 8 weather parameters : MOCAGE prediction, temperature, humidity, NO2 and NO concentration, location, wind speed and wind direction.



**Ozone forecast on the 17th of June 2016 for the 21th of June 2016.**

*Source: http://www.meteofrance.fr/actualites/*

# Mathematical context and terminology

- **Goal**: explain/predict an output $y \in \mathcal{Y}$ from inputs $\mathbf{x} \in \mathcal{X}$.
- **Available data**: $d_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ where the $y_i \in \mathcal{Y}$ denote the outputs and the $\mathbf{x}_i \in \mathcal{X}$ denote the inputs.

# Mathematical context and terminology

- **Goal**: explain/predict an output $y \in \mathcal{Y}$ from inputs $\mathbf{x} \in \mathcal{X}$.
- **Available data**: $d_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ where the $y_i \in \mathcal{Y}$ denote the outputs and the $\mathbf{x}_i \in \mathcal{X}$ denote the inputs.

**Terminology**

- $y$: the response, the output, the target.
- $\mathbf{x} = (x_1, \ldots, x_p)$: the inputs, a vector of $p$ predictors/features.
  ➜ Use of bold print to emphasize that $\mathbf{x}$ is a vector.
- $x_1, \ldots, x_j, \ldots, x_p$: the components of vector $\mathbf{x}$, the $p$ predictors.
  ➜ Components of $\mathbf{x}$ can be continuous, categorical, curve, image, etc.

See for example the book [Hastie et al., 2009].

# Mathematical context and terminology

**Two problems**

- If $y$ is continuous, so $y$ takes values in a real interval $\mathcal{Y} \subseteq \mathbb{R}$.
  ➜ It is a regression problem.
- If $y$ is categorical, so $y$ takes values in a set of labels $\mathcal{Y} = \{1, \ldots, K\}$.
  ➜ It is a classification problem.

# Statistical framework

Statistical assumptions:

- One observation = one random variable $(\mathbf{X}, Y)$.
- $Y \in \mathcal{Y}$ : a random variable, the response, the target.
- $\mathbf{X} = (X_1, \ldots, X_d) \in \mathcal{X}$: a random vector, the predictors, the features.
- $\mathbb{P}$: the probability distribution of $(\mathbf{X}, Y)$.
- $\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$: a sample with $n$ indepedent copies of $(\mathbf{X}, Y)$ where the inputs of the ith observation is $\mathbf{X}_i = (X_{i,1}, \ldots, X_{i,p})$.

*Convention: uppercase for random variables and lowercase for realisations, observed data.*

# Statistical framework

Statistical assumptions:

- One observation = one random variable $(\mathbf{X}, Y)$.
- $Y \in \mathcal{Y}$ : a random variable, the response, the target.
- $\mathbf{X} = (X_1, \ldots, X_d) \in \mathcal{X}$: a random vector, the predictors, the features.
- $\mathbb{P}$: the probability distribution of $(\mathbf{X}, Y)$.
- $\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$: a sample with $n$ indepedent copies of $(\mathbf{X}, Y)$ where the inputs of the ith observation is $\mathbf{X}_i = (X_{i,1}, \ldots, X_{i,p})$.

**Goal**

**Explain/predict** a variable $Y$ based on $\mathbf{X}$.

/!\ Problem: $\mathbb{P}$ unknown.

*Convention: uppercase for random variables and lowercase for realisations, observed data.*

# Statistical framework

**Statistician's strategy**

To learn the relationship between $\mathbf{X}$ and $Y$, statisticians use a sample of observed data

$$d_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\},$$

where $(\mathbf{x}_i, y_i)$ denotes a realisation of $(\mathbf{X}_i, Y_i)$.

# Statistical framework

**Statistical problem**

Find a function $f : \mathcal{X} \to \mathcal{Y}$ such as

$$f(\mathbf{x}_i) \approx y_i,\ i = 1, \ldots, n,$$

where $f(x_i)$ denotes the prediction and $y_i$ the observation (the true value).

# Statistical framework

**Statistical problem**

Find a function $f : \mathcal{X} \to \mathcal{Y}$ such as

$$f(\mathbf{x}_i) \approx y_i, \; i = 1, \ldots, n,$$

where $f(x_i)$ denotes the prediction and $y_i$ the observation (the true value).

➜ We want a *good* function = function that is performant with respect to a criterion.

➜ We have to choose a performance criterion to measure the loss (the error) between one prediction $f(x)$ and one observation $y$.

**Performance measure of $f$**

The performance of $f : \mathcal{X} \to \mathcal{Y}$ is measured by its risk $\mathcal{R}(f)$ defined as

$$\mathcal{R}(f) = \mathbf{E}_{\mathbb{P}}[\ell(Y, f(\mathbf{X}))]$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is a loss function (error) such that

$$\begin{cases} \ell(y, y') & = & 0 \text{ if } y = y' \\ \ell(y, y') & > & 0 \text{ if } y \neq y' \end{cases}$$

**Statistical problem: theory**

For a given loss function $\ell$, we would like to find the function $f^\star$ that solves

$$f^\star = \operatorname*{argmin}_f \mathcal{R}(f). \tag{1}$$

$f^\star$ is called the optimal function that is unknown.

In practice, it is very difficult to compute the risk $\mathcal{R}(f)$ (so problem (1) too).
➜ Find a good proxy of $f^\star$ and $\mathcal{R}(f)$.

To approximate the risk $\mathcal{R}(f)$, we use the empirical risk (=computed on available data) defined as

$$\mathcal{R}_{emp}(f, \mathcal{T}) = \frac{1}{n_{\mathcal{T}}} \sum_{i=1}^{n_{\mathcal{T}}} \ell(y_i, f(\mathbf{x}_i)),$$

where $\mathcal{T}_n$ is a test set (independent from the train set $\mathcal{D}_n$) with size $n_{\mathcal{T}}$.

/!\ Reminder: the empirical risk evaluated on the train set $\mathcal{D}_n$ is not a good proxy (too optimistic).

➜ The empirical risk must be evaluated on a test set or by using cross validation, leave-one-out, etc.

# Statistical framework, reminder

| | **Regression task**<br><br>($y$ continuous) | **Classification task**<br><br>($y$ categorical) |
|---|---|---|
| **Loss function**<br><br>$\ell$ | The quadratic loss<br><br>$\ell(y, f(\mathbf{x})) = (y - f(x))^2$ | The misclassification error<br><br>$\ell(y, f(\mathbf{x})) = \mathbb{1}_{y \neq f(\mathbf{x})}$ |
| **Empirical risk**<br><br>$\mathcal{R}_{emp}(f, \mathcal{T})$ | $\mathcal{R}_{emp}(f, \mathcal{T}) = \frac{1}{n_{\mathcal{T}}} \sum_{i=1}^{n_{\mathcal{T}}} (y_i - f(\mathbf{x}_i))^2$ | $\mathcal{R}_{emp}(f, \mathcal{T}) = \frac{1}{n_{\mathcal{T}}} \sum_{i=1}^{n_{\mathcal{T}}} \mathbb{1}_{y_i \neq f(\mathbf{x}_i)}$ |
| **Optimal function**<br><br>$f^\star$ | The regression estimate<br><br>$f^\star(\mathbf{x}) = \mathbf{E}_{\mathbb{P}}[Y \vert \mathbf{X} = \mathbf{x}]$ | The Bayes rule[1]<br><br>$f^\star(\mathbf{x}) = \underset{l \in \{1,\dots,K\}}{\operatorname{argmax}} \mathbb{P}[Y = l \vert \mathbf{X} = \mathbf{x}]$ |

[1] The most likely label, the label with the highest probability.

Lots of approaches developped in supervised learning :

- linear regression,
- logistic regression,
- decision trees, random forests
- k-NN,
- etc.

# Part I : Decision trees

# Content

# What are decision trees ?

- Supervised learning methods.
- Popular methods that have been applied in many fields such as, for example, electrical engineering, biology, medical research and financial topics.
- Methods with a wide applicability: they can be applied to a wide range of data.
  ➜ Trees can handle both continuous and categorical inputs.
  ➜ Methods generally can deal with large $n$ and/or large $p$ (large datasets).

# What are decision trees ?

- Simple methods: easy to implement and understand.
- Methods that can be easily interpretated thanks to visualization tools: trees can be displayed graphically.
- Existence of multiple decision tree algorithms, such for instance: CART, CHAID, C4.5, CARTGV., etc.
  $\hookrightarrow$ See for example the handbook for a revieuw of the tree-based methods [Rokach and Maimon, 2005]).

➜ **Focus on the CART algorithm** [Breiman et al., 1984] which is one of the most widely used tree algorithms.

# CART

- **CART** for **C**lassification **A**nd **R**egression **T**ree.
  ➜ An algorithm in supervised learning for both regression and classification problems

- Method introduced by Leo Breiman (1984).

- Very famous algorithm : part of the top 10 data mining algorithms indentifed by the IEEE International Conference on Data Mining (ICDM) in December 2006 [Wu et al., 2008]

- Non-parametric method: no assumption required for the couple $(\mathbf{X}, Y)$.
  ➜ No need to check some "validity" conditions/hypothesis such as :
  - Linear relationship between $\mathbf{X}$ and $Y$ ?
  - Correlation or collinearity between predictors ?
  - Gaussian distribution for $\mathbf{X}$ and/or $Y$ ?
  - *etc.*

# Statistical framework

Notations and context: the "classical" supervised learning framework with

- $Y \in \mathcal{Y}$ = the response variable, the output.
- $\mathbf{X} = (X_1, \ldots, X_p) \in \mathcal{X} \subseteq \mathbb{R}^p$ = the vector of $p$ predictors, the inputs.
- $Y$ can be a real ($\mathcal{Y} \subseteq \mathbb{R}$) or a label ($\mathcal{Y} = \{1, \ldots, K\}, K \in \mathbb{N}^+$).
- $X_1, \ldots, X_p$ can be continuous, categorical or both.
- $p$ could be large.

Problem: explain/predict the response $Y \in \mathcal{Y}$ by the vector $\mathbf{X} \in \mathcal{X}$ (the objective in supervised learning).

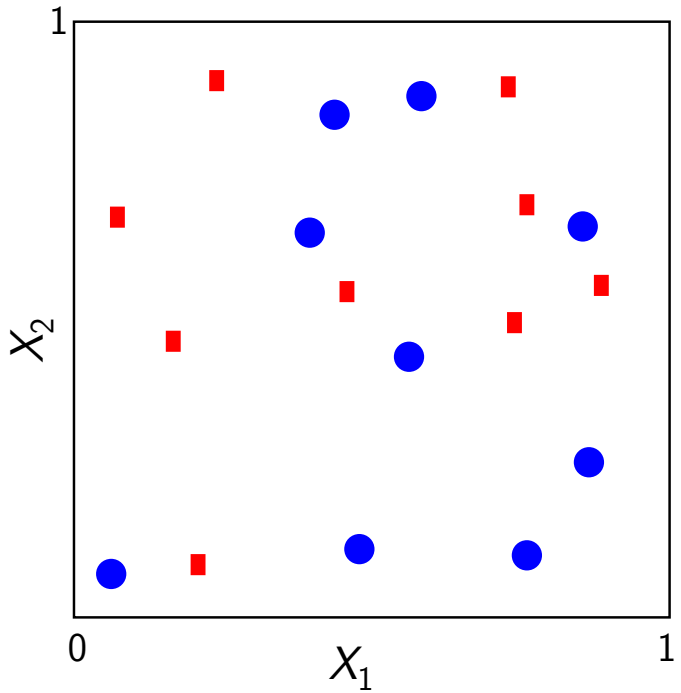
**CART principle**

Build prediction rules by means of recursive and binary splits of the predictor space $\mathcal{X}$.

➜ Let's start with a toy example: a simple problem in binary classification.

# CART: a toy example in classification

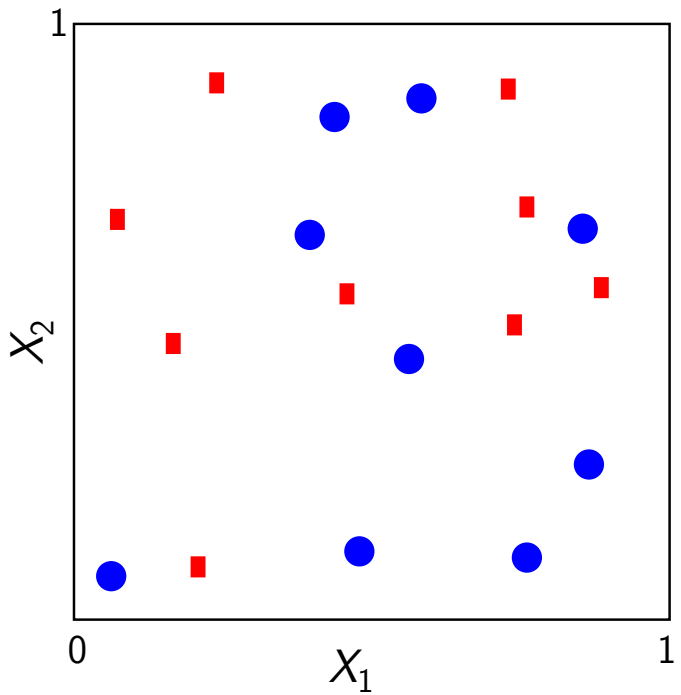
Data: a sample $\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \dots (\mathbf{X}_n, Y_n)\}$ with $\mathbf{X}_i \in [0,1]^2$ and $Y_i \in \{$ ■, ● $\}$.




**Objective: well discriminate blues dots and red squares.**

# CART: a toy example in classification

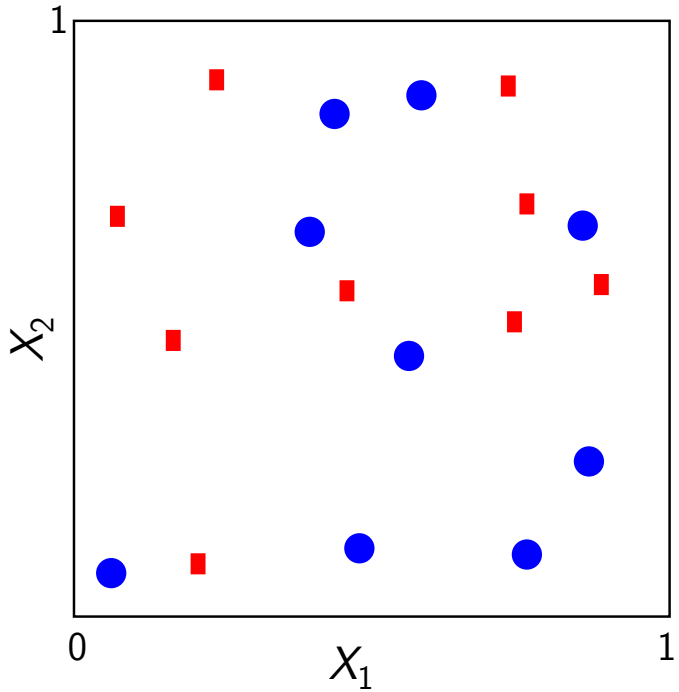
Data: a sample $\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \dots (\mathbf{X}_n, Y_n)\}$ with $\mathbf{X}_i \in [0,1]^2$ and $Y_i \in \{$ ■, ● $\}$.




