

# Growing decision trees

STAT 471

November 2, 2021

# Rolling into a new unit!

✓ **Unit 1:** Intro to modern data mining

✓ **Unit 2:** Tuning predictive models

✓ **Unit 3:** Regression-based methods

**Unit 4:** Tree-based methods

**Unit 5:** Deep learning

**Lecture 1:** Growing decision trees

**Lecture 2:** Tree pruning and bagging

**Lecture 3:** Random forests

**Lecture 4:** Boosting

**Lecture 5:** Unit review and quiz in class

Homework 4 due the following **Wednesday**.

# Leaving the land of linearity

Most methods covered so far based on  $\hat{\beta}_0 + \hat{\beta}_1 X_1 + \cdots + \hat{\beta}_p X_p$  in some way:

- Linear regression
- Logistic regression
- Ridge, lasso, elastic net

Notable exception: K-nearest neighbors (recall Unit 2)

In Unit 4 we will leave the land of linearity.

# Entering the land of trees and forests

**Decision trees** (lectures 1 and 2) are predictive models based on recursively partitioning the feature space.

Their prediction rules can be nicely illustrated and are very **interpretable**.

However, trees are somewhat unstable and do not give the best prediction performance.

Nevertheless, trees can be used as building blocks for state-of-the-art prediction performance:

- **Random forests** (lecture 3)
- **Boosting** (lecture 4)



Predicting baseball players' salaries based on years played and number of hits in the previous year.

# Tree-based models versus linear models

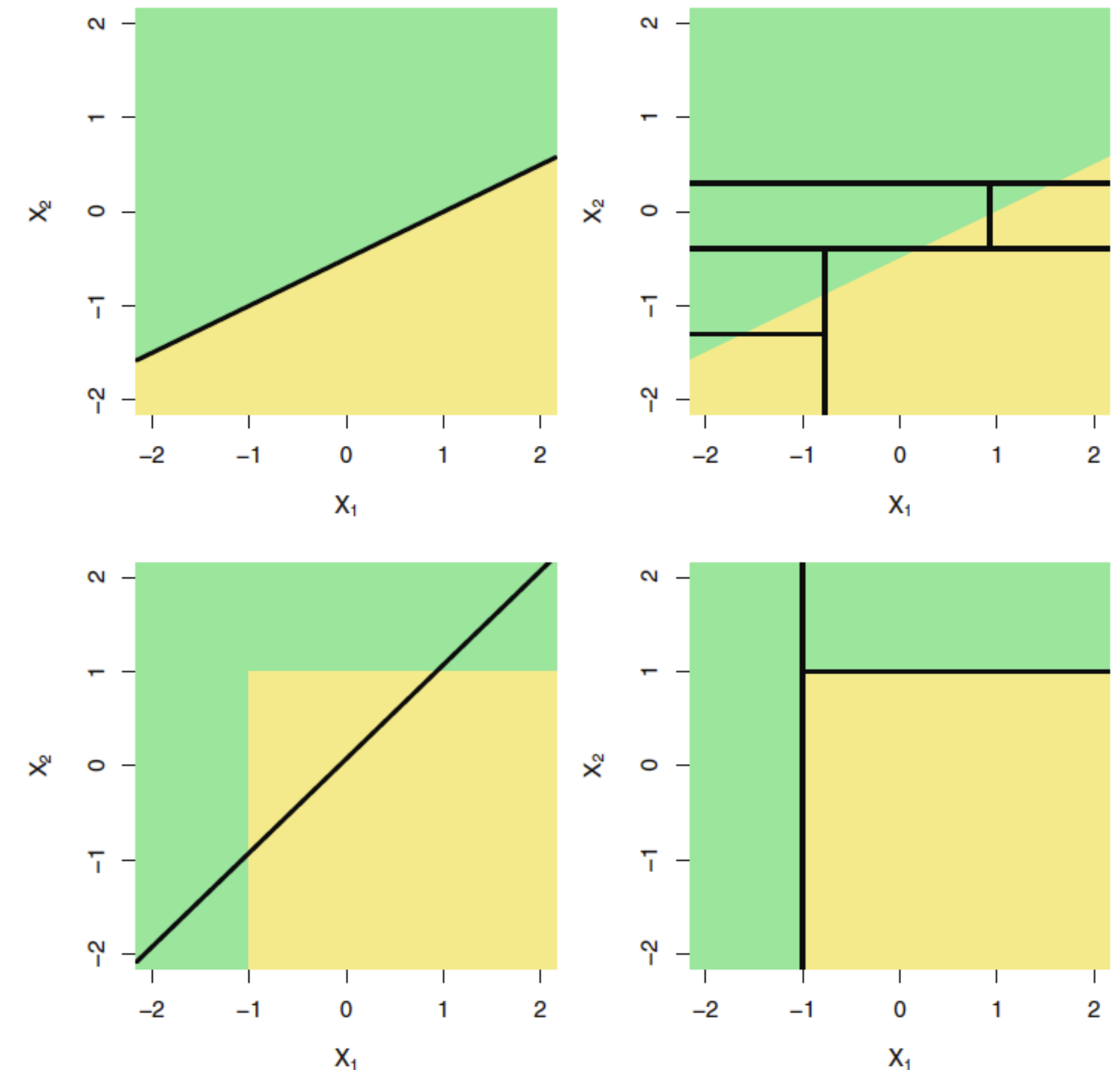
## Which perform better?

Neither tree-based nor linear models dominate the other.

Each prediction method works better when the underlying trend in the data matches its modeling choice.

E.g. for classification:

- Linear model → linear decision boundary
- Decision tree → unions of rectangles



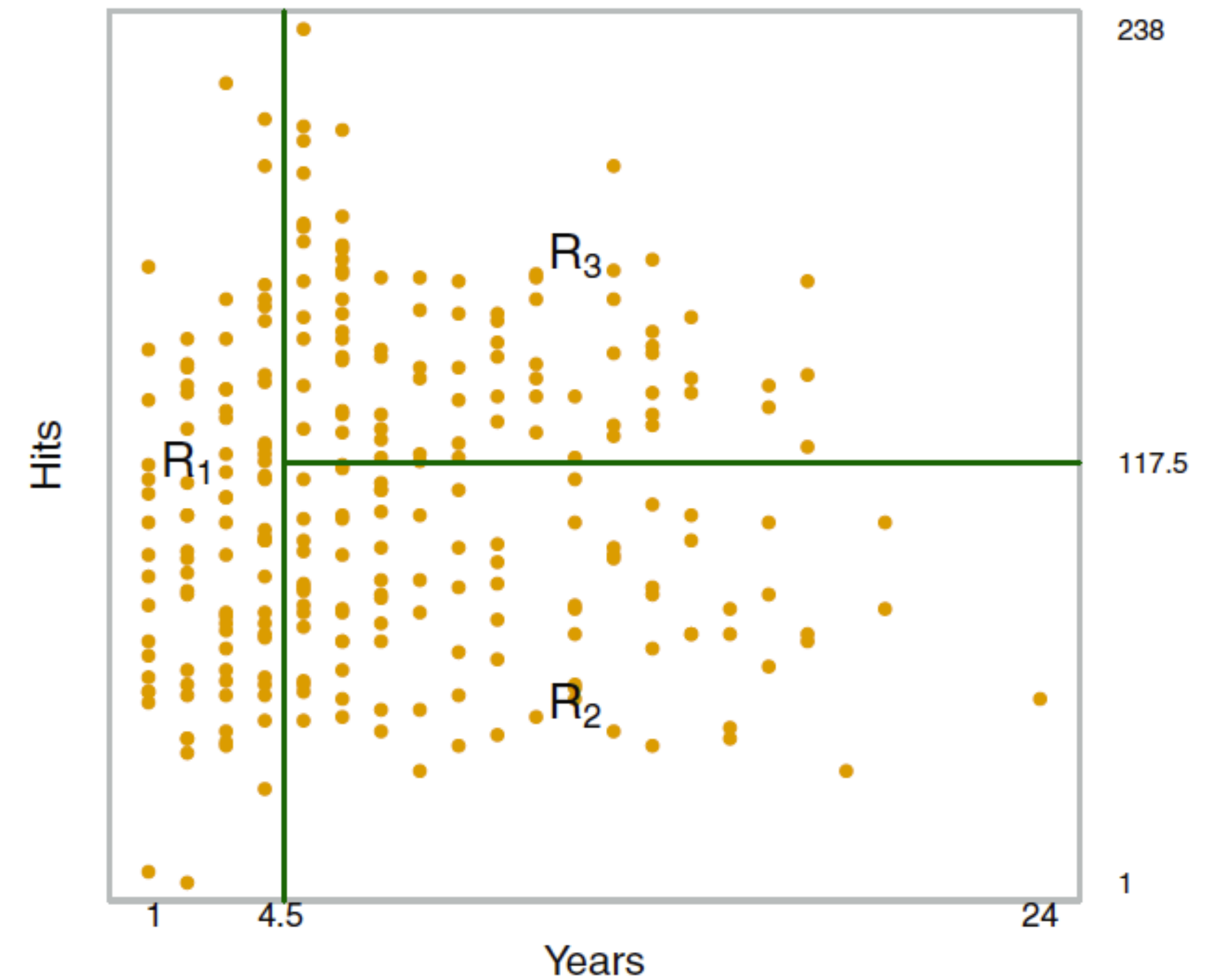
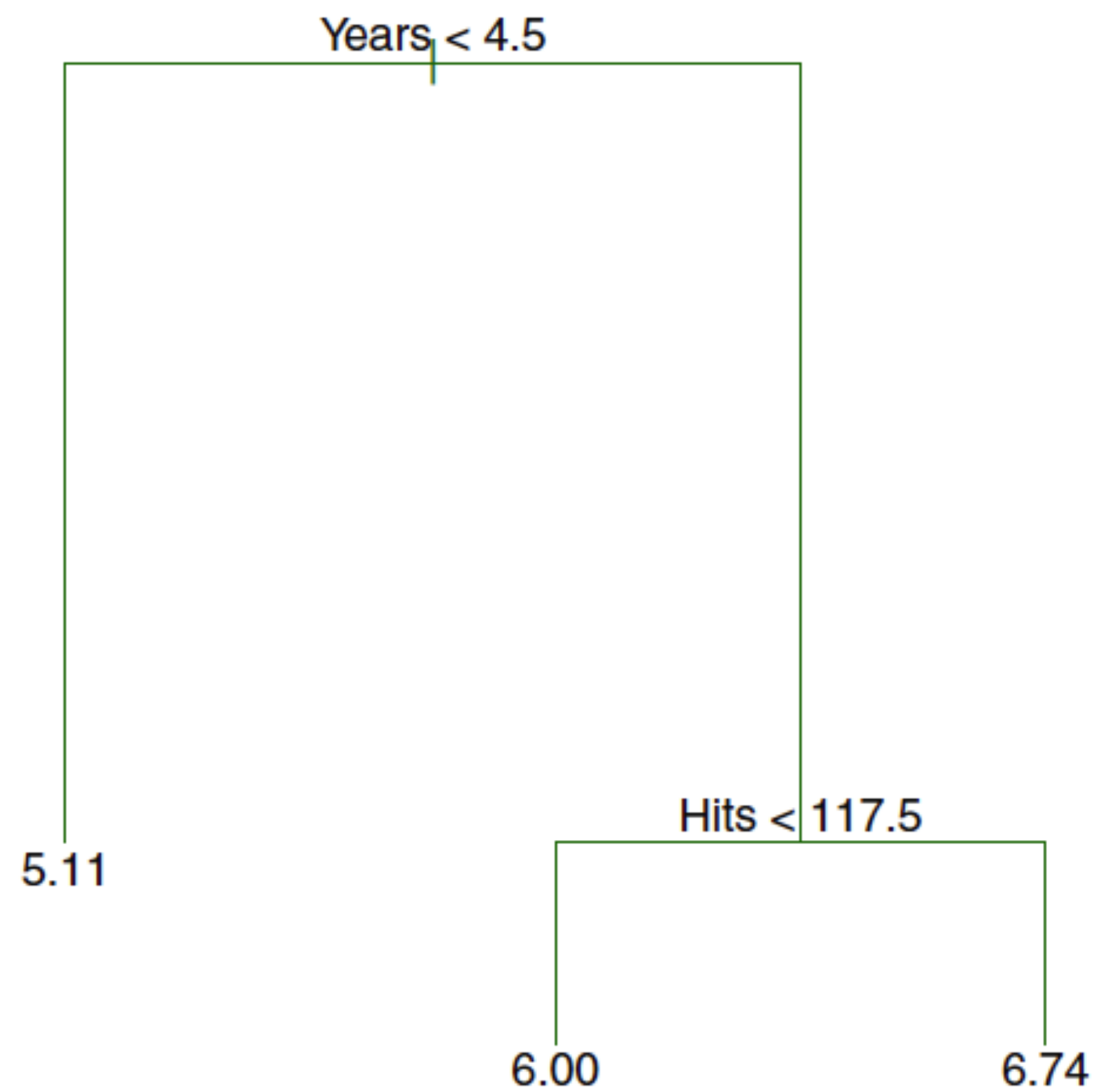
Classification based on two features  
(colors indicate the two classes).

# Hitters data

Major League Baseball Data from the 1986 and 1987 seasons.

