# Trees and random forests

**Hicham Janati** · *INSEA* · hjanati@insea.ac.ma

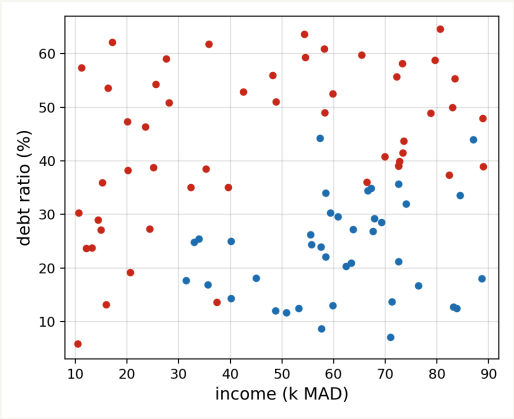
FSML · based on chapter 7 of the lecture notes

# Decision trees

# Predicting default

A bank wants to predict a future **loan default** (when a client does not reimburse) from two features.

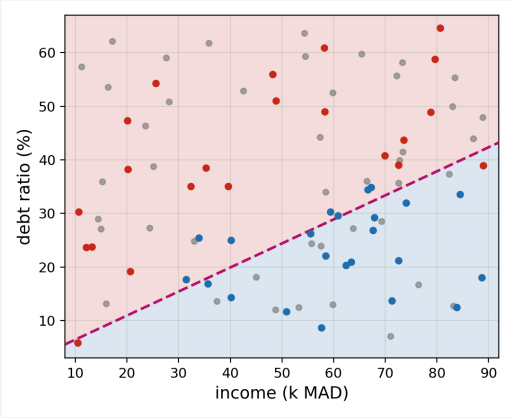
| income (kDh) | debt (%) | default |
|---|---|---|
| 80 | 59 | 1 |
| 33 | 25 | 0 |
| 58 | 49 | 1 |
| 72 | 56 | 1 |
| ⋮ | ⋮ | ⋮ |
| 62 | 29 | ? |



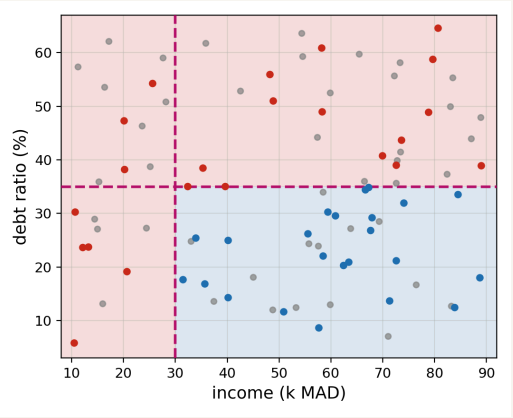
# Predicting default

A bank wants to predict a future **loan default** (when a client does not reimburse) from two features.

| income (kDh) | debt (%) | default |
|---|---|---|
| 80 | 59 | 1 |
| 33 | 25 | 0 |
| 58 | 49 | 1 |
| 72 | 56 | 1 |
| ⋮ | ⋮ | ⋮ |
| 62 | 29 | ? |




linear model: train **20%**, test **16%**




best function: train **0%**, test **9%**

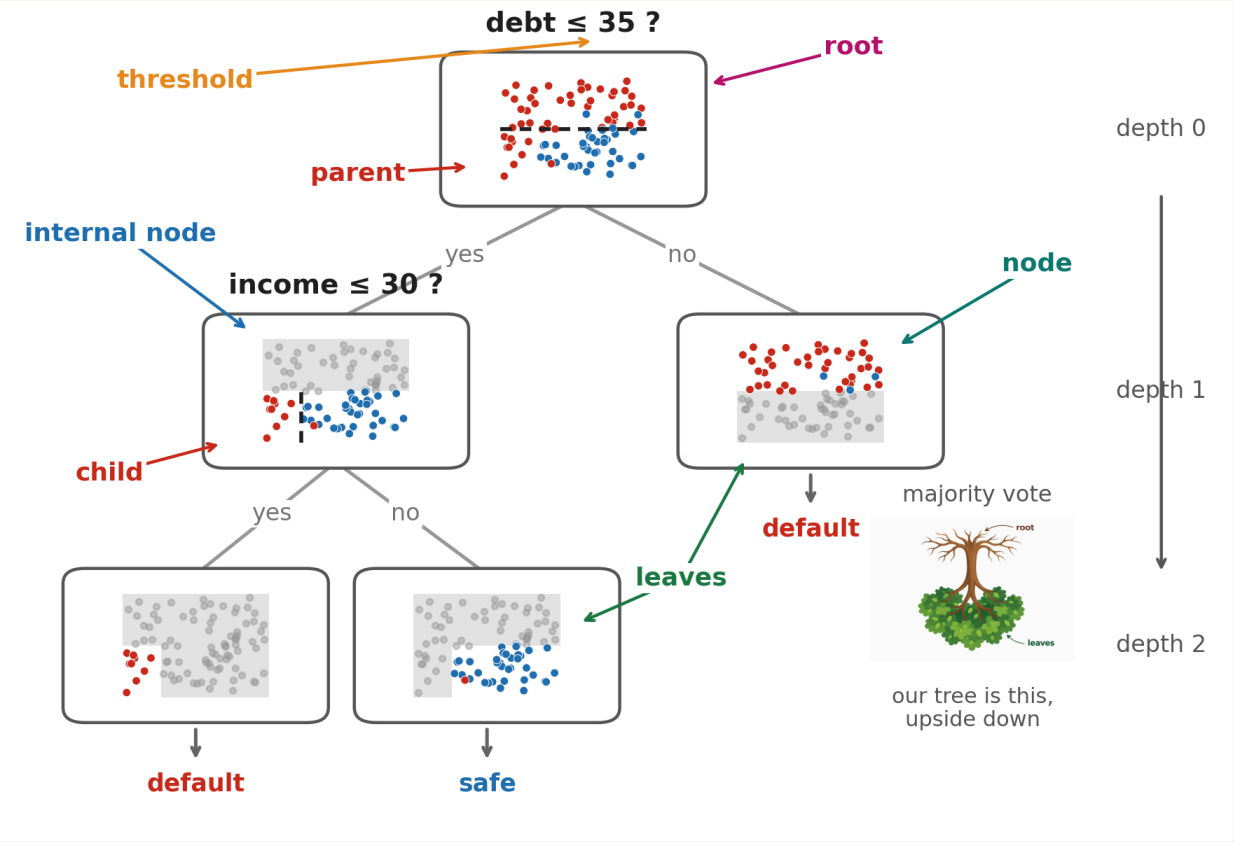
How can we define such a function ?