**CART's objective**

Build a partition of the feature space that discriminates "*at best*" the blue dots and the red squares.

# CART: a toy example in classification

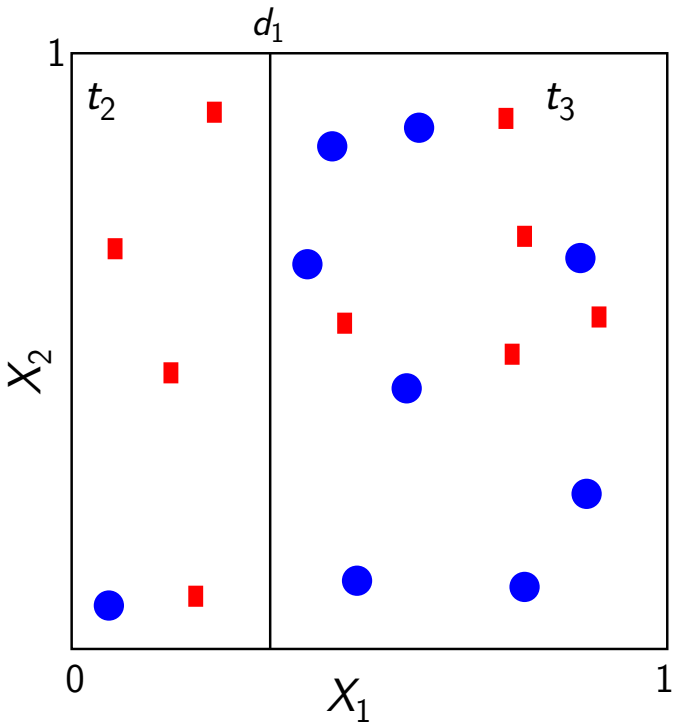


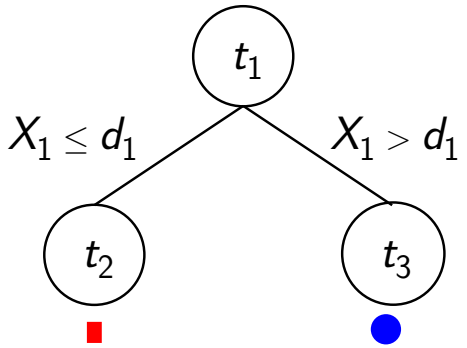

How to build a CART partition ?

**Use recursive and binary splits of the feature space.**

- Splits parallel to the axes.
- A split = choice of a **split variable** and **a split threshold**.

# CART: a toy example in classification

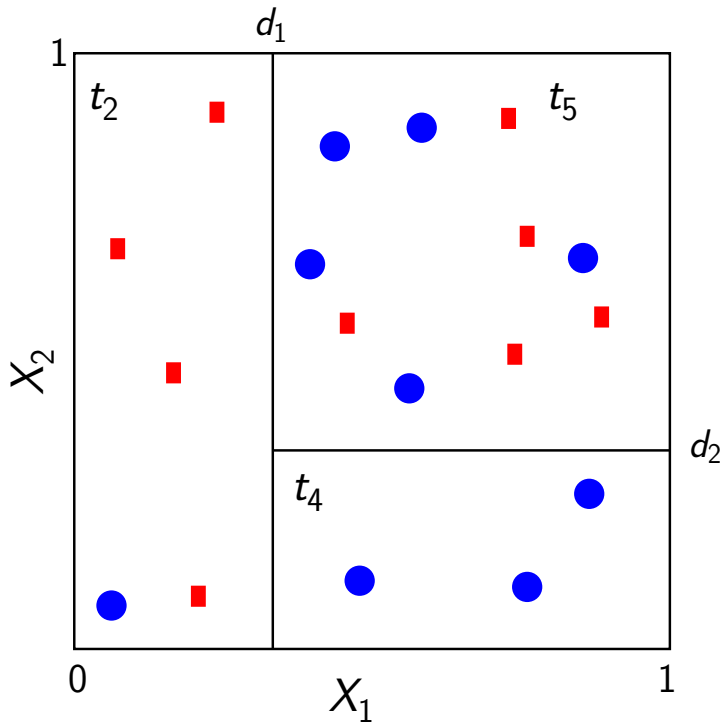


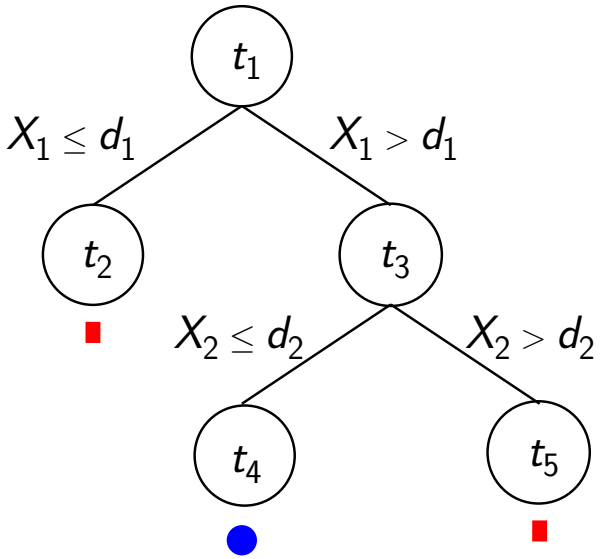




**How to build a CART partition ?**

- At each step, the method splits the data into two regions (or nodes) according to a **split variable** and a **split threshold**.
- The partition can be represented by means of a tree.

# CART: a toy example in classification

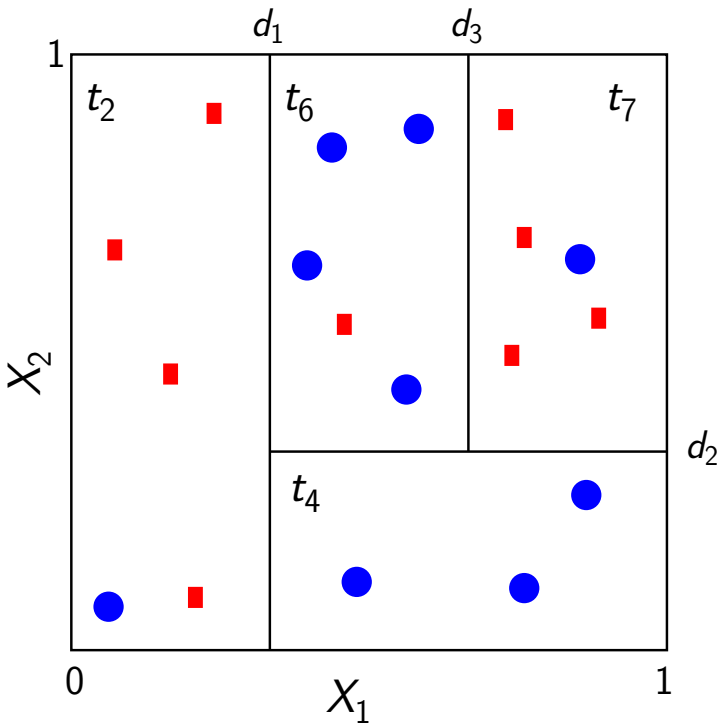


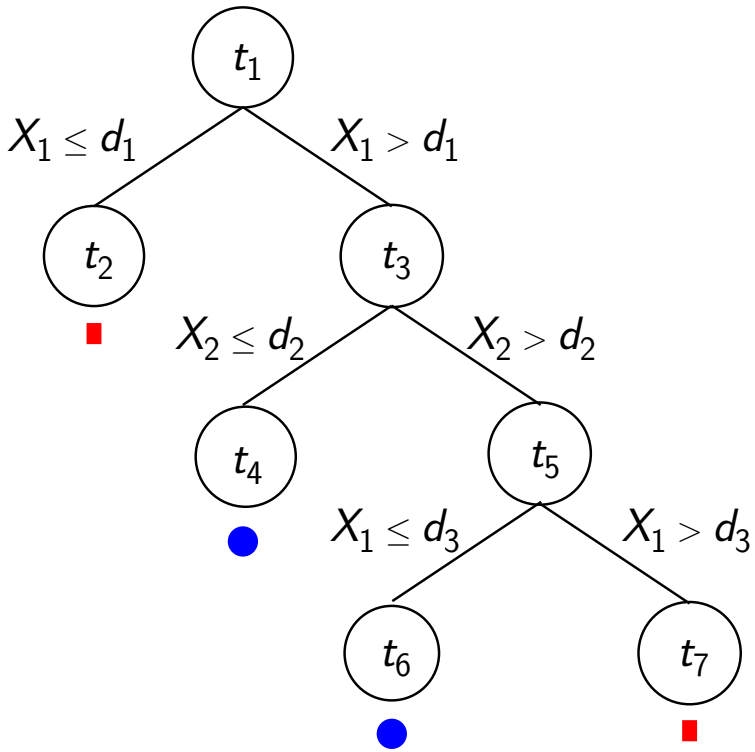




**How to build a CART partition ?**

- At each step, the method splits the data into two regions (or nodes) according to a **split variable** and a **split threshold**.
- The partition can be represented by means of a tree.

# CART: a toy example in classification

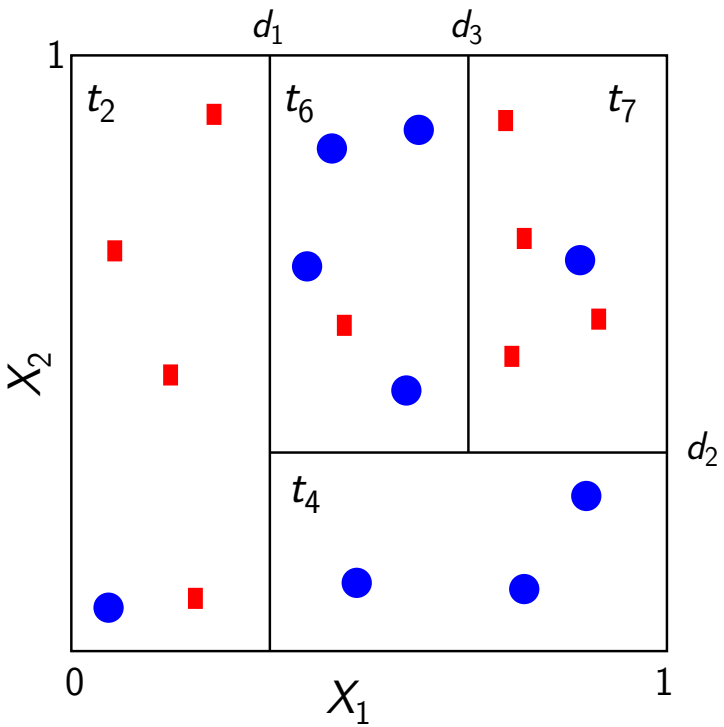


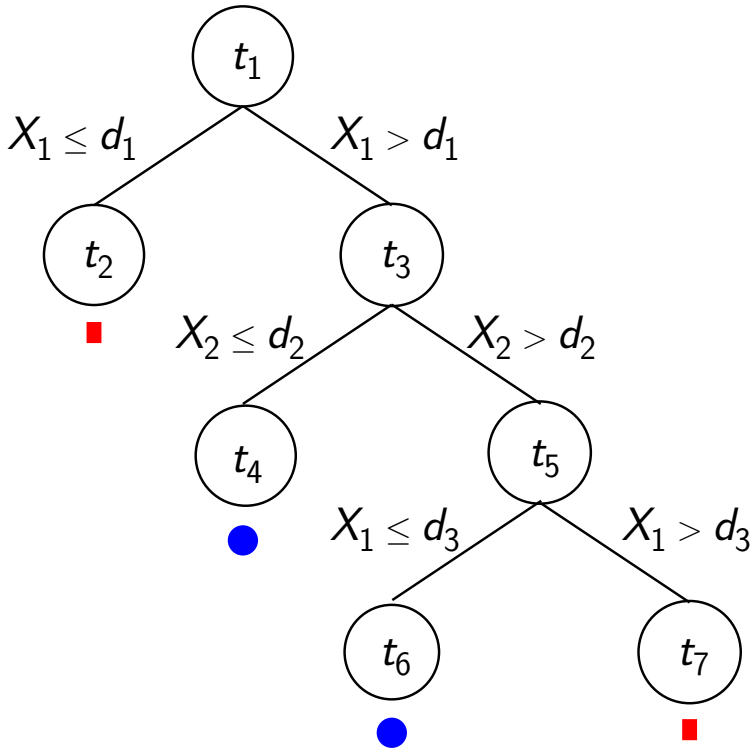




**How to build a CART partition ?**

- At each step, the method splits the data into two regions (or nodes) according to a **split variable** and a **split threshold**.
- The partition can be represented by means of a tree.

