Growing decision trees

STAT 4710

October 27, 2022

Rolling into a new unit!

- **Unit 1:** R for data mining
- **Unit 2:** Prediction fundamentals
- **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

Leaving the land of linearity

Leaving the land of linearity

Most methods covered so far based on $\widehat{\beta}_0 + \widehat{\beta}_1 X_1 + \cdots + \widehat{\beta}_{p-1} X_{p-1}$:

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

Leaving the land of linearity

Most methods covered so far based on $\widehat{\beta}_0 + \widehat{\beta}_1 X_1 + \cdots + \widehat{\beta}_{p-1} X_{p-1}$:

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

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

Leaving the land of linearity

Most methods covered so far based on $\widehat{\beta}_0 + \widehat{\beta}_1 X_1 + \cdots + \widehat{\beta}_{p-1} X_{p-1}$:

- 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.

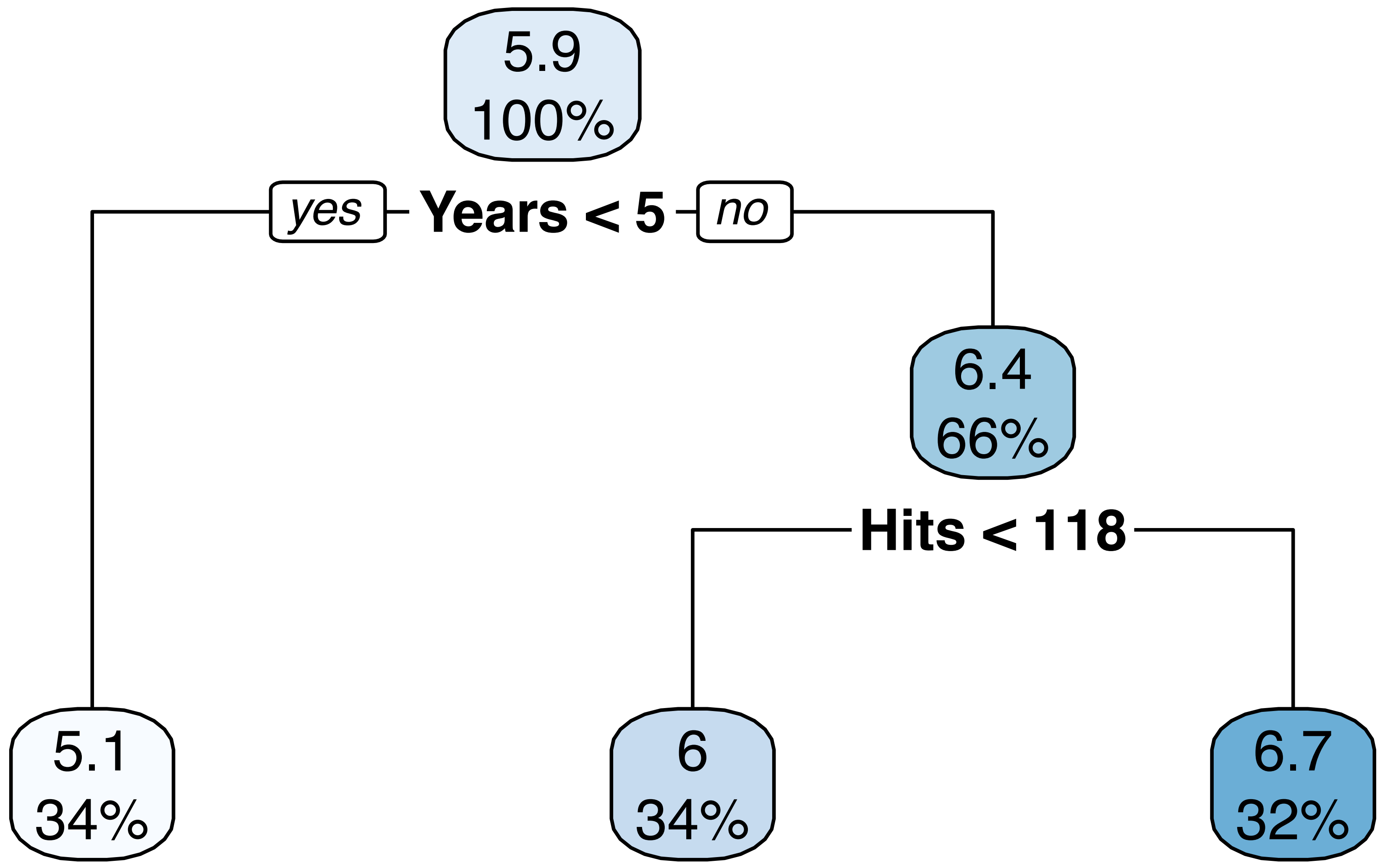
Entering the land of trees and forests

Entering the land of trees and forests

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

Entering the land of trees and forests

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

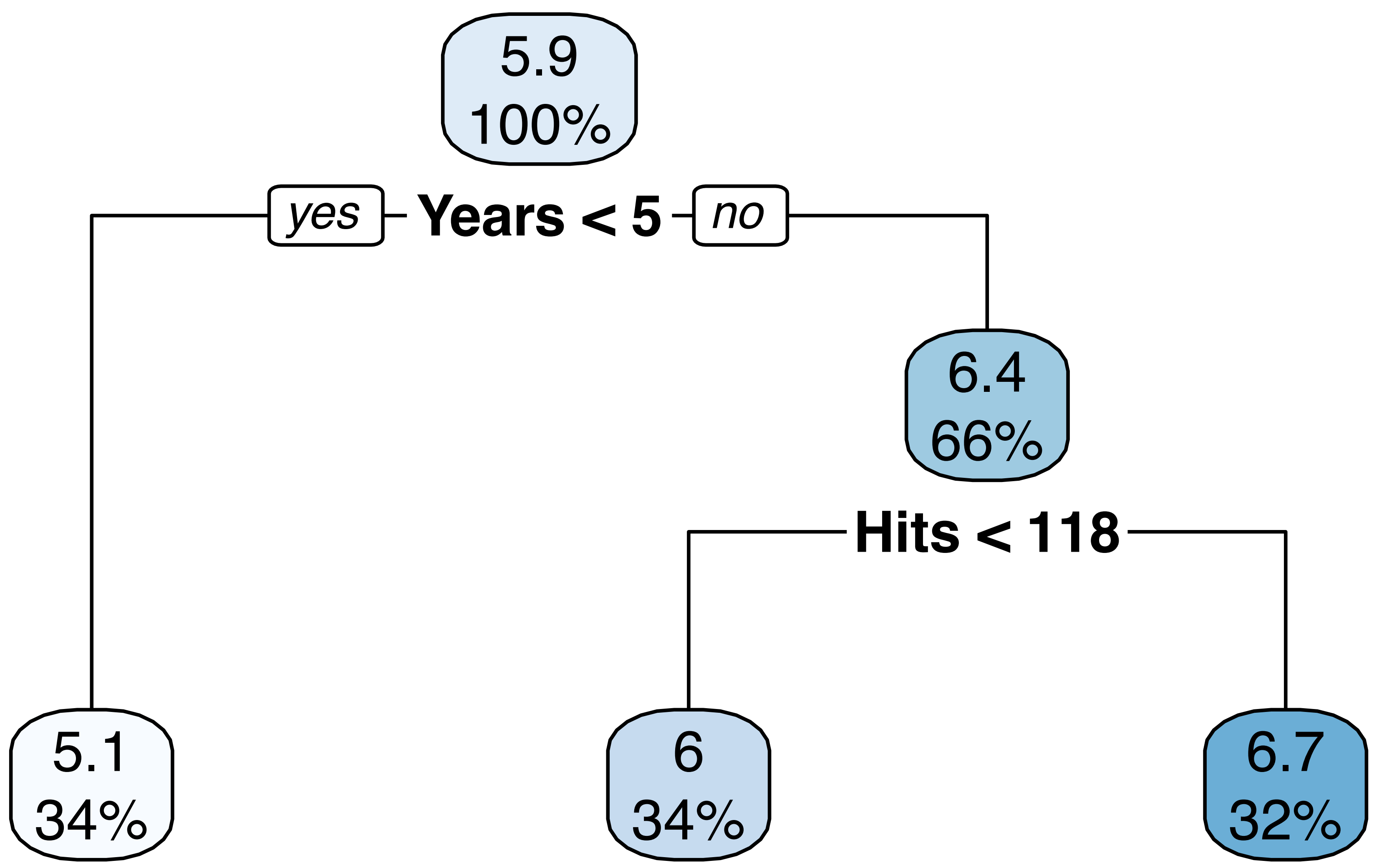


Predicting baseball players’ salaries based on years played and number of hits in the previous year.

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.



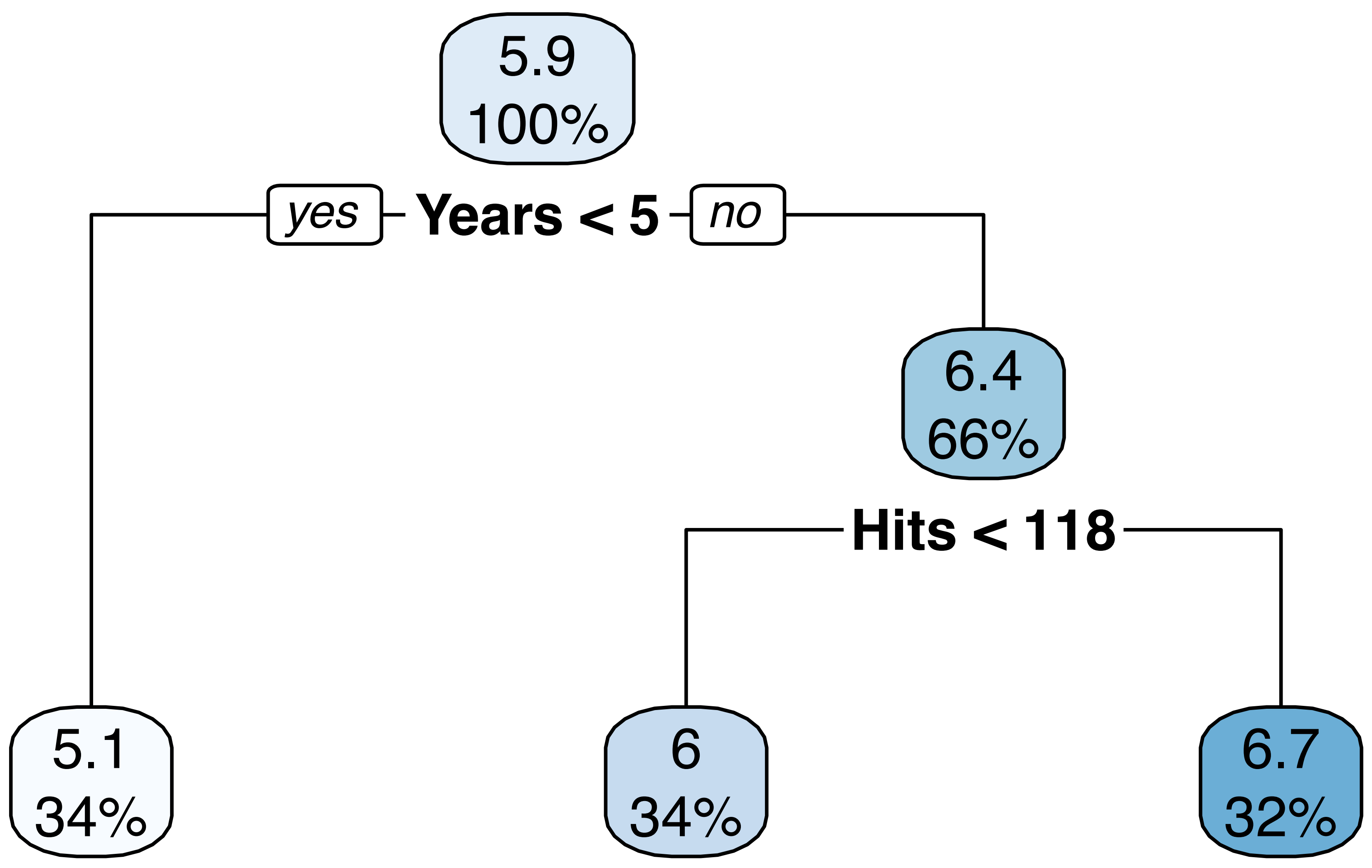
Predicting baseball players' salaries based on years played and number of hits in the previous year.

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.



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

Entering the land of trees and forests

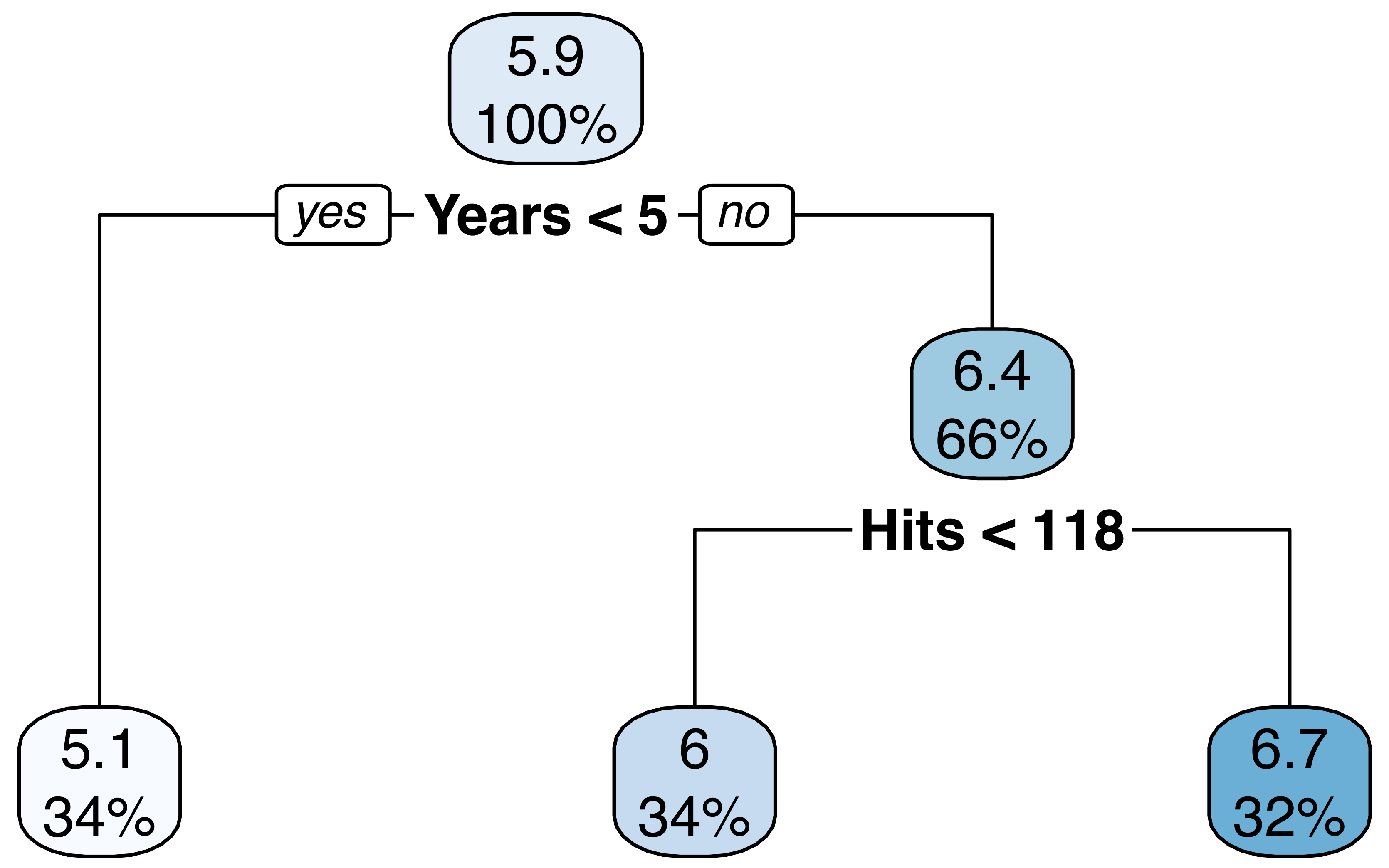
Decision trees (lectures 1 and 2) are predictive models based on recursively partitioning the feature space.

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

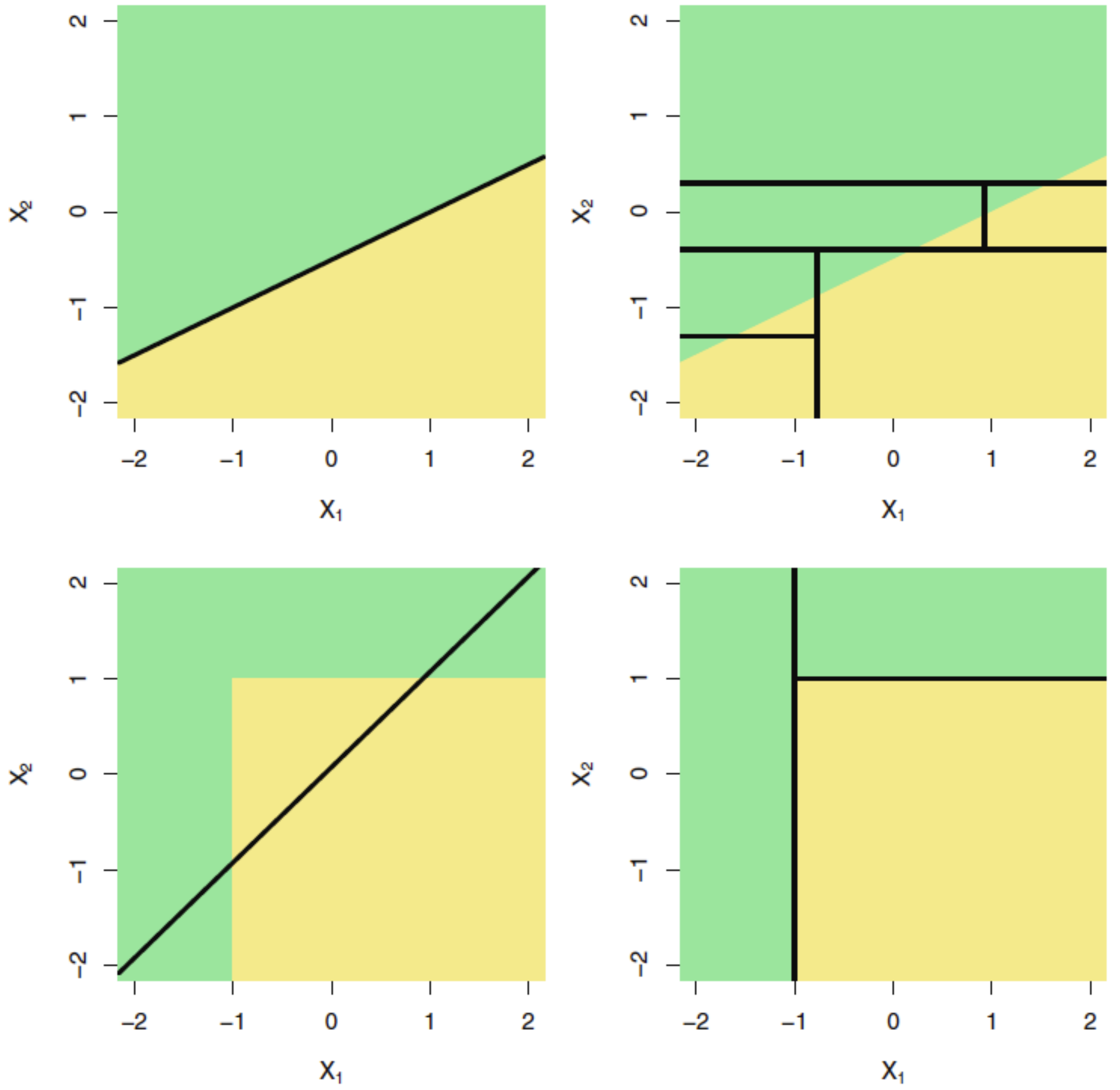
Which perform better?

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).

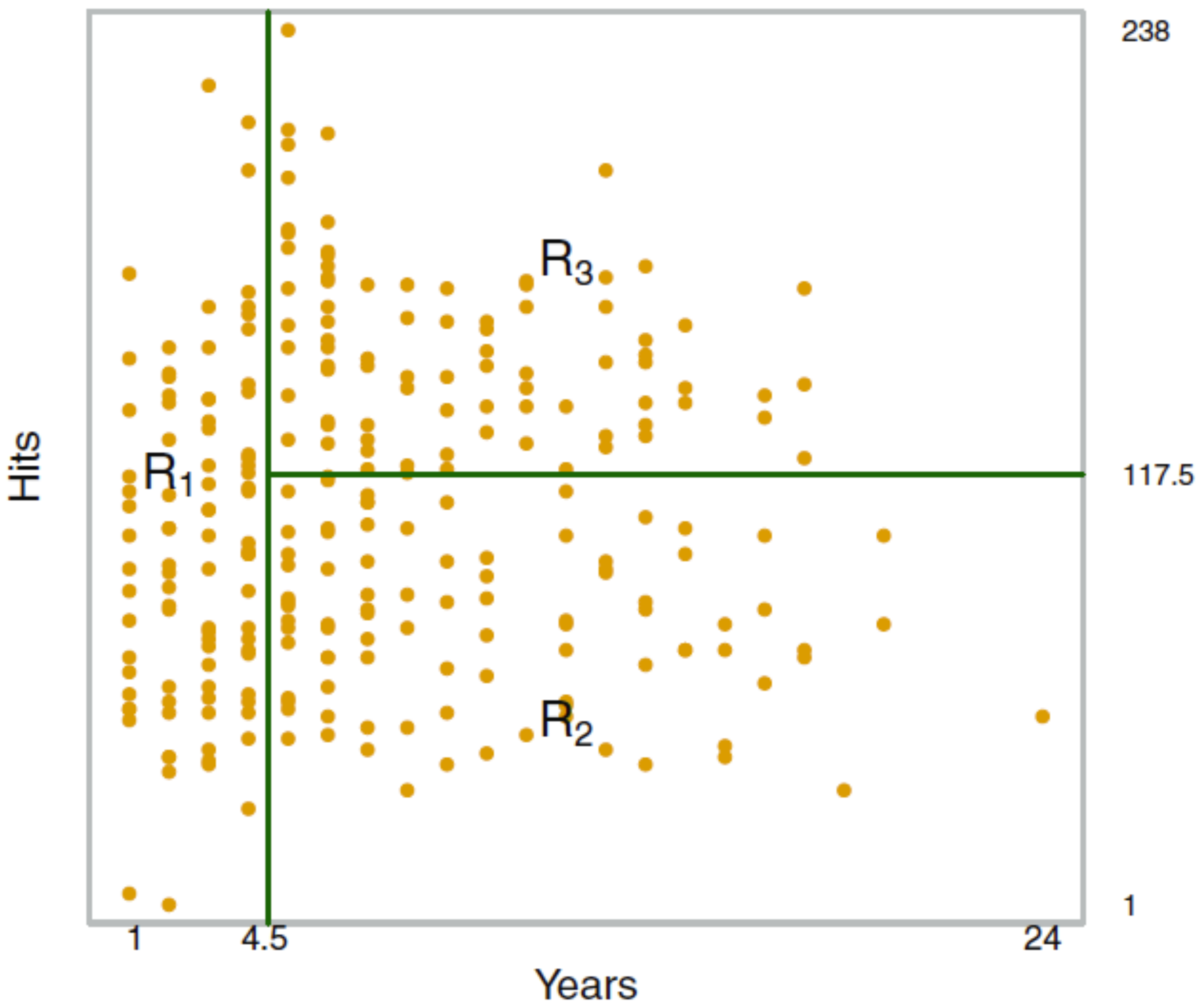
Hitters data

Major League Baseball Data from the 1986 and 1987 seasons.