$$f(x) = 1 \Leftrightarrow \text{income} \leq 30 \text{ or } \text{debt} \geq 35$$

$$x_1 \leq 30 \text{ or } x_2 \geq 35$$

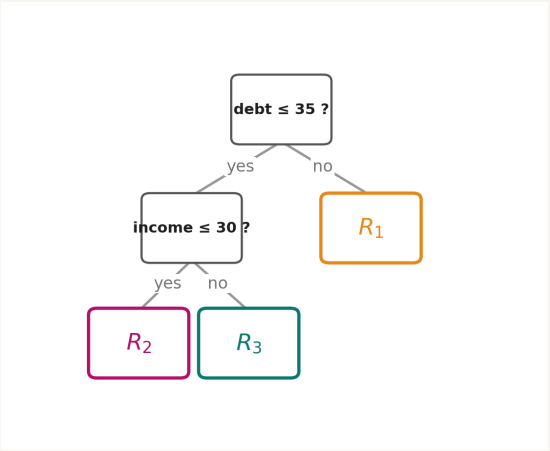
By thresholding the feature space: **a decision tree**.

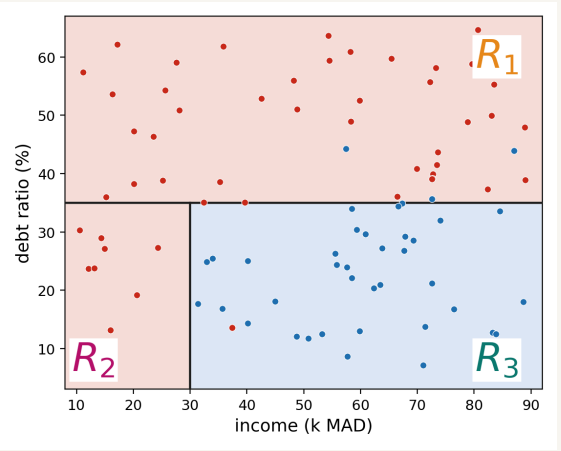
# Visualizing a classification tree



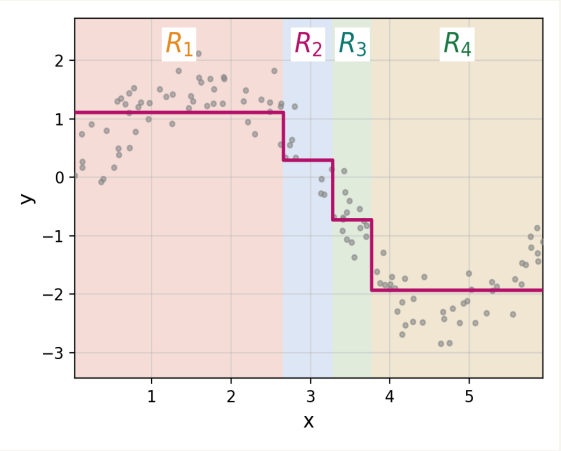
# Decision trees: piecewise-constant functions

A decision tree carves the space into a **partition of regions**: its **leaves**.




Classification




Regression

On each region $R_j$: what is the value of $f$ ?

**Classification:** the region's majority class.

**Regression:** the region's average, $\mathbf{x} \in R_j \Rightarrow f(\mathbf{x}) = \bar{y}_{R_j}$

# The best regression tree, formally

What is the **best regression tree** for a partition of $J$ regions ?

$$\min_{R_1,\dots,R_J} \sum_{j=1}^{J} \sum_{i:\, \mathbf{x}_i \in R_j} \left(y_i - \bar{y}_{R_j}\right)^2$$

The unknown is the **partition itself**, combinatorially many candidates. NP-hard problem.