- Observations: 322 MLB players
- Response: Salary (1987 annual salary on opening day in thousands of dollars)
- Features: Assists, AtBat,...,Hits,...,Years (19 total)

# Tree $\iff$ Partition into nested, axis-aligned rectangles





# Mathematical expression of the prediction rule

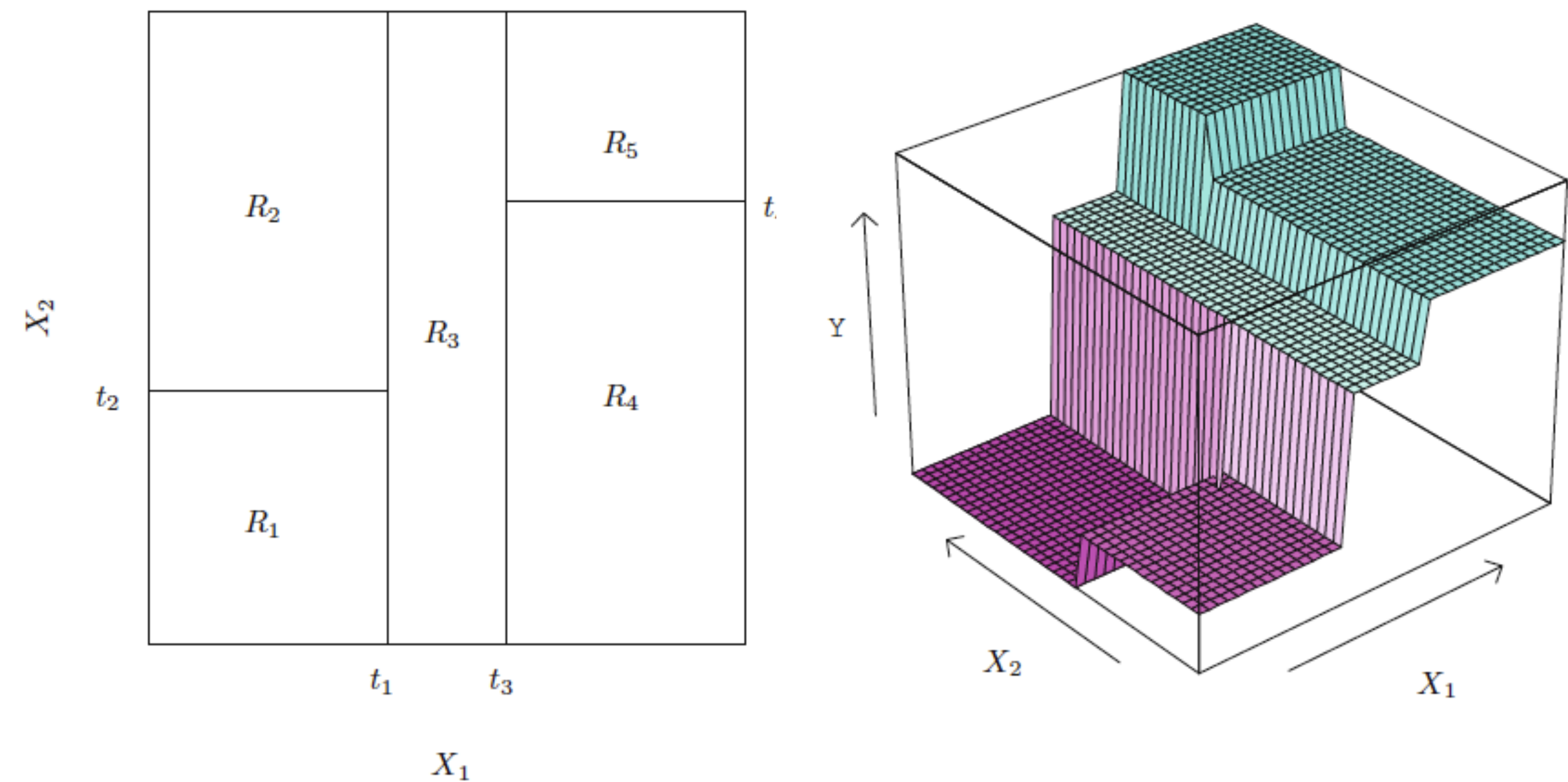
A trained tree consists of:

- $M$  regions  $\hat{R}_1, \dots, \hat{R}_M$
- response values  $\hat{c}_1, \dots, \hat{c}_M$

For a new feature vector  $X^{\text{test}}$ , predict the constant value  $\hat{c}_m$  for region  $\hat{R}_m$ :

$$\hat{y}^{\text{test}} = \sum_{m=1}^M \hat{c}_m \cdot I(X^{\text{test}} \in \hat{R}_m).$$

(continuous or categorical response)





# Partitioning for continuous and categorical features

Suppose we partition on  $X_j$ .

- If  $X_j$  is continuous, we just find a split point  $s$  and split into  $\{X : X_j < s\}$  and  $\{X : X_j \geq s\}$ .
- If  $X_j$  is categorical, e.g. with levels  $\{a, b, c, d, e\}$ , then we need to split the levels into two groups, e.g.  $\{a, c\}$  and  $\{b, d, e\}$ , giving the partitions  $\{X : X_j \in \{a, c\}\}$  and  $\{X : X_j \in \{b, d, e\}\}$ .

# Training a regression tree

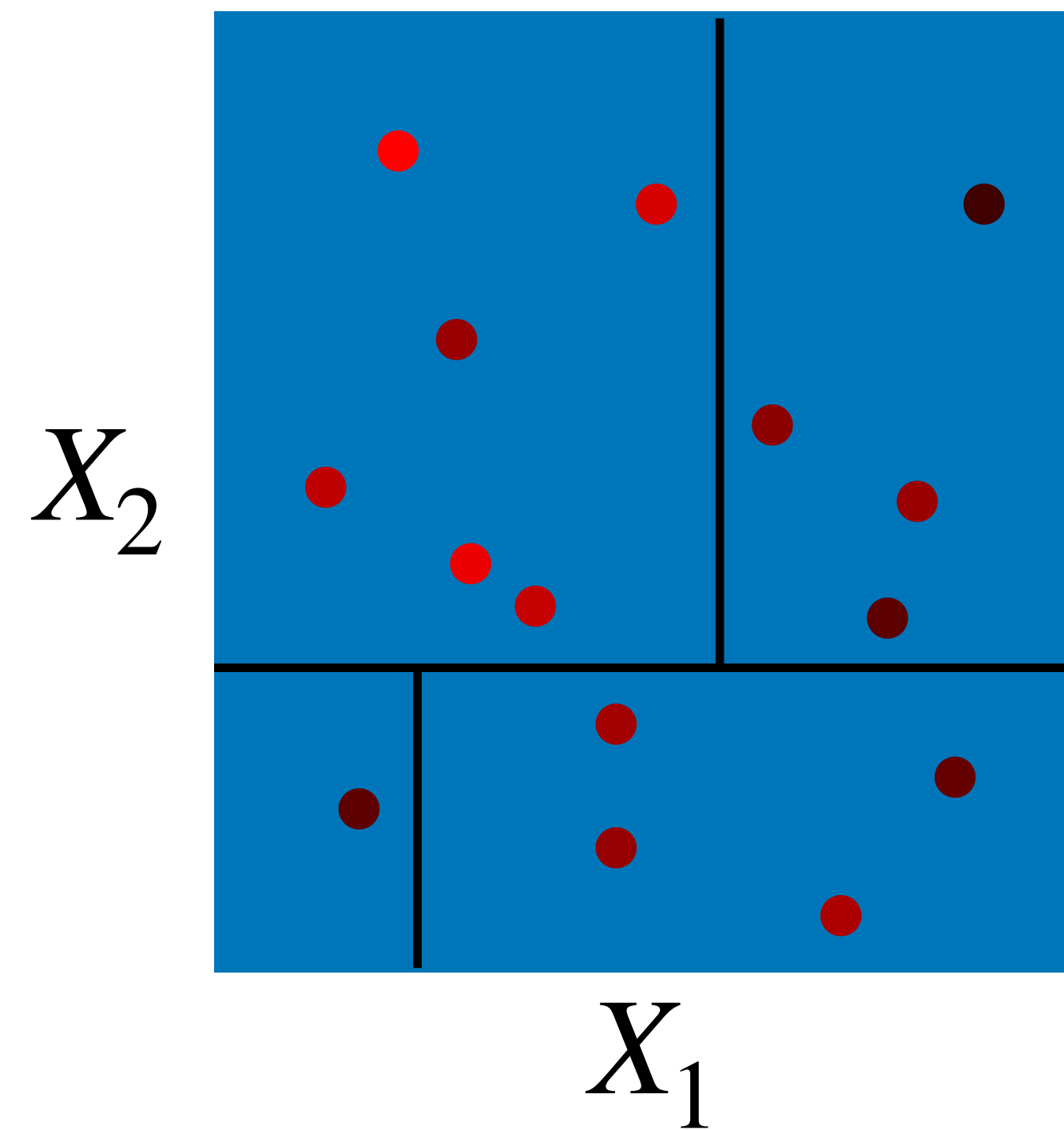
## The squared error objective

As usual, we are given a training dataset  $(X_1, Y_1), \dots, (X_n, Y_n)$ .