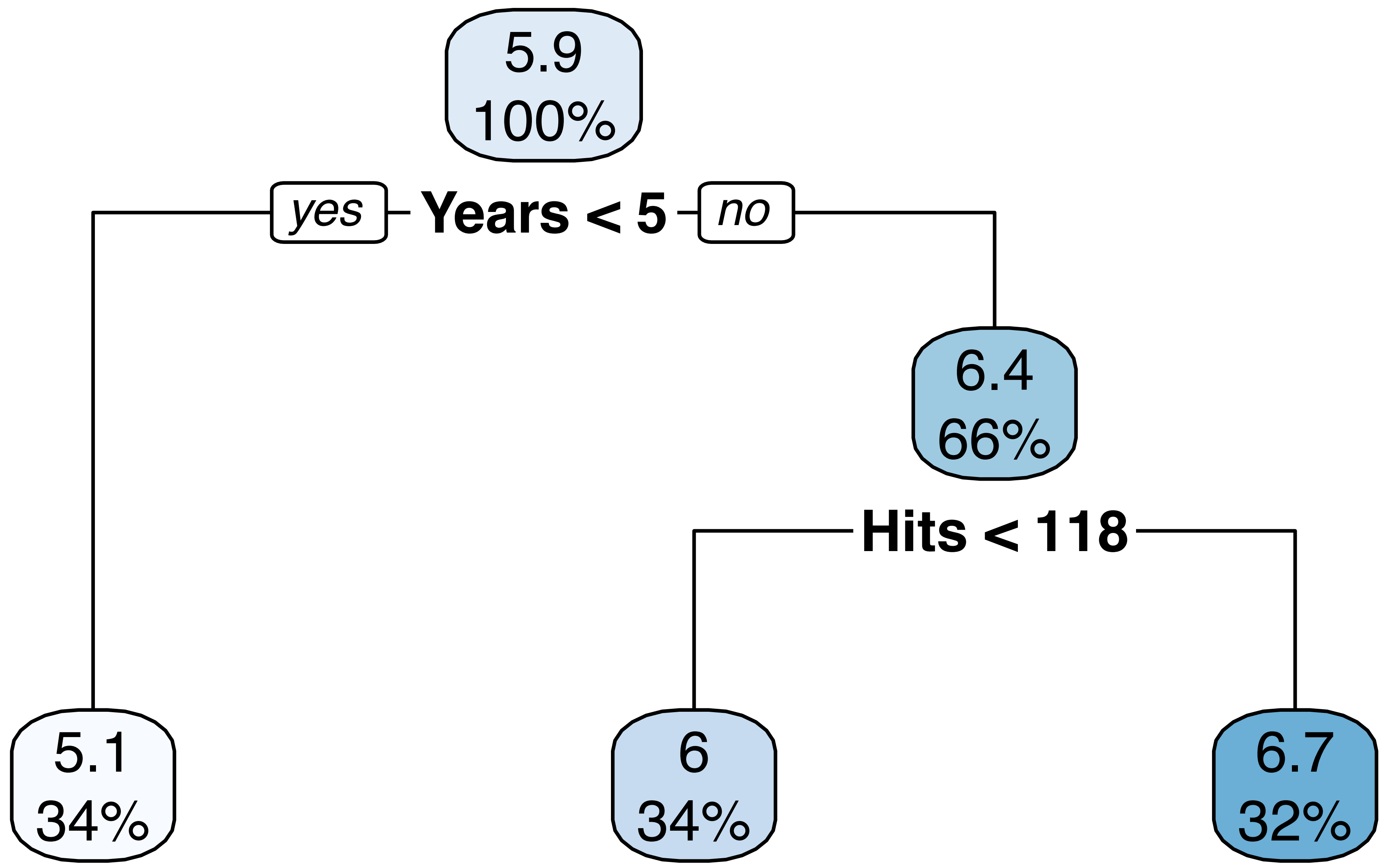
- Observations: 322 MLB players
- Response: Salary (1987 annual salary on opening day in thousands of dollars)
- Features: Assists, AtBat,…,Hits,…,Years (19 total)

Tree $\Longleftrightarrow$ Partition into nested, axis-aligned rectangles

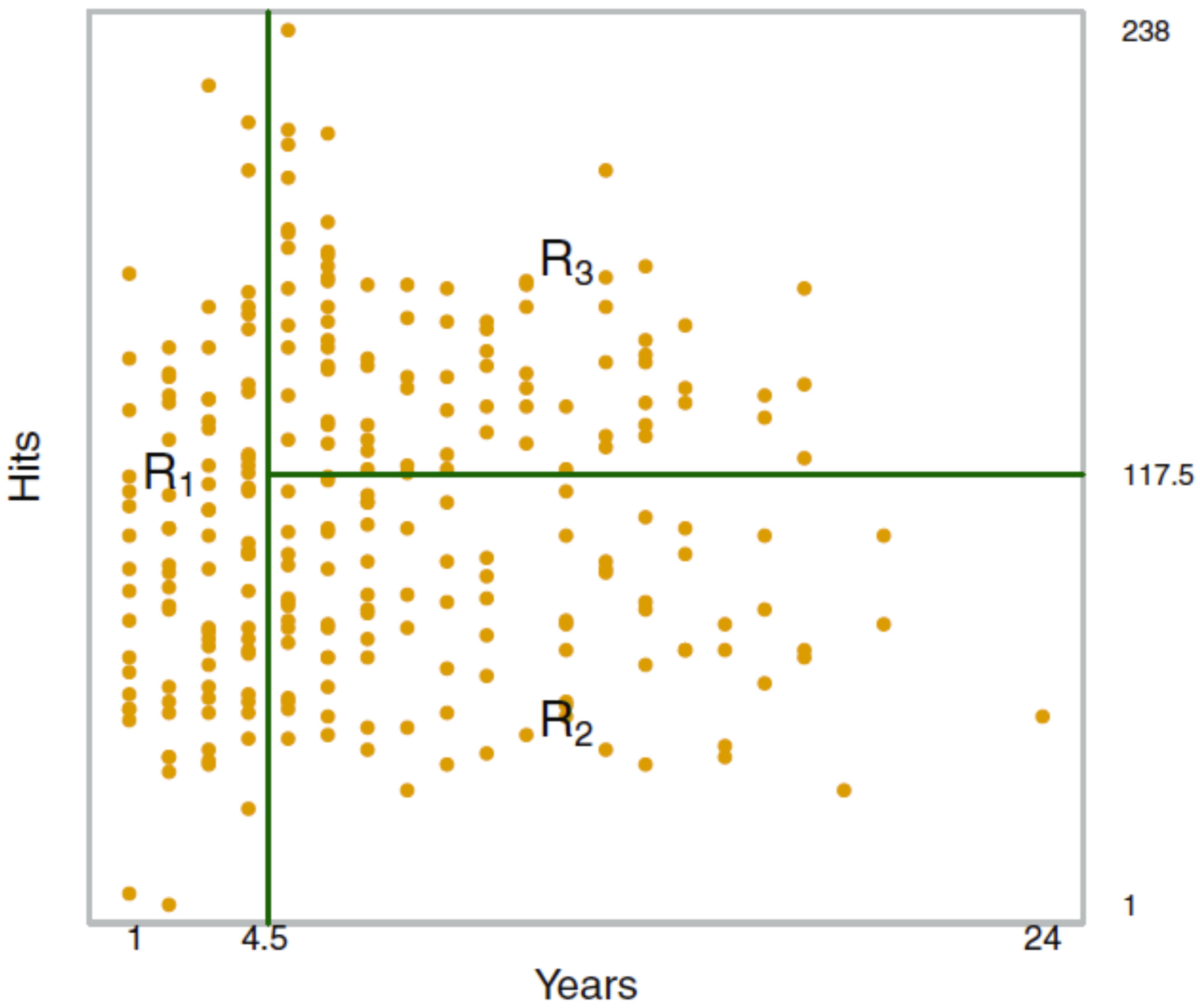
$\Longleftrightarrow$



Tree $\Longleftrightarrow$ Partition into nested, axis-aligned rectangles



$\Longleftrightarrow$



Mathematical expression of the prediction rule

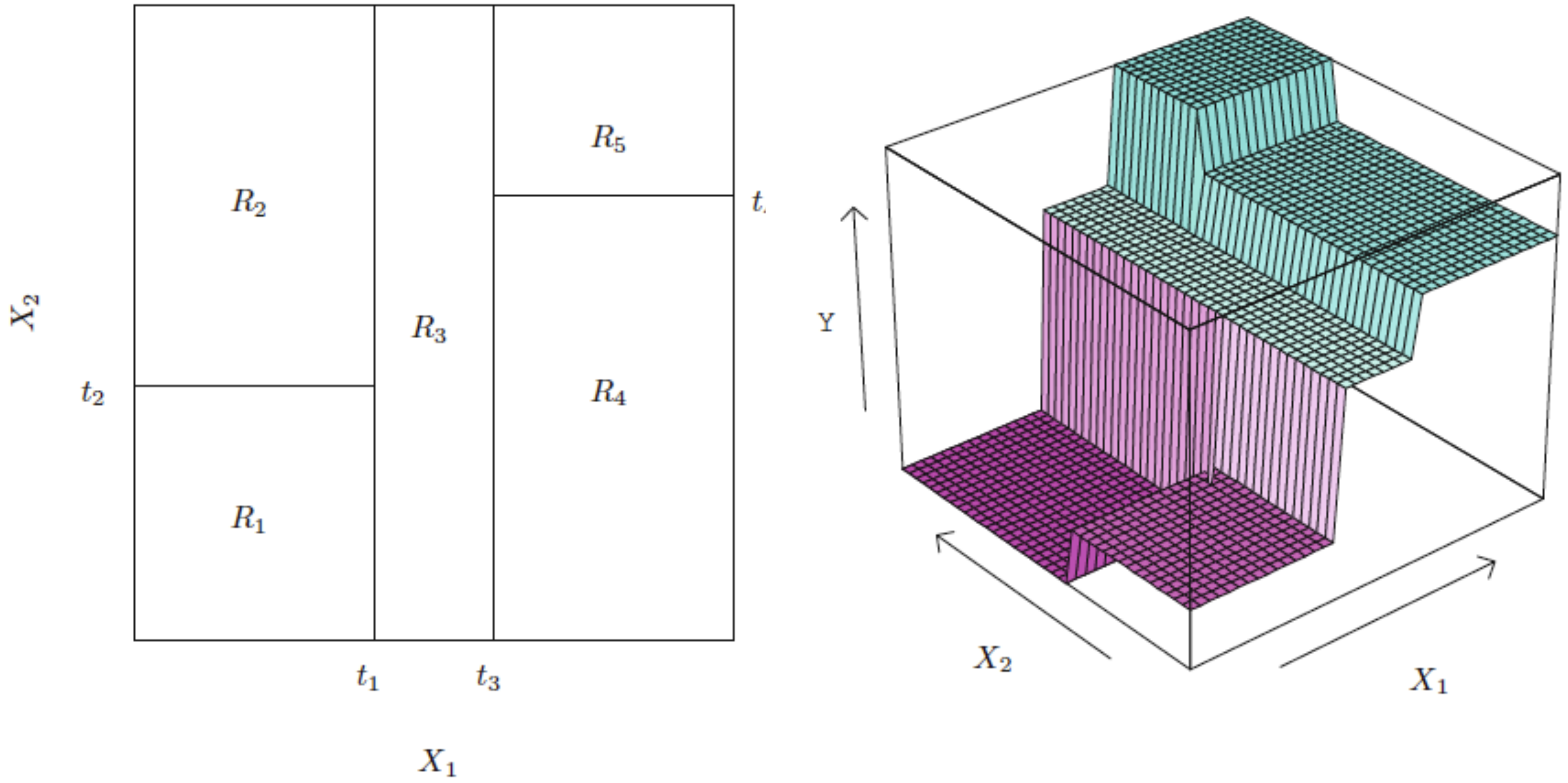
A trained tree consists of:

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

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

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

(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

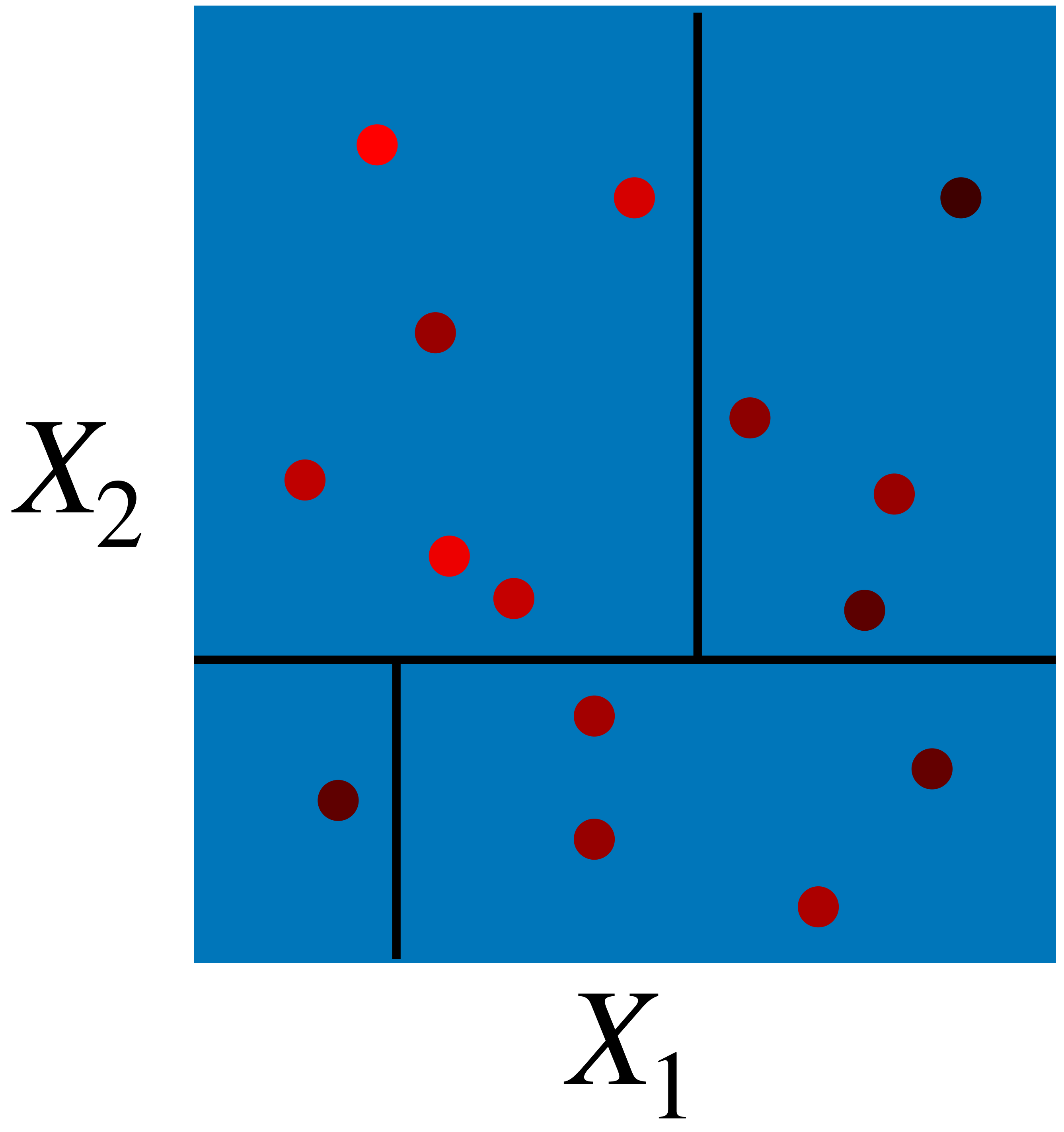
The squared error objective

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

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

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

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



Training a regression tree

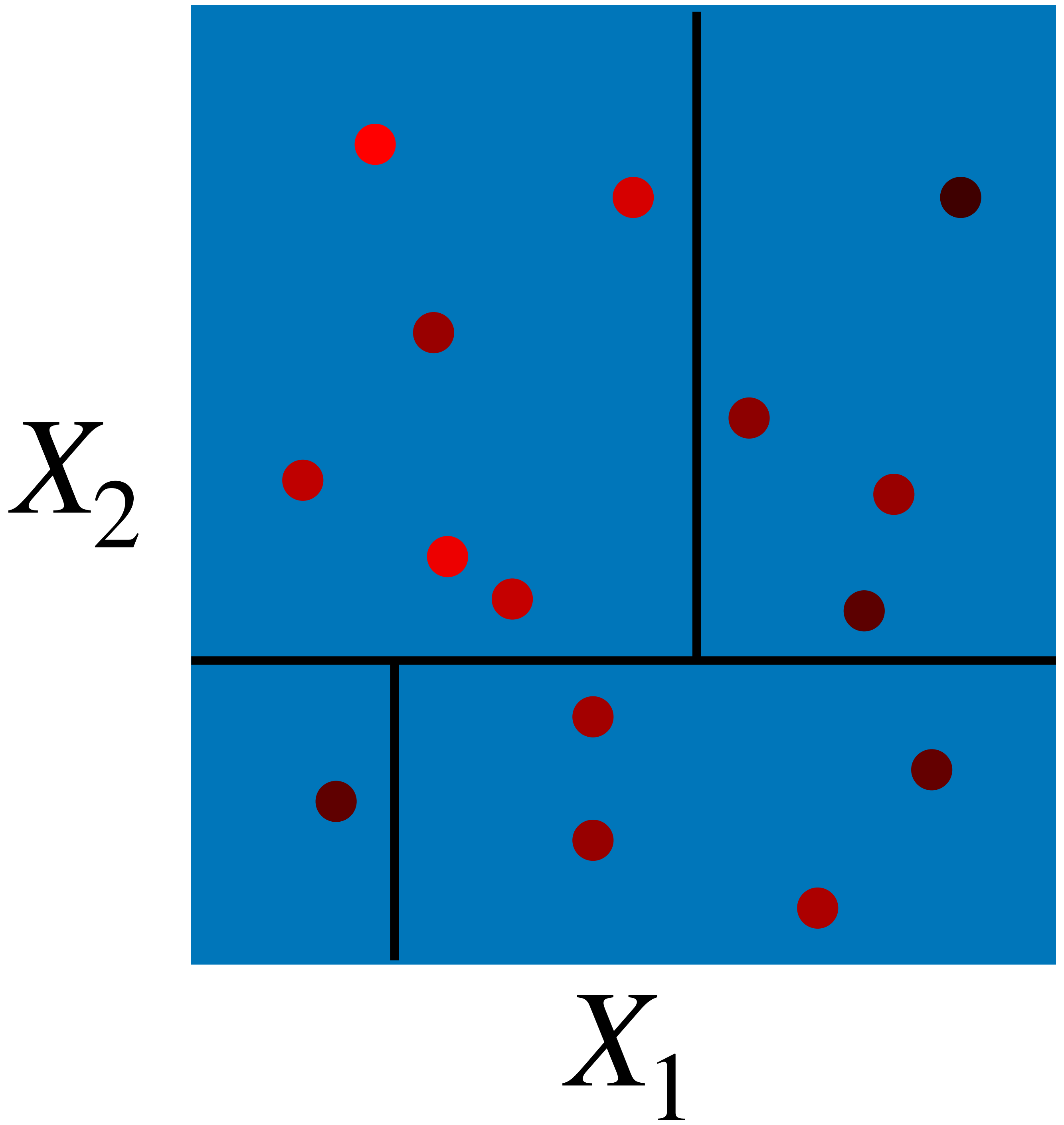
Optimal $\widehat{c}_m$ given $\widehat{R}_m$

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

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

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

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

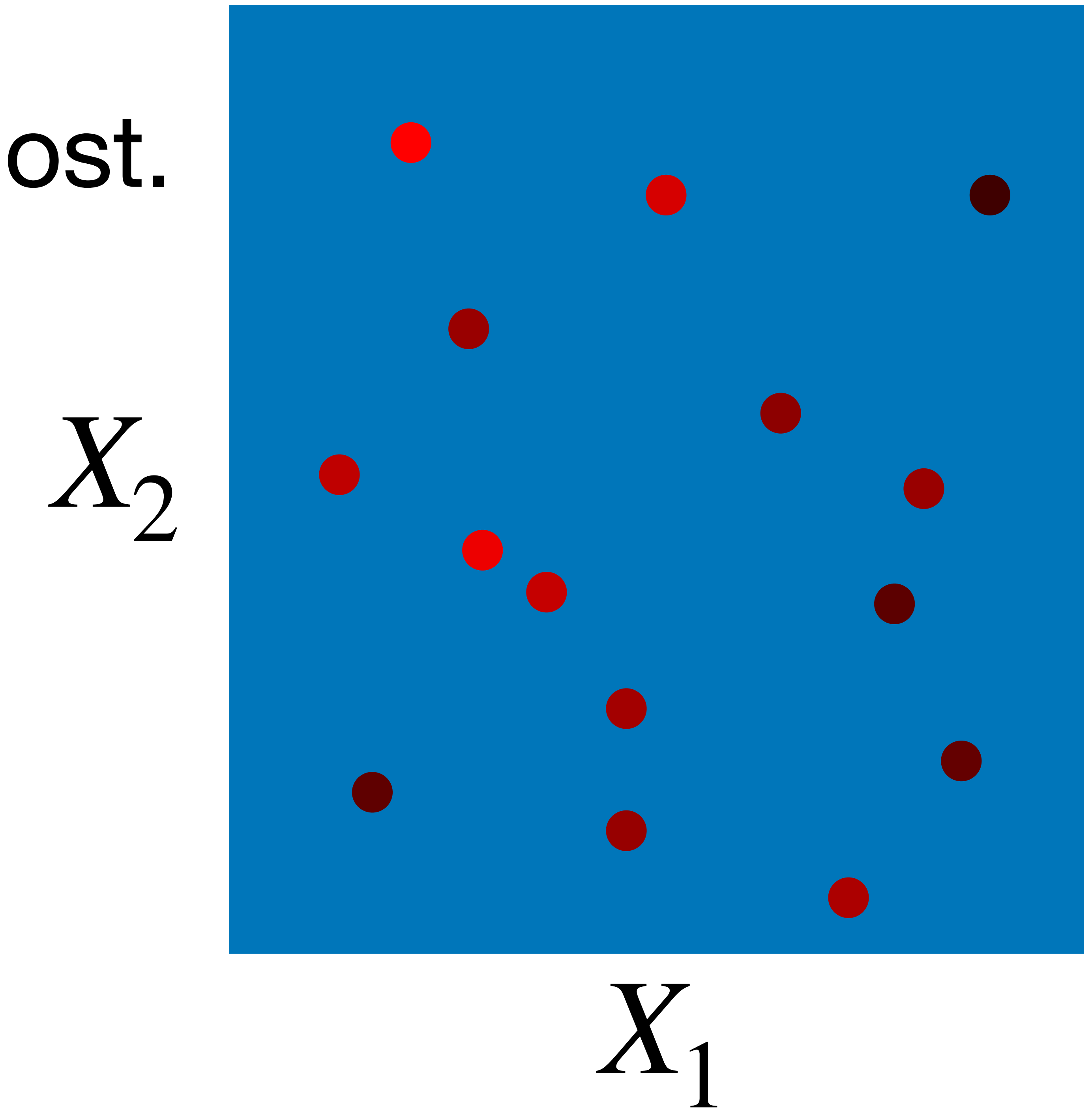


Training a regression tree

Finding the rectangles $\widehat{R}_m$

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

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.

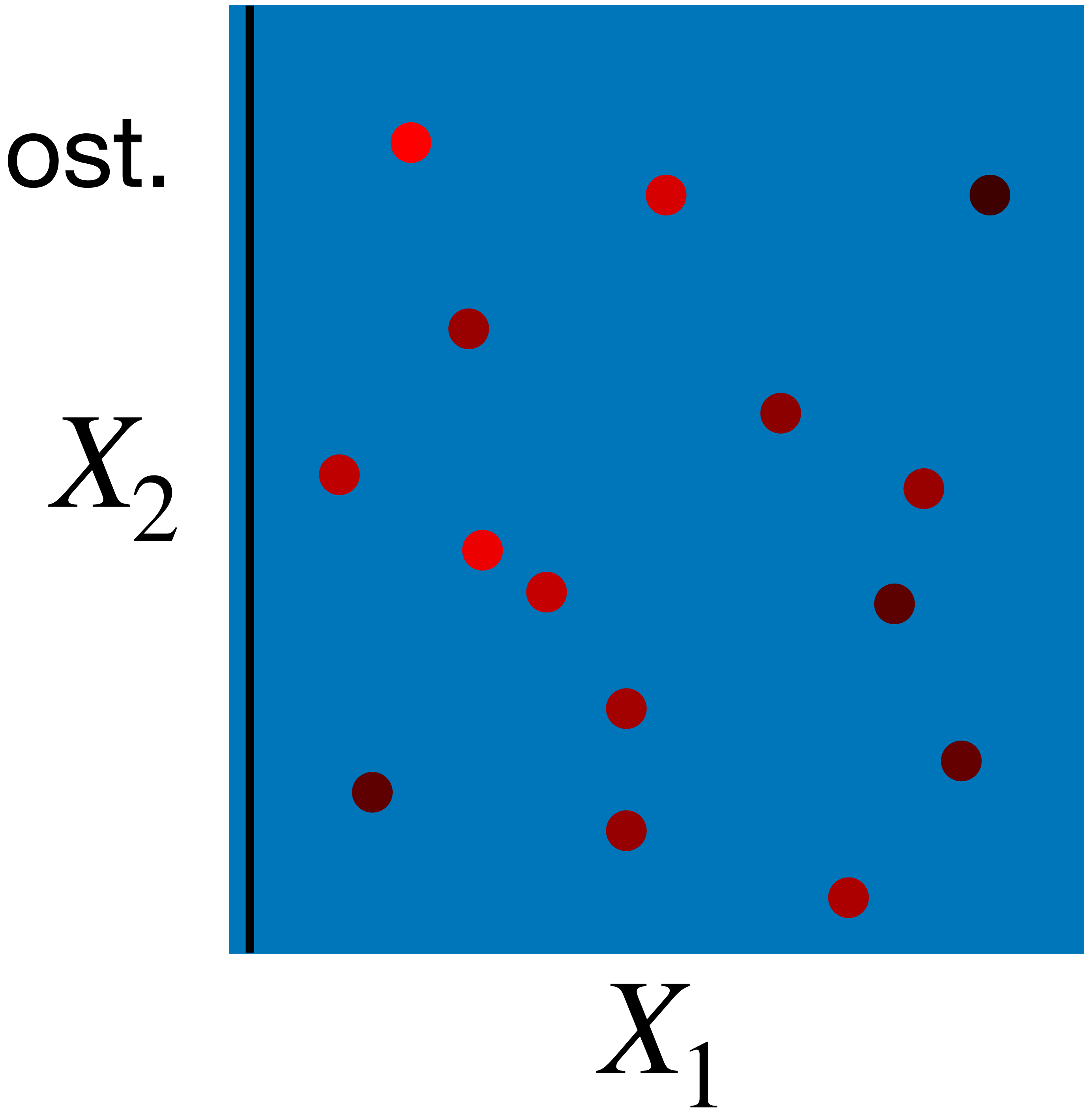


Training a regression tree

Finding the rectangles $\widehat{R}_m$

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

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.

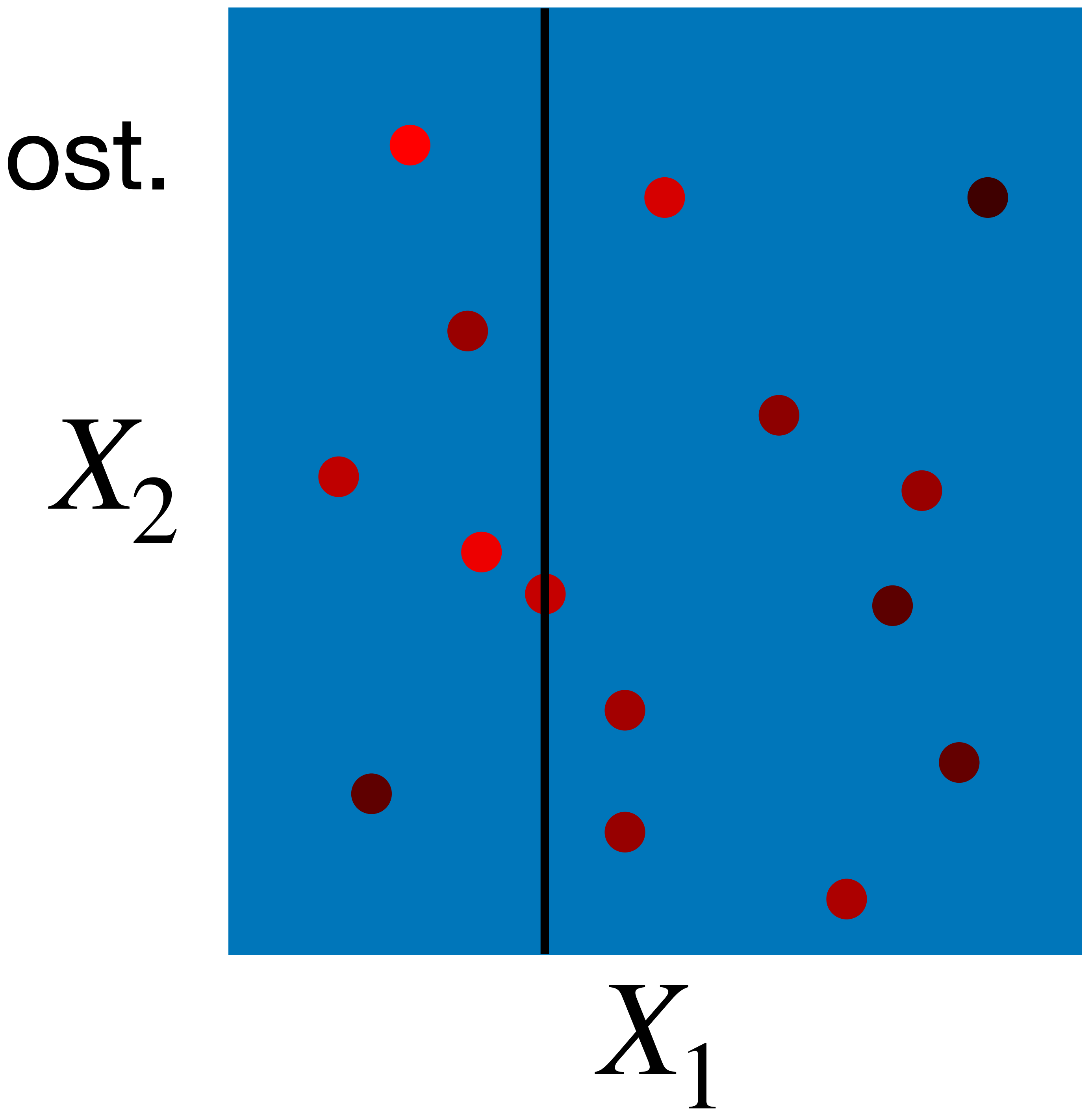


Training a regression tree

Finding the rectangles $\widehat{R}_m$

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

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.