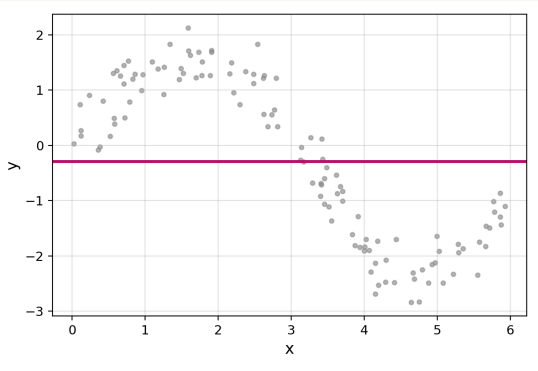
*Greedy* optimization (Algorithme *glouton*):

0. Start with root node = $\mathbb{R}^d$
1. find the best split at the *current* node
2. apply it to get two **children** nodes
3. repeat steps 1 and 2 on each child node

# CART algorithm

# Regression trees

What about **regression**: data $(\mathbf{x}_i, y_i)$ with $y_i \in \mathbb{R}$. What does a **tree** $f$ look like ?



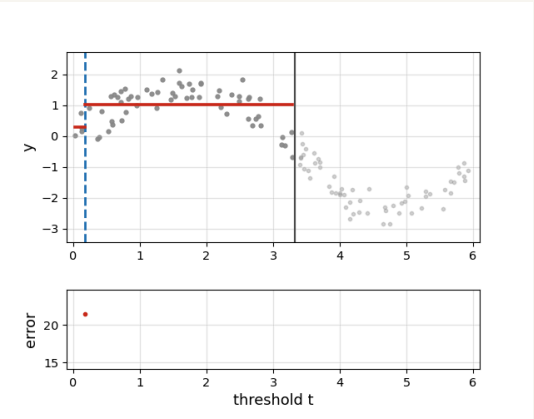

Start simple: fix **depth = 0**: no split allowed. What is the *best* possible tree ?

A **constant** ! One leaf, predicting the global average $\bar{y}$

**Depth 1**: one split allowed. We only have one variable $x$. **Where** do we cut it ?

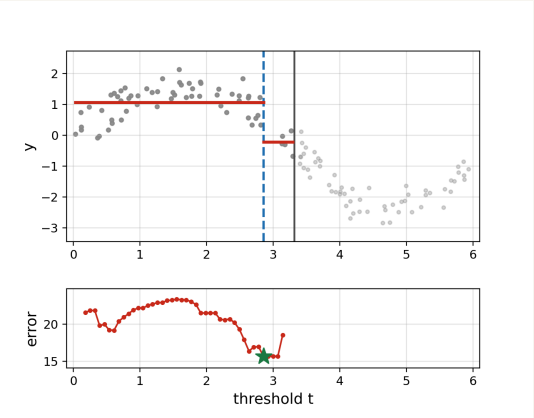
# The first split, live

# Recurse: left, then right

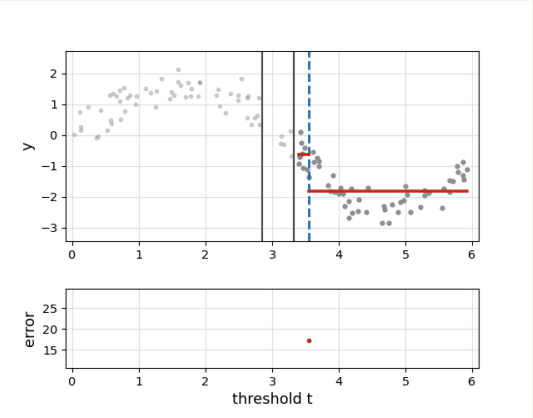



the same sweep, **inside the left** region

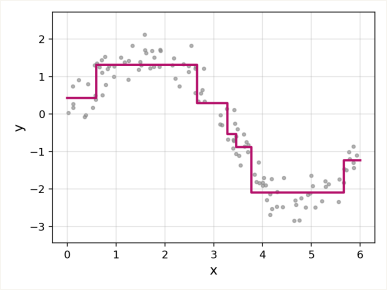
# Recurse: left, then right



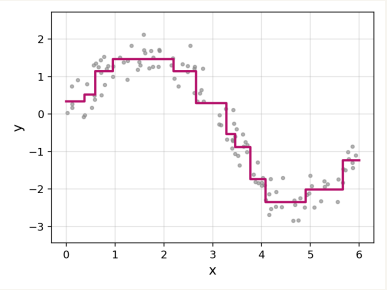

the same sweep, **inside the left** region



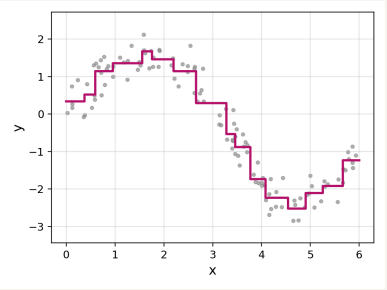

then **inside the right**




... and again: **depth 3**




**depth 5**



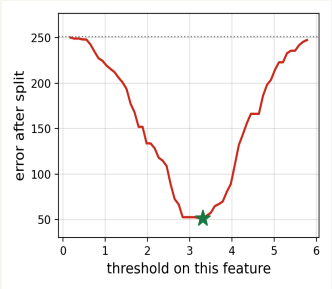

**depth 10**

# Several features ?

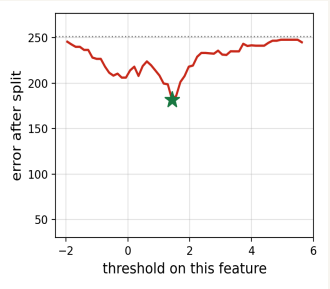
That was one variable. With $d$ features: **which one** gets the cut ?