For a fixed  $M$ , we seek rectangles  $\hat{R}_1, \dots, \hat{R}_M$  and values  $\hat{c}_1, \dots, \hat{c}_M$  to minimize the **residual sum of squares**:

$$\hat{R}_1, \dots, \hat{R}_M, \hat{c}_1, \dots, \hat{c}_M = \arg \min_{R_1, \dots, R_M, c_1, \dots, c_M} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2;$$

$$\hat{Y}_i = \sum_{m=1}^M c_m \cdot I(X_i \in R_m).$$



# Training a regression tree

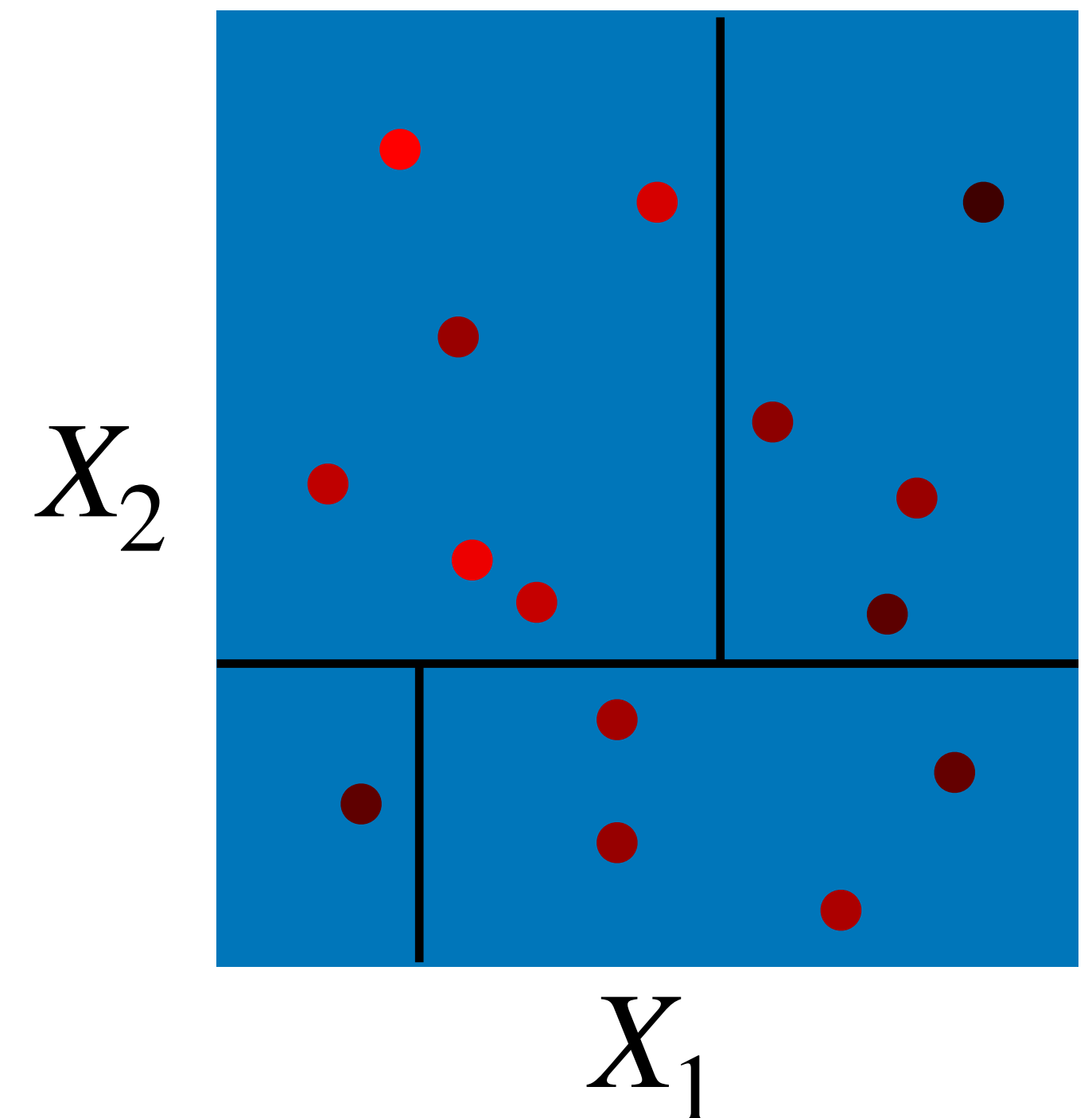
Optimal  $\hat{c}_m$  given  $\hat{R}_m$

First let's consider a simpler problem, where rectangles  $\hat{R}_1, \dots, \hat{R}_M$  are given:

$$\hat{c}_1, \dots, \hat{c}_M = \arg \min_{c_1, \dots, c_M} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2; \quad \hat{Y}_i = \sum_{m=1}^M c_m \cdot I(X_i \in \hat{R}_m).$$