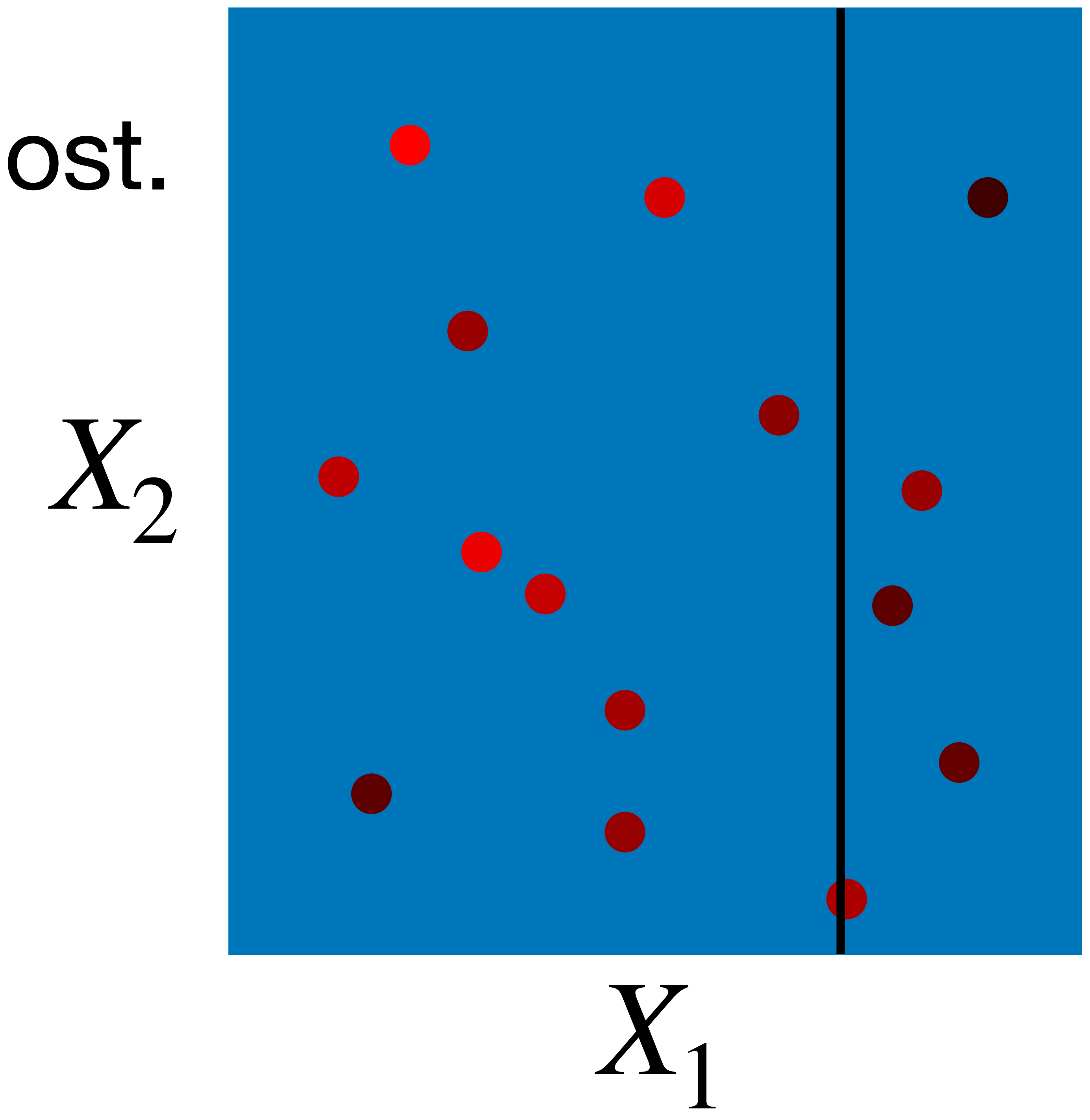


Training a regression tree

Finding the rectangles $\widehat{R}_m$

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

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

Finding the rectangles $\widehat{R}_m$

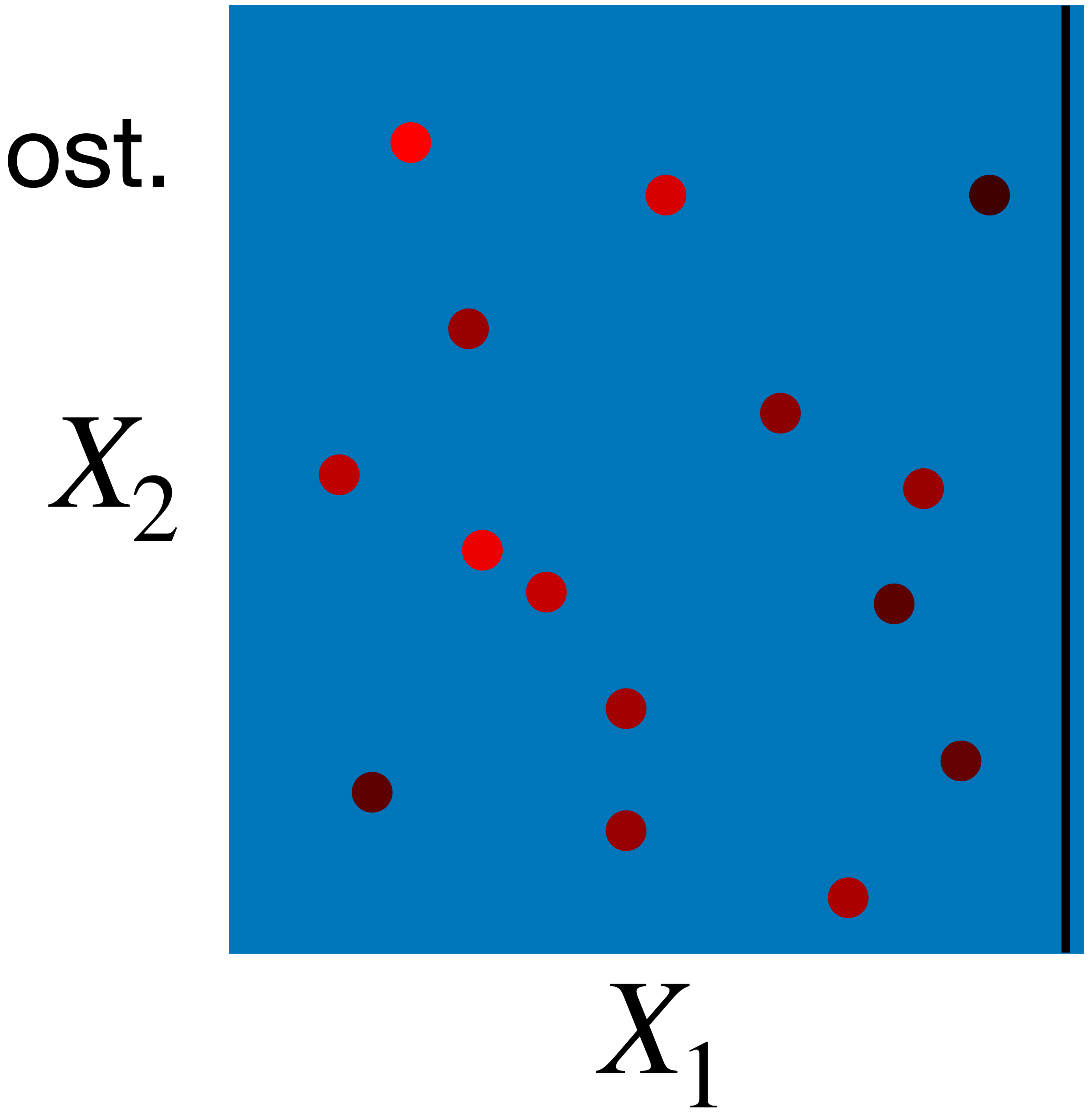
The optimal set of rectangles is computationally intractable to find. In practice, we employ a greedy top-down algorithm:

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.

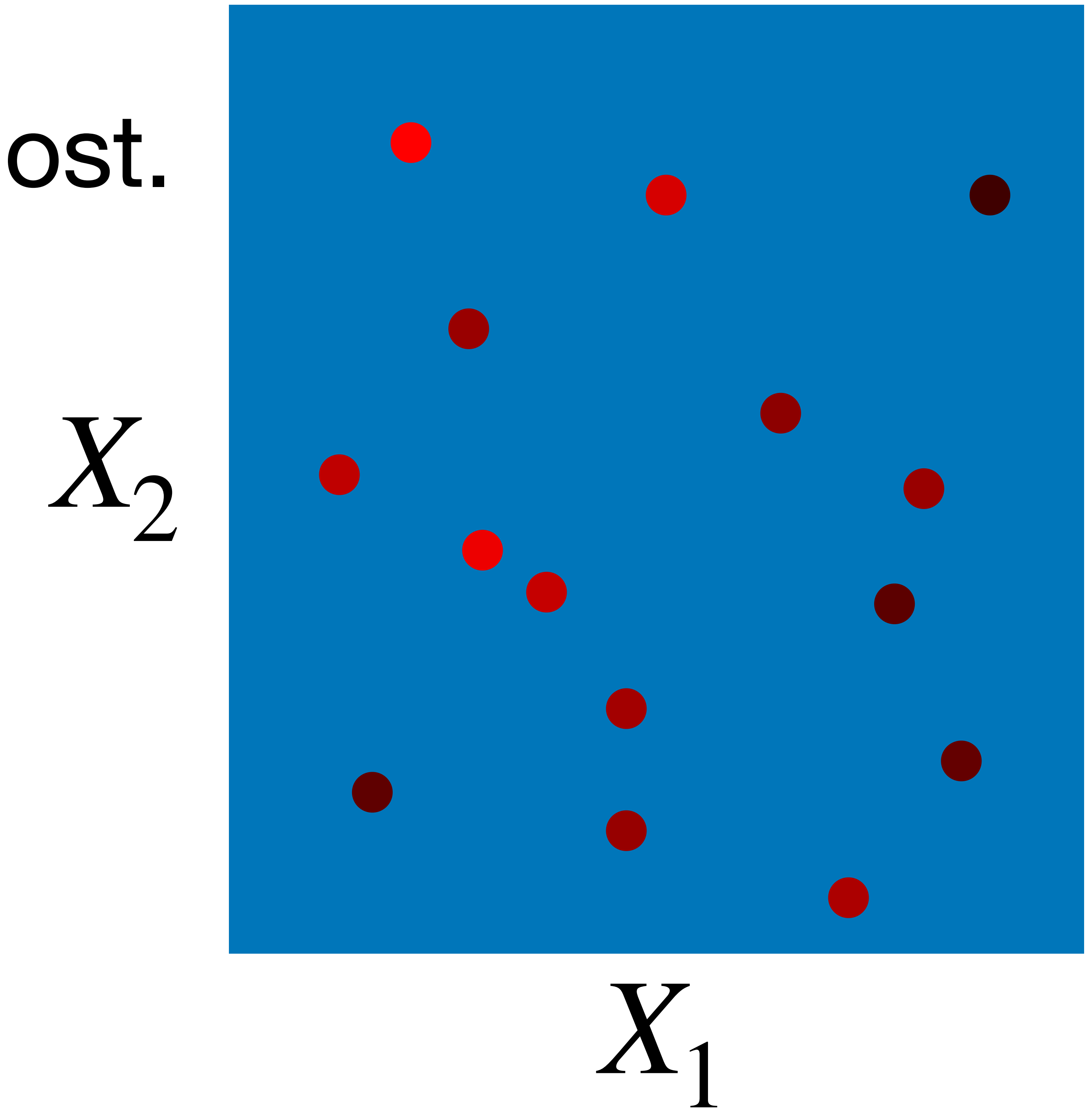


Training a regression tree

Finding the rectangles $\widehat{R}_m$

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

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

Finding the rectangles $\widehat{R}_m$

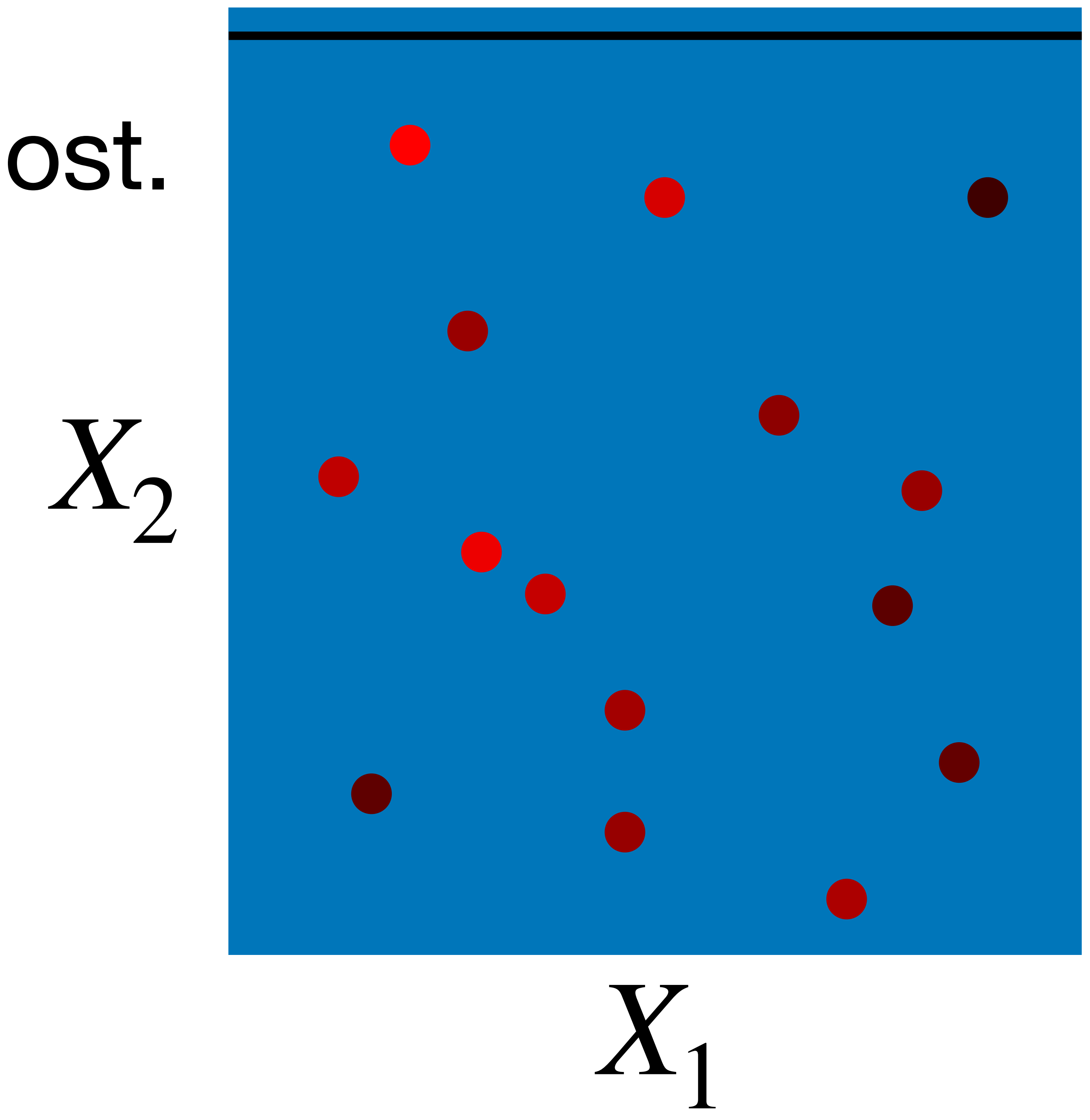
The optimal set of rectangles is computationally intractable to find. In practice, we employ a greedy top-down algorithm:

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.

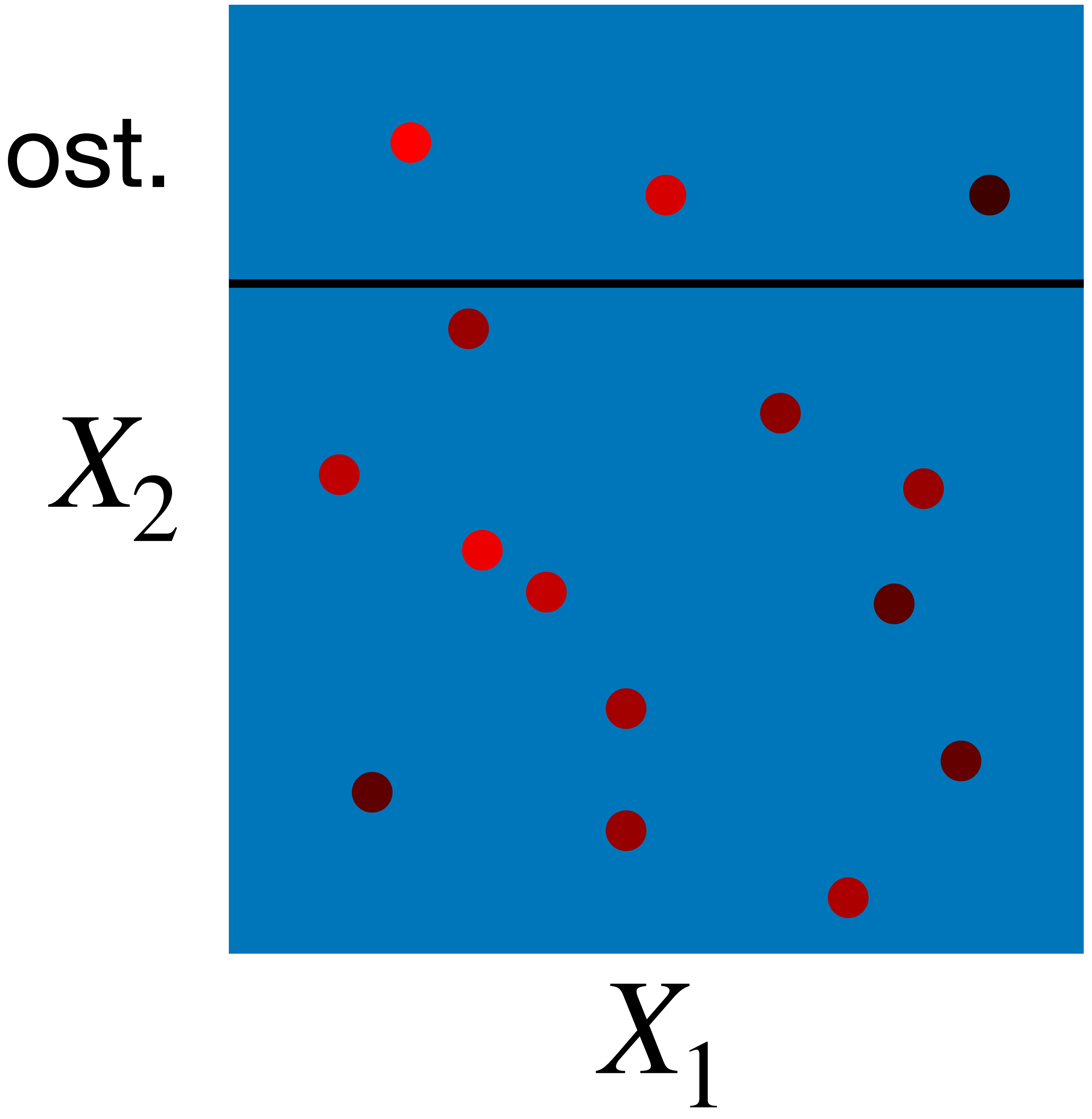


Training a regression tree

Finding the rectangles $\widehat{R}_m$

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

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.

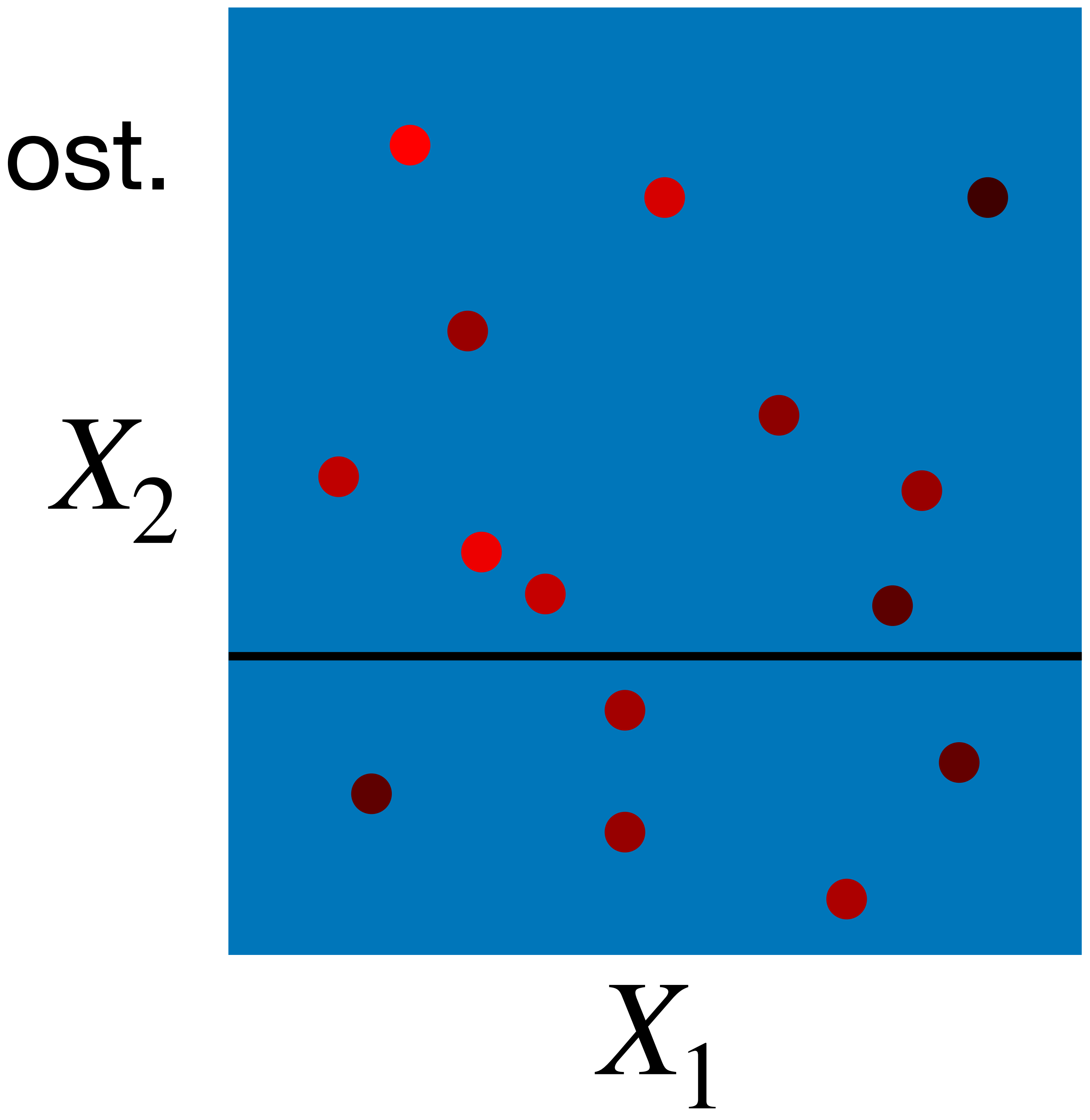


Training a regression tree

Finding the rectangles $\widehat{R}_m$

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

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

Finding the rectangles $\widehat{R}_m$

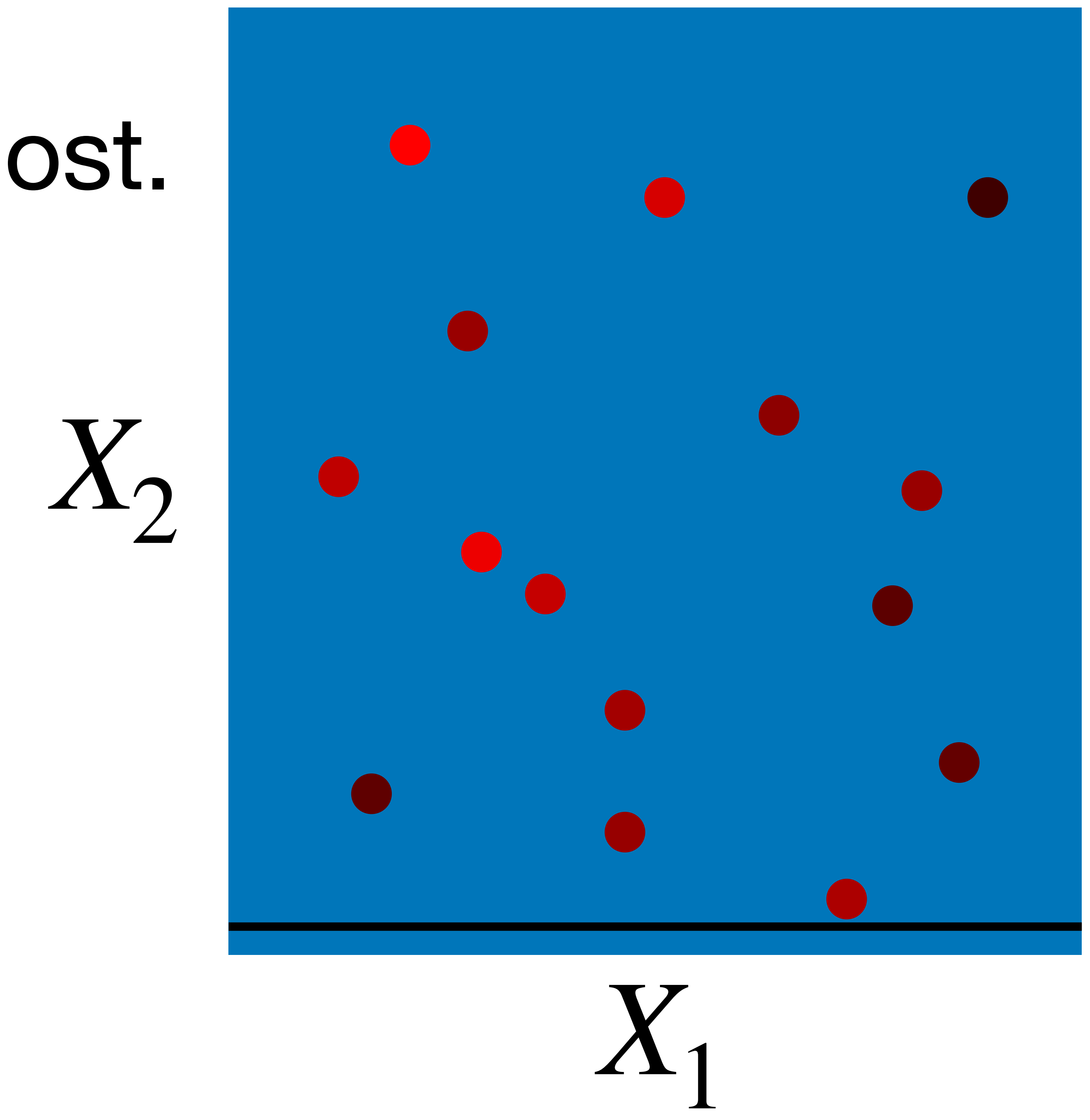
The optimal set of rectangles is computationally intractable to find. In practice, we employ a greedy top-down algorithm:

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.