# CART: a toy example in classification



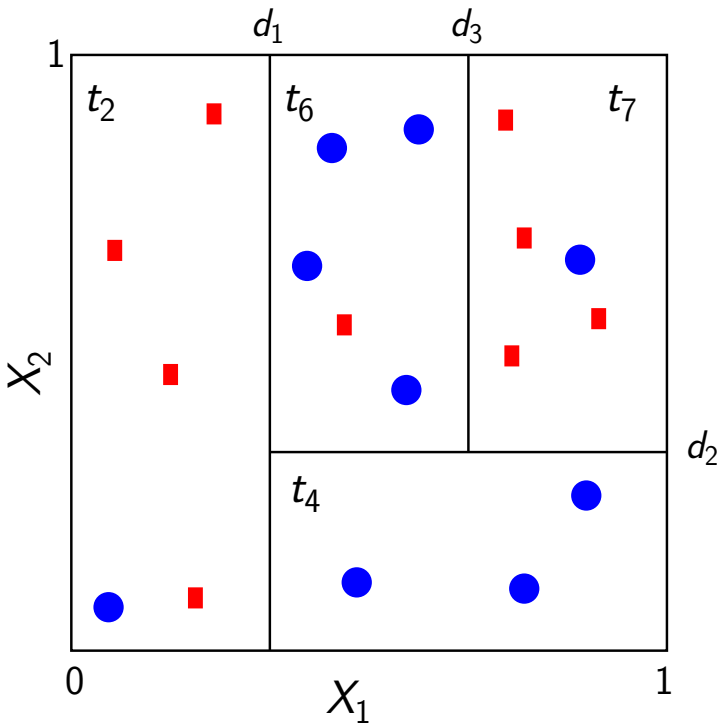



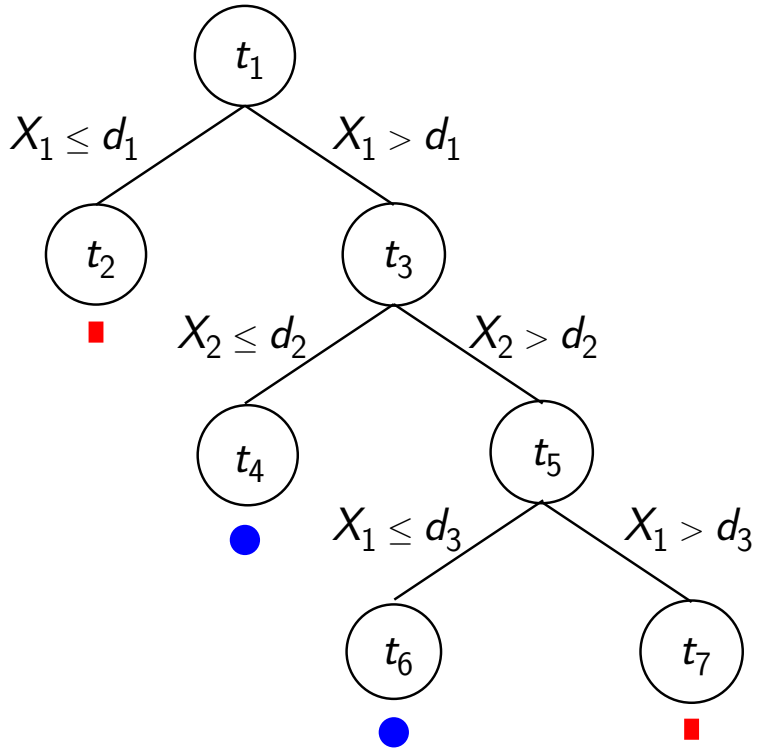

**Final tree $T_n$**

The final tree $T_n$ = a partition $\mathcal{P}$ of $\mathbb{R}^d$ made of $\sharp\mathcal{P}$ distinct regions called the terminal nodes or the leaves.

$\rightsquigarrow$ *In the example above, $\mathcal{P}$ is made of 4 leaves.*

# CART: a toy example in classification



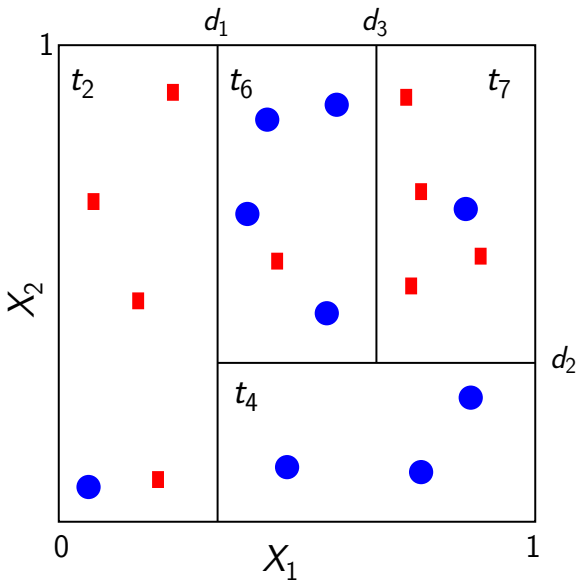

**Prediction rule based on the final tree $T_n$**

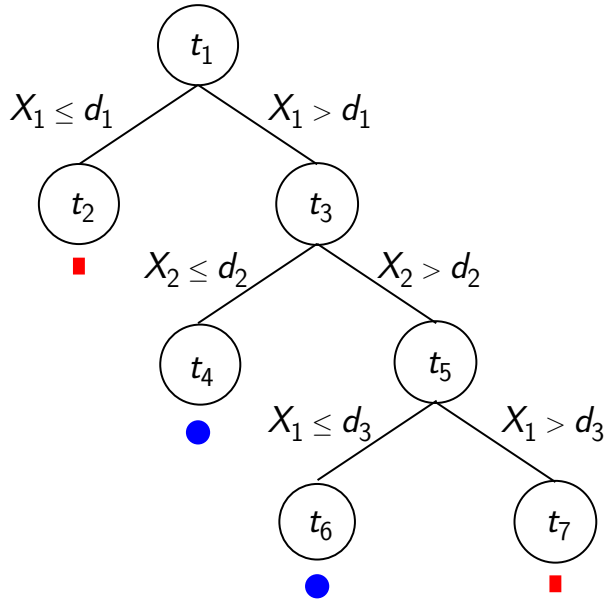
- In classification: at the end, we do a **majority vote** for $Y$ in each region of the final partition.
- In regression: at the end, we do the **mean** for $Y$ in each region of the final partition

# Vocabulary

- Each split defines two child nodes: the left and right child nodes.
- The terminal nodes (or the leaves) = the regions of the final partition.
- The root = the first node (= the whole feature space $\mathcal{X}$).

# Prediction rule of a tree




**Prediction rule based on the final tree $T_n$**

At the end,

- In classification: do the **majority vote** for $Y$ in each region of the final partition.
- In regression: do the **mean** for $Y$ in each region of the final partition.

# Prediction rule of a tree

- **Reminder**: The final tree $T_n$ = a partition $\mathcal{P}$ of the feature space $\mathbb{R}^d$.

**Prediction rule**

For any observation $\mathbf{x} \in \mathbb{R}^p$, the prediction function of a $T_n$ can be defined as

$$f_n(\mathbf{x}) = \sum_{t \in \mathcal{P}} \hat{c}_t \mathbb{1}_{\mathbf{x} \in t}.$$

➜ Predictions are constant in the terminal nodes (leaves).

➢ In regression: $\hat{c}_t$ is the **mean** of $Y$ in node $t$

$$\hat{c}_t = \bar{y}_t = \frac{1}{n_t} \sum_{i:\mathbf{x}_i \in t} y_i$$

➢ In classification: $\hat{c}_t$ is the **majority class** of $Y$ in node $t$

$$\hat{y}_i = \hat{k}(t) = \underset{k \in 1,\dots,K}{\operatorname{argmax}} \sum_{i:\mathbf{x}_i \in t} \mathbb{1}_{y_i = k},$$

where $k \in \{1, \dots, K\}$ are the labels (or classes) of $Y$.

# CART: a toy example in regression

**Reminder**

CART can be applied to both classification and regression problems.

➜ Now, let's see a **toy example in regression**.

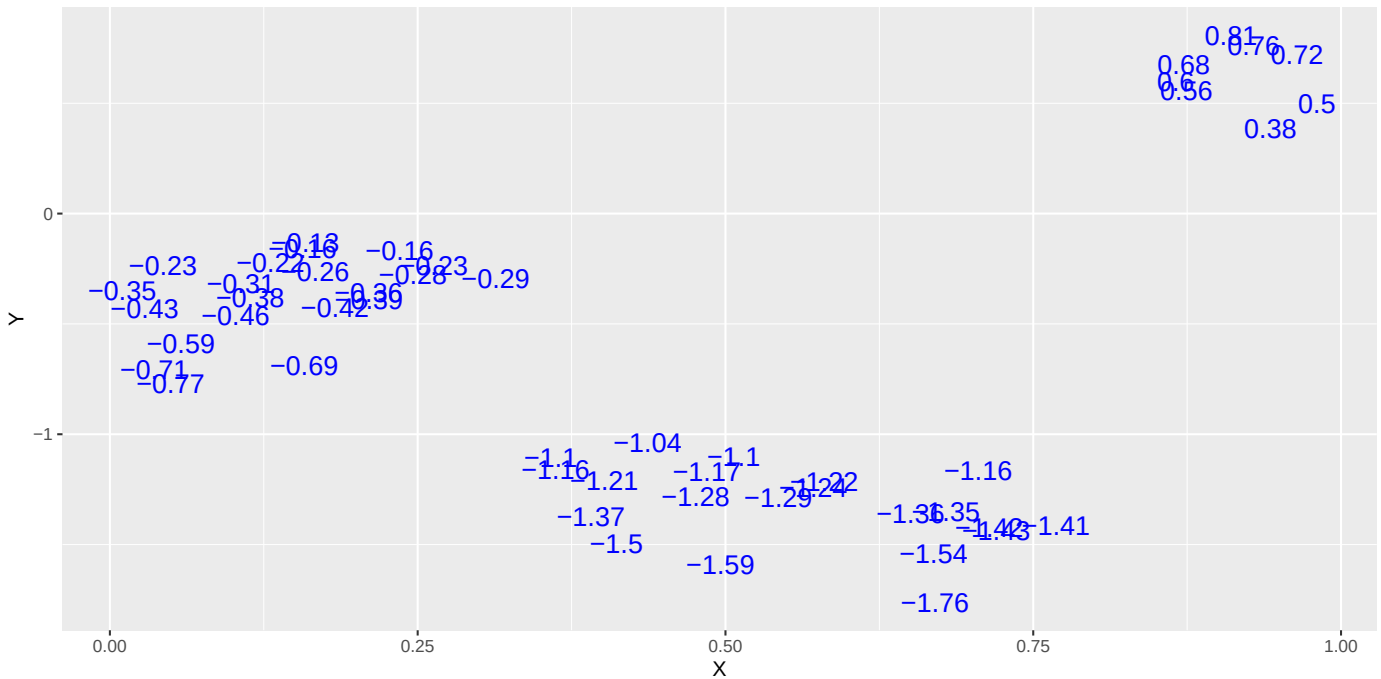
# CART: a toy example in regression

Problem: explain a continuous variable $Y$ based on one predictor $X$.

# CART: a toy example in regression

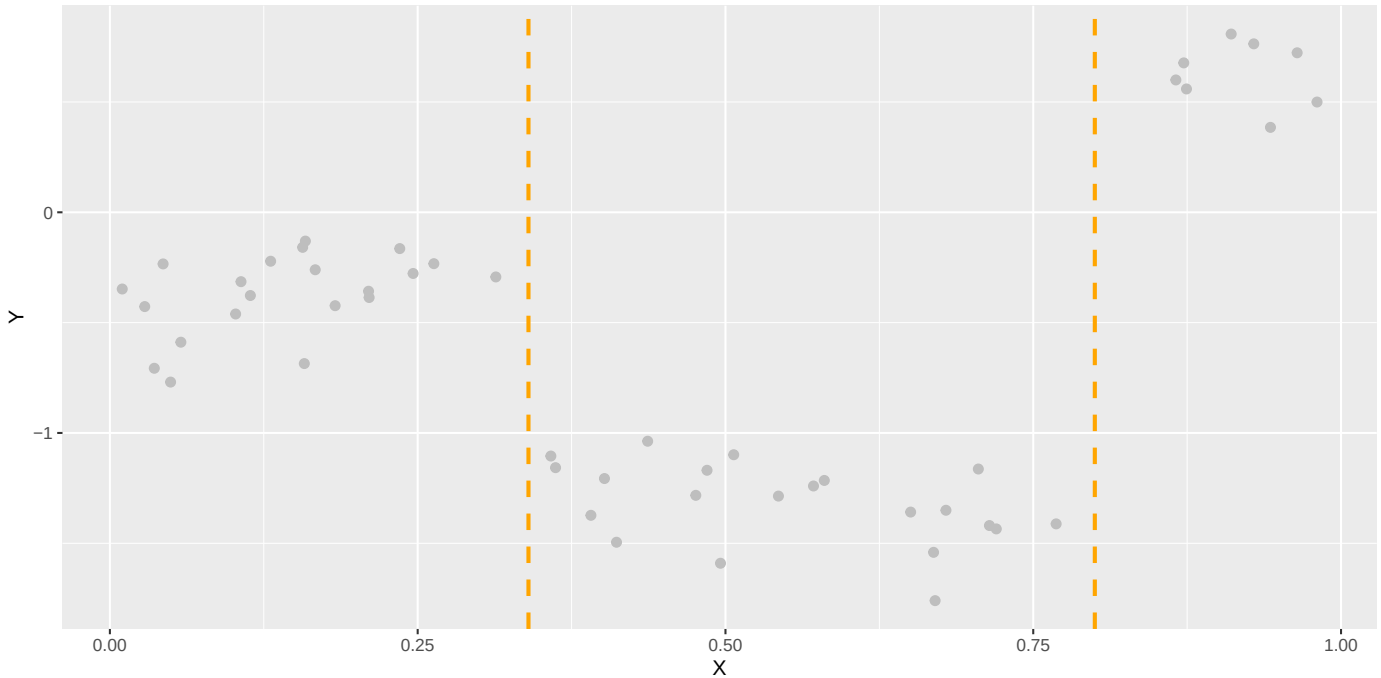
Problem: explain a continuous variable $Y$ based on one predictor $X$.

Data: a sample $\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \dots (\mathbf{X}_n, Y_n)\}$ with $X_i \in [0, 1]$, $Y_i \in [-2, 1]$ and $n = 50$.