We're fitting a constant to each region, so the solution is

$$\hat{c}_m = \text{mean} \left( \{Y_i : X_i \in \hat{R}_m\} \right).$$



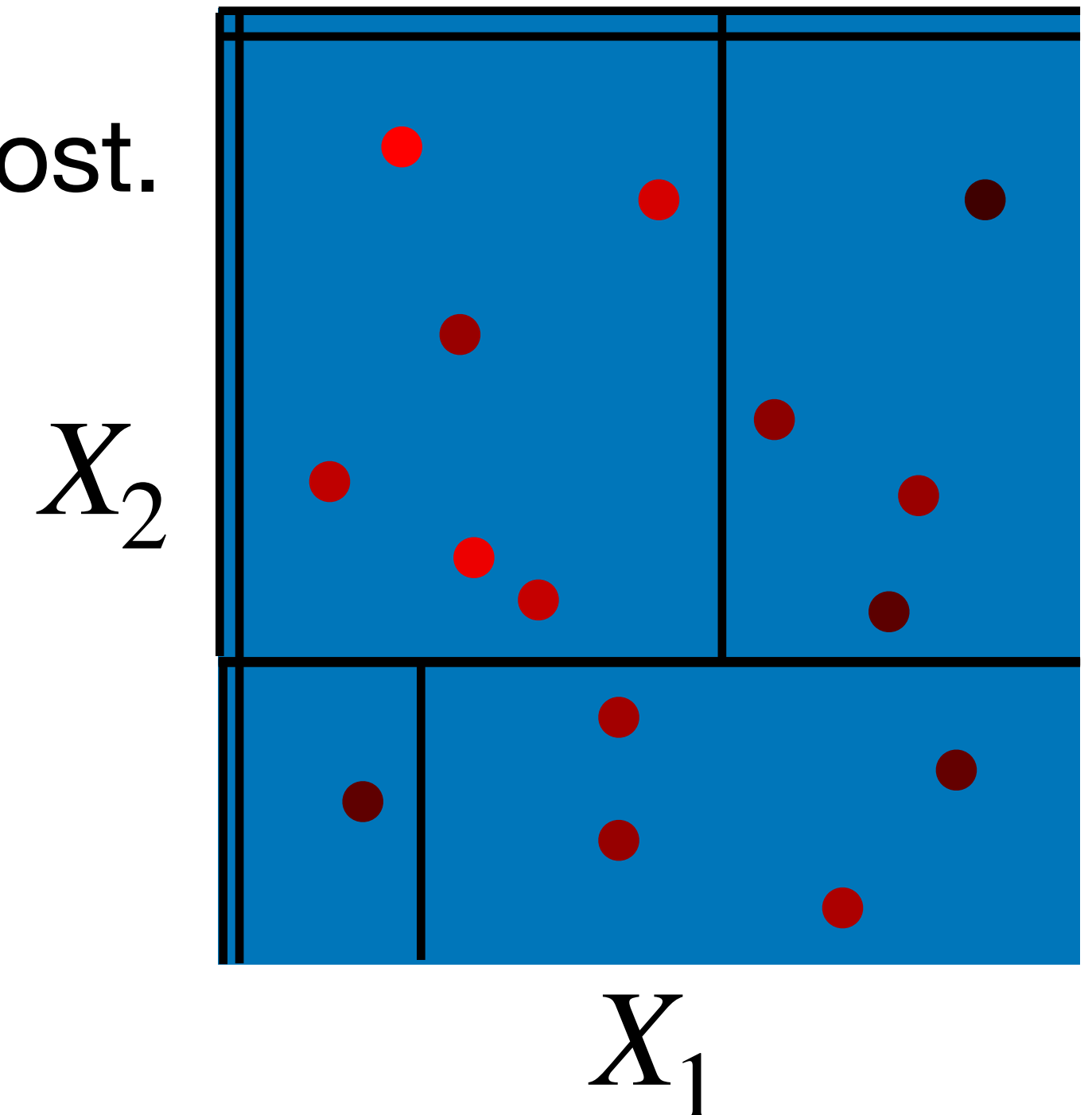
# Training a regression tree

Finding the rectangles  $\hat{R}_m$

The optimal set of rectangles is computationally intractable to find. In practice, we employ a **greedy top-down algorithm**:

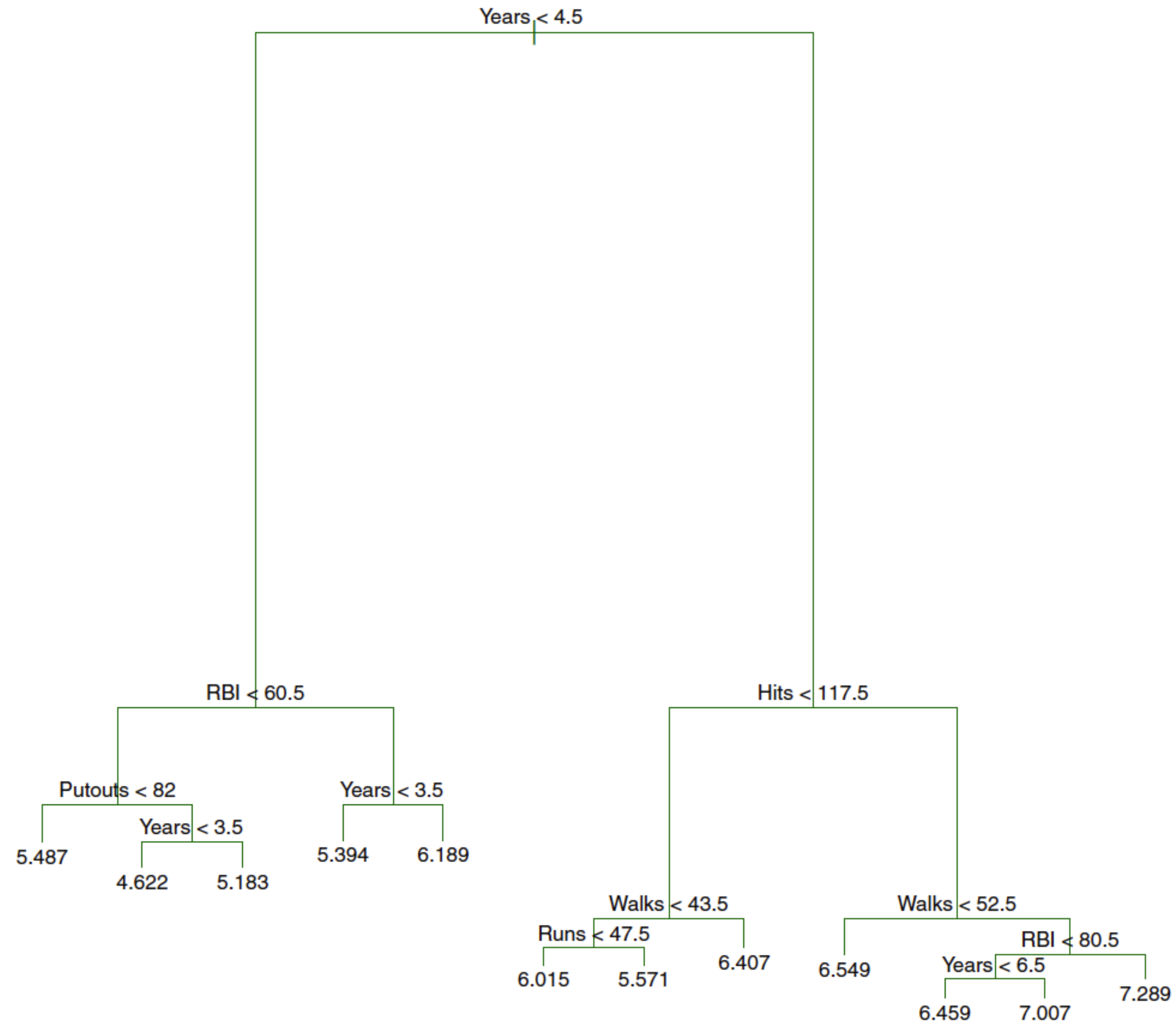
greedy bc  $\emptyset$   
think about future  
splits

1. Fit constant model to the entire space and calculate RSS.
2. Find split of the whole region that decreases RSS the most.
3. Find next split that decreases the RSS the most.
4. Repeat until there are  **$M$  regions**.