CART is *greedy*: THE BEST ONE in terms of **decreasing prediction error**.

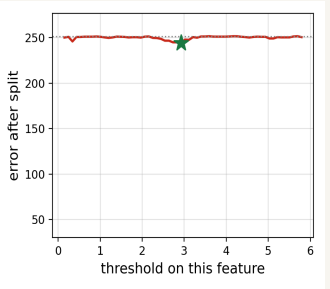
So we need to try to split **every single** feature:



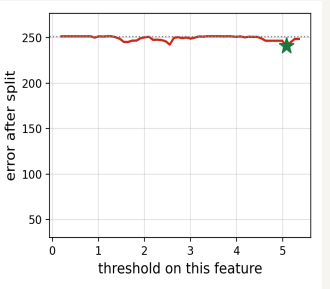

feature 1




feature 2




feature 3



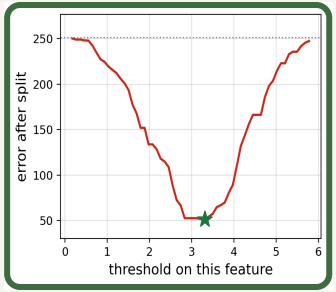

feature 4

# Several features ?

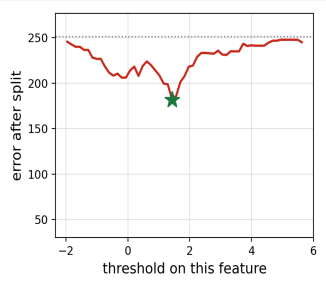
That was one variable. With $d$ features: **which one** gets the cut ?

CART is *greedy*: THE BEST ONE in terms of **decreasing prediction error**.

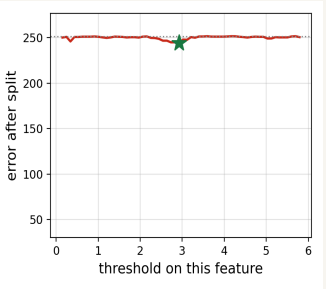
So we need to try to split **every single** feature:



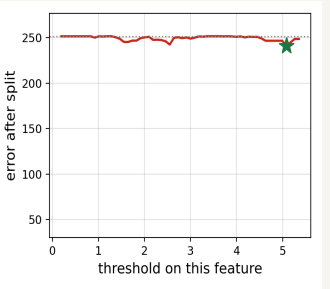

**feature 1**




feature 2




feature 3

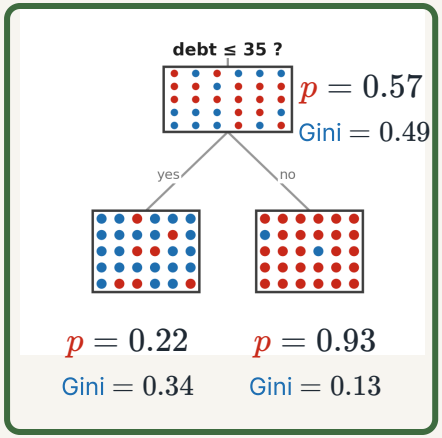



feature 4

Find the best **threshold** for every **feature**, select the (feature, threshold) pair with the **largest gain**.

# Scoring a classification split

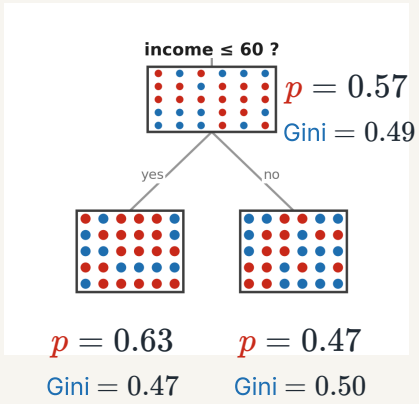
Back to classification: the **same parent node**, two candidate cut:




**split A**

average Gini $\approx 0.24$

$\Delta$ Gini $= -0.25$




**split B**

average Gini $\approx 0.48$

$\Delta$ Gini $= -0.01$

**Which split is better ?**

**Split A**: its children are *cleaner*.

How can we define a score for *cleanliness* ?

$p$ = the frequency of class $1$ in a leaf node

$$\mathbb{P}(y_1 \neq y_2) = 2p(1-p)$$

*Gini Impurity*

If we draw samples at random $\mathbb{P}(y_1 \neq y_2)$ must be low.