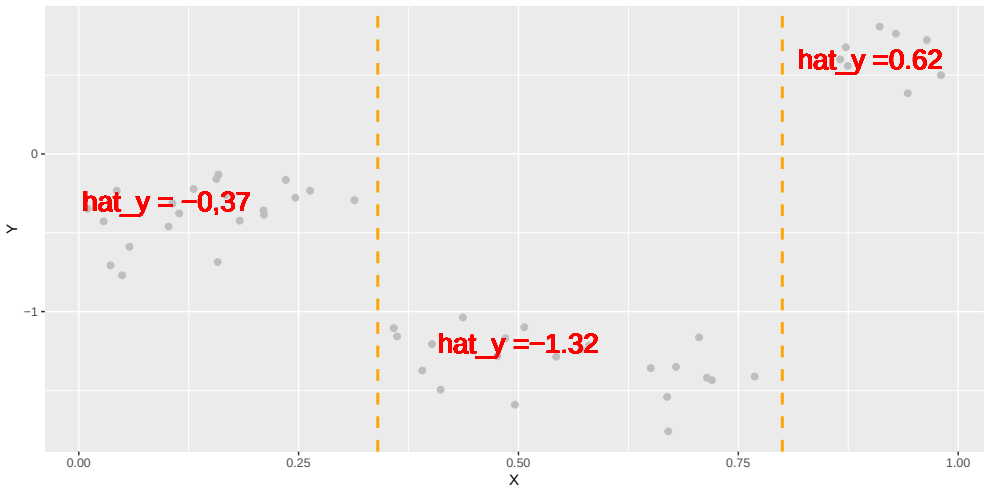

**The data**

# CART: a toy example in regression

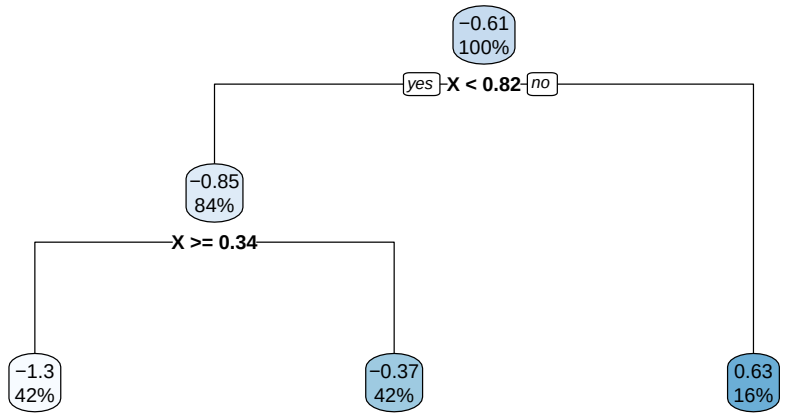



**A partition of the data**

# CART: a toy example in regression




A partition of the data




The associated CART tree

# CART method

To build a CART tree, we need to know the answer to these **two crucial questions**:

Question 1: How to choose an *"optimal"* split ?

Question 2: When to stop splitting ?

# Content

# How do we choose an "optimal" split ?

**CART principle**

Build prediction rules by means of recursive and binary splits of the feature space $\mathbb{R}^d$.

# How do we choose an "optimal" split ?

**CART principle**

Build prediction rules by means of recursive and binary splits of the feature space $\mathbb{R}^d$.

At each step, we split a node $t$ into two child nodes (*Left* and *Right*) w.r.t a split variable $X_j$ and a split threshold $d$ :

$$t_L(j,d) = \{\mathbf{X} \in t | X_j \leq d\} \quad \text{and} \quad t_R(j,d) = \{\mathbf{X} \in t | X_j > d\}.$$

➜ Constraint: we want that child nodes are as *pure* as possible (pure = homogeneous for $Y$).

➜ Conclusion: cut $t$ = find a couple $(j,d)$ minimizing an impurity criterion.

# How do we choose an "optimal" split ?

**CART principle**

Build prediction rules by means of recursive and binary splits of the feature space $\mathbb{R}^d$.

At each step, we split a node $t$ into two child nodes (*Left* and *Right*) w.r.t a split variable $X_j$ and a split threshold $d$ :

$$t_L(j,d) = \{\mathbf{X} \in t | X_j \leq d\} \quad \text{and} \quad t_R(j,d) = \{\mathbf{X} \in t | X_j > d\}.$$

➜ Constraint: we want that child nodes are as *pure* as possible (pure = homogeneous for $Y$).

➜ Conclusion: cut $t$ = find a couple $(j,d)$ minimizing an impurity criterion.

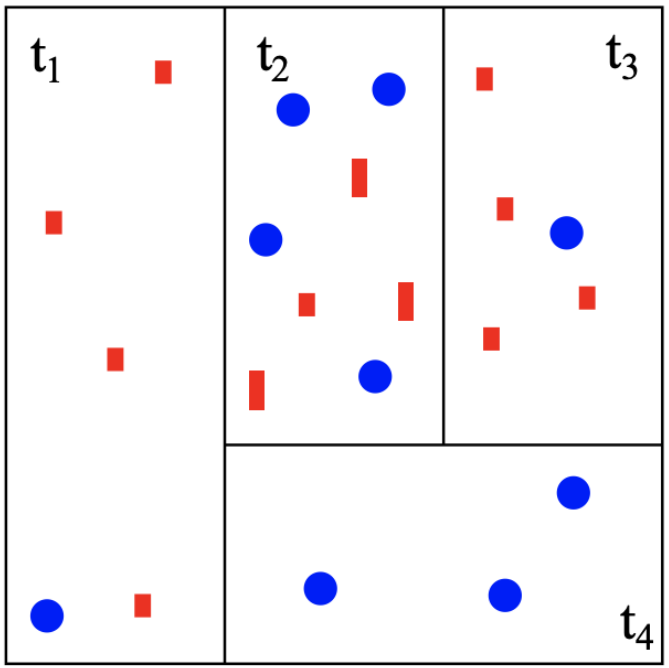
$\rightsquigarrow$ *What is an impurity criterion ?*

# Impurity

- **Definition**: an impurity function measures the homogeneity of the output variable $Y$ within a set (here a node).

- For a node, the impurity is:
  - Small when the node is homogeneous (according to $Y$): values of $Y$ are close to each other in the node.
  - Large when the node is heterogeneous (according to $Y$): values of $Y$ are different from each other in the node.

# Impurity: illustration




- $t_4$ is pure: impurity in $t_4$ is very small (equals 0).
- $t_2$ is less pure than the other nodes: impurity in $t_2$ is larger.

# Impurity

**Reminder**:

At each step, we split a node $t$ into two child nodes w.r.t a split variable $X_j$ and a split threshold $d$ :

$$t_L(j, d) = \{\mathbf{X} \in t | X_j \leq d\} \quad \text{and} \quad t_R(j, d) = \{\mathbf{X} \in t | X_j > d\}.$$

➜ **Constraint**: we want child nodes as *pure* as possible.

# Impurity

**Reminder**:

At each step, we split a node $t$ into two child nodes w.r.t a split variable $X_j$ and a split threshold $d$ :

$$t_L(j,d) = \{\mathbf{X} \in t | X_j \leq d\} \quad \text{and} \quad t_R(j,d) = \{\mathbf{X} \in t | X_j > d\}.$$

➜ **Constraint**: we want child nodes as *pure* as possible.

**CART split criterion**

We seek the couple $(j,d)$ that solves

$$\max_{j,d} \left[ \mathrm{I}(t) - (\hat{p}_{t_L} \mathrm{I}(t_L) + \hat{p}_{t_R} \mathrm{I}(t_R)) \right], \tag{2}$$

where

- $n_t$: the number of observations in $t$,
- $\hat{p}_{t_s} = \frac{n_{t_s}}{n_t}$: the proportion of observations in $t$ that fall into $t_s$, for $s \in \{L, R\}$.

➜ We select the couple $(j^\star, d^\star)$ that maximizes the impurity decrease (2) .

# Content

# Impurity in regression

**In regression**: $Y$ is continuous.

➜ Usually, $\mathrm{I}$ is the variance in the node defined as

$$\mathrm{I}(t) = \frac{1}{n_t} \sum_{i:\mathbf{x}_i \in t} (y_i - \bar{y}_t)^2,$$

with

- $n_t$: the number of observations into $t$,
- $\bar{y}_t = \frac{1}{n_t} \sum_{i:\mathbf{x}_i \in t} y_i$: the mean of $Y$ in the node $t$.

# Choice of a split in regression

**Reminder**:

At each step, we split a node $t$ into two child nodes w.r.t a split variable $X_j$ and a split threshold $d$ :

$$t_L(j,d) = \{\mathbf{X} \in t | X_j \leq d\} \quad \text{and} \quad t_R(j,d) = \{\mathbf{X} \in t | X_j > d\}.$$

➜ **Constraint**: we want child nodes as *pure* as possible.

**CART split criterion**

We choose the couple $(j,d)$ that solves

$$\max_{j,d} \left[ \mathrm{I}(t) - (\hat{p}_{t_L} \mathrm{I}(t_L) + \hat{p}_{t_R} \mathrm{I}(t_R)) \right].$$

# Choice of a split in regression

**Reminder**:

At each step, we split a node $t$ into two child nodes w.r.t a split variable $X_j$ and a split threshold $d$ :

$$t_L(j,d) = \{\mathbf{X} \in t | X_j \leq d\} \quad \text{and} \quad t_R(j,d) = \{\mathbf{X} \in t | X_j > d\}.$$

➜ **Constraint**: we want child nodes as *pure* as possible.

**CART split criterion**

We choose the couple $(j,d)$ that solves

$$\max_{j,d} \left[ \frac{1}{n_t} \sum_{i:\mathbf{x}_i \in t} (y_i - \bar{y}_t)^2 - \left( \hat{p}_{t_L} \sum_{i:\mathbf{x}_i \in t_L} (y_i - \bar{y}_{t_L})^2 + \hat{p}_{t_R} \sum_{i:\mathbf{x}_i \in t_R} (y_i - \bar{y}_{t_R})^2 \right) \right],$$

with $\bar{y}_{t_s}$ = the mean of $Y$ in $t_s$, for $s \in \{L, R\}$.

# Choice of a split in regression

**Reminder**:

At each step, we split a node $t$ into two child nodes w.r.t a split variable $X_j$ and a split threshold $d$ :

$$t_L(j,d) = \{\mathbf{X} \in t | X_j \leq d\} \quad \text{and} \quad t_R(j,d) = \{\mathbf{X} \in t | X_j > d\}.$$

➜ **Constraint**: we want child nodes as *pure* as possible.

**CART split criterion**

We choose the couple $(j,d)$ that solves

$$\max_{j,d} \left[ \underbrace{\sum_{i:\mathbf{x}_i \in t} (y_i - \bar{y}_t)^2}_{\mathrm{I}(t)} - \left( \underbrace{\sum_{i:\mathbf{x}_i \in t_L} (y_i - \bar{y}_{t_L})^2}_{\hat{p}_{t_L}\mathrm{I}(t_L)} + \underbrace{\sum_{i:\mathbf{x}_i \in t_R} (y_i - \bar{y}_{t_R})^2}_{\hat{p}_{t_R}\mathrm{I}(t_R)} \right) \right],$$

with $\bar{y}_{t_s}$ = the mean of $Y$ in $t_s$, for $s \in \{L, R\}$.

# Content

# Impurity in classification

**In classification**: $Y$ is a label, $Y \in \{1, \ldots, L\}$.

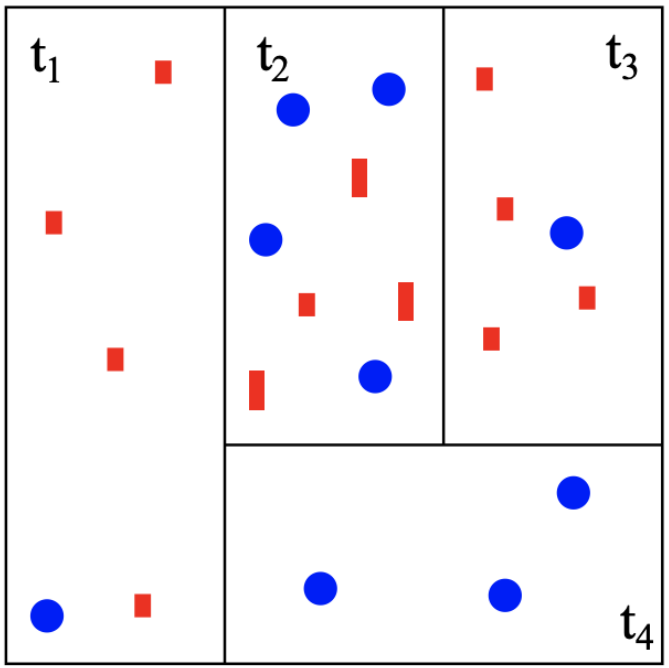
**Reminder**

The impurity measure $\mathrm{I}$ should be:

- Small when the node is homogeneous for $Y$: one label of $Y$ appears more frequently in $t$ compared to the others.
  ➜ $t$ pur $\Leftrightarrow$ only one unique label in $t$ $\Leftrightarrow$ $\mathrm{I}(t) = 0$.

- Large when the node is heterogeneous for $Y$: there is no "majority" label of $Y$ in $t$.

# Impurity: illustration




- $t_4$ is pure: impurity in $t_4$ is very small (equals 0).
- $t_2$ is less pure than the other nodes: impurity in $t_2$ is larger.

# Impurity in classification

Notation: the proportion of observations with label $k$ in $t$

$$\hat{p}_{kt} = \frac{1}{n_t} \sum_{i:\mathbf{x}_i \in t} \mathbb{1}_{\{y_i = k\}}, \quad k = 1, \ldots, K,$$

with $\mathbb{1}_{\{y_i = k\}} = \begin{cases} 0 & \text{if } y_i = k, \\ 1 & \text{otherwise.} \end{cases}$

# Impurity in classification

Notation: the proportion of observations with label $k$ in $t$

$$\hat{p}_{kt} = \frac{1}{n_t} \sum_{i:\mathbf{x}_i \in t} \mathbb{1}_{\{y_i = k\}}, \quad k = 1, \ldots, K,$$

with $\mathbb{1}_{\{y_i=k\}} = \begin{cases} 0 & \text{if } y_i = k, \\ 1 & \text{otherwise.} \end{cases}$

Impurity function $\mathrm{I}$ for classification: two main impurity functions

- Gini impurity: $\mathrm{I}(t) = \sum_{k=1}^{K} \hat{p}_{kt}(1 - \hat{p}_{kt})$.
- Cross-entropy: $\mathrm{I}(t) = -\sum_{k=1}^{K} \hat{p}_{kt} \log(\hat{p}_{kt})$.