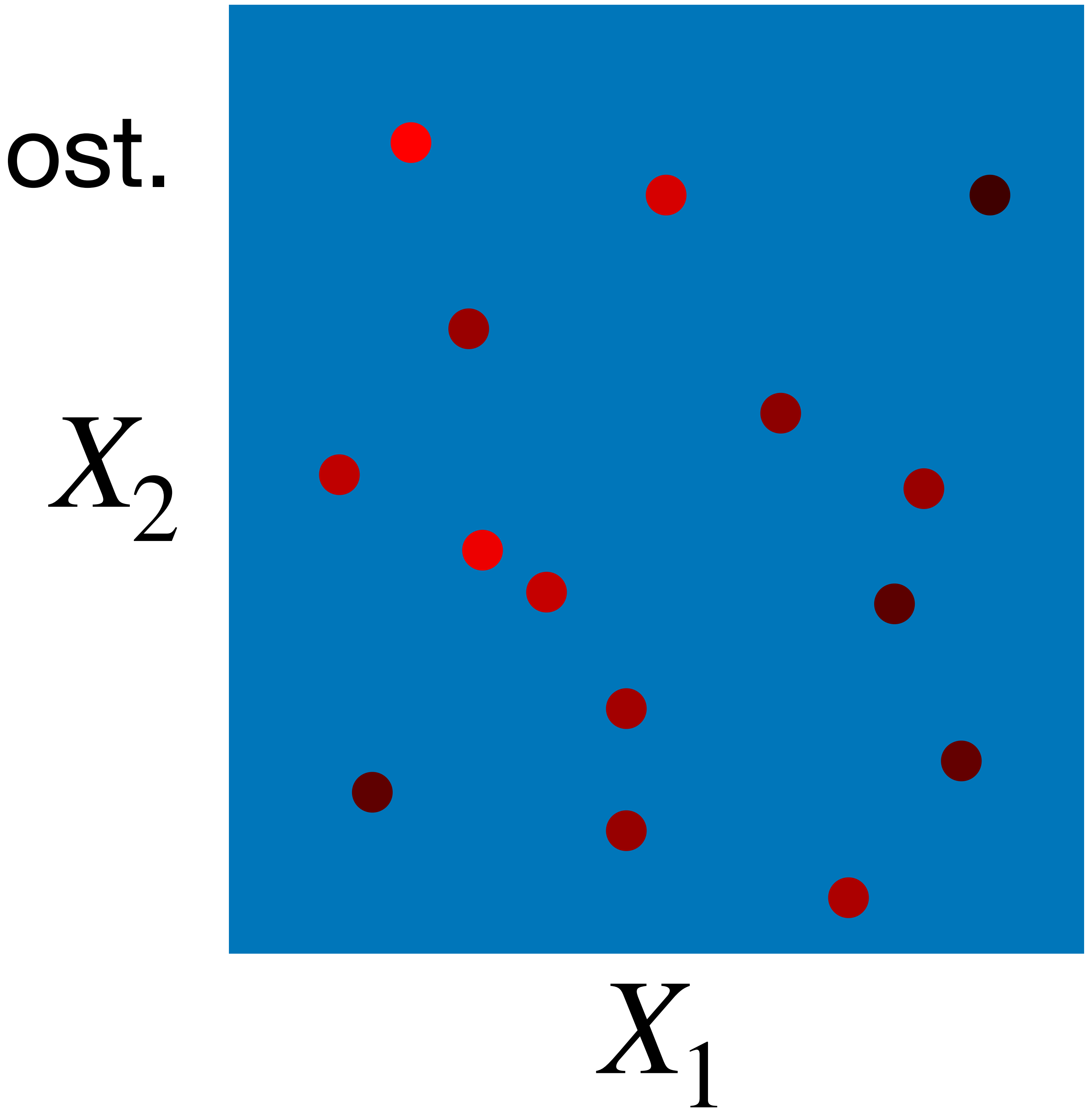


Training a regression tree

Finding the rectangles $\widehat{R}_m$

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

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.

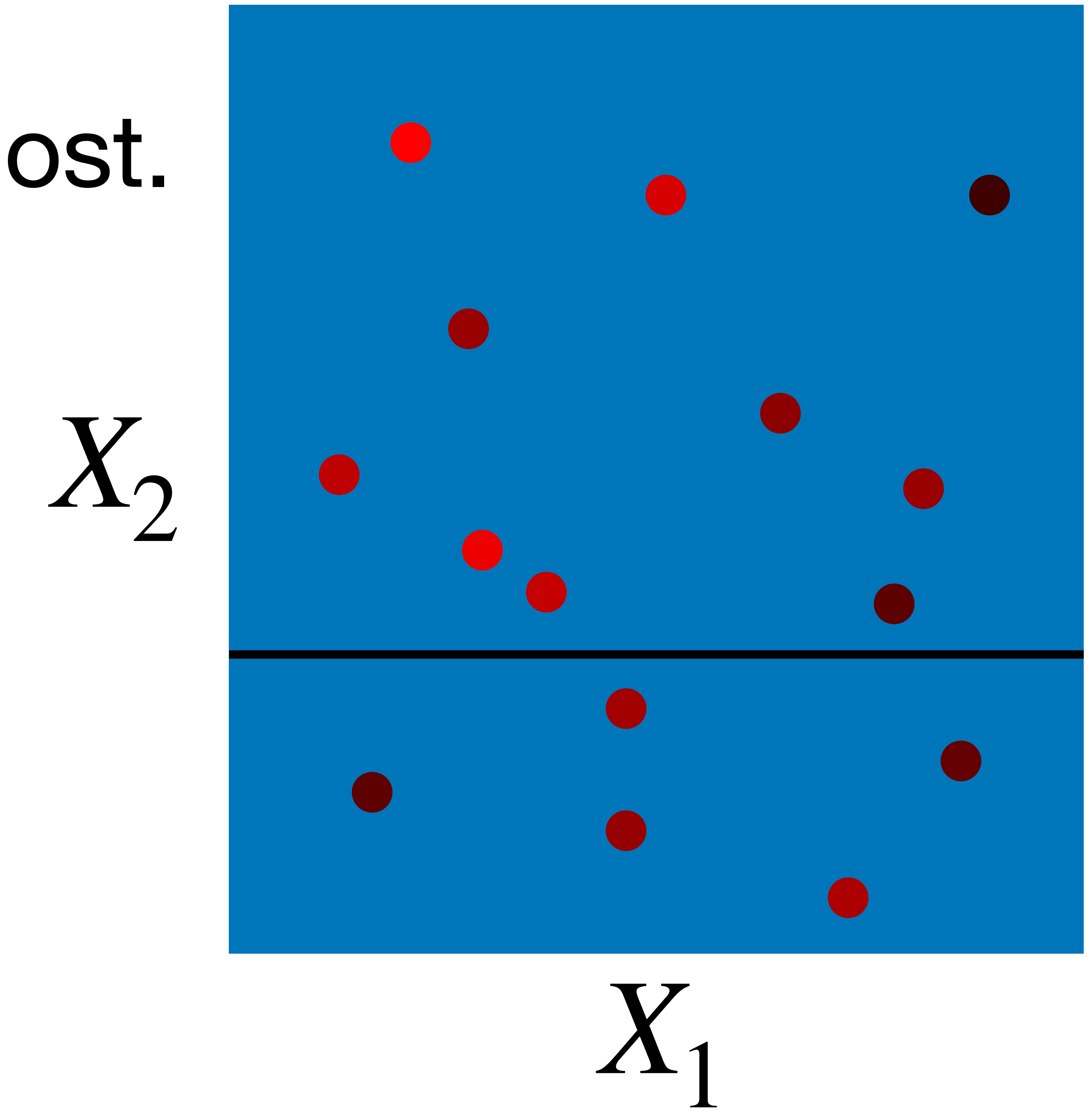


Training a regression tree

Finding the rectangles $\widehat{R}_m$

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

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.

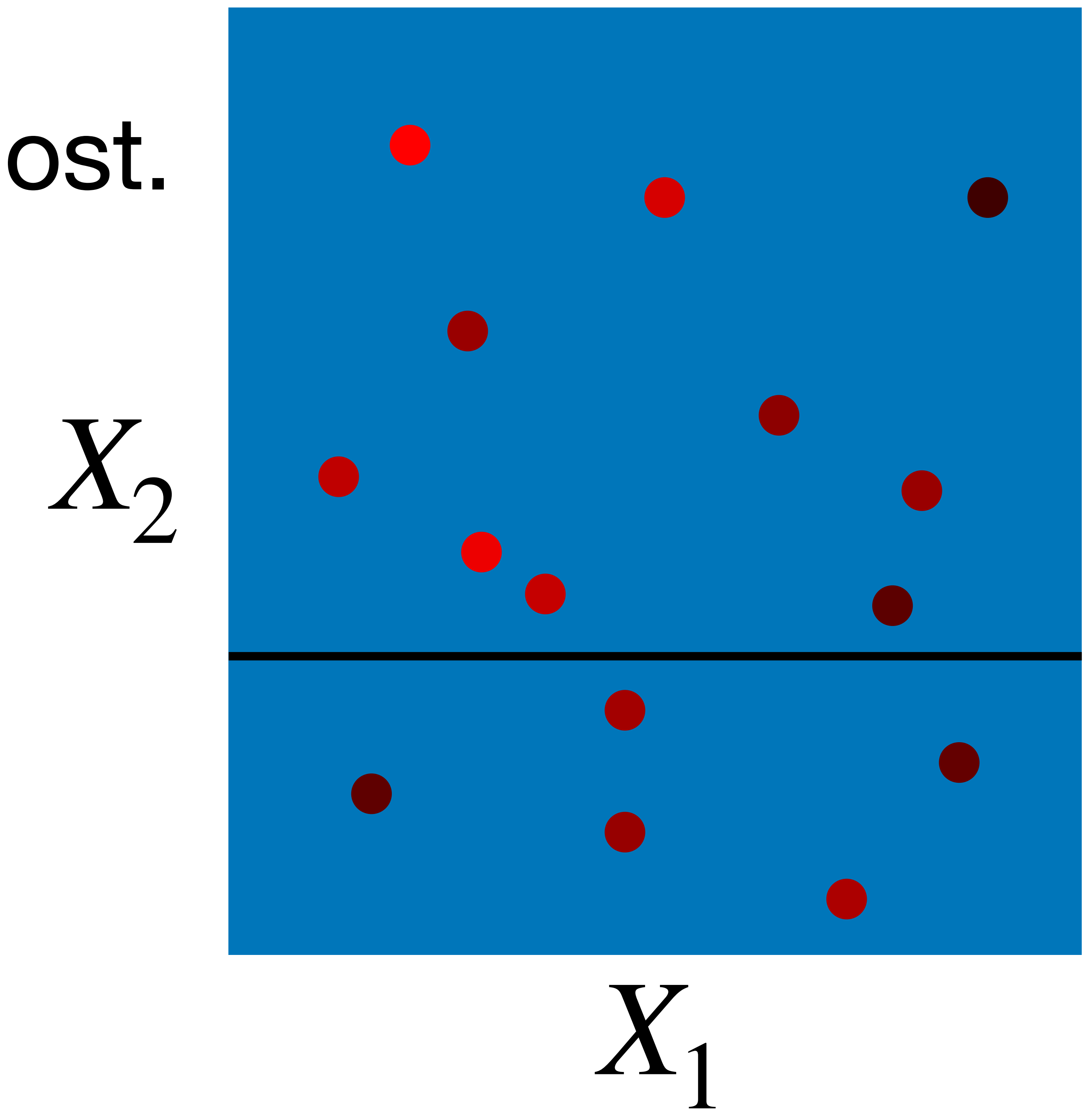


Training a regression tree

Finding the rectangles $\widehat{R}_m$

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

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.

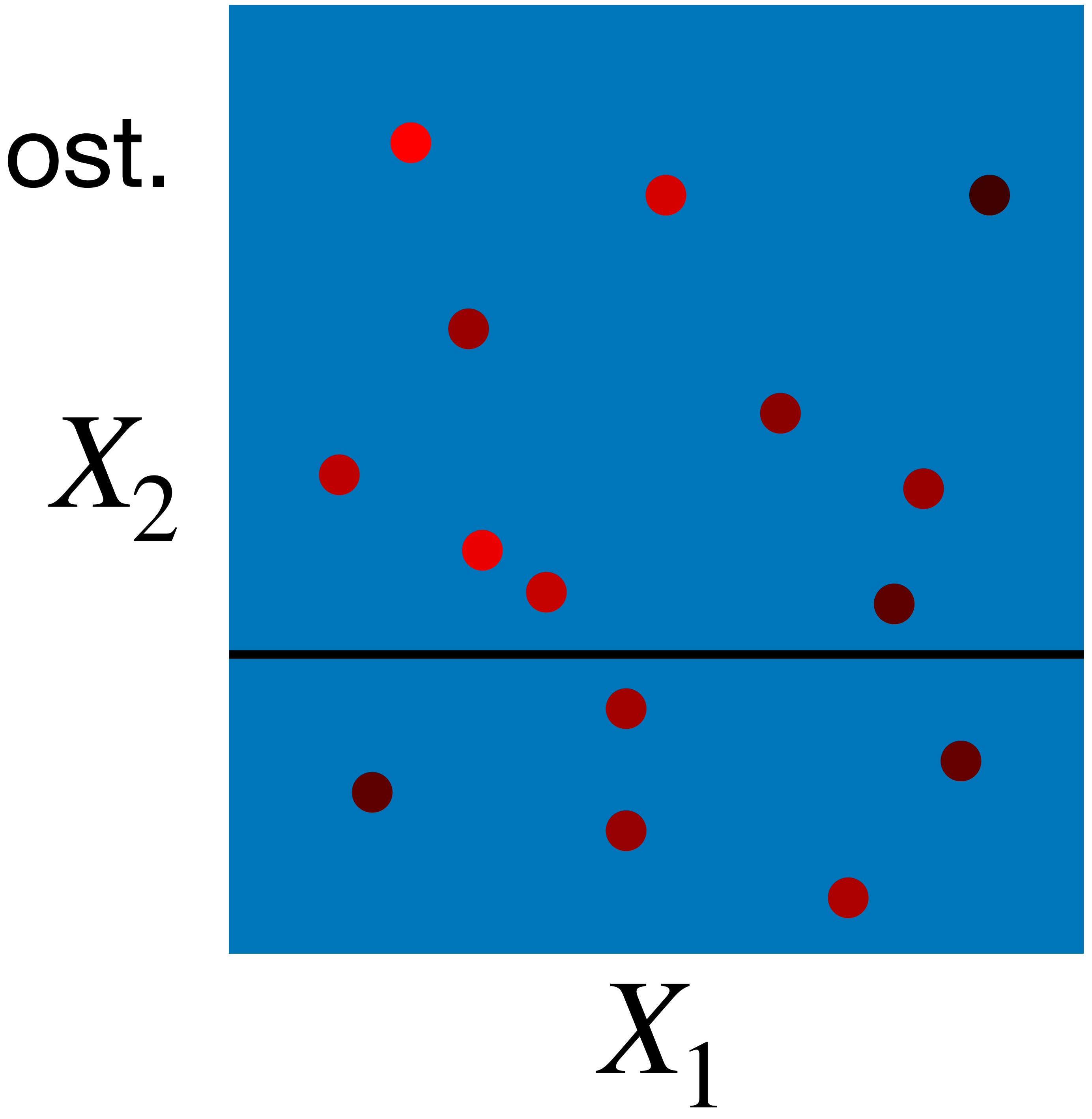


Training a regression tree

Finding the rectangles $\widehat{R}_m$

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

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.

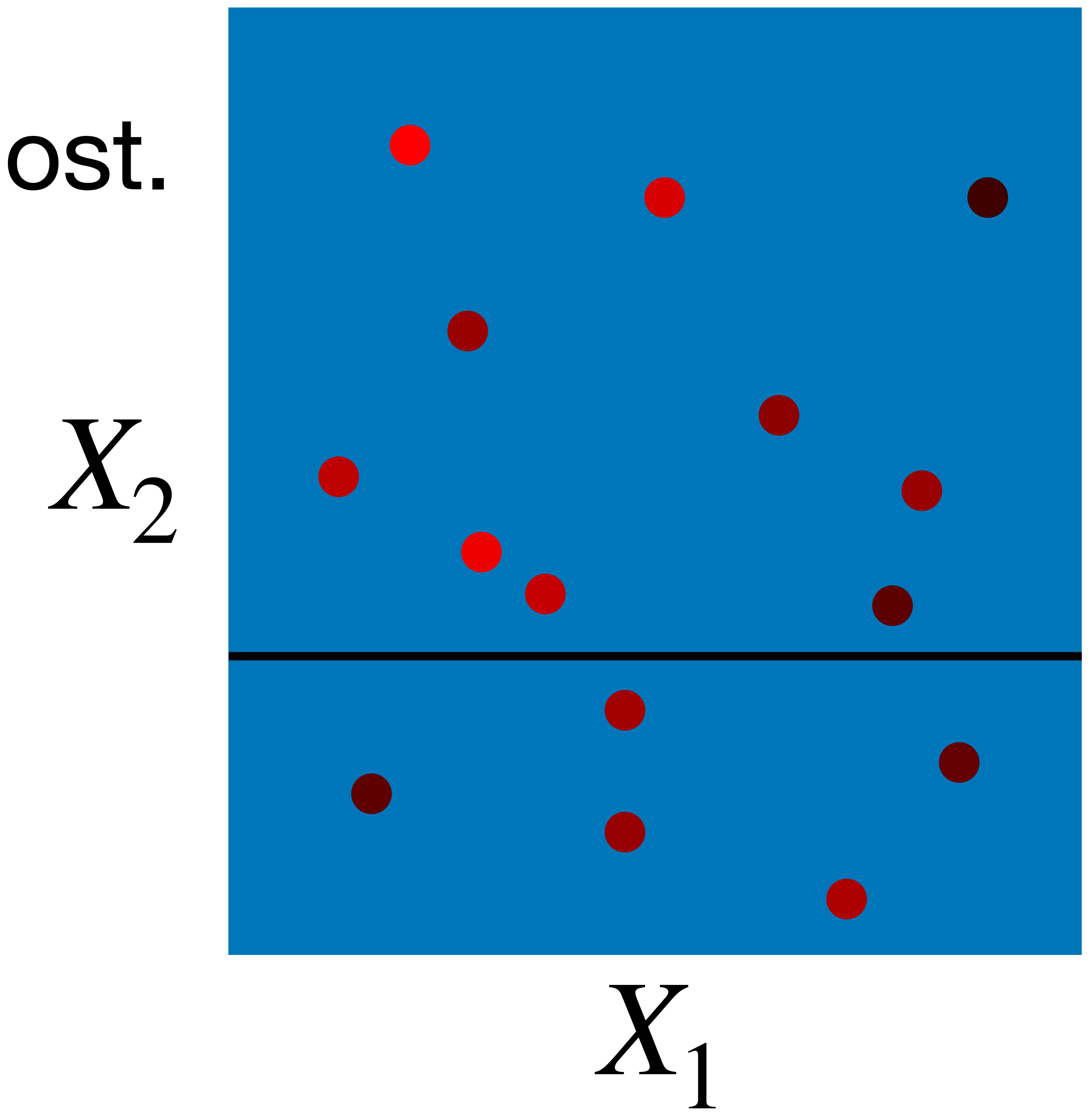


Training a regression tree

Finding the rectangles $\widehat{R}_m$

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

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.

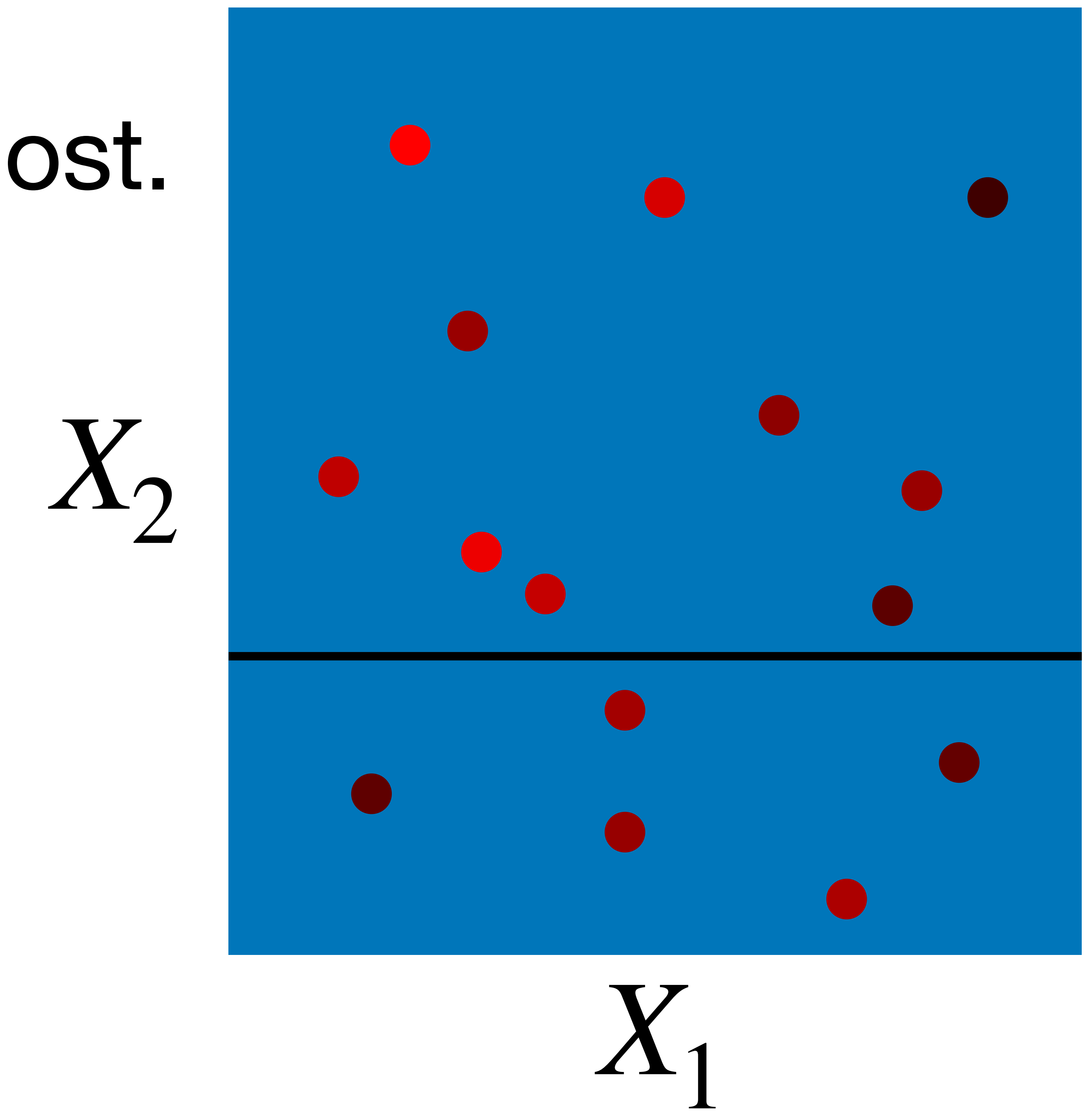


Training a regression tree

Finding the rectangles $\widehat{R}_m$

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

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.

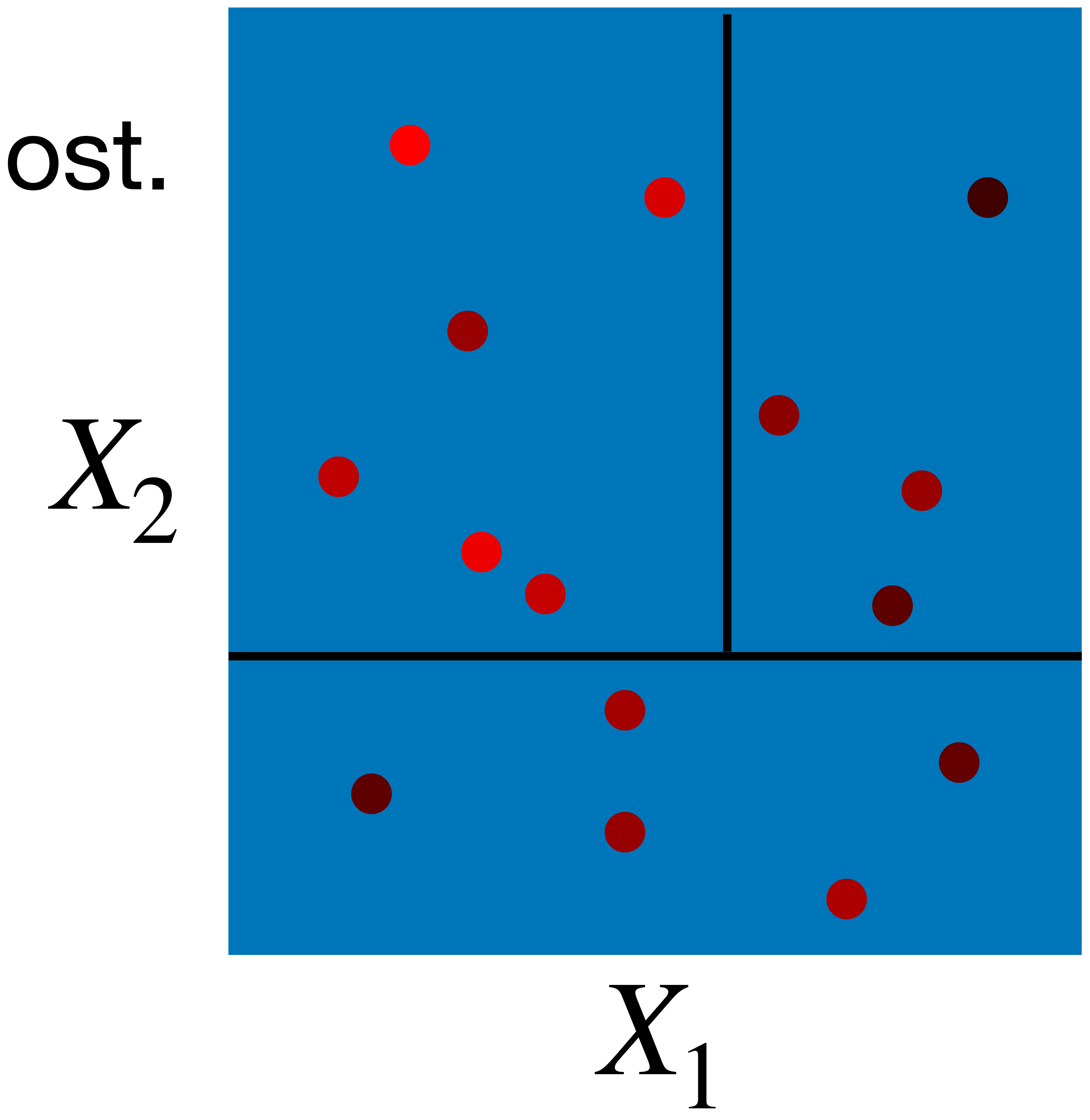


Training a regression tree

Finding the rectangles $\widehat{R}_m$

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

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.