# CART, unified

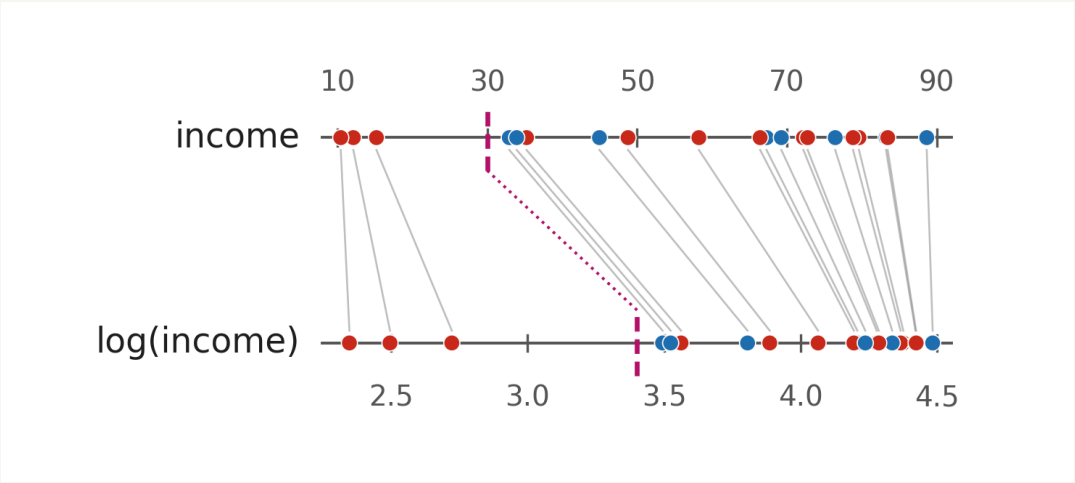
The same greedy recursion runs both classification and regression trees:

| | regression | classification |
|---|---|---|
| **split score** | children's squared error | children's Gini $2p(1-p)$ |
| **leaf output** | the leaf's average $\bar{y}$ | majority class / proportions |
| **the recursion** | scan every (feature, threshold), keep the largest gain, repeat on the children | |

CART algorithm: *Classification And Regresion Trees* (1984)

# Trees in practice

# Rescaling a feature ?




the split $\text{income} \leq 30$

Chapter 5 standardized the columns before any penalty. What happens to a **tree** if we use $\log(\text{income})$ instead ?

the split $\log(\text{income}) \leq 3.40$: the **same clients** on each side

$$x_j \leq t \iff \varphi(x_j) \leq \varphi(t) \quad \text{for any strictly increasing } \varphi \text{ (log, z-score, min-max)}$$

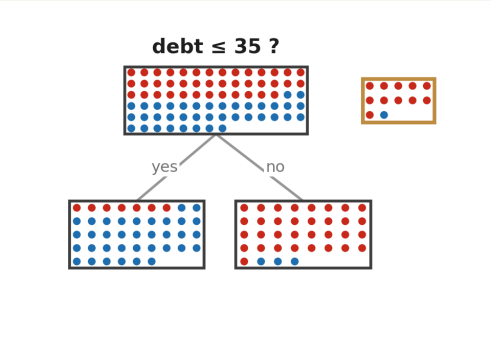
Same order, same splits, **same tree**: a tree needs no standardization.

# Missing values ?

Some clients left their **debt** blank. The root asks $\text{debt} \leq 35$: where do **they** go ?

| income (kDh) | debt (%) | default |
|---|---|---|
| 80 | 59 | 1 |
| 33 | 25 | 0 |
| 58 | ? | 1 |
| 72 | ? | 1 |
| ⋮ | ⋮ | ⋮ |

- debt missing: 12 of 90 clients

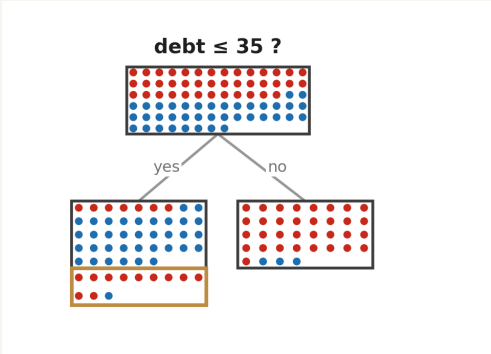


# Missing values ?

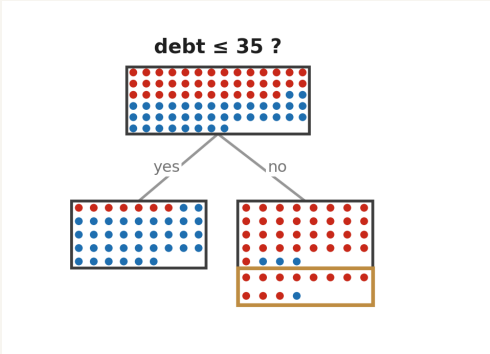
Some clients left their **debt** blank. The root asks $\text{debt} \leq 35$: where do **they** go ?

| income (kDh) | debt (%) | default |
|---|---|---|
| 80 | 59 | 1 |
| 33 | 25 | 0 |
| 58 | ? | 1 |
| 72 | ? | 1 |
| ⋮ | ⋮ | ⋮ |