# Training a regression tree

## Final output



# Training a classification tree

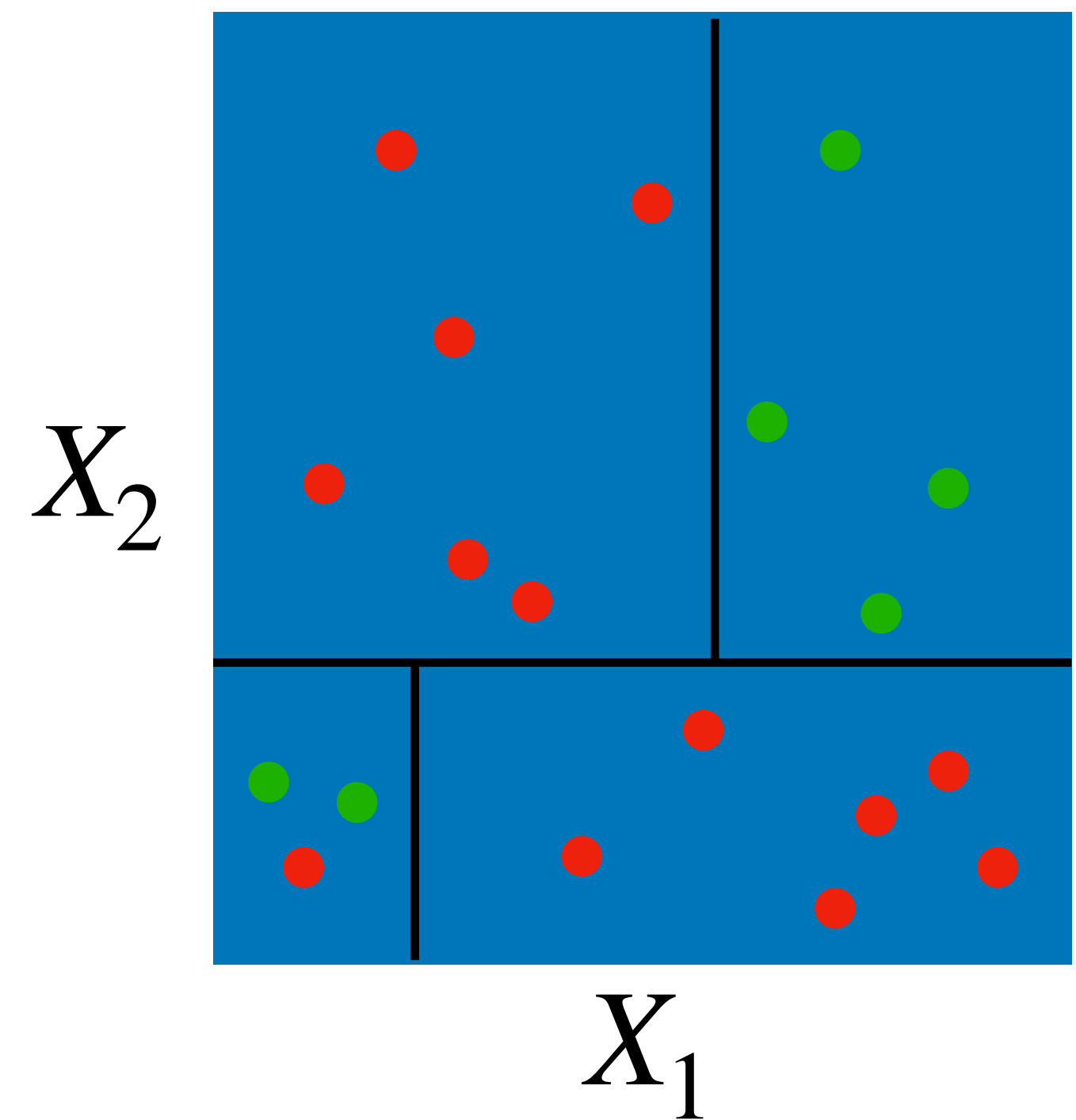
## The misclassification error objective

As usual, we are given a training dataset  $(X_1, Y_1), \dots, (X_n, Y_n)$ , where the response  $Y$  is binary.

For a fixed  $M$ , we seek rectangles  $\hat{R}_1, \dots, \hat{R}_M$  and values  $\hat{c}_1, \dots, \hat{c}_M$  to minimize the misclassification loss:

$$\hat{R}_1, \dots, \hat{R}_M, \hat{c}_1, \dots, \hat{c}_M = \arg \min_{R_1, \dots, R_M, c_1, \dots, c_M} \frac{1}{n} \sum_{i=1}^n I(Y_i \neq \hat{Y}_i);$$

$$\hat{Y}_i = \sum_{m=1}^M c_m \cdot I(X_i \in R_m).$$



# Training a classification tree

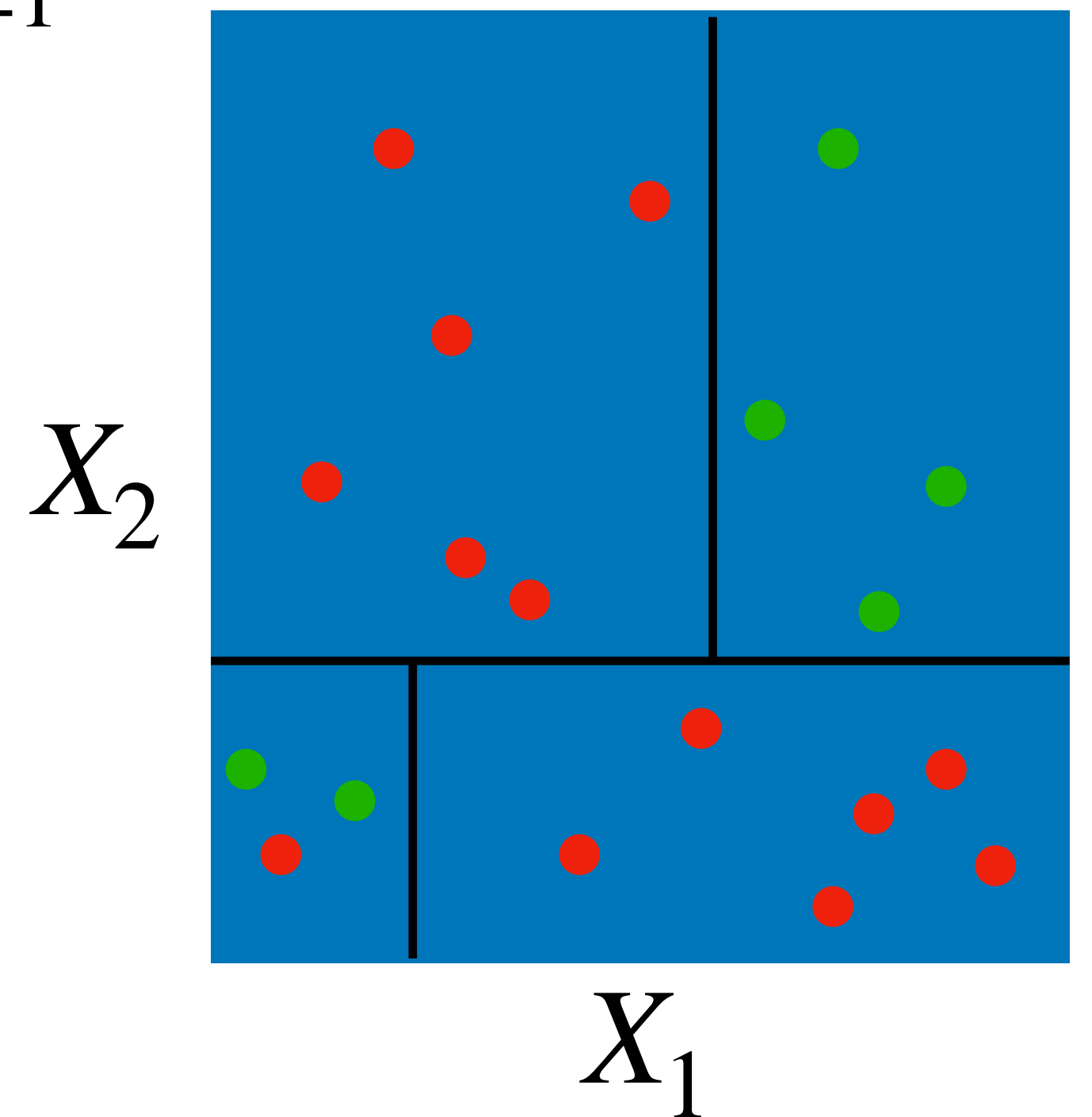
Optimal  $\hat{c}_m$  given  $\hat{R}_m$