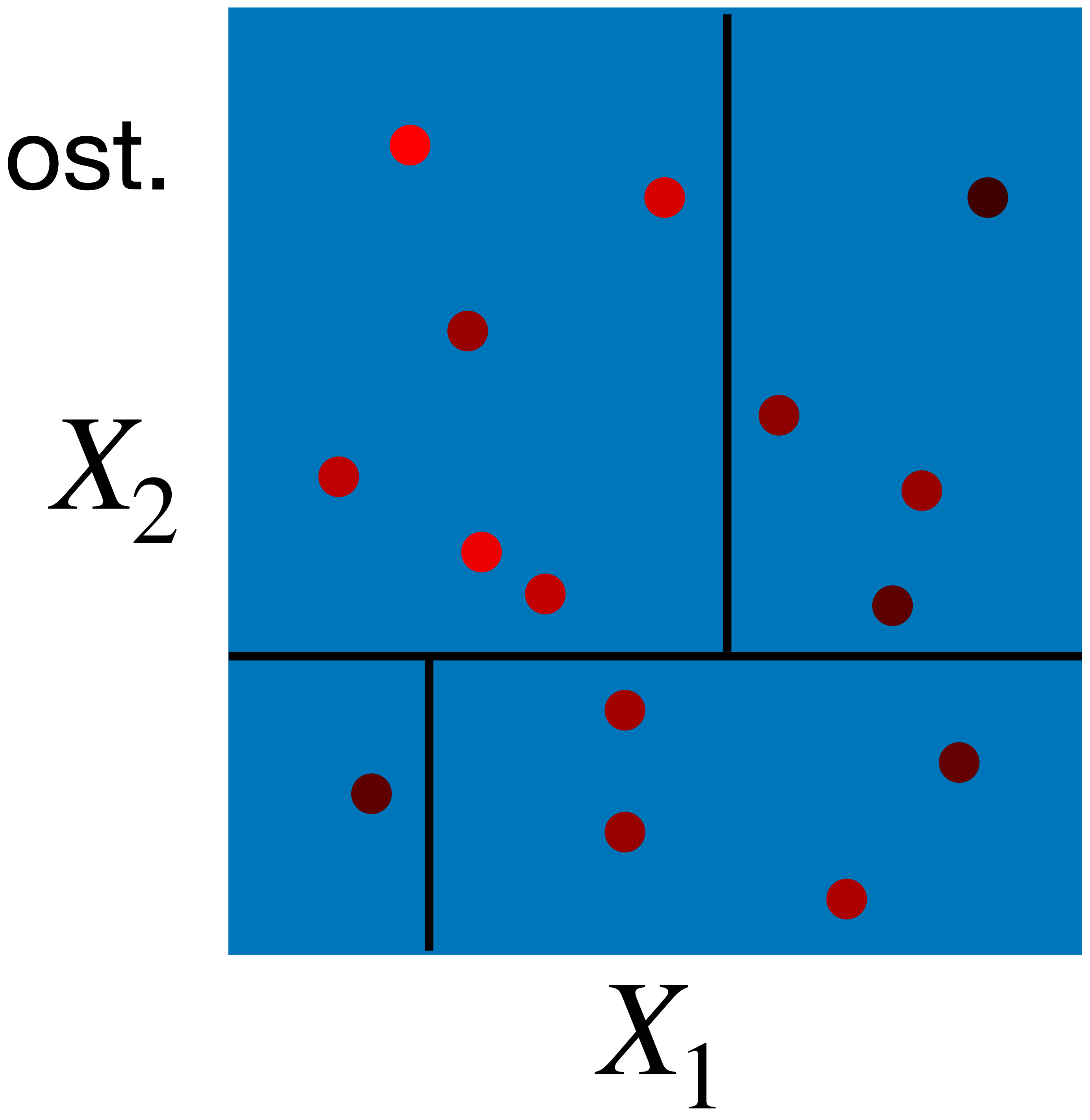


Training a regression tree

Finding the rectangles $\widehat{R}_m$

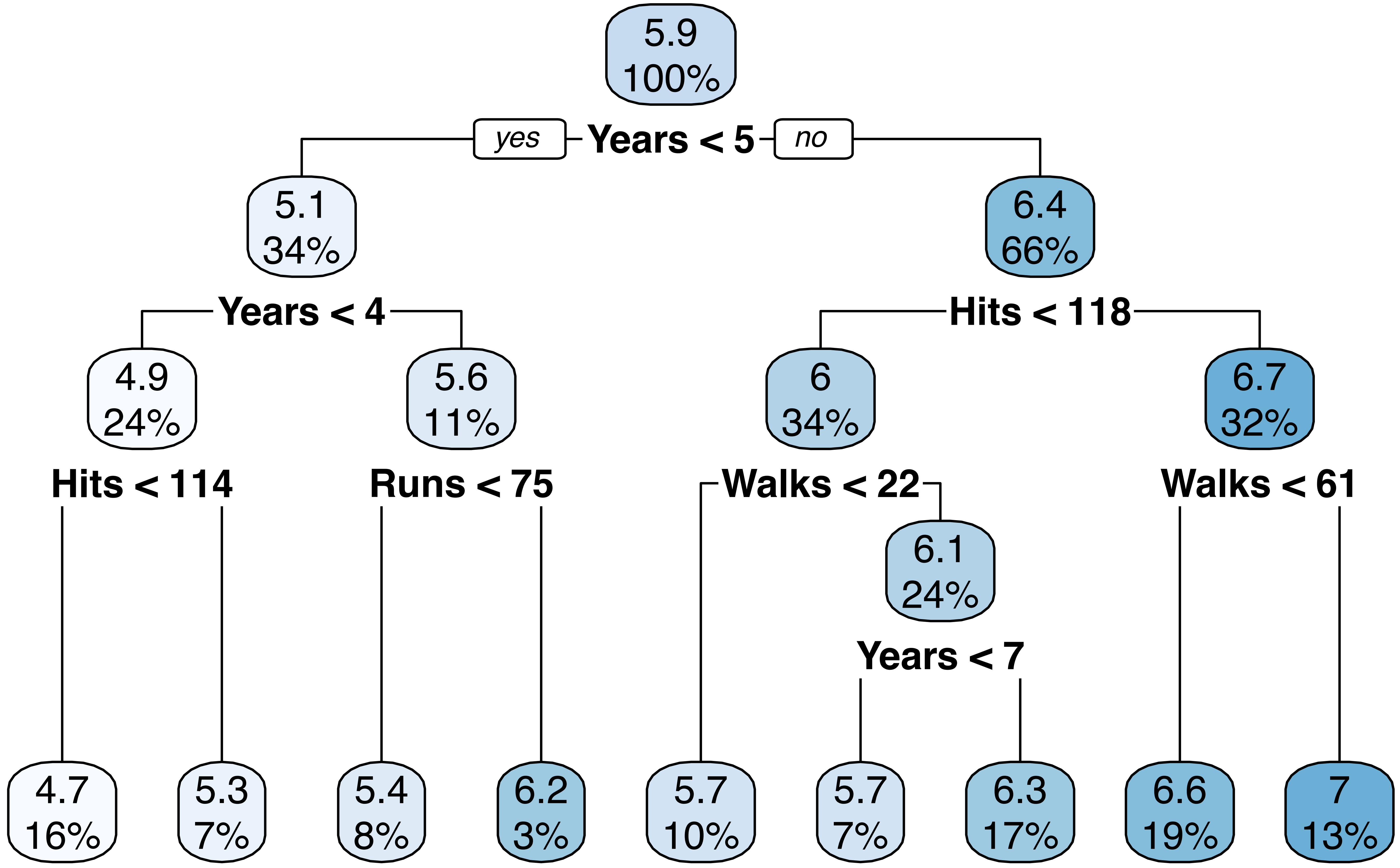
The optimal set of rectangles is computationally intractable to find. In practice, we employ a greedy top-down algorithm:

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

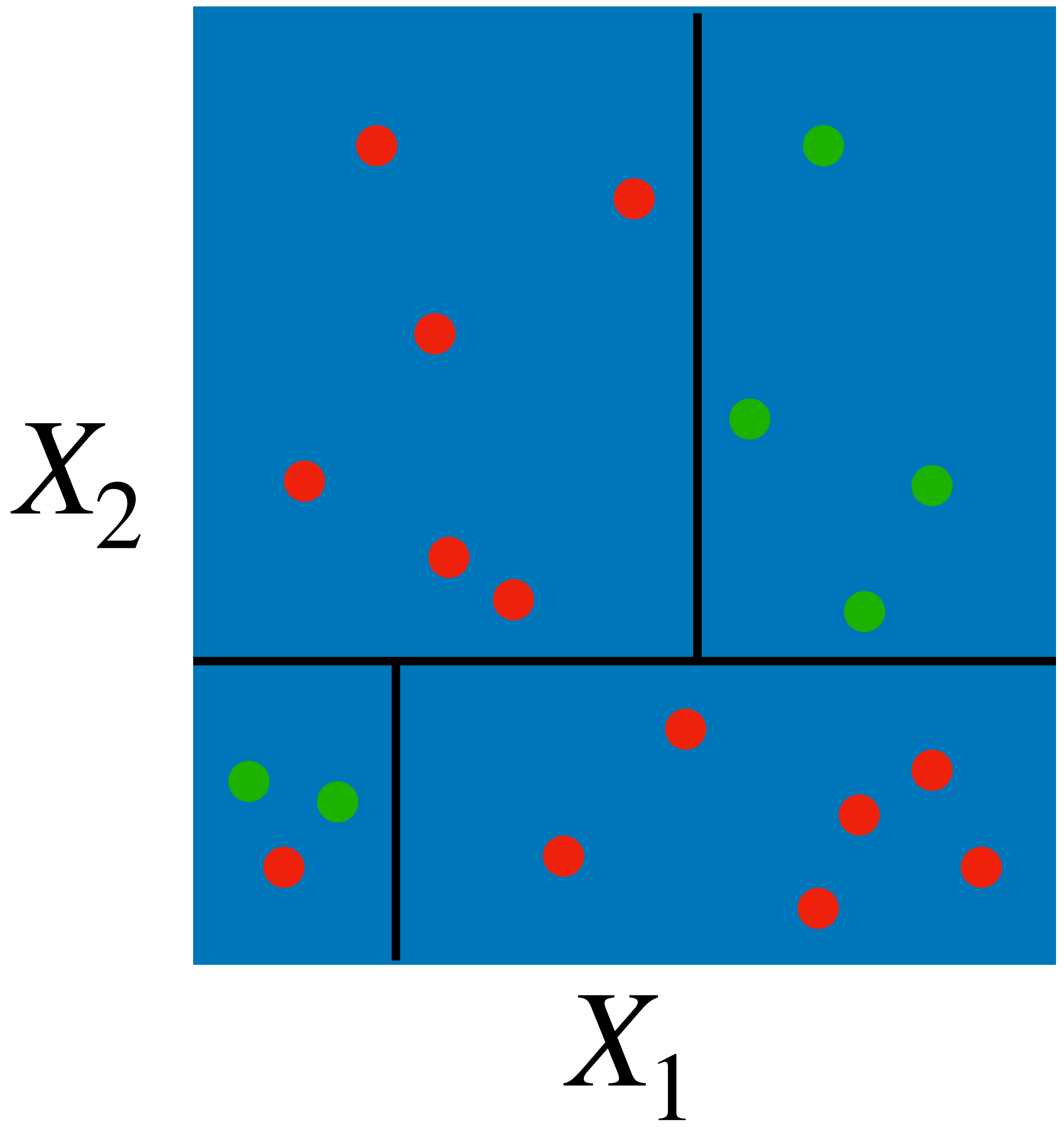
The misclassification error objective

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 $\widehat{R}_1, \dots, \widehat{R}_M$ and values $\hat{c}_1, \dots, \hat{c}_M$ to minimize the misclassification loss:

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

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



Training a classification tree

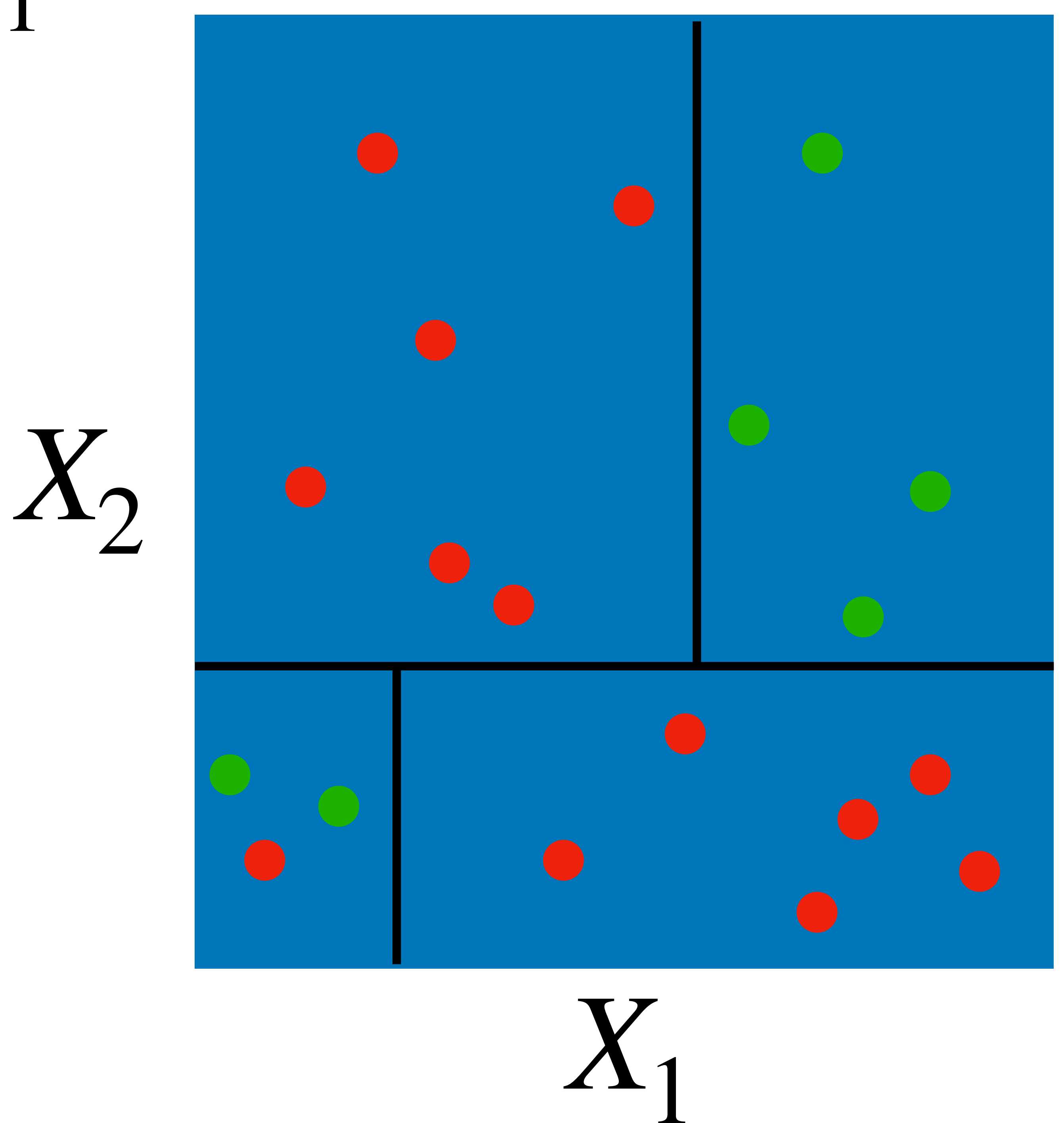
Optimal $\widehat{c}_m$ given $\widehat{R}_m$

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

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

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

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



Training a classification tree

Finding the rectangles $\widehat{R}_m$

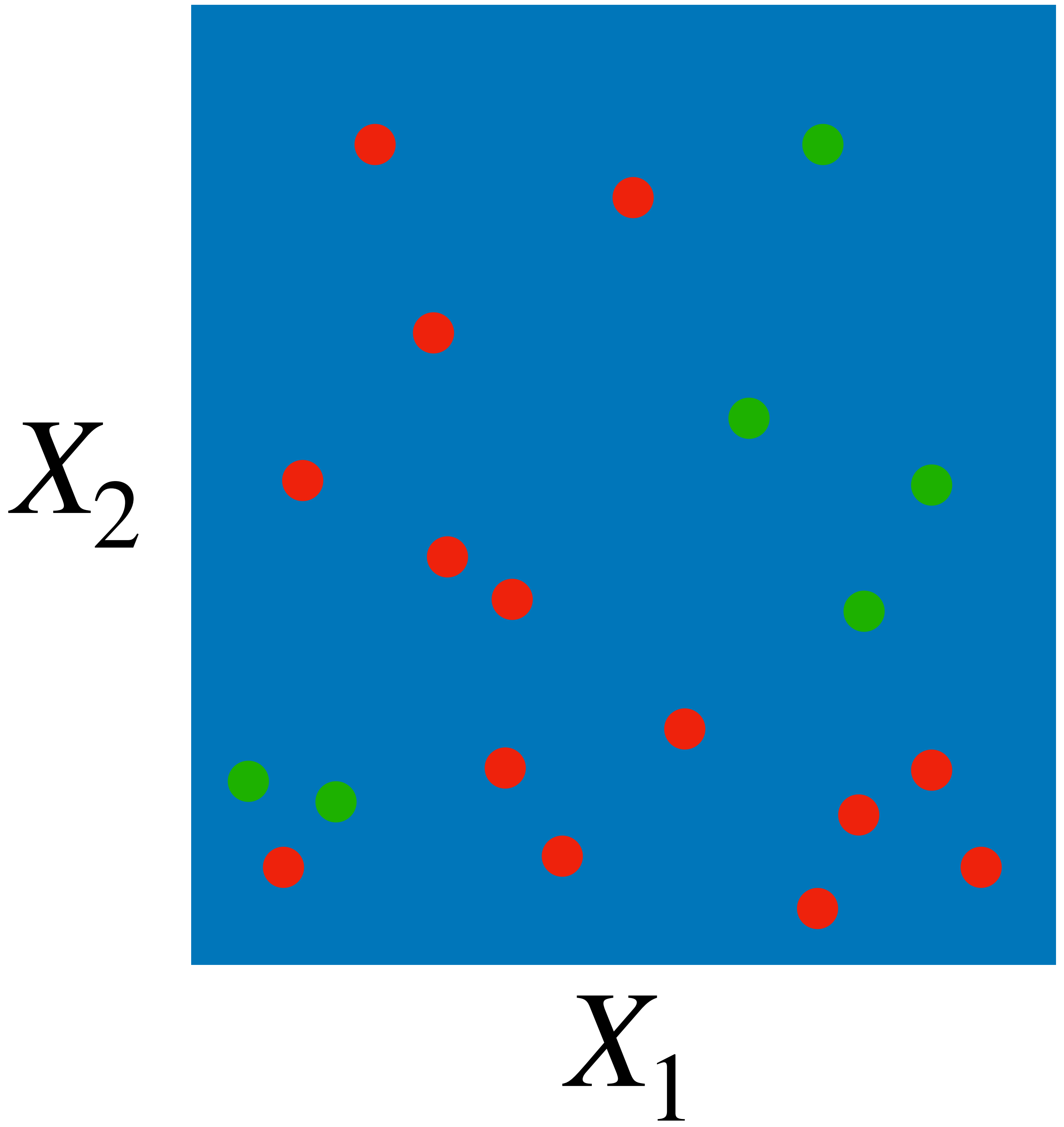
We use a very similar greedy recursive splitting algorithm.

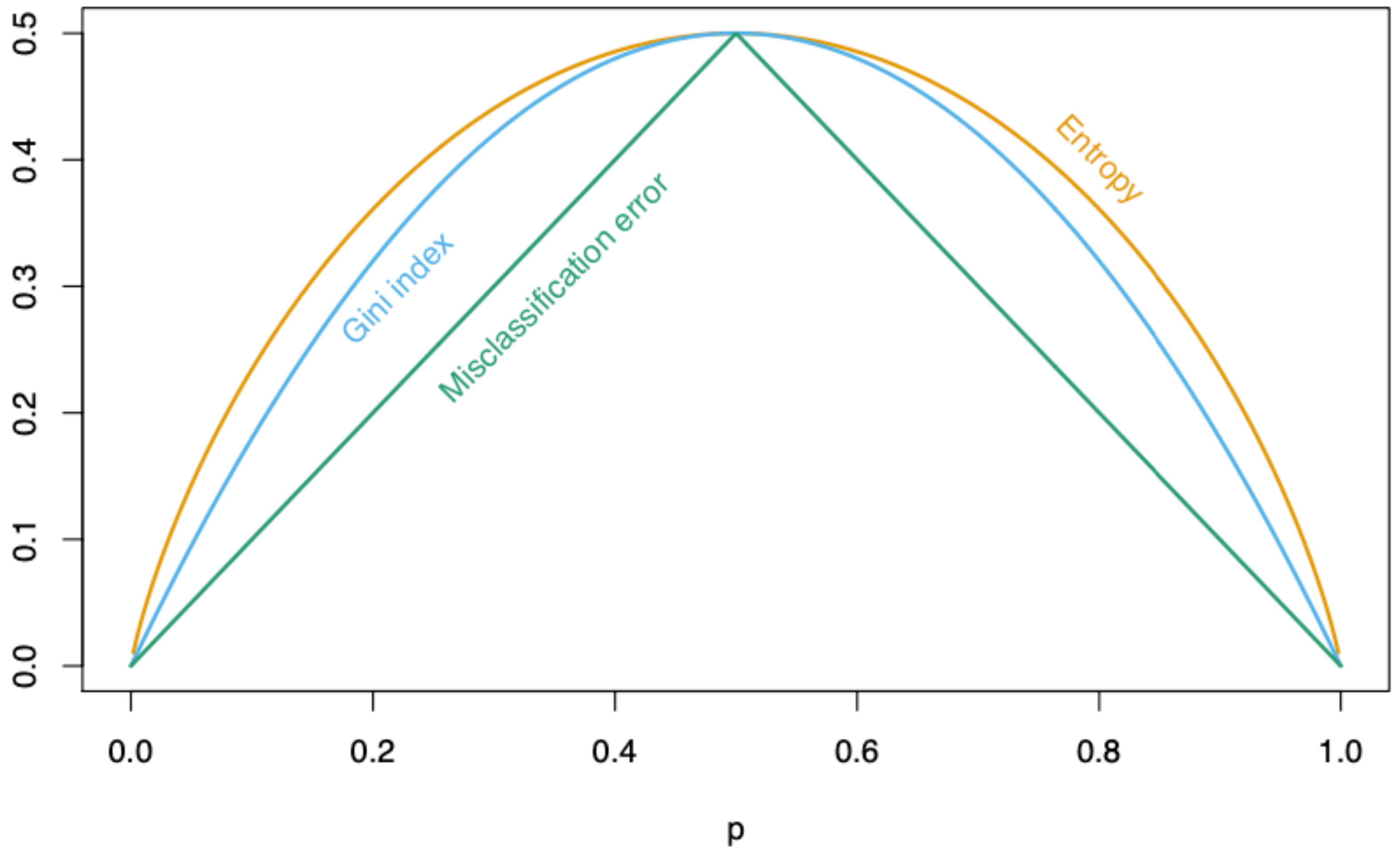
Let $\widehat{p}_m$ be proportion of class 1 in region m. The misclassification error in that region is $\min(\widehat{p}_m, 1 - \widehat{p}_m)$.

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

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

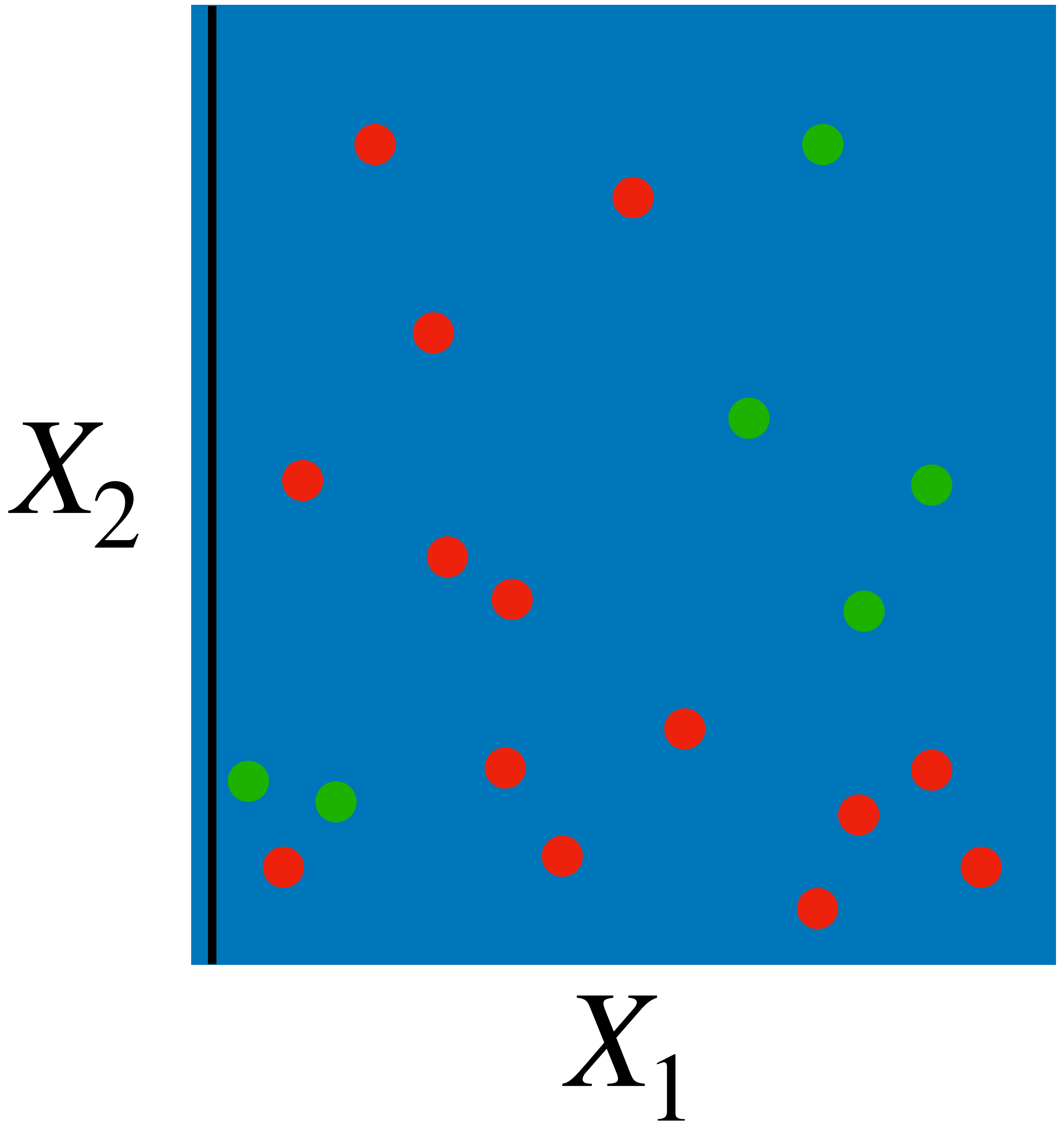
Find split that minimizes the total impurity of the regions.