- debt missing: 12 of 90 clients




missing → **left**: average Gini **0.33**

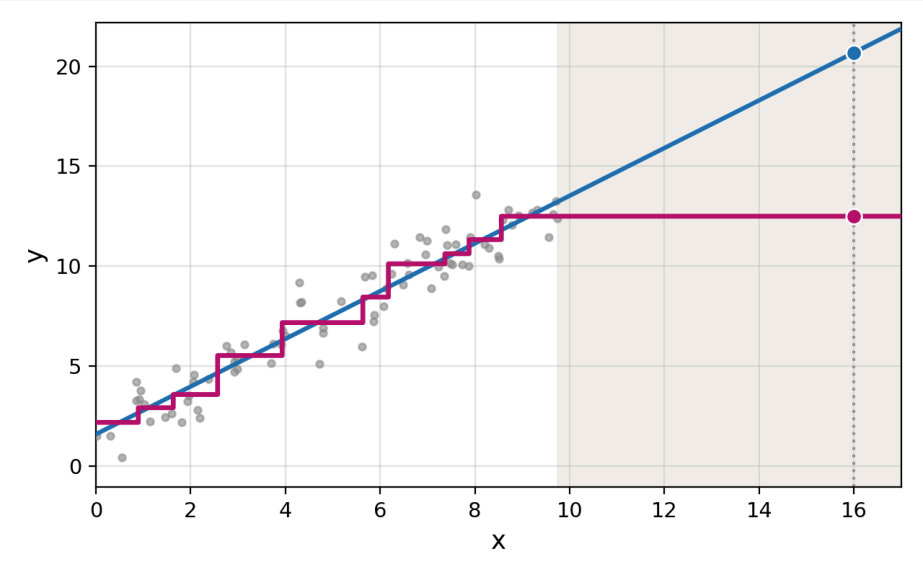



missing → **right**: average Gini **0.21**

The lower-loss side becomes the split's **default direction**, stored with it: a new client with a missing debt follows it.

No imputation: scikit-learn trees and random forests accept `NaN` directly, and boosting libraries do the same.

# Extrapolation ?




training data up to $x = 9.7$; the shaded region holds **no data**

A new point at $x = 16$ (dotted line): what do a **line** and a **tree** predict ?

**line**: $20.7$, it follows the trend

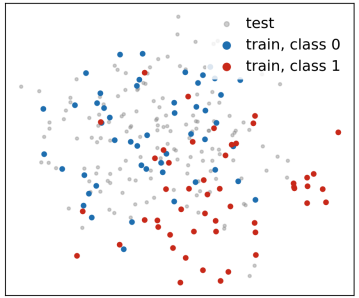
**tree**: $12.5$, exactly its value at $x = 9.7$

Past the last training value, a tree **freezes** at its boundary leaf's constant; a line keeps moving.

# Complexity of a decision tree

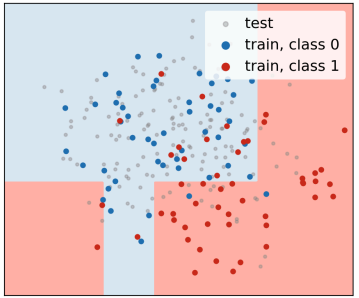
# Depth

Can a tree overfit ?

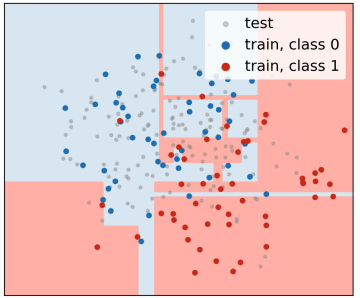



the data

**depth 2**




**depth 3**




**depth 10**

**train** / **test** error

**0.17** / **0.23**

**0.15** / **0.24**

**0.01** / **0.28**

What happens if when we increase depth further ?

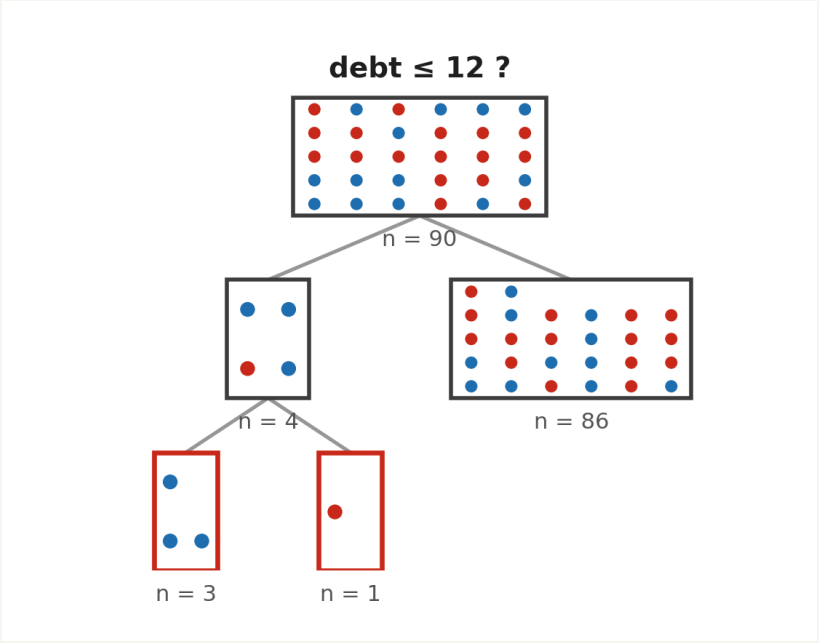
We keep splitting until **all leaves** are pure !

# Depth is not enough

So we should **restrict max depth** to avoid overfitting.

What about **this** depth-2 tree ?

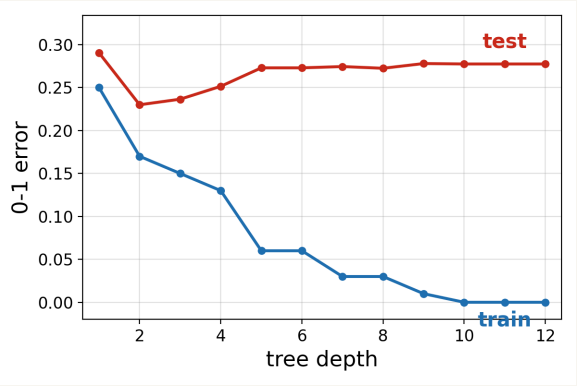
We need to avoid having very small leaves.