*Remark: CART uses Gini by default.*

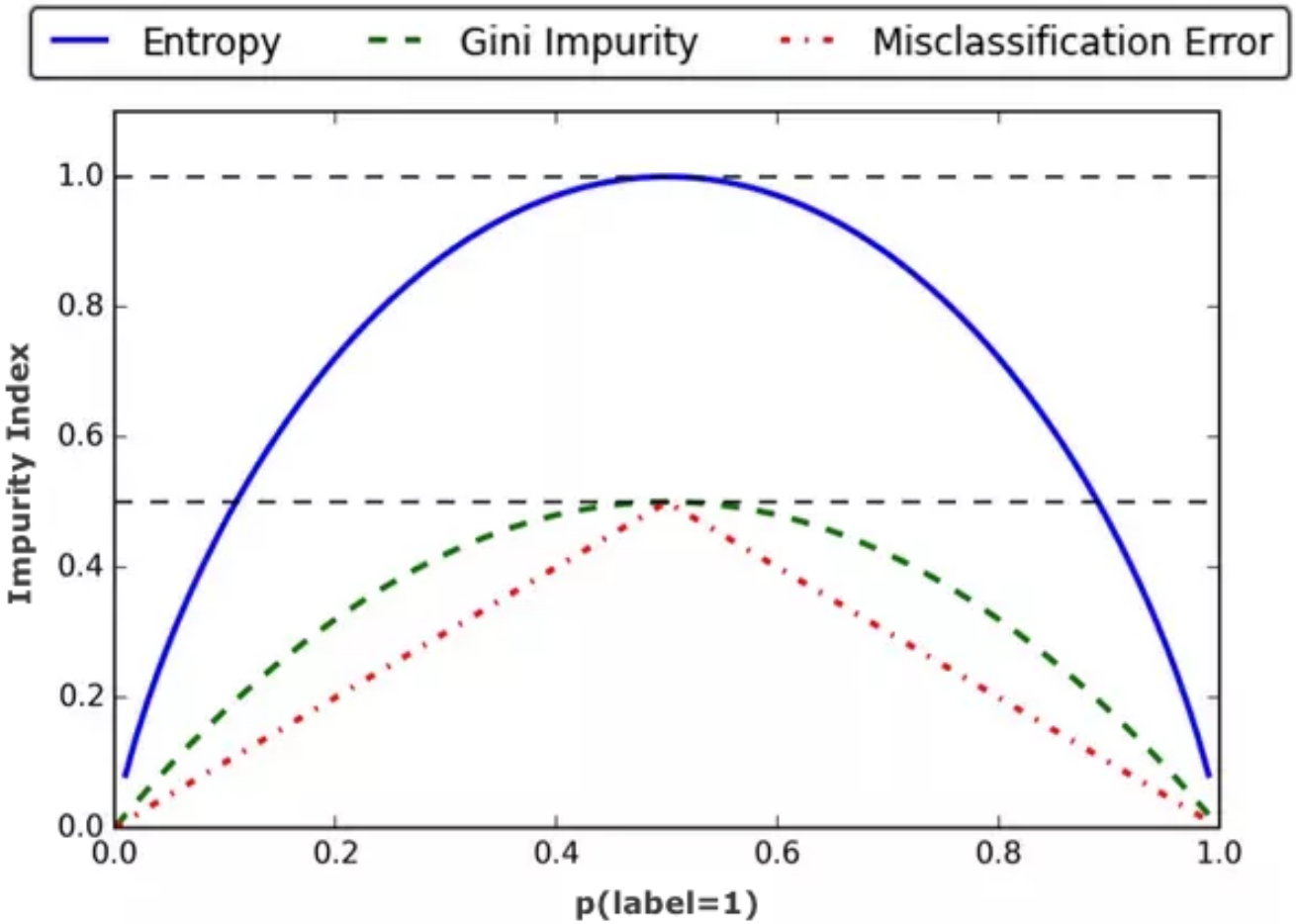
# Impurity in classification

**Binary classification**: only two labels, $Y = 0$ or $1$.

**Impurity measures in binary classification**

- Gini impurity: $\mathrm{I}(t) = 2\hat{p}_{1t}(1 - \hat{p}_{1t})$.
- Cross-entropy: $\mathrm{I}(t) = -\hat{p}_{1t}\log\left(\hat{p}_{1t}\right) - \hat{p}_{0t}\log\left(\hat{p}_{0t}\right)$.

# Choice of a split in classification




**Impurity functions for binary classification**

*Remark: use preferably Gini or equivalenthly cross-entropy.*

# Choice of a split in classification

**Reminder**:

At each step, we split a node $t$ into two child nodes w.r.t a split variable $X_j$ and a split threshold $d$ :

$$t_L(j,d) = \{\mathbf{X} \in t | X_j \leq d\} \quad \text{and} \quad t_R(j,d) = \{\mathbf{X} \in t | X_j > d\}.$$

➜ Constraint: we want child nodes as *pure* as possible.

**Split criterion for classification (with Gini)**

We choose the couple $(j,d)$ that solves

$$\max_{j,d} \left[ \mathrm{I}(t) - (\hat{p}_{t_L} \mathrm{I}(t_L) + \hat{p}_{t_R} \mathrm{I}(t_R)) \right].$$

# Choice of a split in classification

**Reminder**:

At each step, we split a node $t$ into two child nodes w.r.t a split variable $X_j$ and a split threshold $d$ :

$$t_L(j,d) = \{\mathbf{X} \in t | X_j \leq d\} \quad \text{and} \quad t_R(j,d) = \{\mathbf{X} \in t | X_j > d\}.$$

➜ Constraint: we want child nodes as *pure* as possible.

**Split criterion for classification (with Gini)**

We choose the couple $(j,d)$ that solves

$$\max_{j,d} \left[ \sum_{k=1}^{K} \hat{p}_{kt}(1-\hat{p}_{kt}) - (\hat{p}_{t_L} \sum_{k=1}^{K} \hat{p}_{kt_L}(1-\hat{p}_{kt_L}) + \hat{p}_{t_R} \sum_{k=1}^{K} \hat{p}_{kt_R}(1-\hat{p}_{kt_R})) \right].$$

# Choice of a split in classification

**Reminder**:

At each step, we split a node $t$ into two child nodes w.r.t a split variable $X_j$ and a split threshold $d$ :

$$t_L(j,d) = \{\mathbf{X} \in t | X_j \leq d\} \quad \text{and} \quad t_R(j,d) = \{\mathbf{X} \in t | X_j > d\}.$$

➜ Constraint: we want child nodes as *pure* as possible.

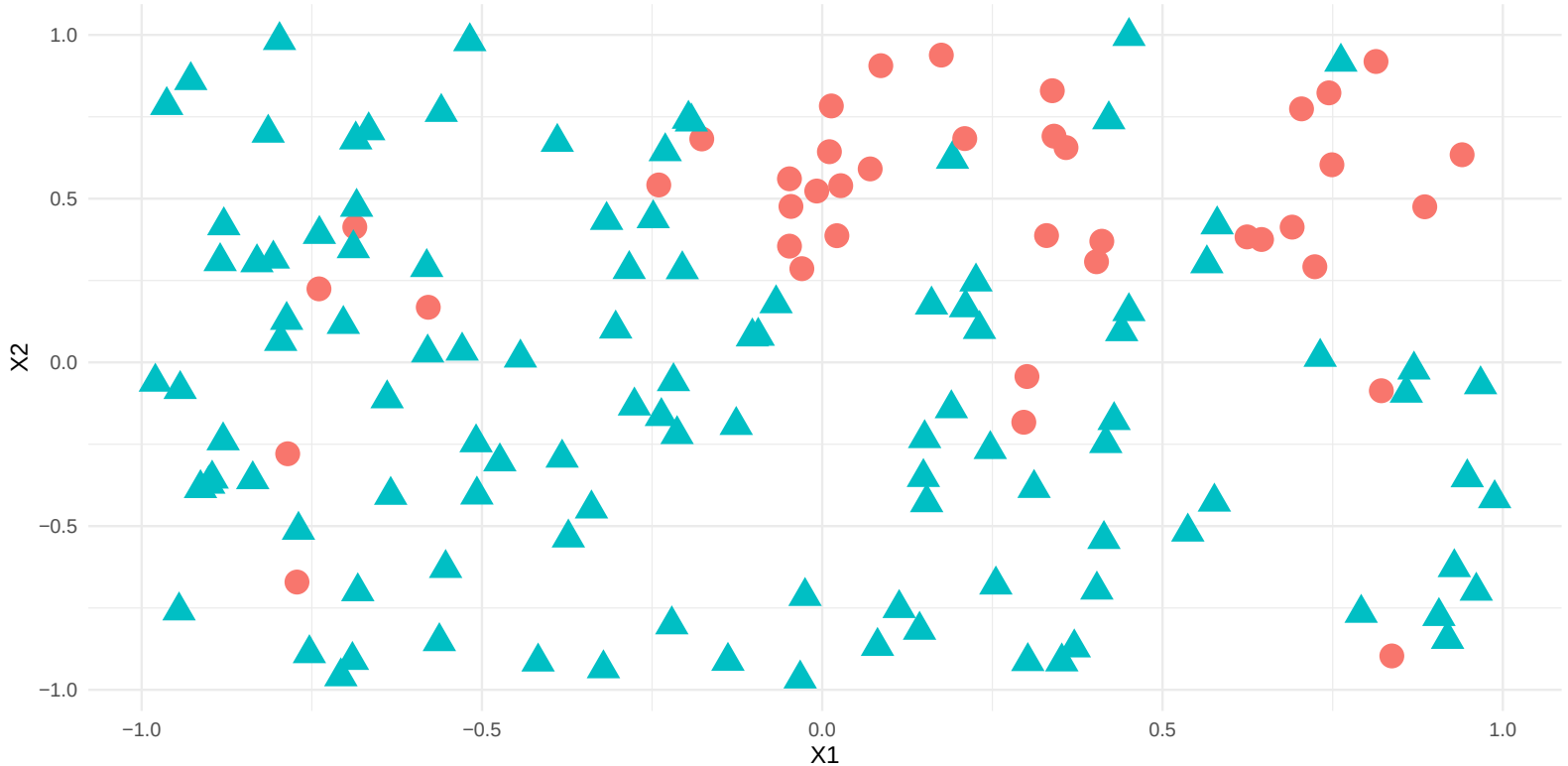
**Split criterion for classification (with Gini)**

We choose the couple $(j,d)$ that solves

$$\max_{j,d} \left[ \underbrace{\sum_{k=1}^{K} \hat{p}_{kt}(1-\hat{p}_{kt})}_{\mathrm{I}(t)} - \left( \underbrace{\hat{p}_{t_L} \sum_{k=1}^{K} \hat{p}_{kt_L}(1-\hat{p}_{kt_L})}_{\hat{p}_{t_L}\mathrm{I}(t_L)} + \underbrace{\hat{p}_{t_R} \sum_{k=1}^{K} \hat{p}_{kt_R}(1-\hat{p}_{kt_R})}_{\hat{p}_{t_R}\mathrm{I}(t_R)} \right) \right],$$

with $\bar{y}_{t_s}$ = the mean of $Y$ in $t_s$, for $s \in \{L, R\}$.

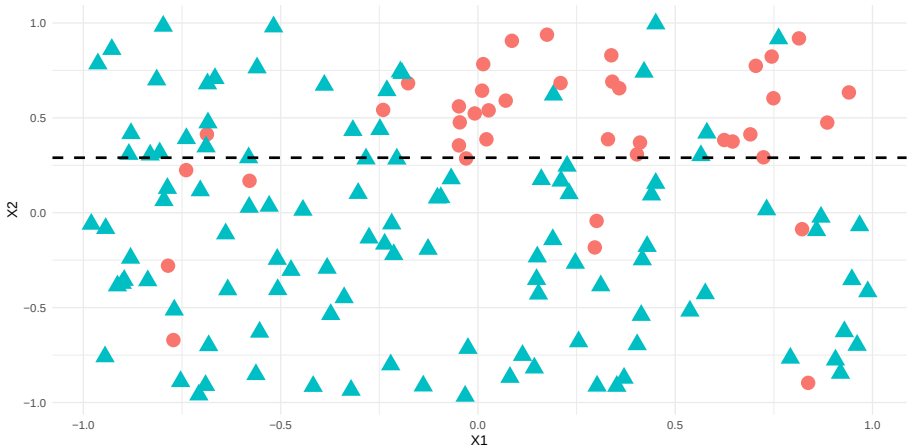
# Choice of a split: example

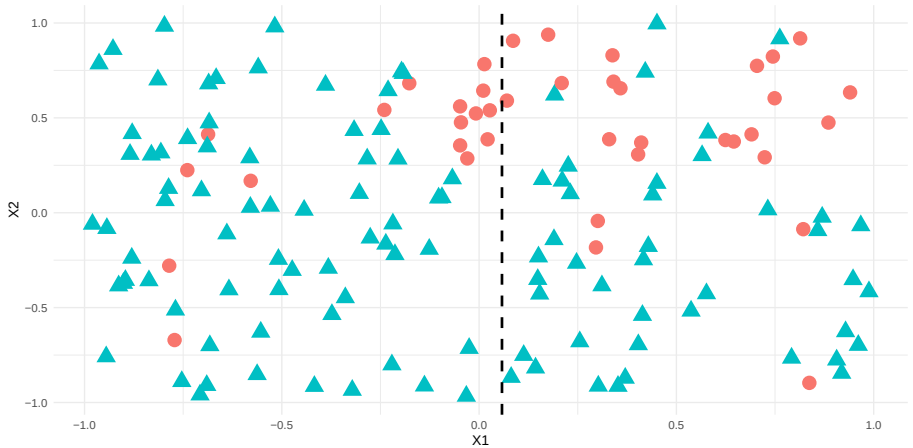



$$\mathrm{I}(t) = 0.3942$$

# Choice of a split: example

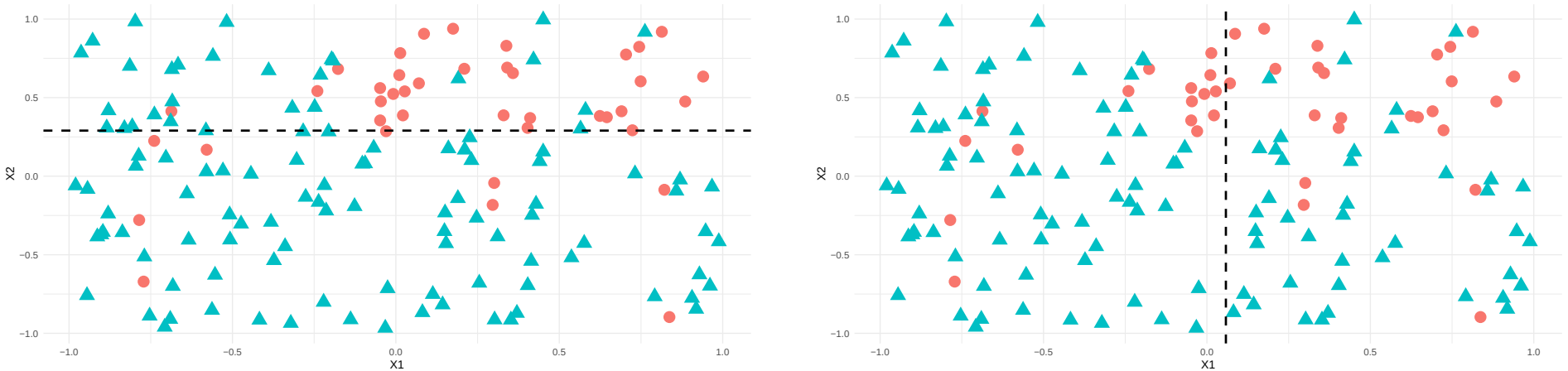
## Two split candidates



# Choice of a split: example

## Two split candidates

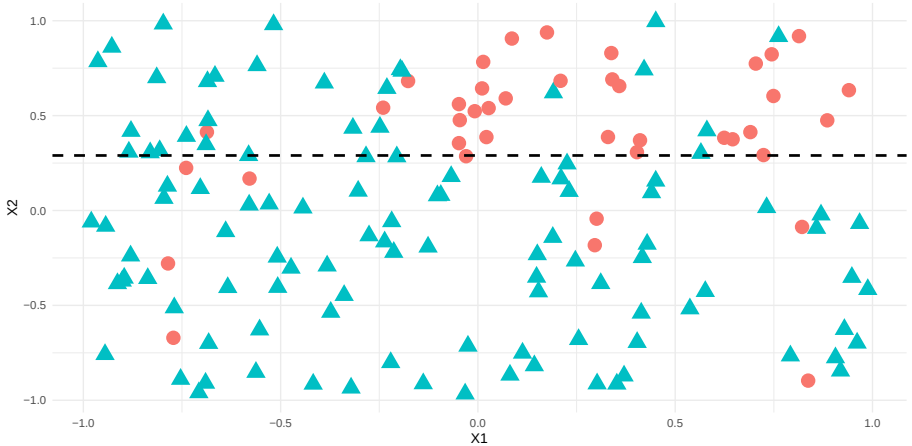



- Split 1: split variable $= X_2$ and the threshold $= 0.29$,
- Split 2: split variable $= X_1$ and the threshold $= 0.058$.