Training a classification tree

Finding the rectangles $\widehat{R}_m$

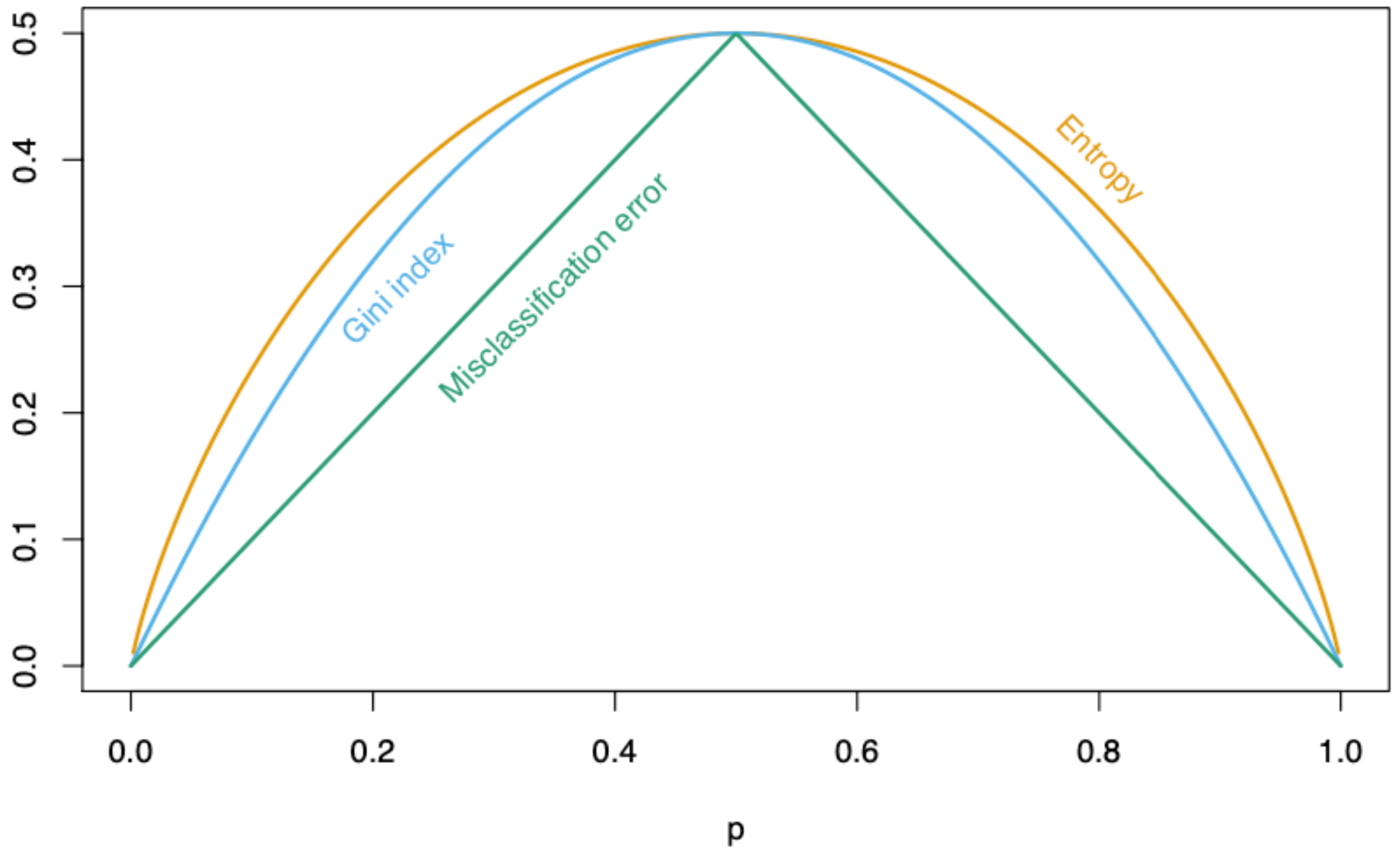


We use a very similar greedy recursive splitting algorithm.

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

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

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



Find split that minimizes the total impurity of the regions.

Training a classification tree

Finding the rectangles $\widehat{R}_m$

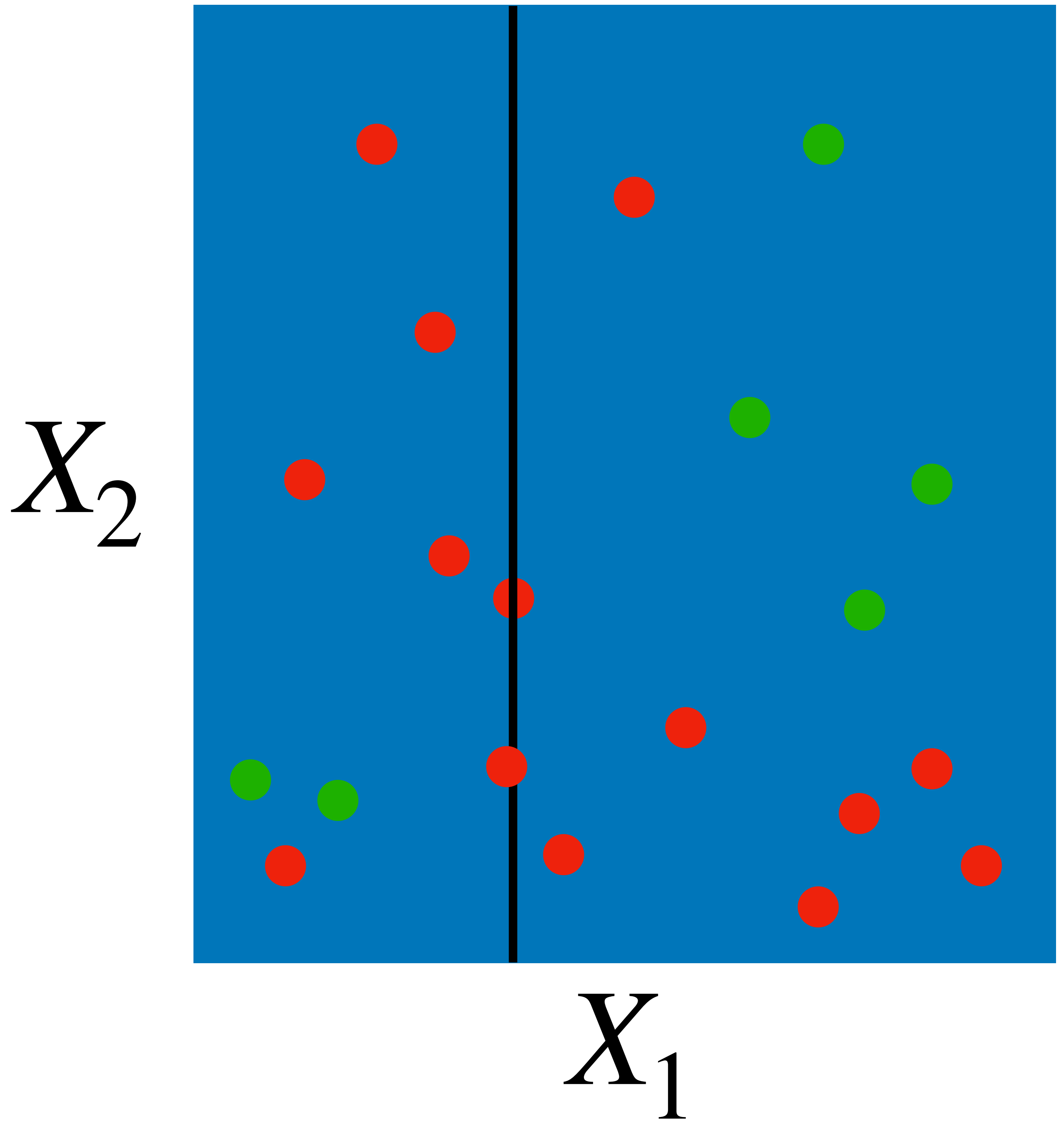
We use a very similar greedy recursive splitting algorithm.

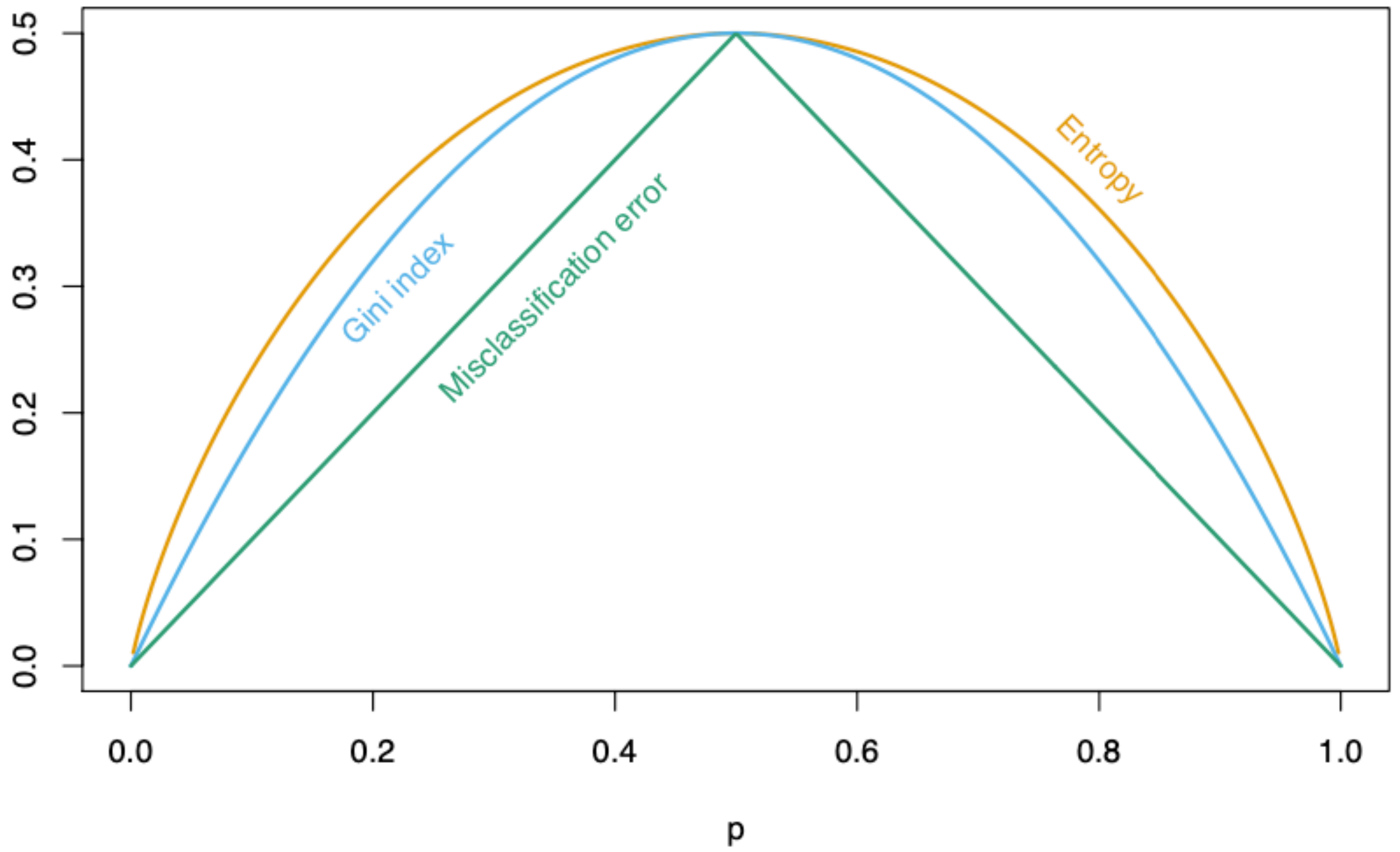
Let $\hat{p}_m$ be proportion of class 1 in region m. The misclassification error in that region is $\min(\hat{p}_m, 1 - \hat{p}_m)$.

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

Finding the rectangles $\widehat{R}_m$

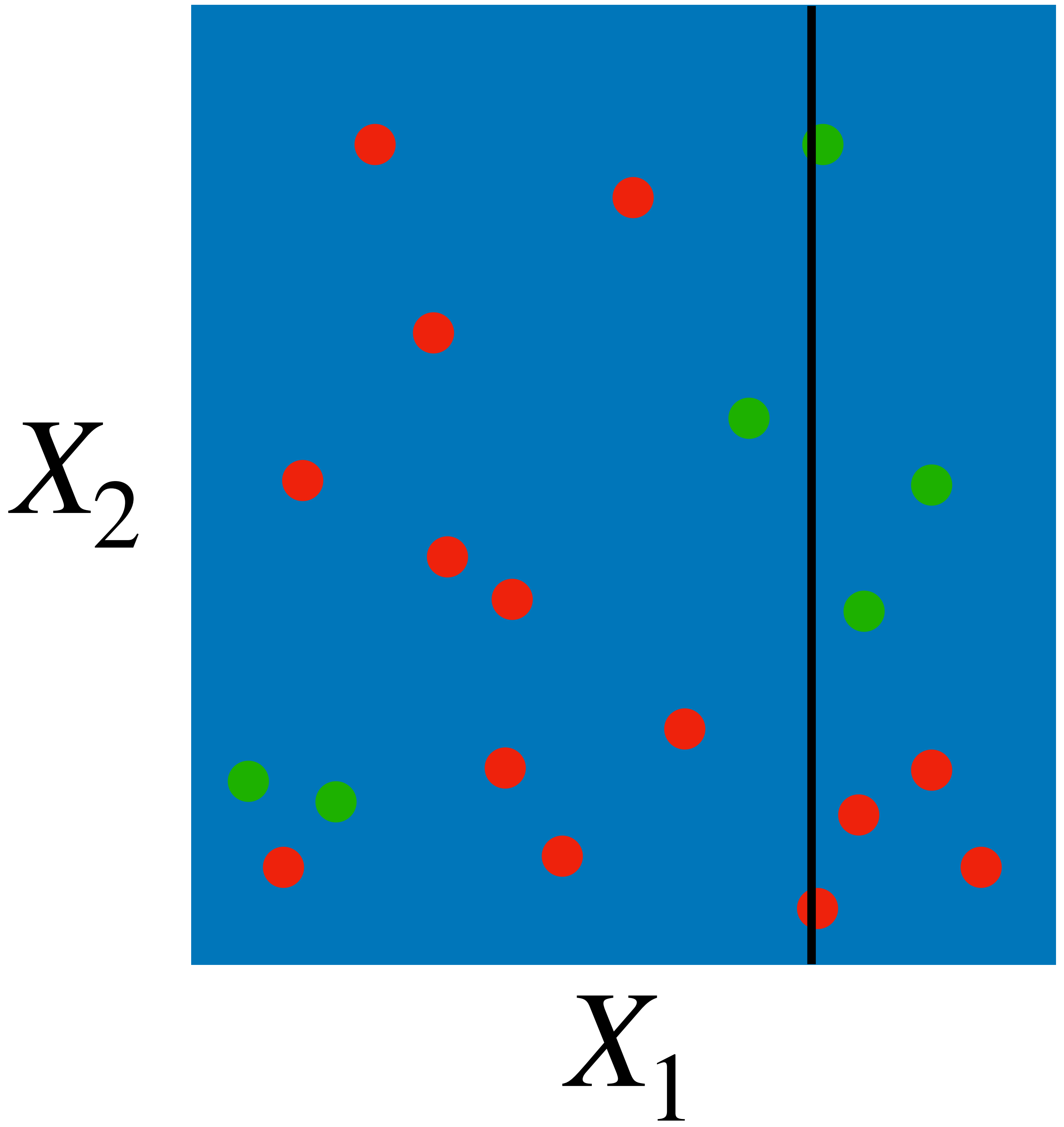
We use a very similar greedy recursive splitting algorithm.

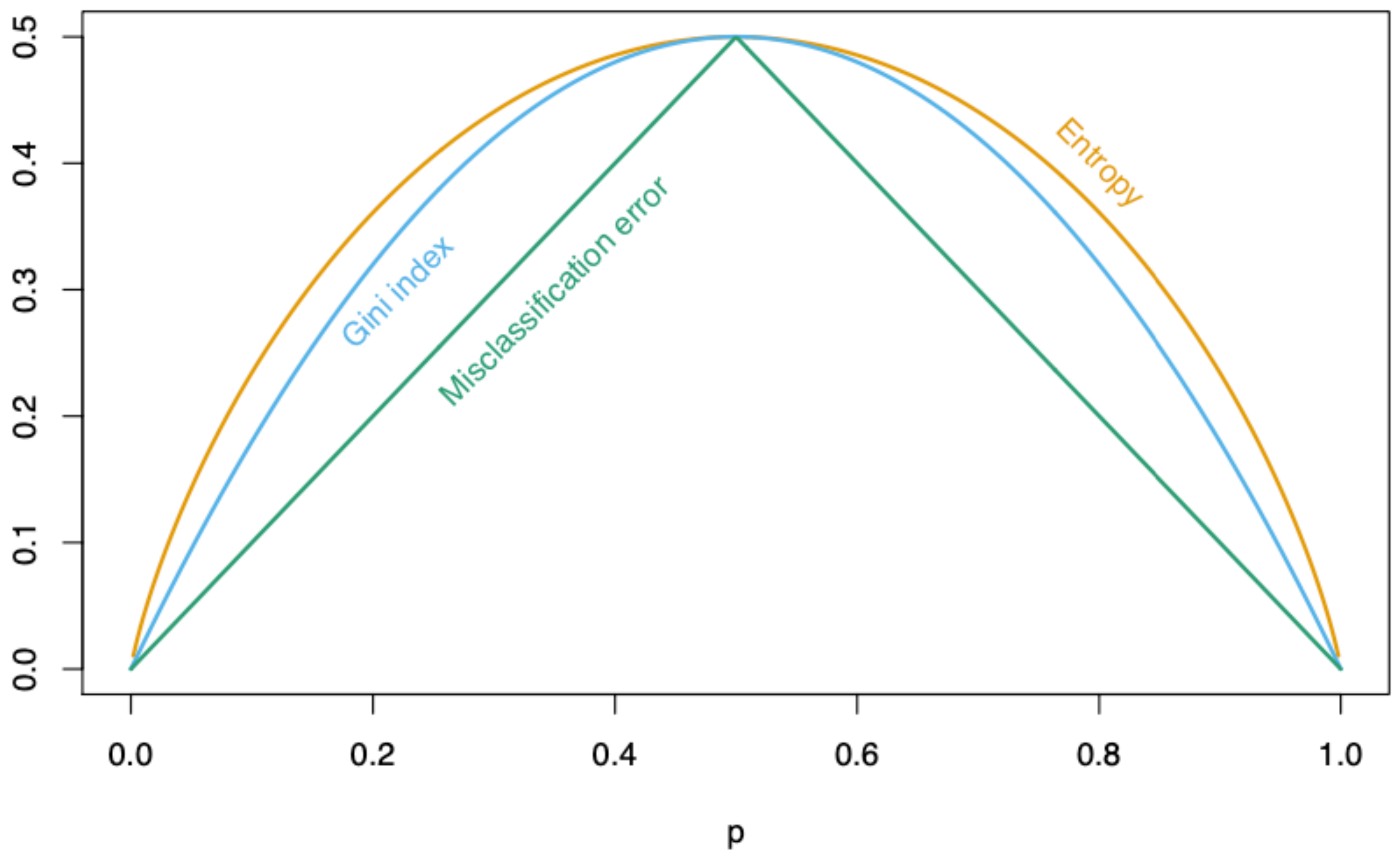
Let $\widehat{p}_m$ be proportion of class 1 in region m. The misclassification error in that region is $\min(\widehat{p}_m, 1 - \widehat{p}_m)$.

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

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

Find split that minimizes the total impurity of the regions.





Training a classification tree

Finding the rectangles $\widehat{R}_m$

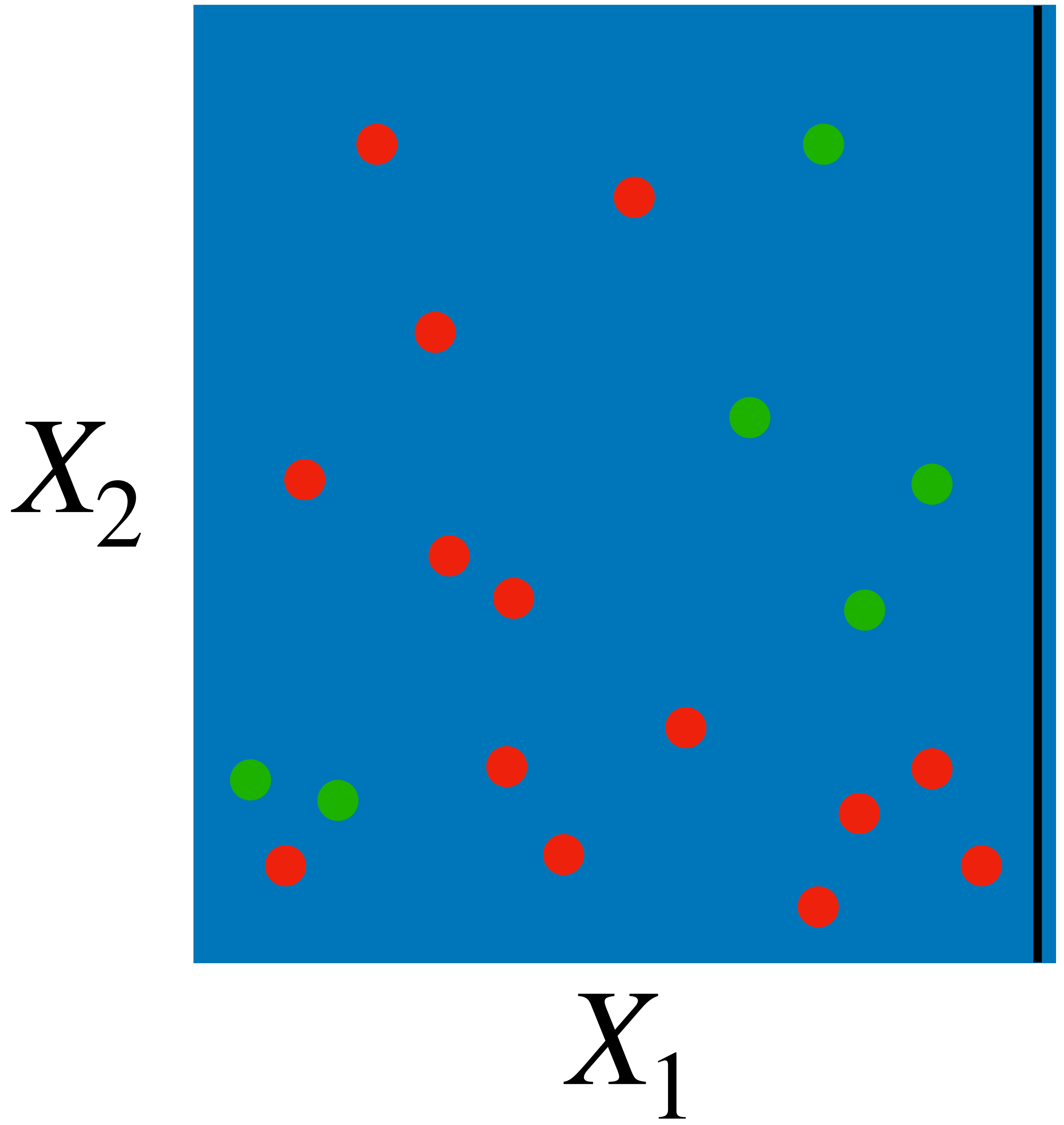
We use a very similar greedy recursive splitting algorithm.

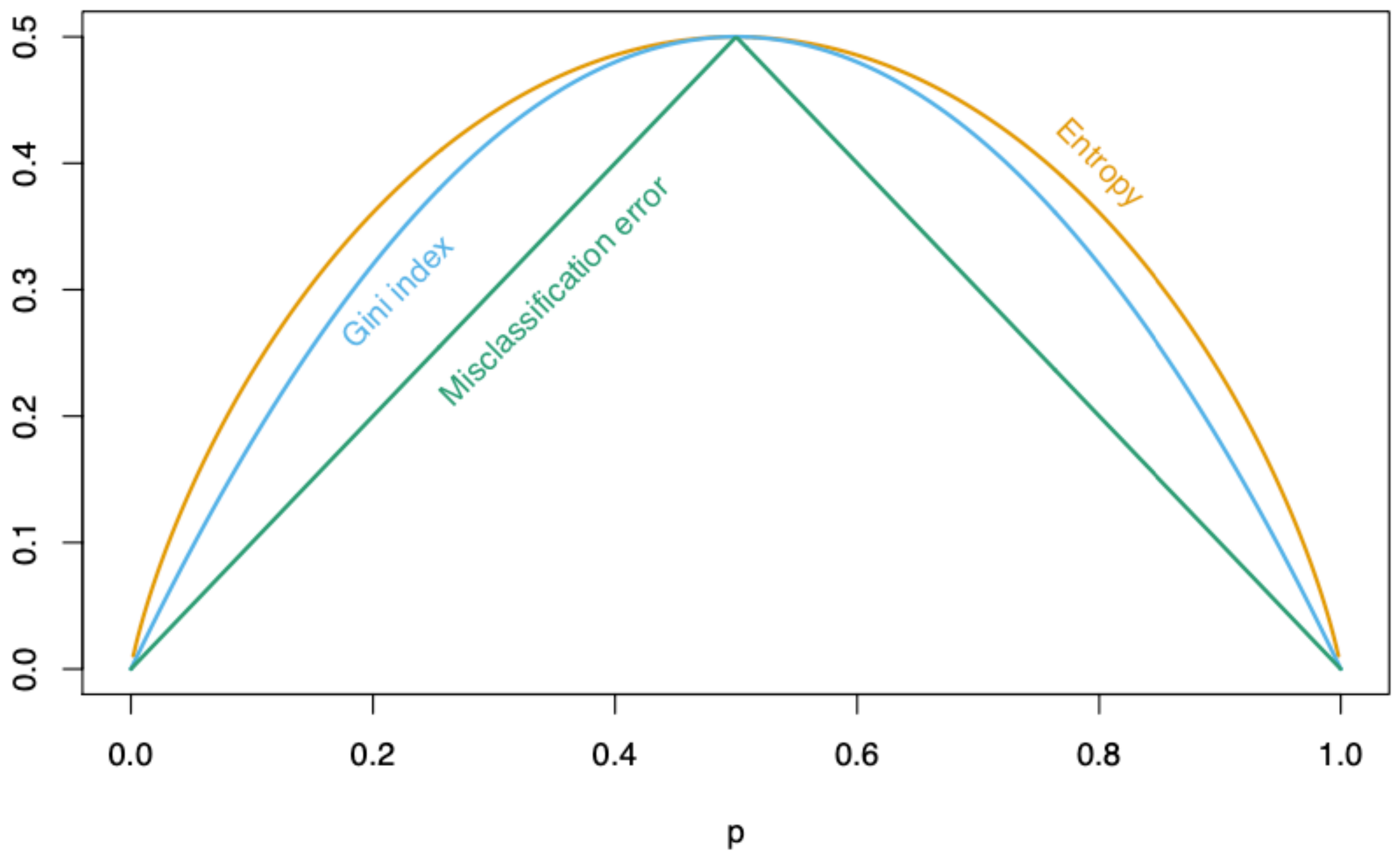
Let $\hat{p}_m$ be proportion of class 1 in region m. The misclassification error in that region is $\min(\hat{p}_m, 1 - \hat{p}_m)$.

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

Finding the rectangles $\widehat{R}_m$

We use a very similar greedy recursive splitting algorithm.