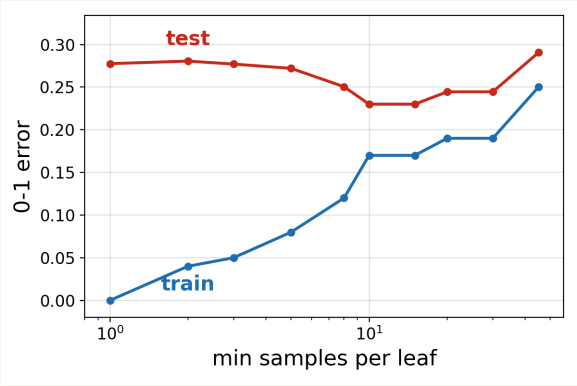



Another hyperparameter to restrict the possible splits: *minimum samples per leaf*

# U-curve

What happens to the **train error** when we increase each hyperparameter ?



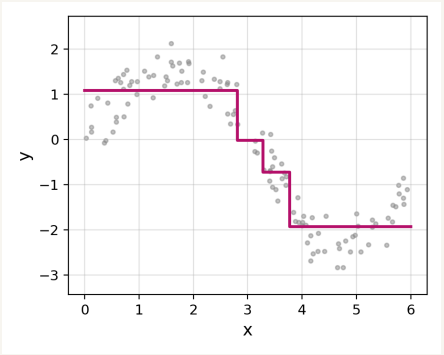

And the **test error** ?

These are (two of) the main hyperparameters of a Decision Tree

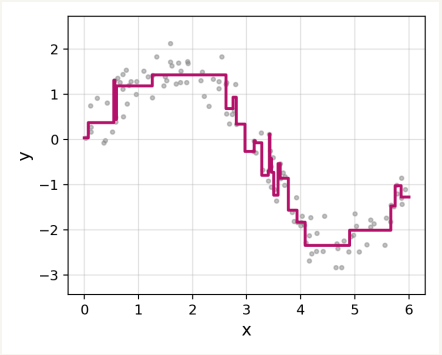
Chosen exactly as in chapter 5: a **grid**, **cross-validation**.

# Trees have high variance

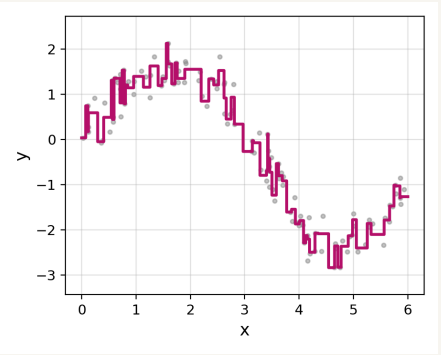
To fit anything interesting, a tree must be **deep**. What is the price ?




**depth 2**: train 0.23, test 0.29




**depth 5**: train 0.07, test 0.17




**depth 12**: train **0.03**, test **0.22**

The **useful** trees are the deep ones, and deep trees **overfit**: they have **high variance**.

# Bagging and random forests

# Bagging: the idea

We want to **reduce the variance**. How do we reduce the variance of a random variable ?

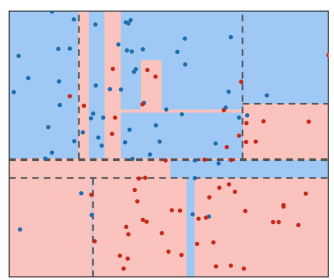
**Average** different estimates:, $\mathbb{V}(\bar{X}) = \sigma^2/n$. But average **what** ? We only have **one** dataset, one tree.

Chapter 4's **bootstrap**: resample the rows **with replacement**, fit one tree per replica.

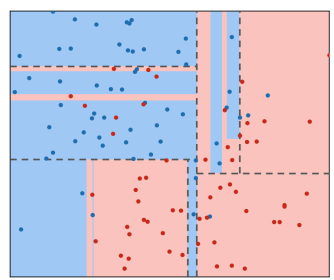
**original row indices**: 1 2 3 4 5 6

↙ ↘

**Bootstrap sample**: 2 **2** 5 1 6 4
→ tree $T_1$

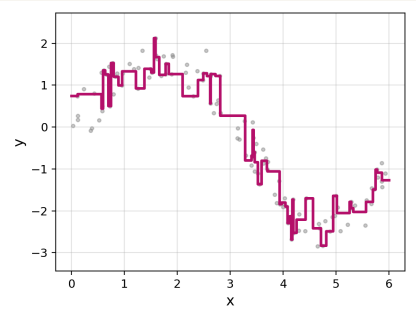


**Bootstrap sample**: 3 6 **6** 2 4 1
→ tree $T_2$

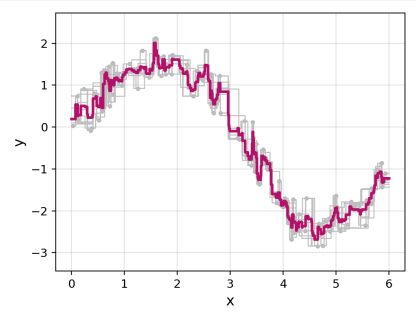


# Bagging: Bootstrap Aggregation

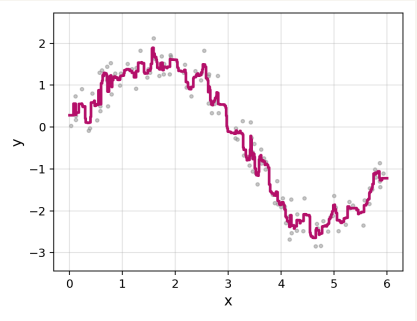
Draw $n$ rows with replacement, fit one **deep** tree; repeat B times; **average** the $B$ predictions.