First let's consider a simpler problem, where rectangles  $\hat{R}_1, \dots, \hat{R}_M$  are given:

$$\hat{c}_1, \dots, \hat{c}_M = \arg \min_{c_1, \dots, c_M} \frac{1}{n} \sum_{i=1}^n I(Y_i \neq \hat{Y}_i); \quad \hat{Y}_i = \sum_{m=1}^M c_m \cdot I(X_i \in \hat{R}_m).$$

We're fitting the same category to each region, so the solution is the **majority vote**:

$$\hat{c}_m = \text{mode} \left( \{Y_i : X_i \in \hat{R}_m\} \right).$$





# Training a classification tree

Finding the rectangles  $\hat{R}_m$

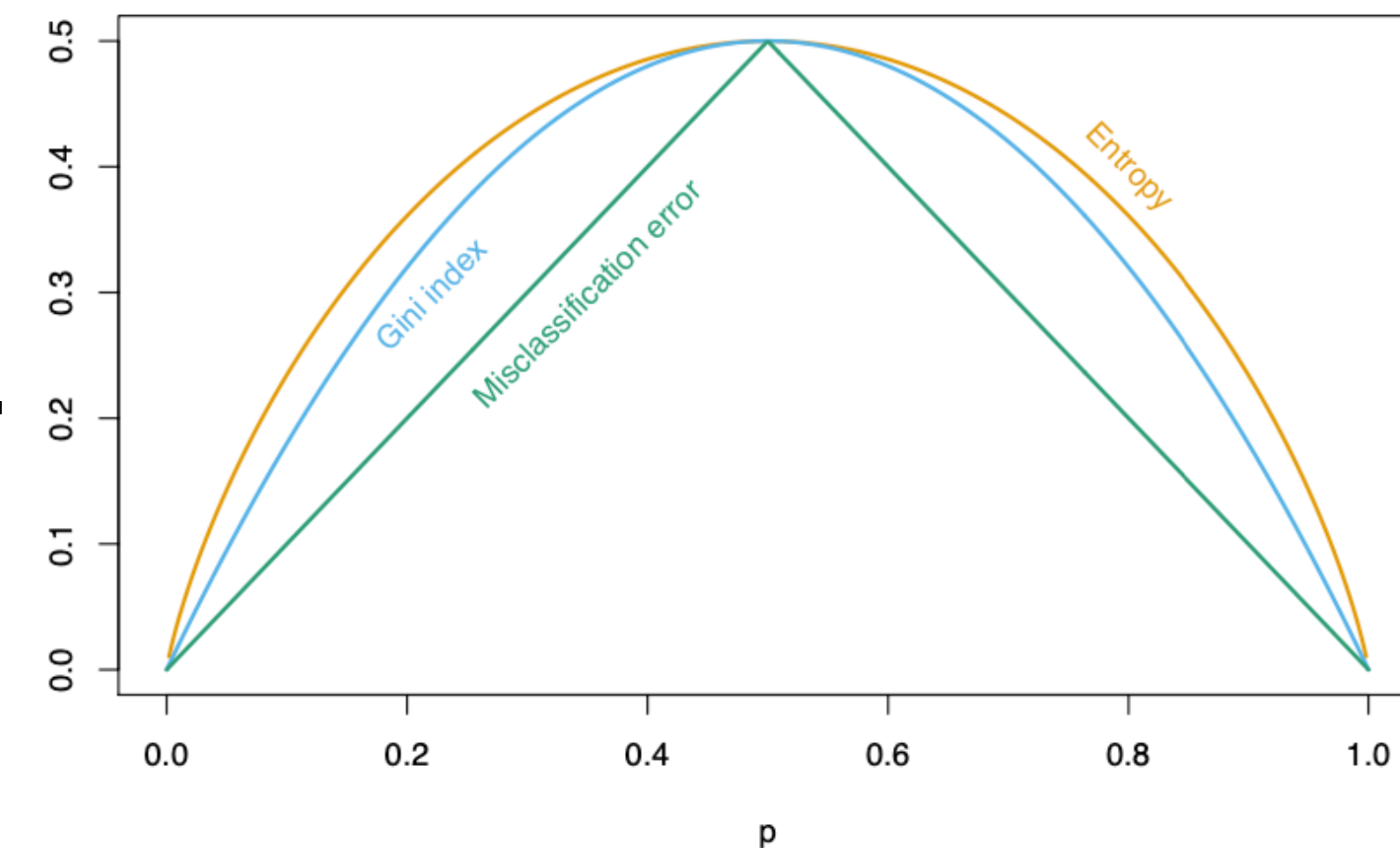
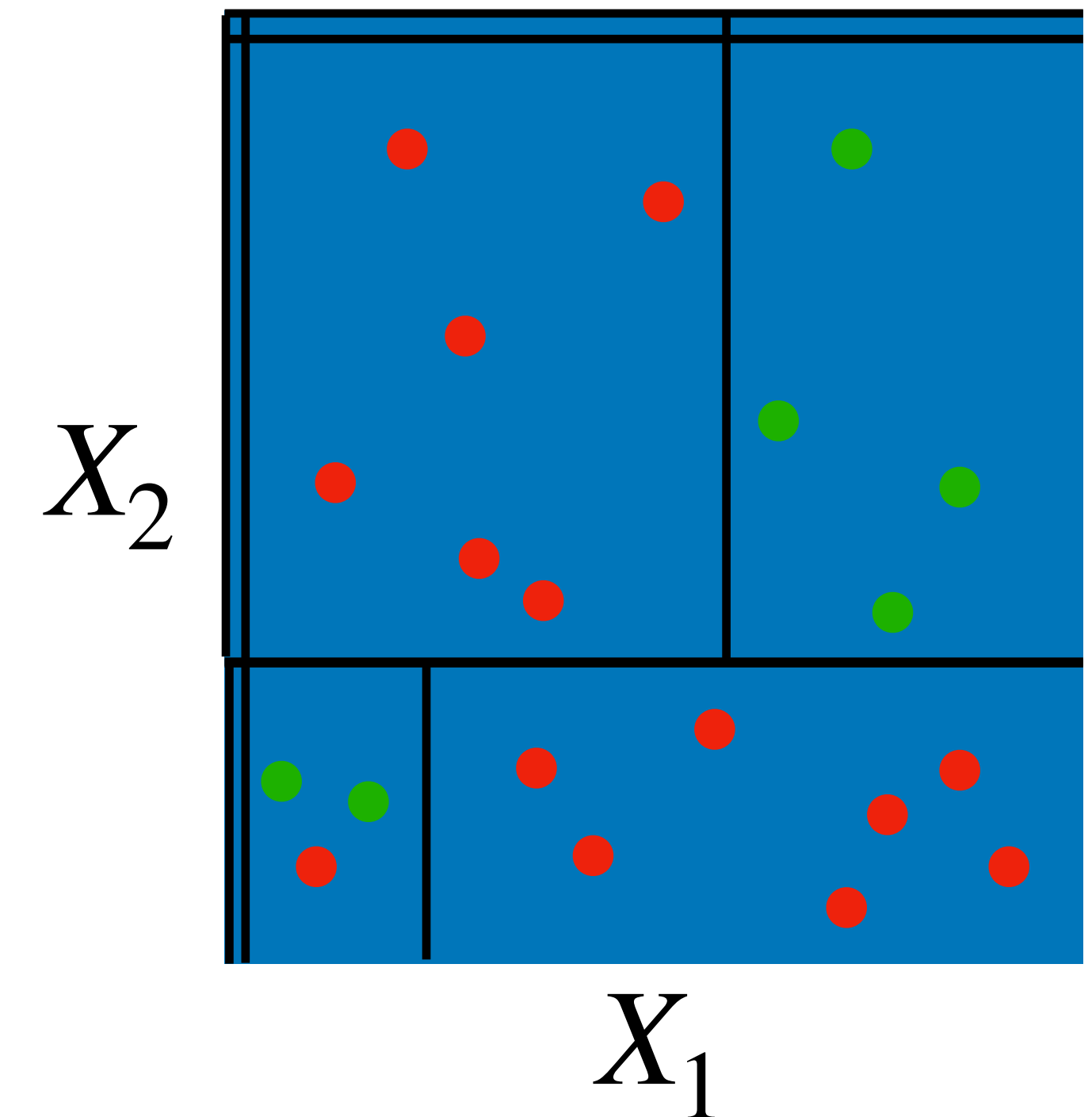
We use a very similar greedy recursive splitting algorithm.