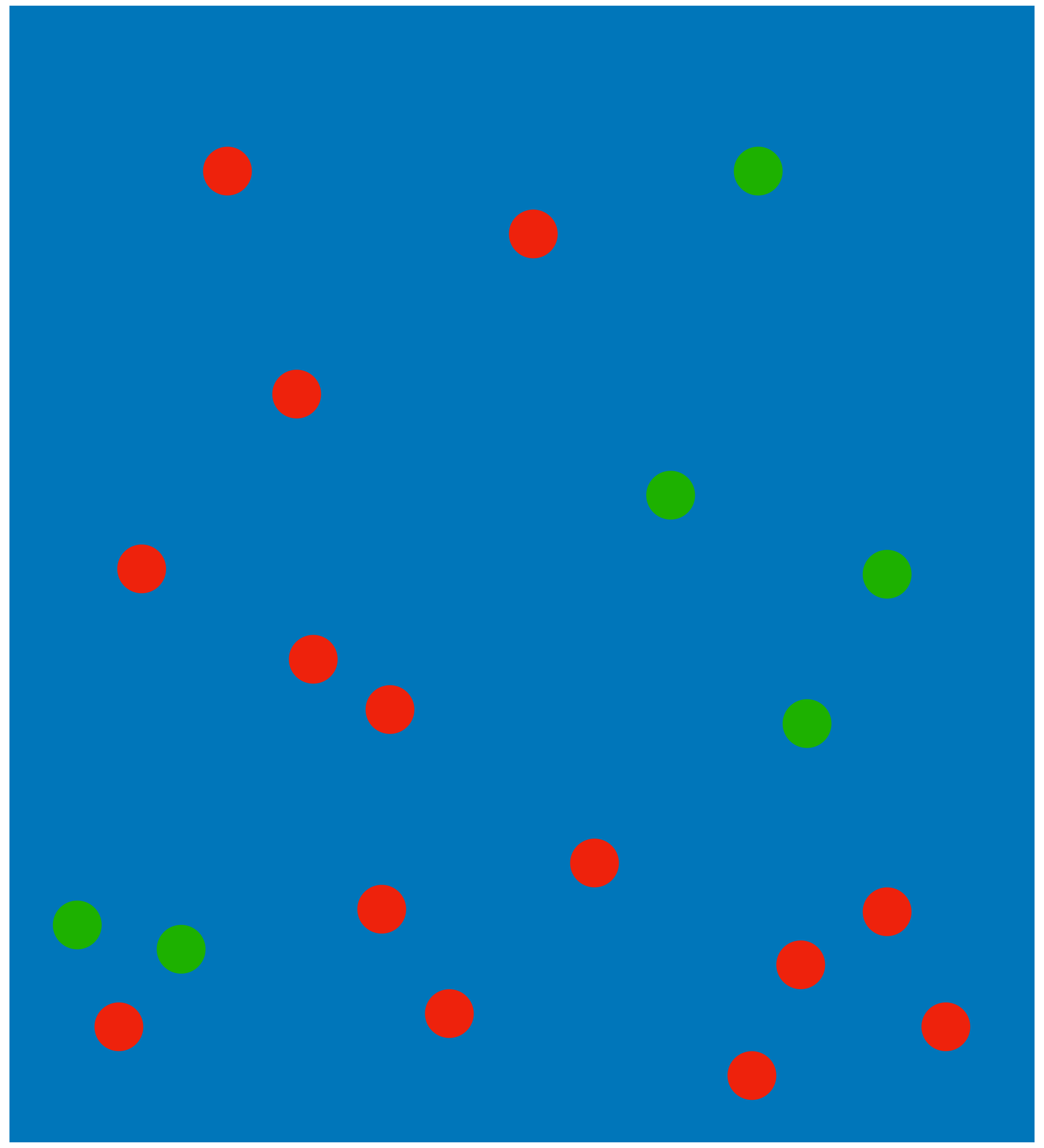
Let $\widehat{p}_m$ be proportion of class 1 in region m. The misclassification error in that region is $\min(\widehat{p}_m, 1 - \widehat{p}_m)$.

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

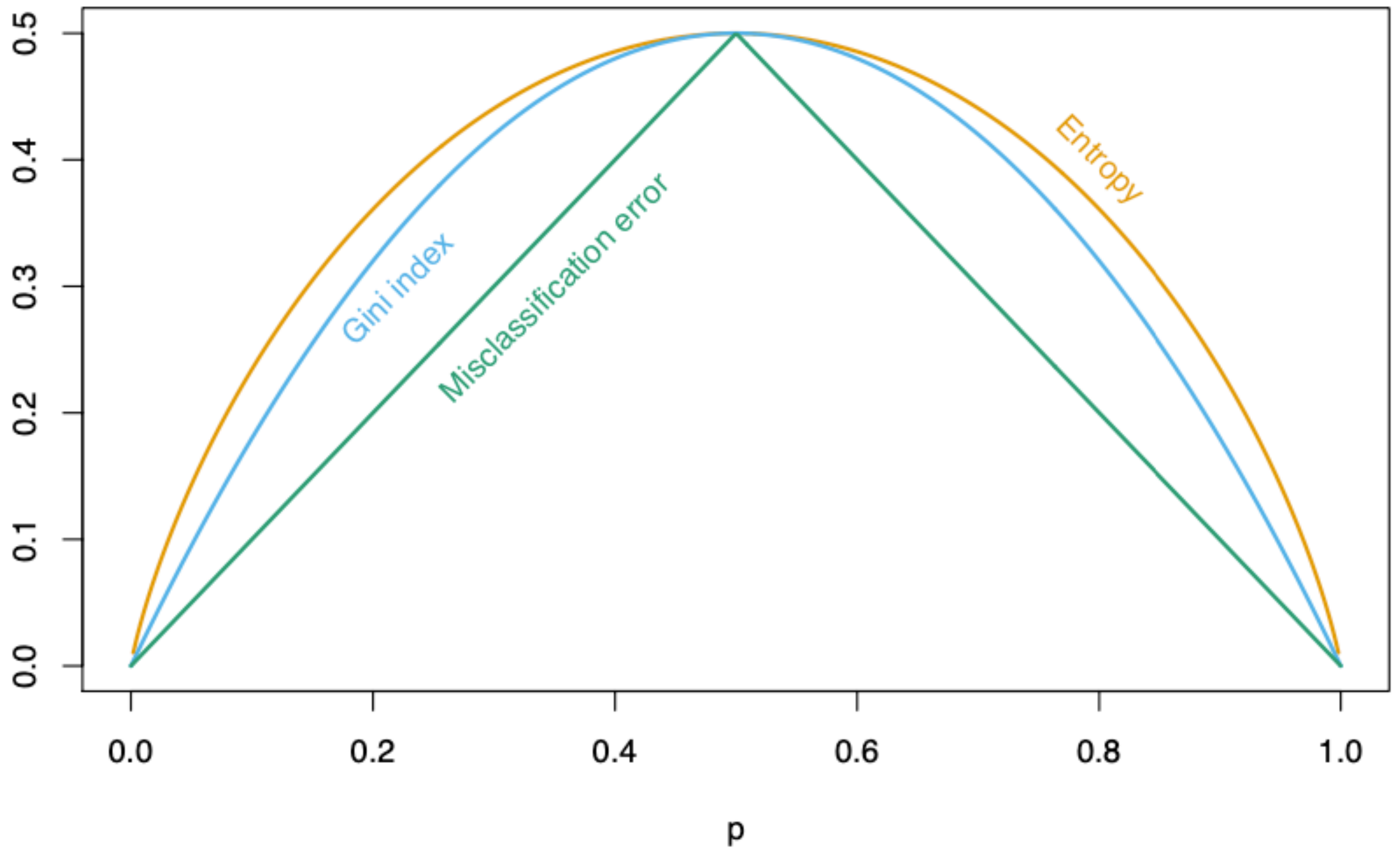
$$\text{Gini index} = 2\widehat{p}_m(1 - \widehat{p}_m).$$

Find split that minimizes the total impurity of the regions.

X_2



X_1



Training a classification tree

Finding the rectangles $\widehat{R}_m$

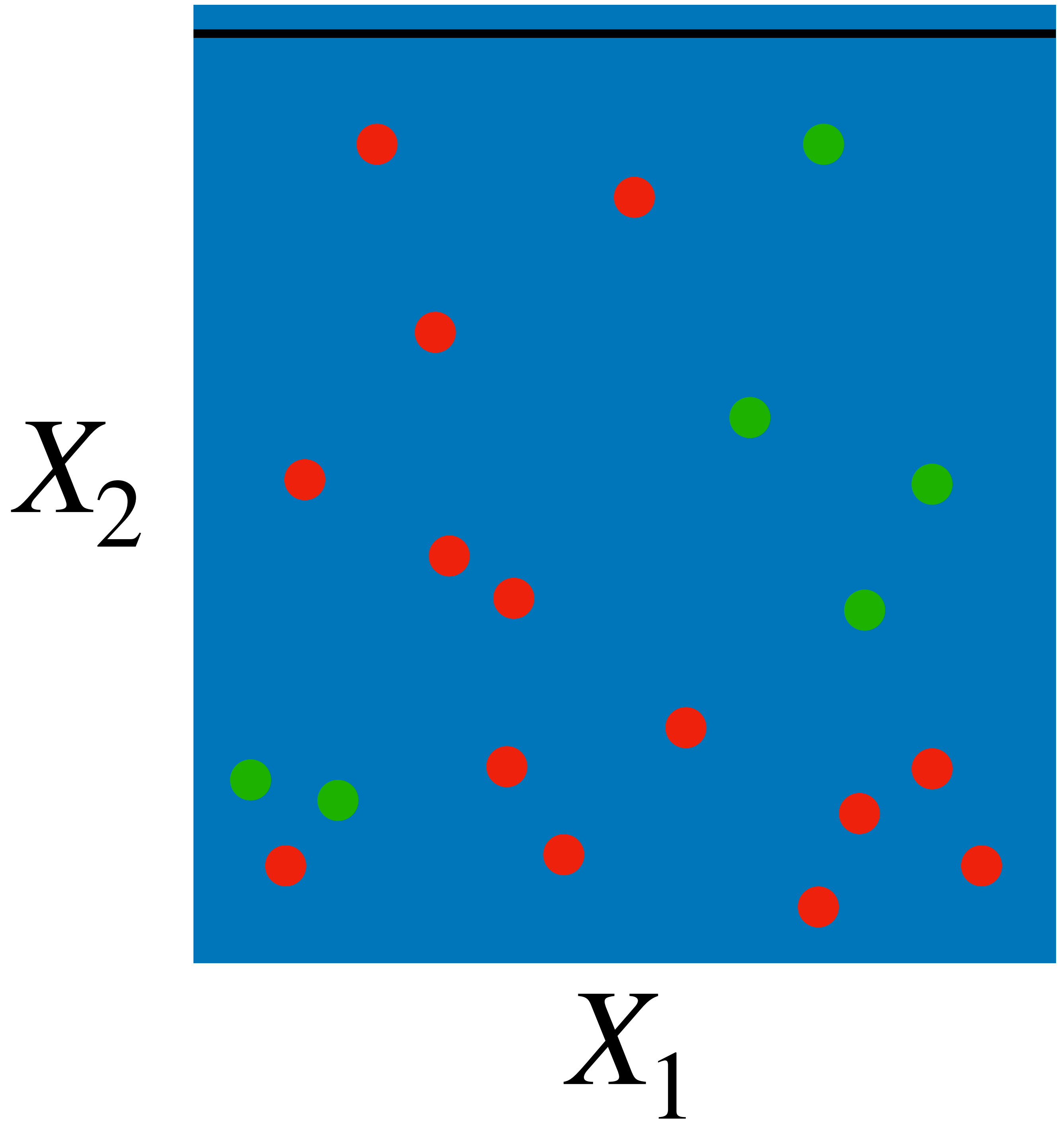
We use a very similar greedy recursive splitting algorithm.

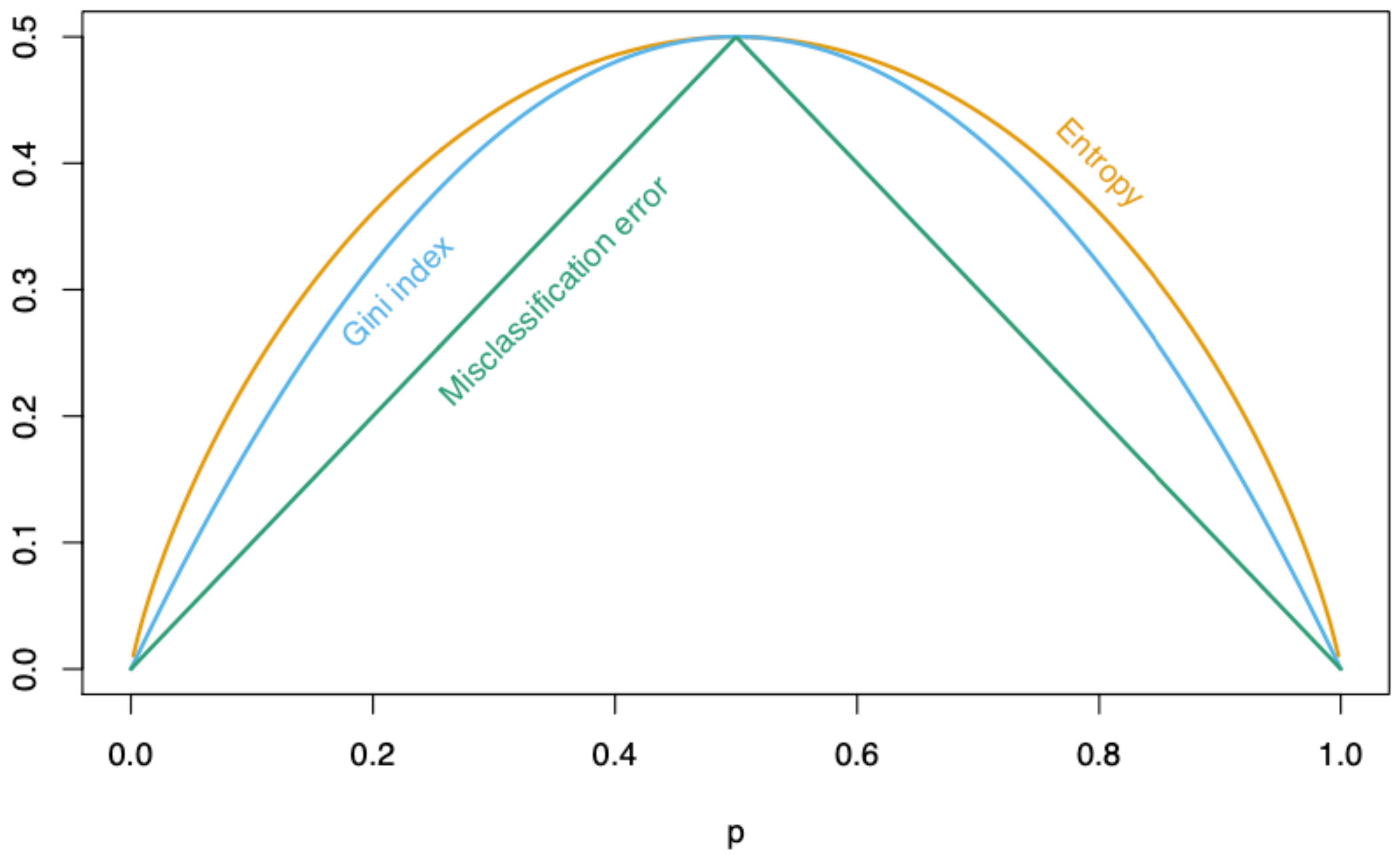
Let $\widehat{p}_m$ be proportion of class 1 in region m. The misclassification error in that region is $\min(\widehat{p}_m, 1 - \widehat{p}_m)$.

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

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

Find split that minimizes the total impurity of the regions.





Training a classification tree

Finding the rectangles $\widehat{R}_m$

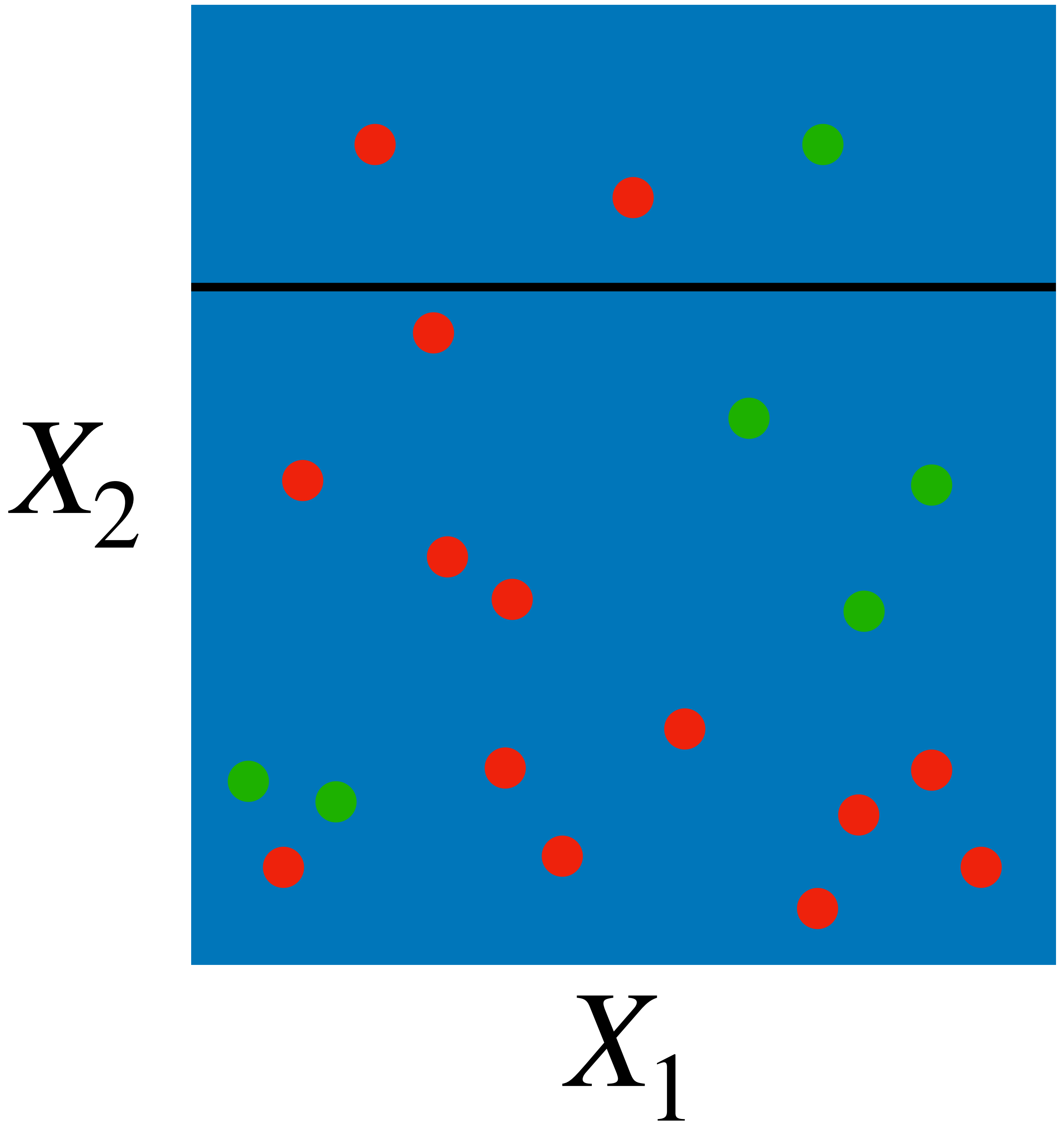
We use a very similar greedy recursive splitting algorithm.

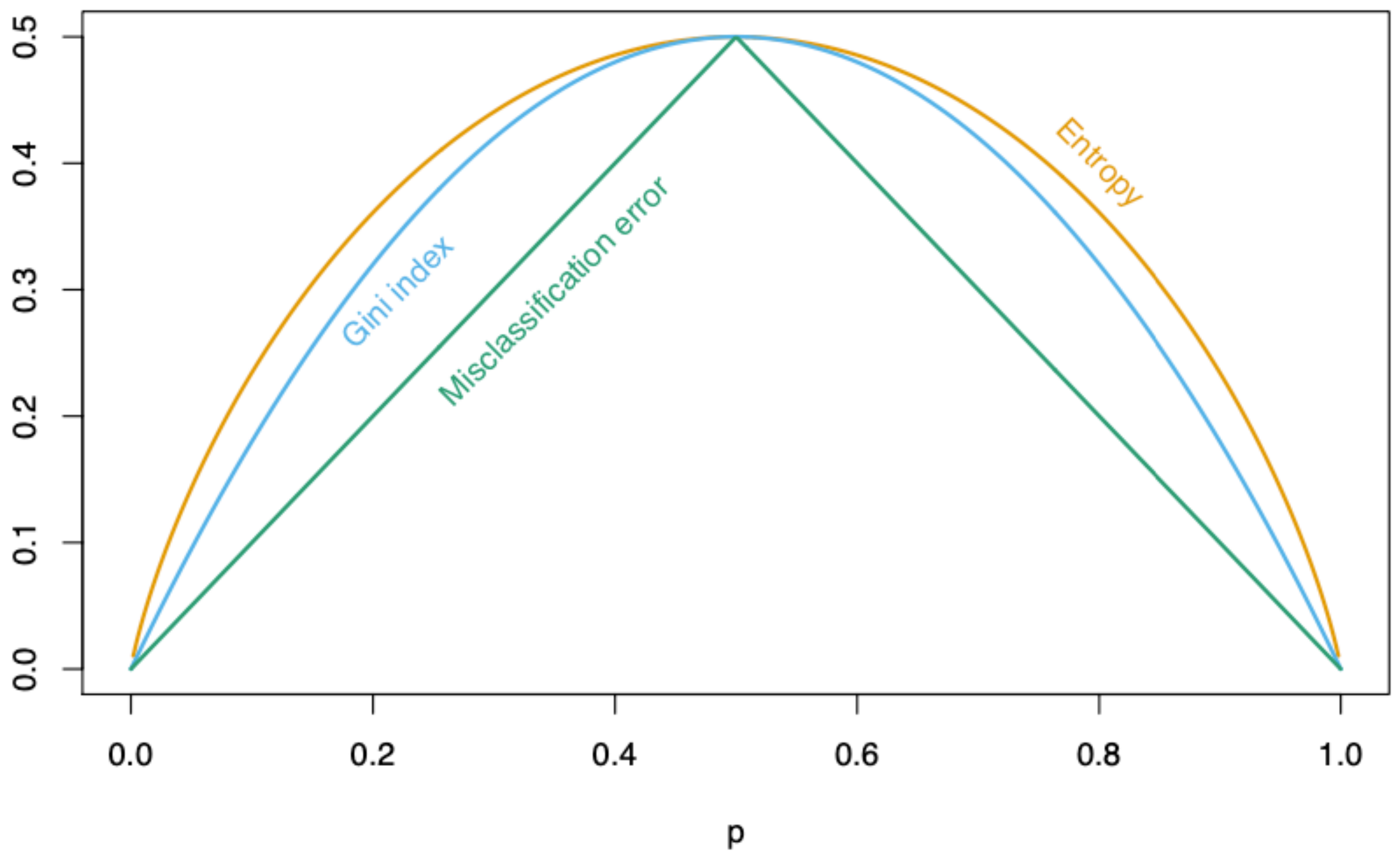
Let $\widehat{p}_m$ be proportion of class 1 in region m. The misclassification error in that region is $\min(\widehat{p}_m, 1 - \widehat{p}_m)$.

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

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

Find split that minimizes the total impurity of the regions.





Training a classification tree

Finding the rectangles $\widehat{R}_m$

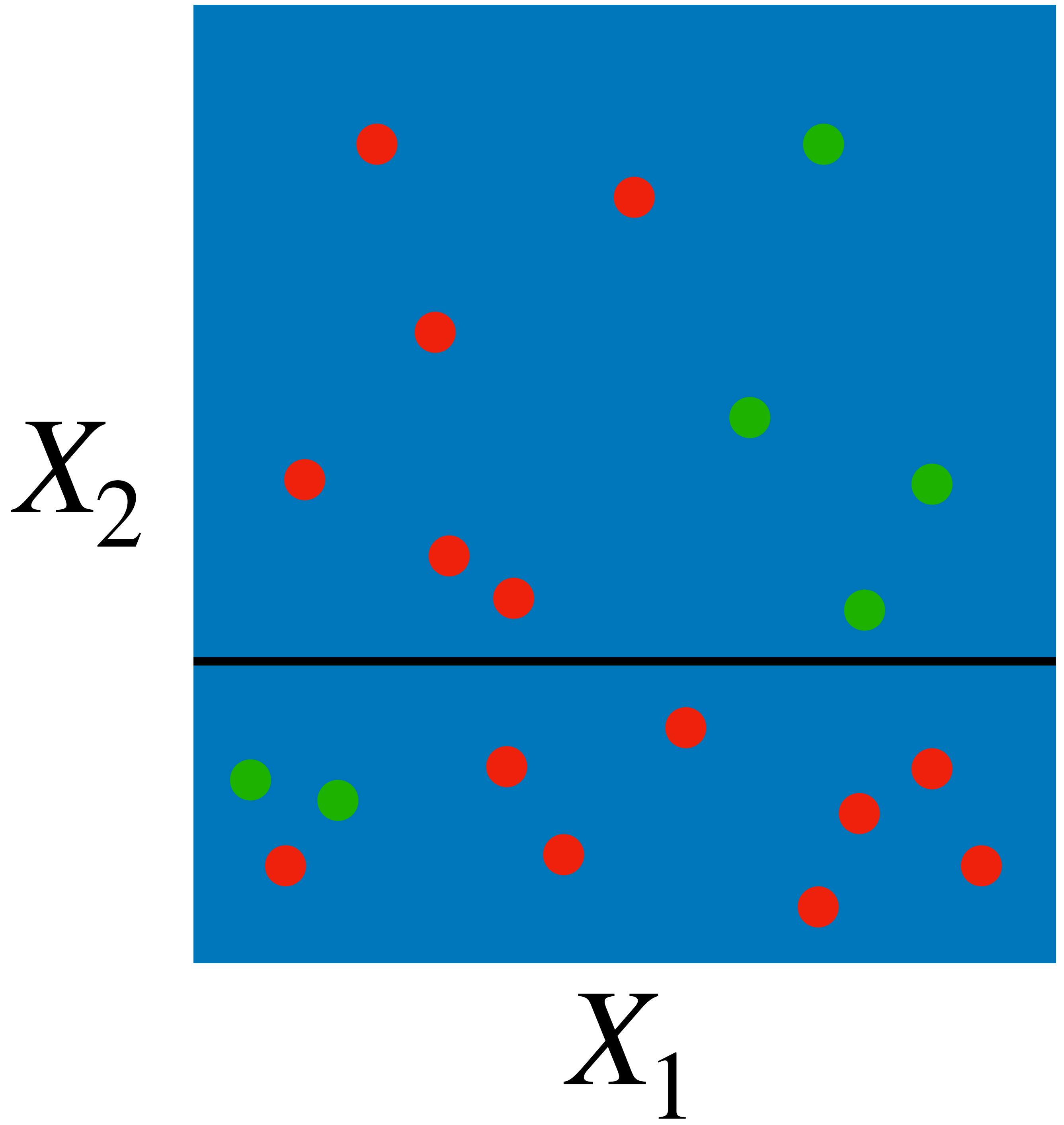
We use a very similar greedy recursive splitting algorithm.