**one** bootstrap tree ($B = 1$)




$B = 12$ **trees** (gray) and their **average**




the average of $B = 200$ trees

Even with B = 200, the variance is still high, why could that be?

# Random forests: decorrelate

Consider $B$ different trees $T_1, \ldots, T_B$ with variance $\sigma^2$ and pairwise correlation $\rho$. How much variance did averaging remove ?

$$\mathbb{V}\left(\frac{1}{B}\sum_{b=1}^{B} T_b\right) = ?$$

# Random forests: decorrelate

Consider $B$ different trees $T_1, \dots, T_B$ with variance $\sigma^2$ and pairwise correlation $\rho$. How much variance did averaging remove ?

$$\mathbb{V}\left(\frac{1}{B}\sum_{b=1}^{B} T_b\right) = \rho\,\sigma^2 + \frac{1-\rho}{B}\,\sigma^2$$

If the **correlation** $\rho$ is high, the variance never dies even if $B \to +\infty$ !

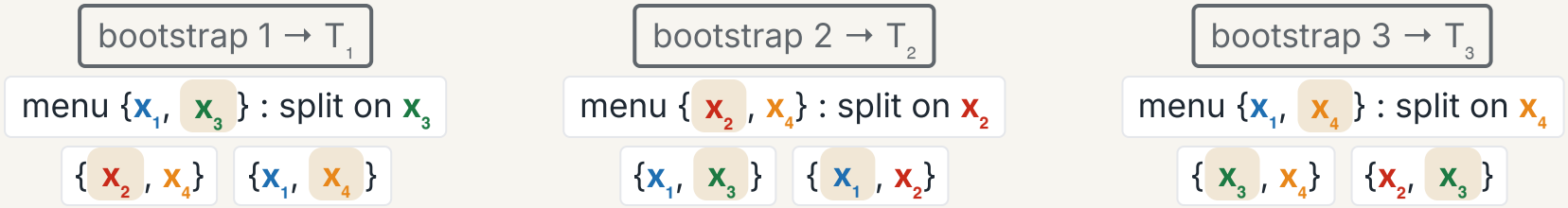
How do we make the trees **uncorrelated** ?

**Don't give them the same features**: for every split consider only a **random subset** of the features !

# The forest, one tree at a time

At **every split**, each node draws a fresh random subset of $m = 2$ of the $d = 4$ features.

**data** (n rows, features $x_1, x_2, x_3, x_4$) → three bootstraps, three trees grown **in parallel**

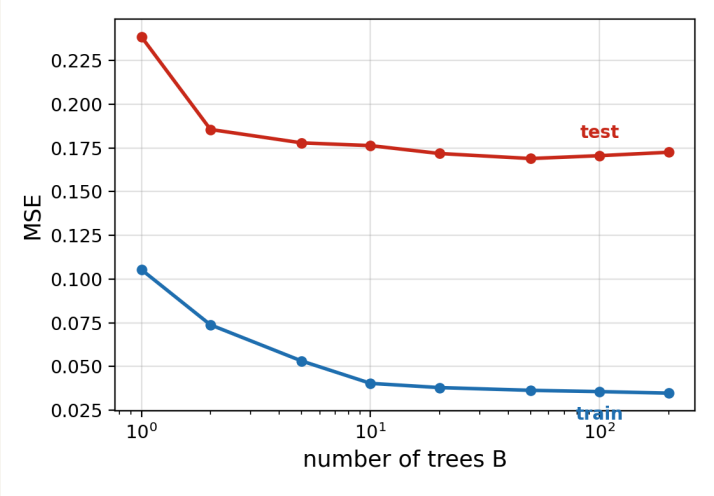



different random subsets → **different splits** → decorrelated trees

then **average**: $\hat{F}_B(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} T_b(\mathbf{x})$. That is a **random forest**.

# Number of trees

What should be the effect of a large $B$ on the test error ?




The error falls and **flattens**. $B$ is not a complexity dial; more trees reduce overfitting.

# What to remember

| | decision tree | bagging | random forest |
|---|---|---|---|
| **what** | one deep tree, axis-aligned regions | average of **B** trees, one per bootstrap replica | bagging + a **random feature menu** at every split |
| **fights** | bias: any partition, no assumption | the **variance** of a single tree | the **correlation** ρ left between trees |
| **dials** | max depth, min samples per leaf | B: not a complexity dial | + m features per split |
| **overfits** | depth (the U-curve) | not in B | not in B |

A deep tree has **low bias** and **high variance**. Averaging removes the variance **without** paying bias; decorrelating removes what averaging alone cannot: the floor $\rho\sigma^2$.

The out-of-bag error, variable importance, and the **other** way to combine trees (boosting): **chapter 7** of the lecture notes.

# Contents

1. Decision trees
2. CART algorithm
3. Trees in practice
4. Complexity of a decision tree
5. Bagging and random forests