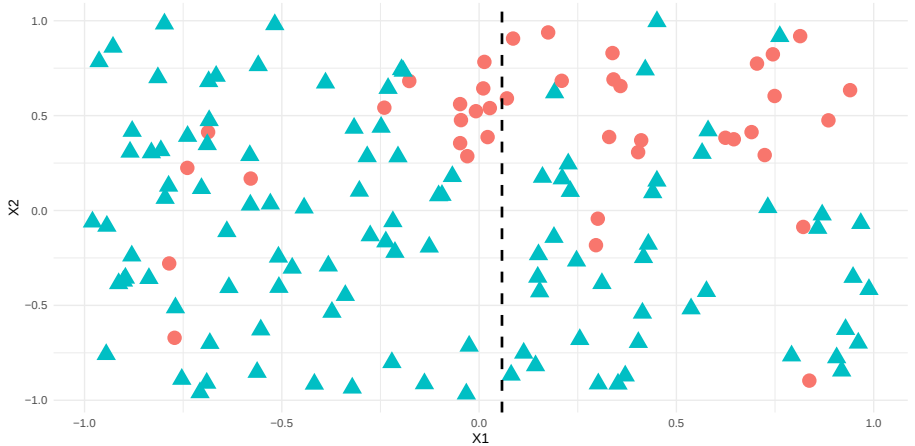
➜ What split do we choose ?

# Choice of a split: example

Split 1

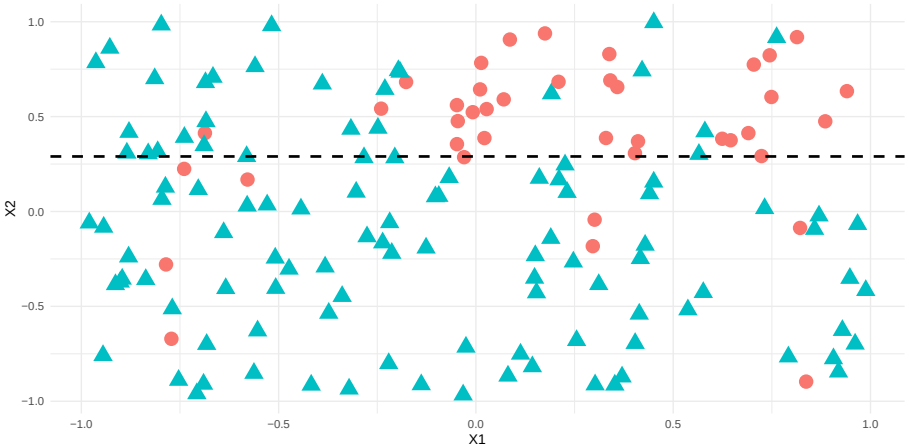



Split 2

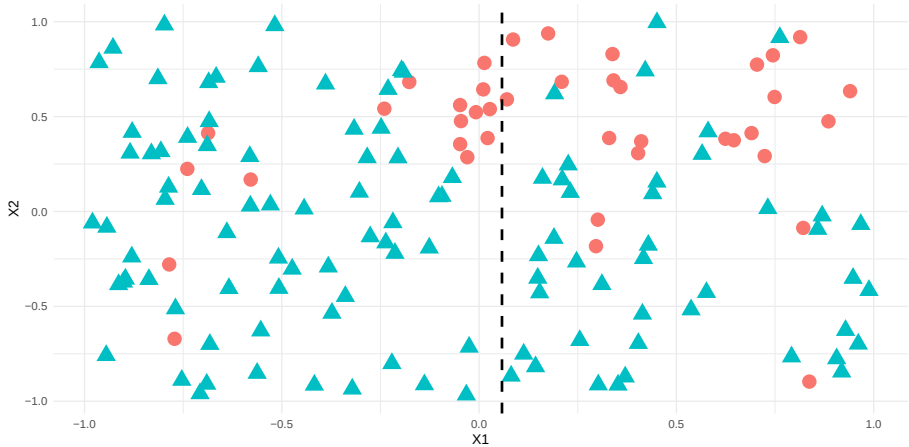


# Choice of a split: example

Split 1




Split 2




| | $\mathcal{I}(t_L)$ | $\mathcal{I}(t_R)$ | $\hat{p}_{t_L}$ | $\hat{p}_{t_R}$ | Impurity gain |
|---|---|---|---|---|---|
| Split 1 | 0.4978 | 0.1619 | 0.4 | 0.6 | 0.094 |
| Split 2 | 0.4898 | 0.1734 | 0.51 | 0.49 | 0.054 |

➜ What split do we choose ? Split 1.

# Content

**One remaining question**

To build a CART, we need to know the answer to these two crucial questions:

1. How to choose an *"optimal"* split ?
2. **When to stop splitting ?**

**One remaining question**

To build a CART, we need to know the answer to these two crucial questions:

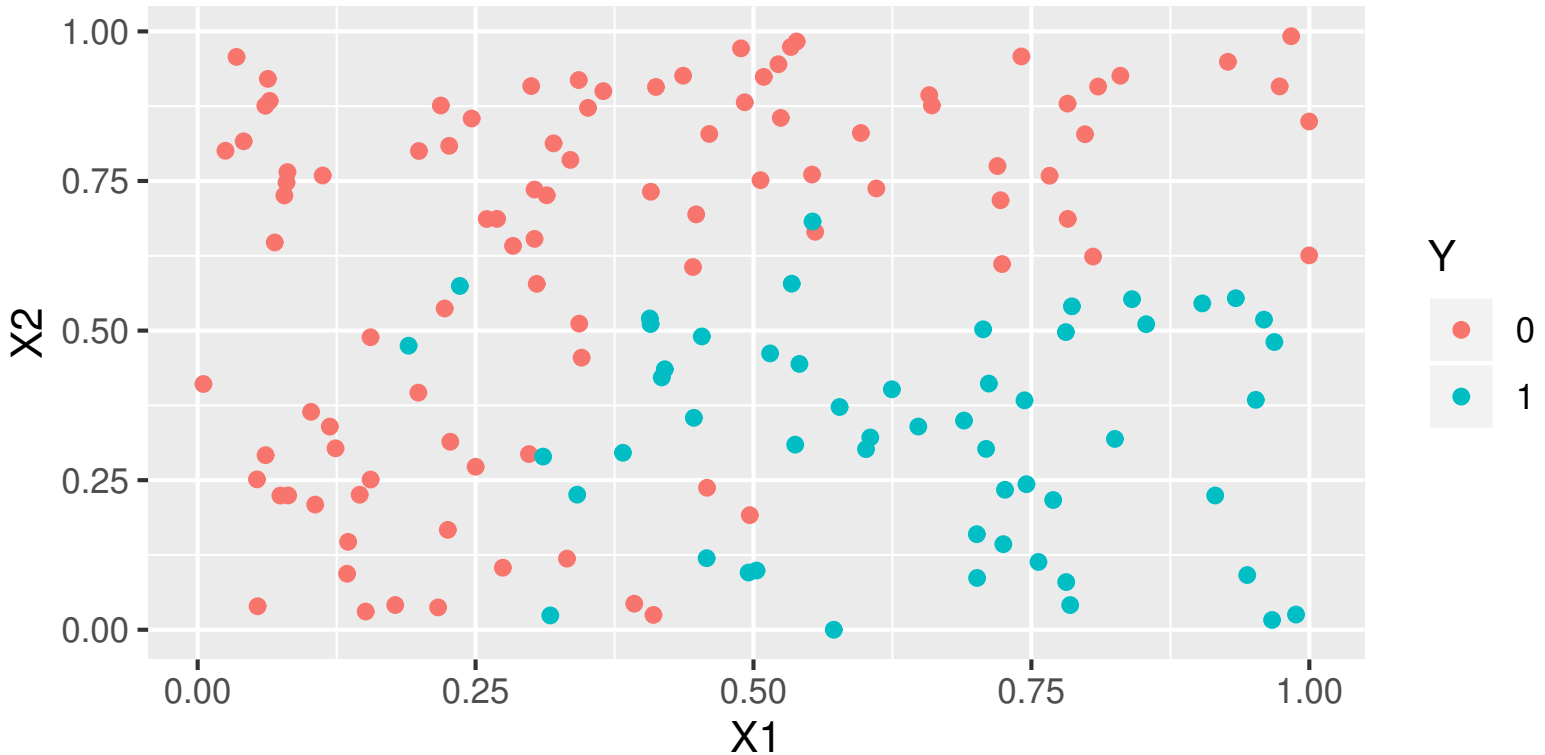
1. How to choose an *"optimal"* split ?
2. **When to stop splitting ?**

Several strategies:

- Do we select a number $m$ and we stop splitting as soon as there are less than $m$ observations in each node ?
- Do we split until all nodes are pur and we choose the maximum tree (or *the deepest tree*) ?

# Pruning: example 1

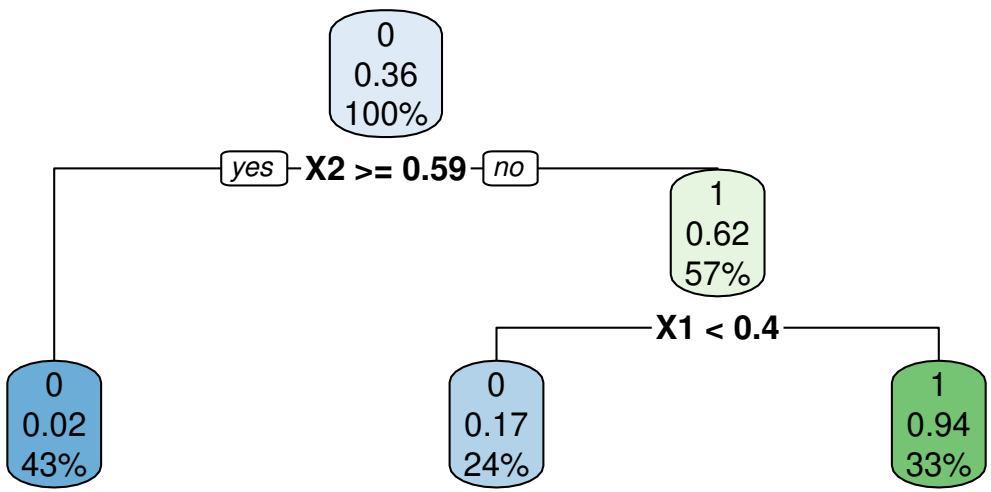
Data: a simple binary classification problem with 150 observations, two inputs $X_1$ and $X_2$.




➜ Here, we might want to split twice.

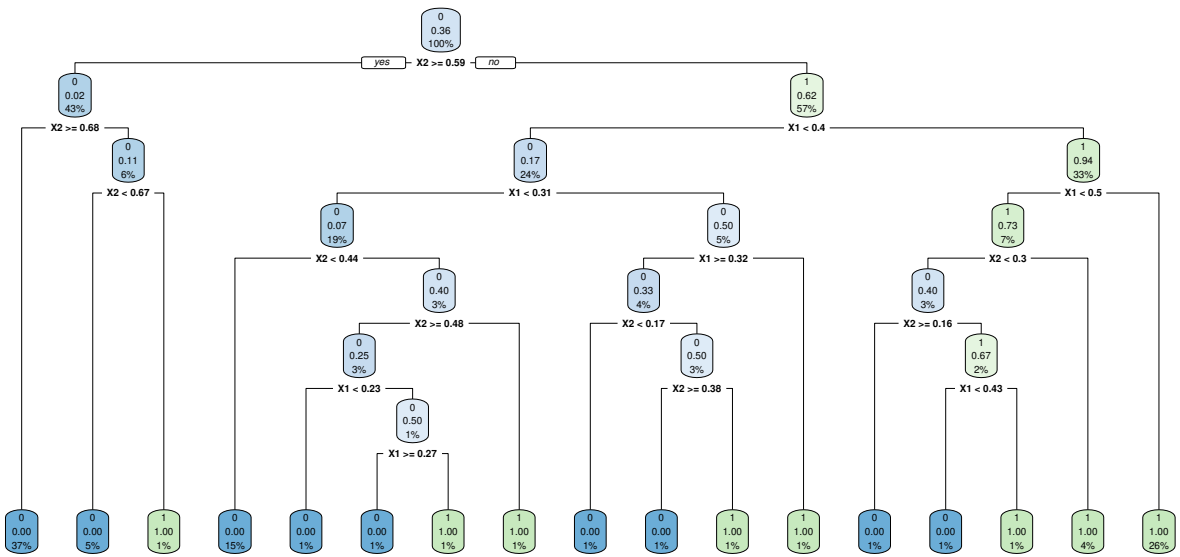
# Pruning: example 1

The associated CART tree with only two splits:



# Pruning: example 1

The maximum CART tree with 16 splits:



# Pruning: example 1

**Comparison**:

| | Misclassification error[2] | Number of splits |
|---|---|---|
| Small tree | 0.31 | 2 |
| Maximal tree | 0.37 | 16 |

➜ Performance is not always improved by the size of the tree.