Let  $\hat{p}_m$  be proportion of class 1 in region  $m$ . The misclassification error in that region is  $\min(\hat{p}_m, 1 - \hat{p}_m)$ .

Misclassification error not sensitive enough to find good split points at each step; instead, evaluate impurity using

$$\text{Gini index} = 2\hat{p}_m(1 - \hat{p}_m).$$

Find split that minimizes the total impurity of the regions.

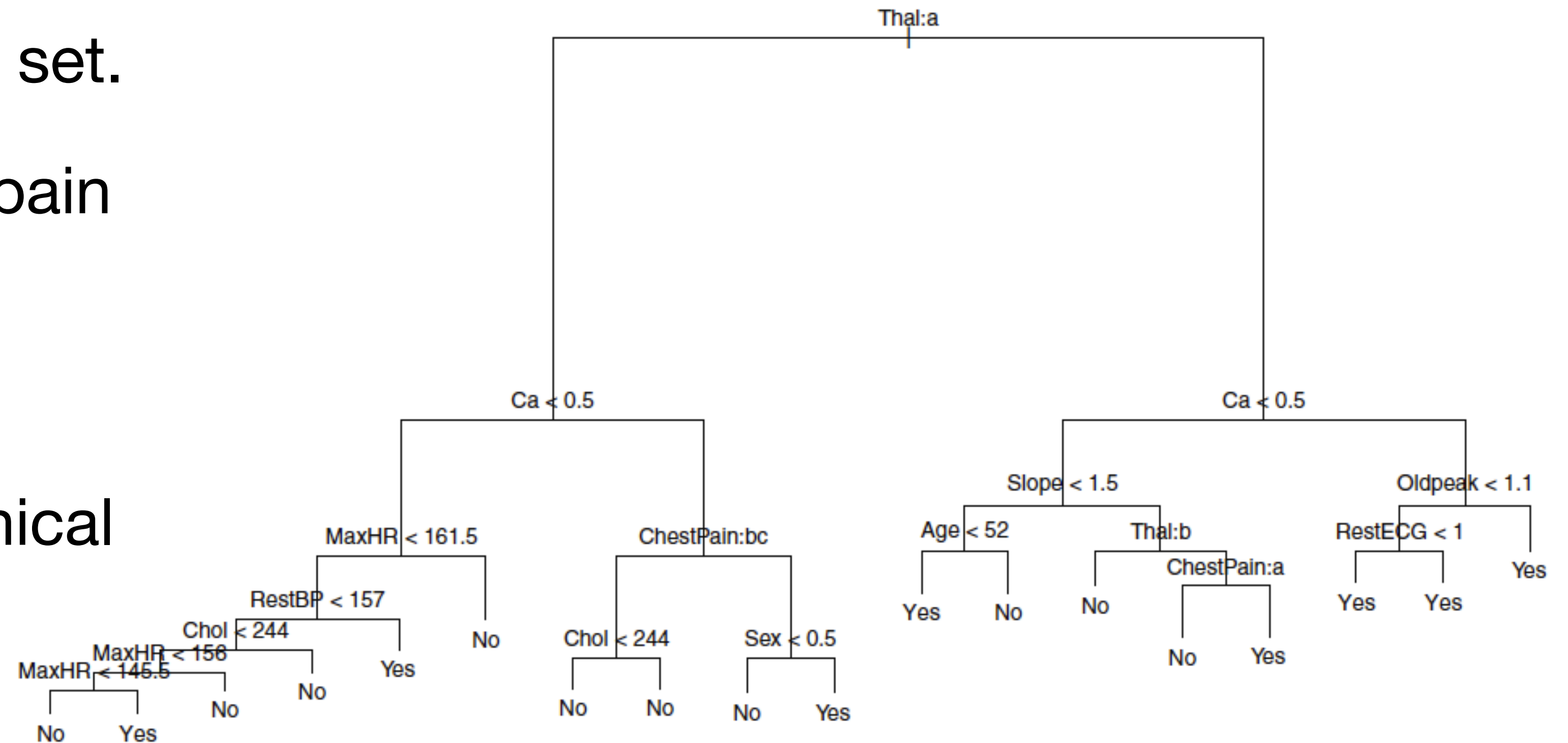


# Training a classification tree

## Final output

Example: Heart disease data set.

- 303 patients with chest pain
- Binary response HD (heart disease)
- 13 demographic and clinical features



Note: Classification trees extend seamlessly to more than two classes!

# Summary

- Decision trees partition the feature space into axis-aligned nested rectangles, producing a constant prediction for feature vectors in each rectangle.
- Decision trees are built by recursively choosing
  - The optimal rectangle to split
  - The optimal feature to split that rectangle on
  - The optimal split-point for that feature
- Regression and classification trees aim to minimize squared error and misclassification losses, respectively.