[2] Misclassification error computed using 10-fold cross validation.

# Pruning

**Selection of the final tree**

We are faced a dilemma:

- Too small tree = robust/stable model (small variance)... but does not fit well the data (large bias) .
  $\hookrightarrow$ Underfitting.
- Too large tree = fits perfectly the data (low bias)... but unstable model (large variance)
  $\hookrightarrow$ Overfitting.

# Pruning

**Selection of the final tree**

We are faced a dilemma:

- Too small tree = robust/stable model (small variance)... but does not fit well the data (large bias) .
  $\hookrightarrow$ Underfitting.
- Too large tree = fits perfectly the data (low bias)... but unstable model (large variance)
  $\hookrightarrow$ Overfitting.

➠ How to find a trade-off ?

➠ How to select an *optimal* size for the tree ?

# Cost-complexity pruning

**CART strategy (Breiman, 1984)**

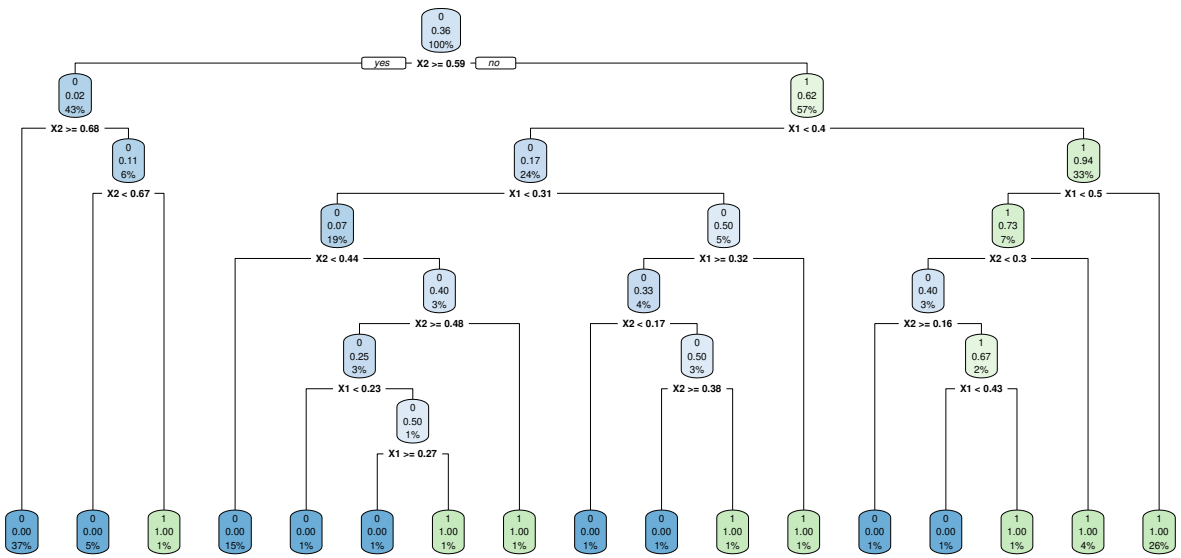
1. Build a maximal tree $T_{\max}$.

2. Select a sequence of *optimal* nested subtrees of $T_{\max}$.

3. Choose the final tree in this sequence: select the tree with the smallest empirical risk $\mathcal{R}_{emp}$.

# Cost-complexity pruning

**Step 1:** Build a maximal tree $T_{\max}$ by using the CART split criterion.



# Cost-complexity pruning

**Step 2:** Select a sequence of *optimal* nested subtrees

$$T_{\max} = T_0 \supset T_1 \supset \ldots \supset T_M = \{\text{the tree root}\}.$$

➜ Optimal w.r.t. a criterion that takes account both of tree performance and tree complexity.

➜ Breiman's theorem (1984) says that this sequence exists and is unique (see appendix).

➜ Each optimal tree is associated to a value for the parameter `ccp_alpha`. `ccp_alpha` depends on the tree complexity.

- $\alpha = 0$ for the deepest tree $T_{\max}$.
- if $\alpha = +\infty$ for the tree with no split = the root $t_1$.

# Cost-complexity pruning

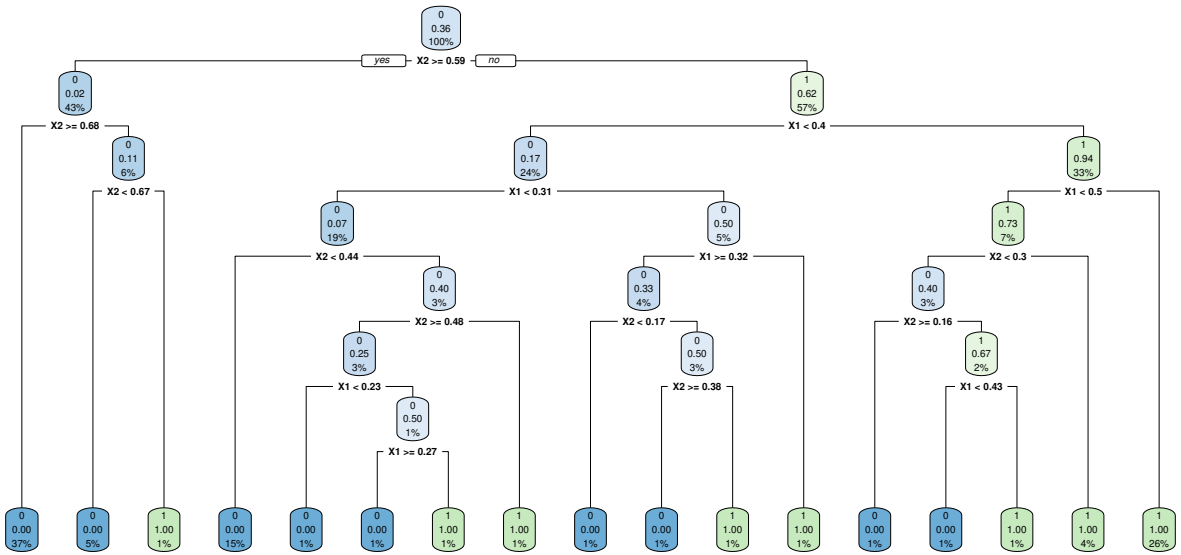
**Step 3:** Choose the final tree in this sequence i.e. select the tree with the smallest empirical risk $\mathcal{R}_{emp}$.

- In regression: the quadratic risk $R_{emp}(T_m, \mathcal{T}) = \frac{1}{n_{\mathcal{T}}} \sum_{i=1}^{n_{\mathcal{T}}} (y_i - T_m(\mathbf{x}_i))^2$
- In classification: the misclassification error $R_{emp}(T_m, \mathcal{T}) = \frac{1}{n_{\mathcal{T}}=} \sum_{i=1}^{n_{\mathcal{T}}} \mathbb{1}_{y_i \neq T_m(\mathbf{x}_i)}$.

*Reminder: evaluate the risk on a test set or by using cross valation, leave-one-out, etc.*

# Example: the sequence of optimal subtrees

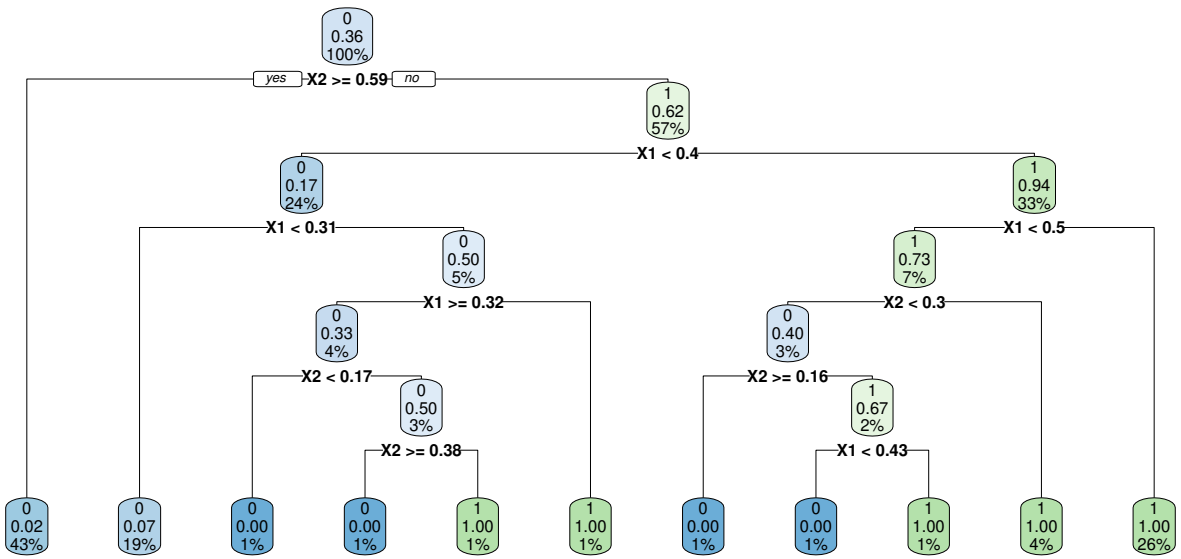
**Let see an example:** step 1 - we grow the deepest tree.




**The deepest tree, $T_{\max}$ ($=T_0$**

# Example: the sequence of optimal subtrees

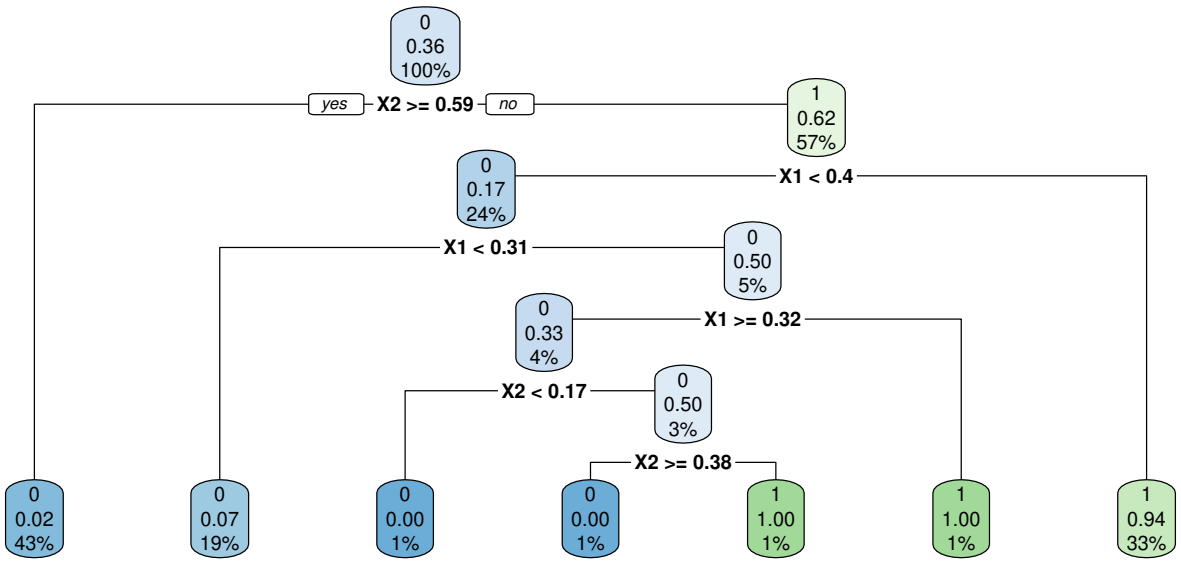
step 2 - we draw the sequence of nested subtrees.




$T_1$

# Example: the sequence of optimal subtrees

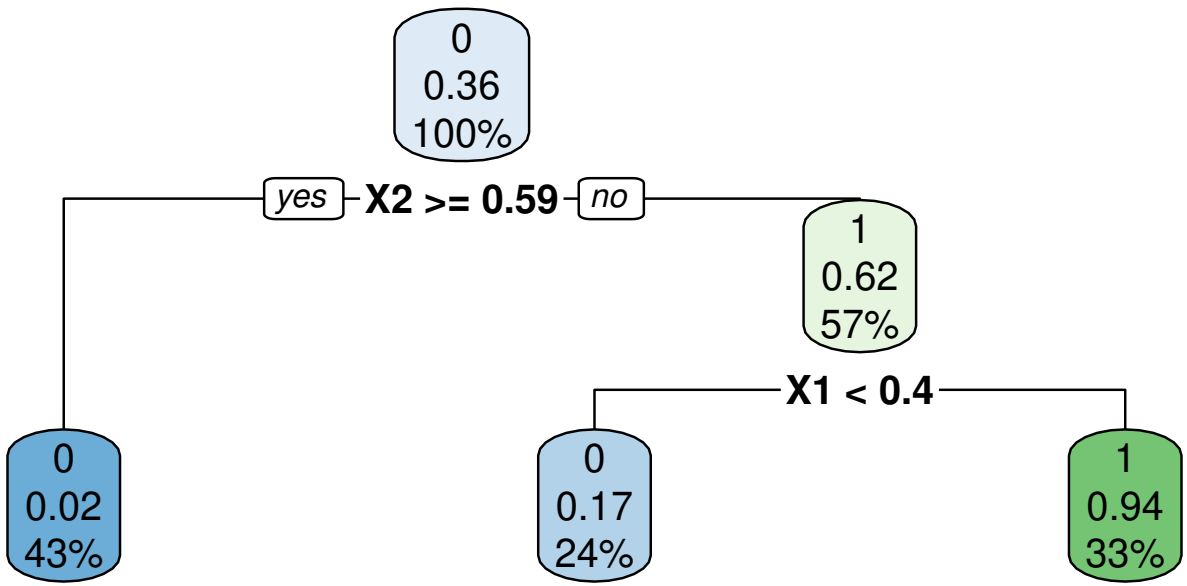
step 2 - we draw the sequence of nested subtrees.




$T_2$

# Example: the sequence of optimal subtrees

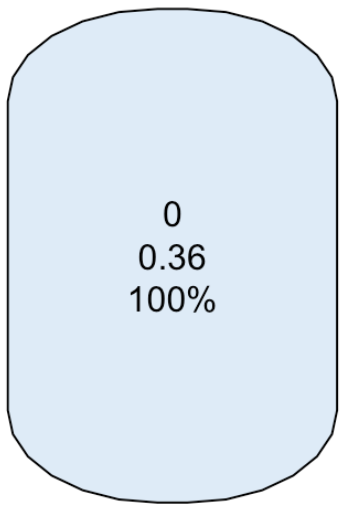
step 2 - we draw the sequence of nested subtrees.




$T_3$

# Example: the sequence of optimal subtrees

step 2 - we draw the sequence of nested subtrees.




$T_4$ **(the root)**

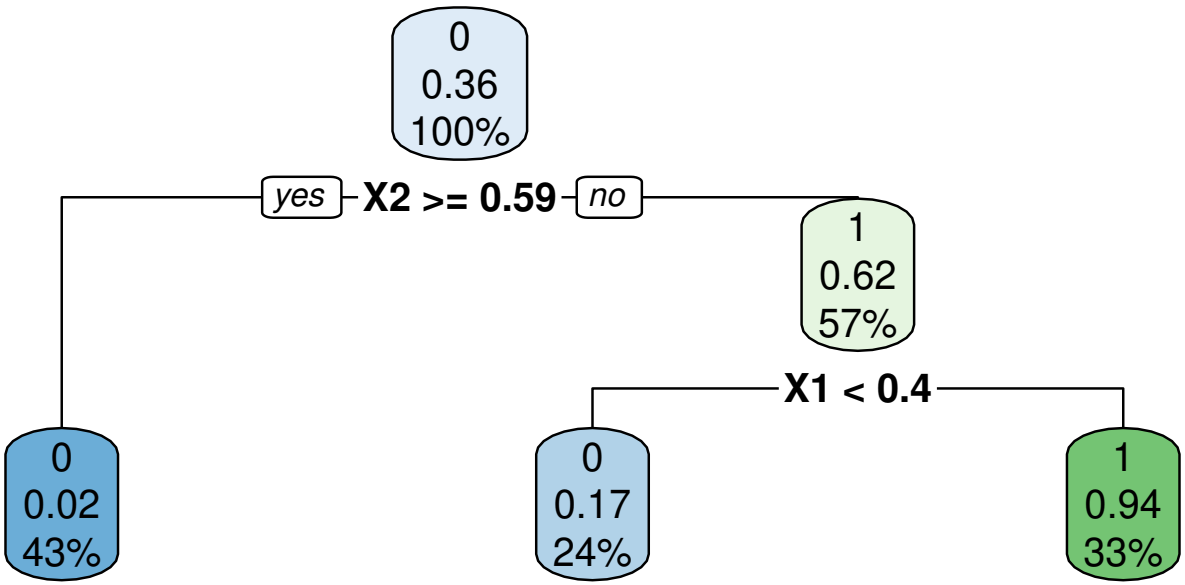
# Example: the sequence of optimal subtrees

step 3 - we select the tree that minimizes the accuracy computed by cross-validation.

| | `ccp_alpha` | Node count | train risk | CV risk |
|---|---|---|---|---|
| $T_{\max}$ | 0 | 17 | 0 | 0.14 |
| $T_1$ | 0.01 | 11 | 0.02 | 0.16 |
| $T_2$ | 0.014 | 7 | 0.04 | 0.15 |
| $T_3$ | 0.019 | 3 | 0.07 | 0.10 |
| the root | 0.41 | 1 | 0.36 | 0.36 |

➜ We select the subtree minimizing the error estimated by crossvalidation (CV risk) (`xerror`) : so we choose $T_3$.

# Example: the final tree




**The final tree**

# Content

# Conclusion

Properties of trees:

- ✓ Can handle categorical and quantitative response variable.
- ✓ Can handle mixed predictors (categorical and quantitative).
- ✓ Easily ignore redundant variables.
- ✓ Small trees are easy to interpret.
- ✓ Handle cleverly missing data.
- ✗ Large trees are not easy to interpret.
  ➜ Sensitive to small changes in the training data.
- ✗ Trees are sensitive to outliers and small changes in the train set.
  ➜ Often lead to lower prediction performance compared to other learning methods.

# Conclusion

- ✓ *Simple* method for both regresssion and classification: *easy* to understand, to implement and to interpret.
- ✓ Wide applicability: non-paramteric method, for both categorical and continuous predictors.
- ✗ Main drawback: algorithm not robust.

# Conclusion

✓ *Simple* method for both regresssion and classification: *easy* to understand, to implement and to interpret.

✓ Wide applicability: non-paramteric method, for both categorical and continuous predictors.

✗ Main drawback: algorithm not robust.

✓✓✓ **...But standard model (baseline model) for lots of averaging models** such as bagging, random forests, boosting, etc.

# References



Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984).
**Classification and regression trees.**
CRC press.


Hastie, T., Tibshirani, R., and Friedman, J. (2009).
**The elements of statistical learning: data mining, inference, and prediction.**
Springer Science & Business Media.


Rokach, L. and Maimon, O. (2005).
**Decision trees.**
In *Data mining and knowledge discovery handbook*, pages 165–192. Springer.


Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Philip, S. Y., et al. (2008).
**Top 10 algorithms in data mining.**
*Knowledge and information systems*, 14(1):1–37.

**The pruning phase**

Find a pruned subtree of $T_{\max}$ that minimizes the cost function

$$\mathcal{R}_\alpha(T) = \mathcal{R}(T, \mathcal{D}_n) + \alpha|\widetilde{T}|, \quad \alpha \in \mathbb{R}^+,$$

where

- $\mathcal{R}(T, \mathcal{D}_n)$= the risk,
- $|\widetilde{T}|$= the number of leaves of $T$,
- $\alpha$= a tuning parameter which manage the bias-variance tradeoff:
  - if $\alpha = 0$ ➠ Choose the deepest tree $T_{\max}$.
  - if $\alpha = +\infty$ ➠ Choose the tree with no split = the root $t_1$.

# Appendix: Cost-complexity pruning

**The pruning phase**

Find a pruned subtree of $T_{\max}$ that minimizes the cost function

$$\mathcal{R}_\alpha(T) = \mathcal{R}(T, \mathcal{D}_n) + \alpha|\widetilde{T}|, \quad \alpha \in \mathbb{R}^+,$$

where

- $\mathcal{R}(T, \mathcal{D}_n)$= the risk,
- $|\widetilde{T}|$= the number of leaves of $T$,
- $\alpha$= a tuning parameter which manage the bias-variance tradeoff:
  - if $\alpha = 0$ ➠ Choose the deepest tree $T_{\max}$.
  - if $\alpha = +\infty$ ➠ Choose the tree with no split = the root $t_1$.

**Two problems:**

➠ For any fixed $\alpha$, how to find the optimal tree?

➠ Who to choose the value for $\alpha$?

**Theorem (Breiman, 1984)**

For any non-trivial and tree $T$ with root $t_1$, there exist a unique sequence

$$0 = \alpha_1 < \ldots < \alpha_K = \infty$$

and a unique sequence of nested subtrees of $T$

$$T \supset T_1 \supset \ldots \supset T_K = \{t_1\}$$

such that for every $1 \leq k < K$,

$$\forall\, \alpha \in [\alpha_k; \alpha_{k+1}[, \quad T_k = \underset{T' \preceq T}{\operatorname{argmin}}\, \mathcal{R}_\alpha(T'), \text{ and}$$

$$\forall\, \alpha \geq \alpha_K, \quad T_K = \underset{T' \preceq T}{\operatorname{argmin}}\, \mathcal{R}_\alpha(T').$$

# Appendix: Cost-complexity pruning

**Theorem (Breiman, 1984)**

For any non-trivial and tree $T$ with root $t_1$, there exist a unique sequence

$$0 = \alpha_1 < \ldots < \alpha_K = \infty$$

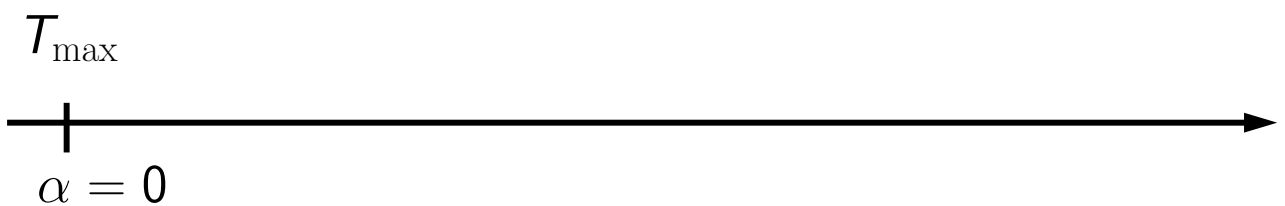
and a unique sequence of nested subtrees of $T$

$$T \supset T_1 \supset \ldots \supset T_K = \{t_1\}$$

such that for every $1 \leq k < K$,

$$\forall\, \alpha \in [\alpha_k; \alpha_{k+1}[, \quad T_k = \operatorname*{argmin}_{T' \preceq T} \mathcal{R}_\alpha(T'), \text{ and}$$

$$\forall\, \alpha \geq \alpha_K, \quad T_K = \operatorname*{argmin}_{T' \preceq T} \mathcal{R}_\alpha(T').$$



# Appendix: Cost-complexity pruning

**Theorem (Breiman, 1984)**

For any non-trivial and tree $T$ with root $t_1$, there exist a unique sequence

$$0 = \alpha_1 < \ldots < \alpha_K = \infty$$

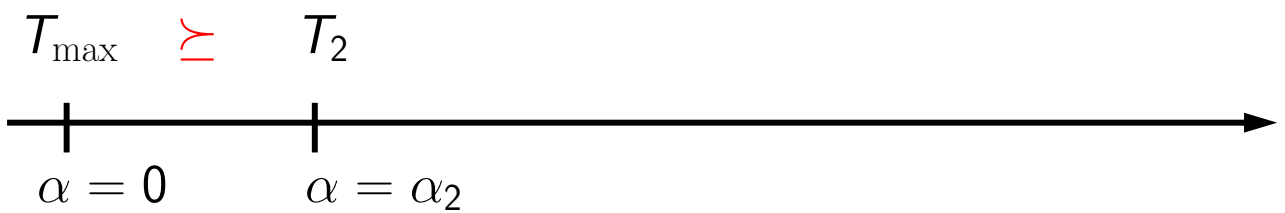
and a unique sequence of nested subtrees of $T$

$$T \supset T_1 \supset \ldots \supset T_K = \{t_1\}$$

such that for every $1 \leq k < K$,

$$\forall\, \alpha \in [\alpha_k; \alpha_{k+1}[, \quad T_k = \underset{T' \preceq T}{\operatorname{argmin}}\, \mathcal{R}_\alpha(T'), \text{ and}$$

$$\forall\, \alpha \geq \alpha_K, \quad T_K = \underset{T' \preceq T}{\operatorname{argmin}}\, \mathcal{R}_\alpha(T').$$



# Appendix: Cost-complexity pruning

**Theorem (Breiman, 1984)**

For any non-trivial and tree $T$ with root $t_1$, there exist a unique sequence

$$0 = \alpha_1 < \ldots < \alpha_K = \infty$$

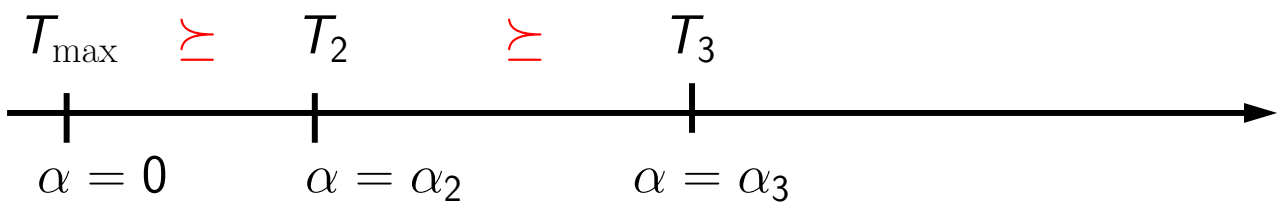
and a unique sequence of nested subtrees of $T$

$$T \supset T_1 \supset \ldots \supset T_K = \{t_1\}$$

such that for every $1 \leq k < K$,

$$\forall\, \alpha \in [\alpha_k; \alpha_{k+1}[, \quad T_k = \underset{T' \preceq T}{\operatorname{argmin}}\, \mathcal{R}_\alpha(T'), \text{ and}$$

$$\forall\, \alpha \geq \alpha_K, \quad T_K = \underset{T' \preceq T}{\operatorname{argmin}}\, \mathcal{R}_\alpha(T').$$



# Appendix: Cost-complexity pruning

**Theorem (Breiman, 1984)**

For any non-trivial and tree $T$ with root $t_1$, there exist a unique sequence

$$0 = \alpha_1 < \ldots < \alpha_K = \infty$$

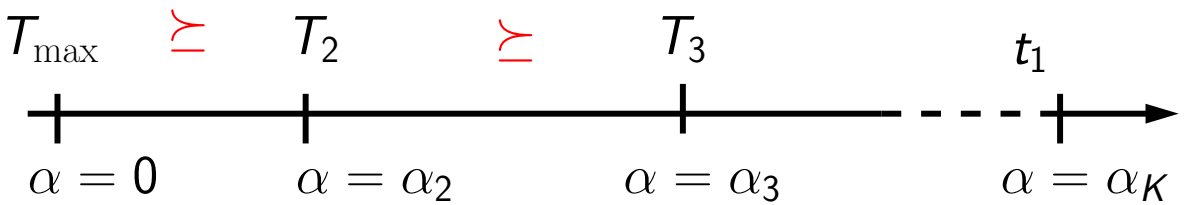
and a unique sequence of nested subtrees of $T$

$$T \supset T_1 \supset \ldots \supset T_K = \{t_1\}$$

such that for every $1 \leq k < K$,

$$\forall\, \alpha \in [\alpha_k; \alpha_{k+1}[, \quad T_k = \operatorname*{argmin}_{T' \preceq T} \mathcal{R}_\alpha(T'), \text{ and}$$

$$\forall\, \alpha \geq \alpha_K, \quad T_K = \operatorname*{argmin}_{T' \preceq T} \mathcal{R}_\alpha(T').$$



# Appendix: Cost-complexity pruning

**CART prunning strategy**

Based on the previous theorem, we know now how to find a final tree:

1. Build the finite sequence of "optimal" subtrees of $T_{\max}$.

2. Choose the final tree in this sequence (or one value for $\alpha$).