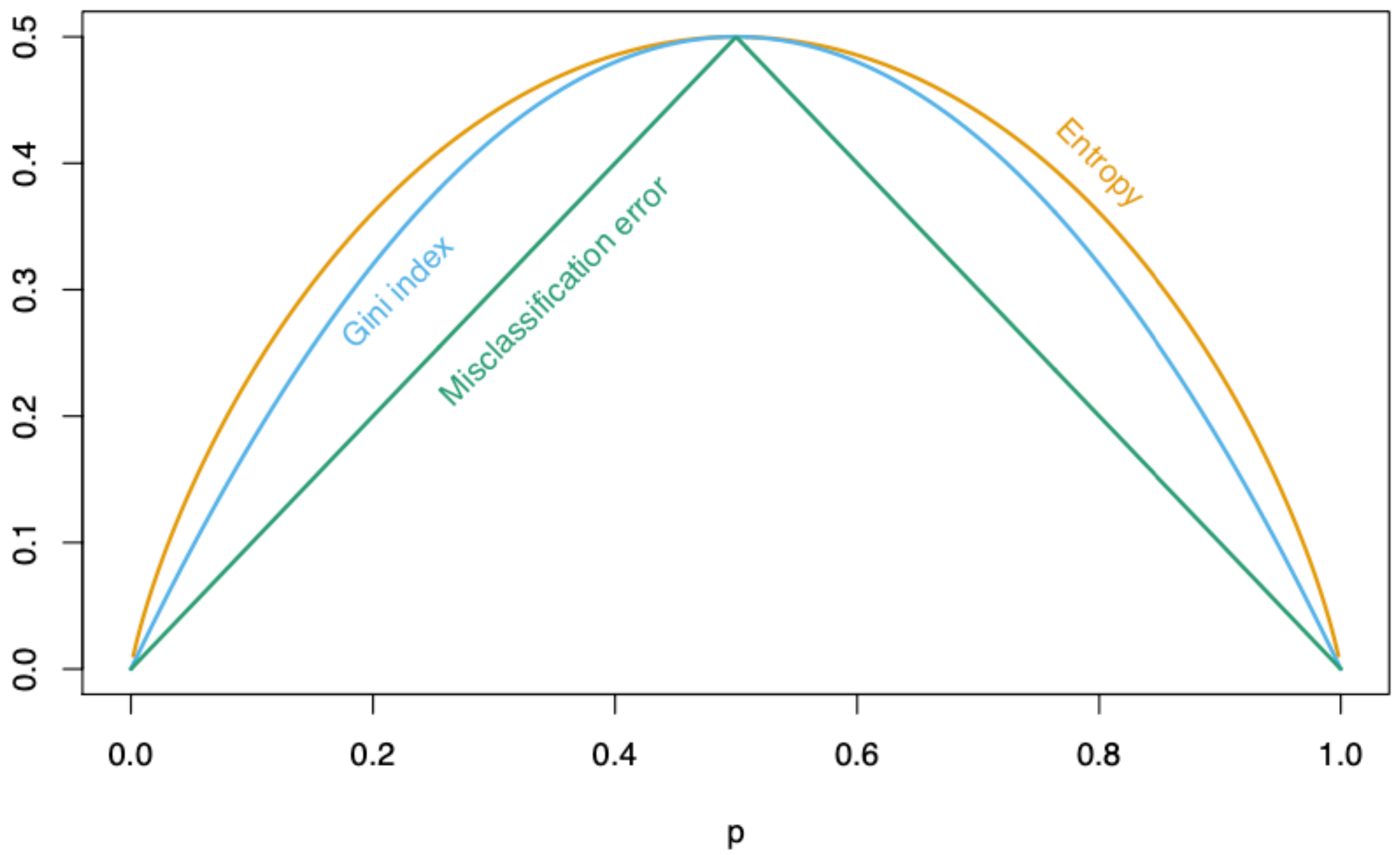
Let $\hat{p}_m$ be proportion of class 1 in region m. The misclassification error in that region is $\min(\hat{p}_m, 1 - \hat{p}_m)$.

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

Finding the rectangles $\widehat{R}_m$

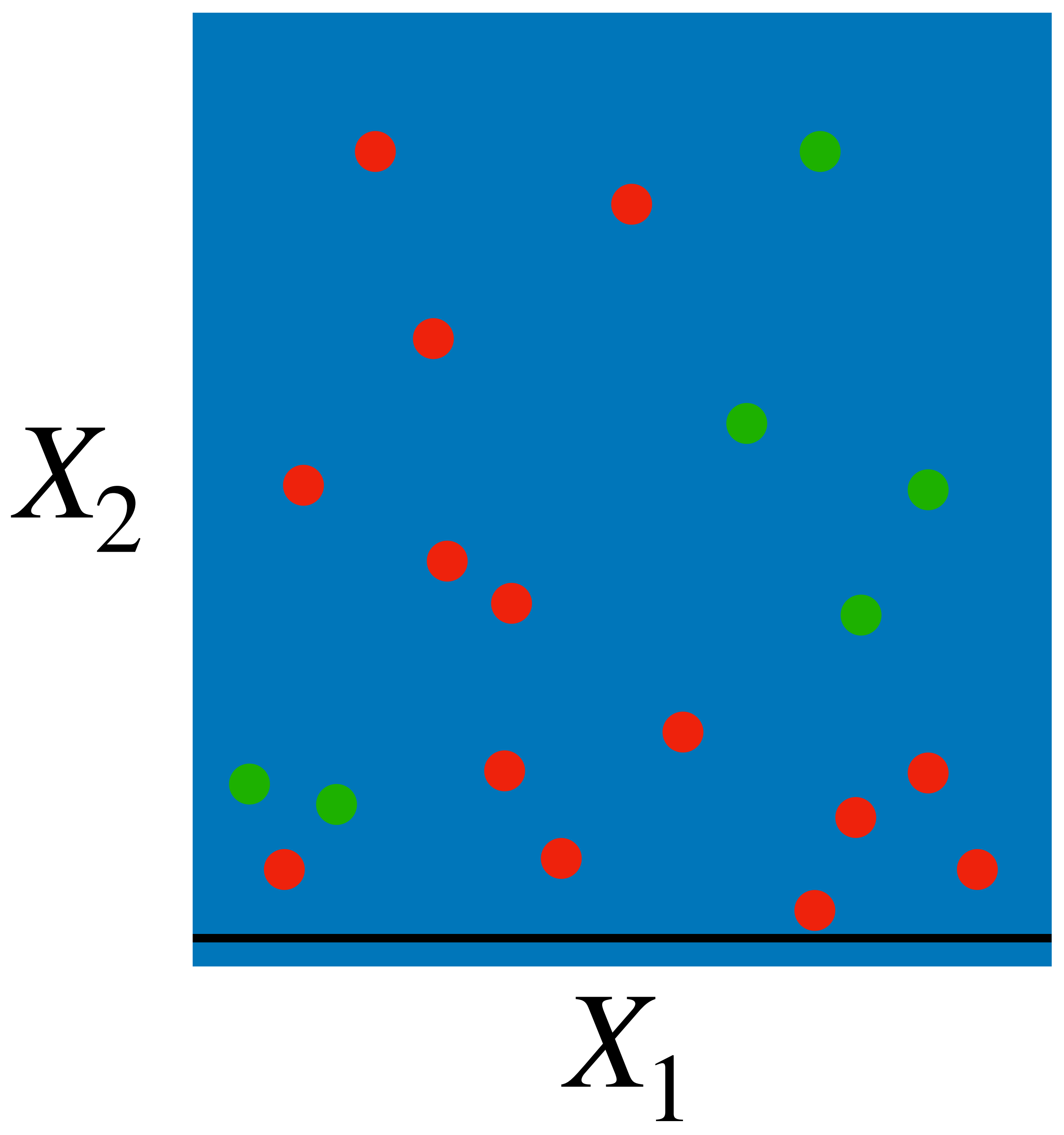
We use a very similar greedy recursive splitting algorithm.

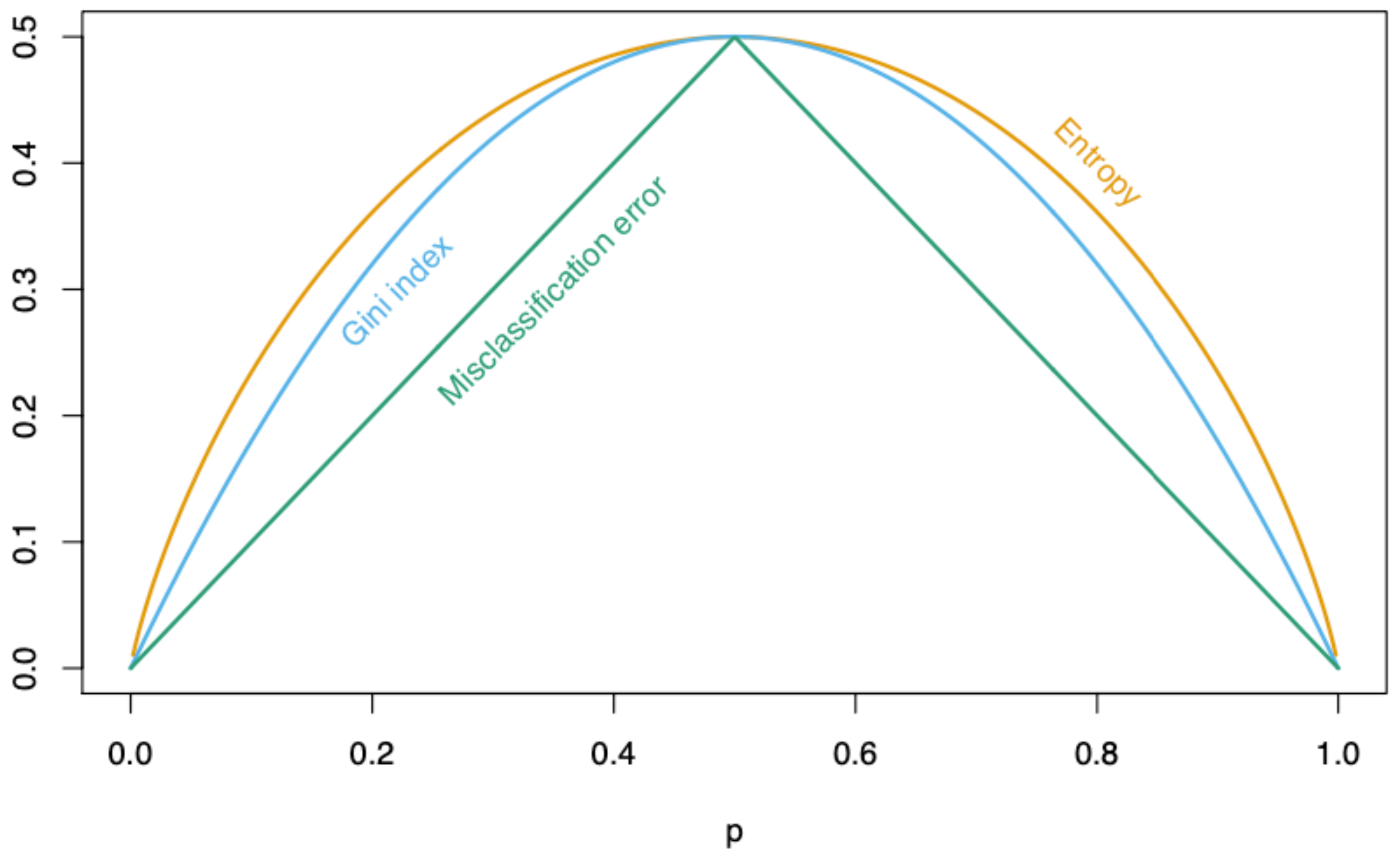
Let $\hat{p}_m$ be proportion of class 1 in region m. The misclassification error in that region is $\min(\hat{p}_m, 1 - \hat{p}_m)$.

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

Finding the rectangles $\widehat{R}_m$

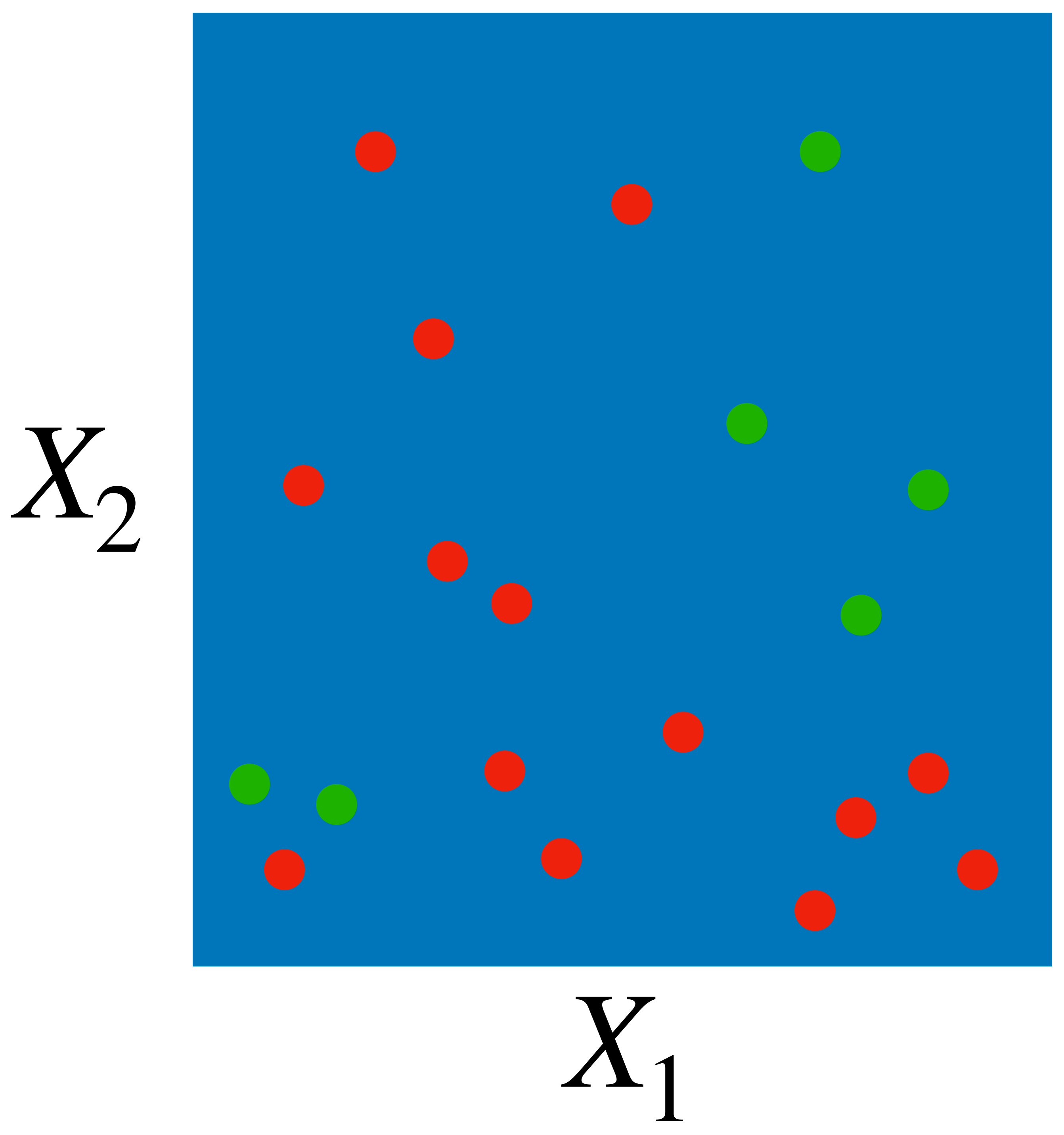
We use a very similar greedy recursive splitting algorithm.

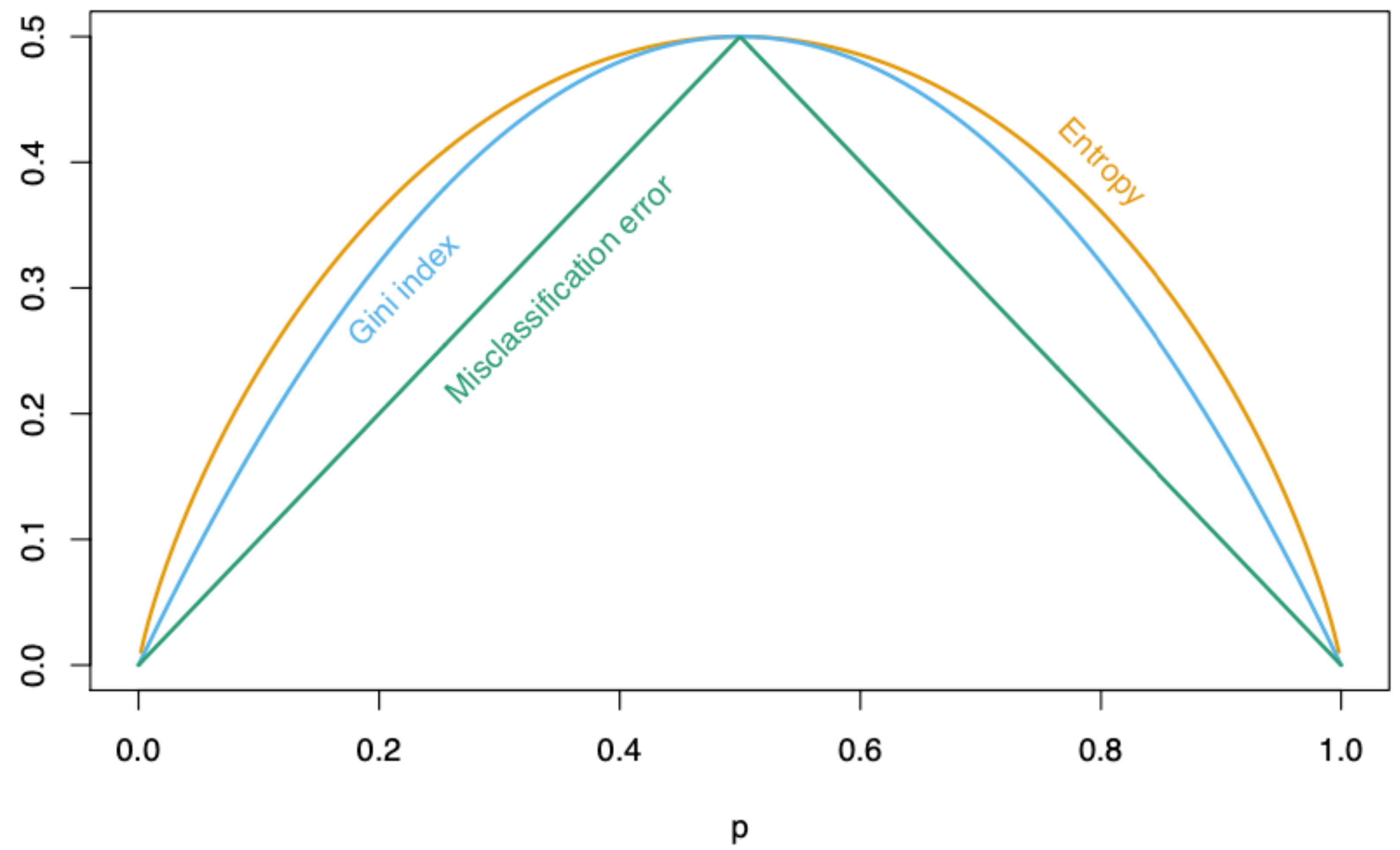
Let $\widehat{p}_m$ be proportion of class 1 in region m. The misclassification error in that region is $\min(\widehat{p}_m, 1 - \widehat{p}_m)$.

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

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

Find split that minimizes the total impurity of the regions.





Training a classification tree

Finding the rectangles $\widehat{R}_m$

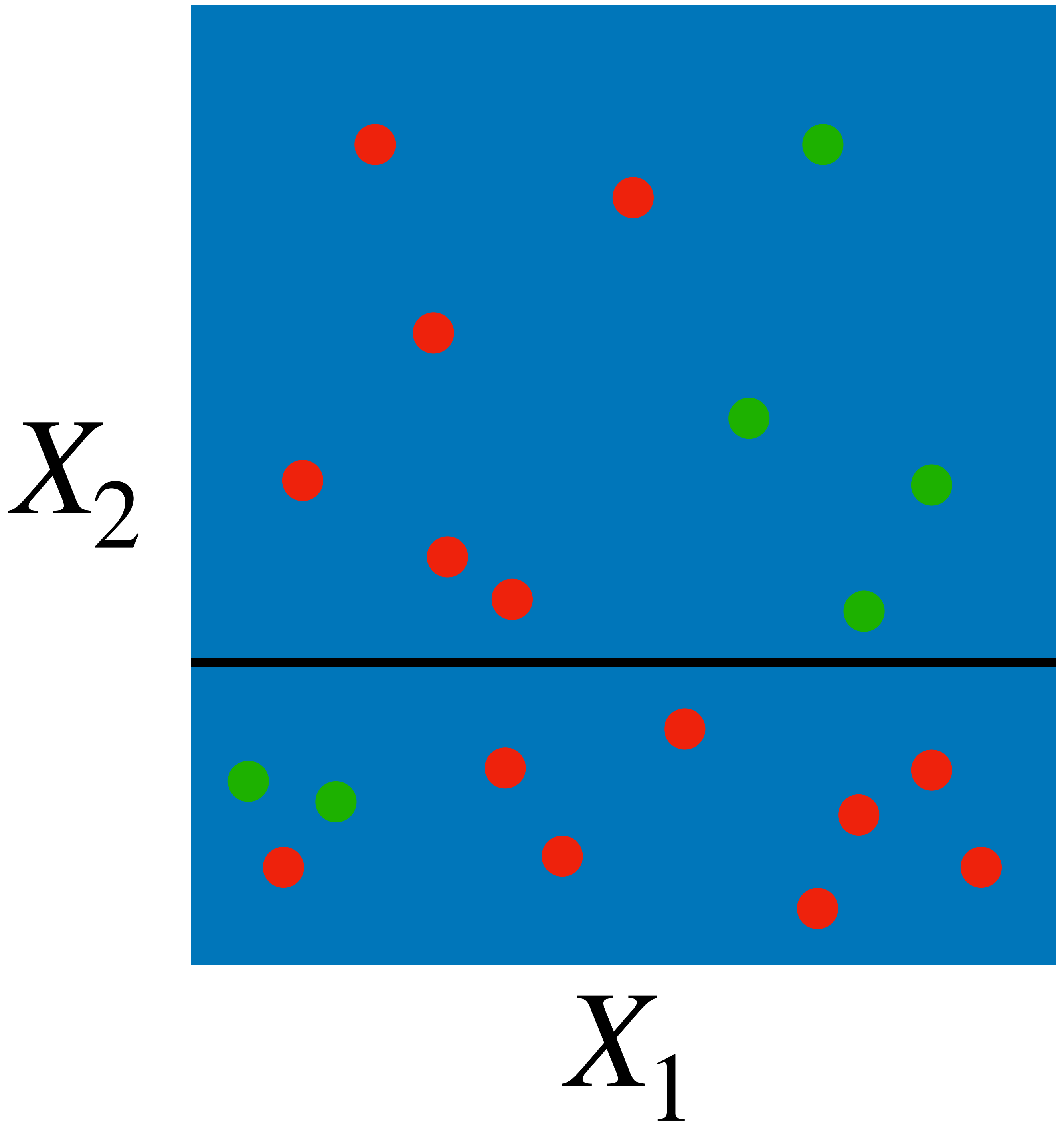
We use a very similar greedy recursive splitting algorithm.

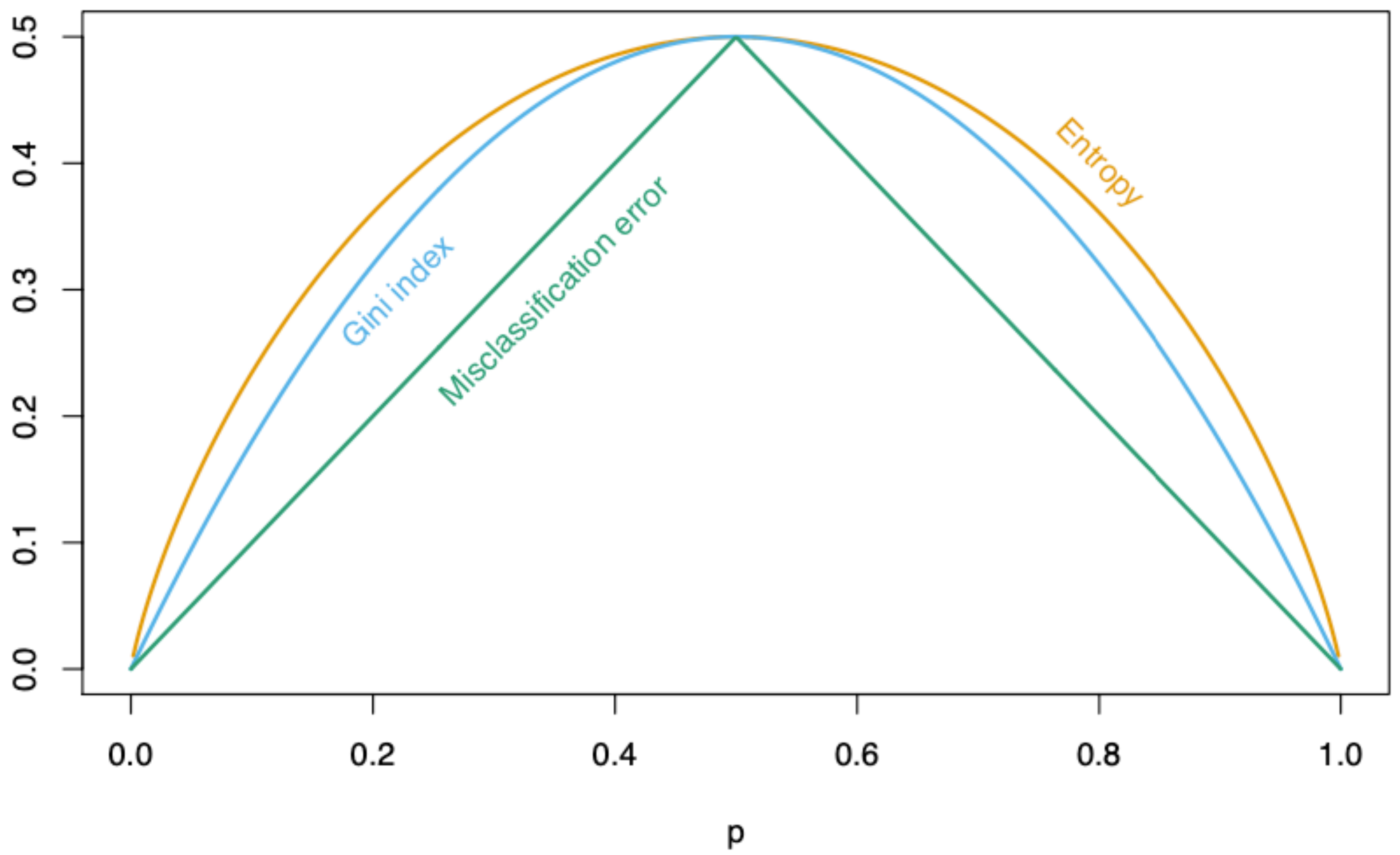
Let $\widehat{p}_m$ be proportion of class 1 in region m. The misclassification error in that region is $\min(\widehat{p}_m, 1 - \widehat{p}_m)$.

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

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

Find split that minimizes the total impurity of the regions.





Training a classification tree

Finding the rectangles $\widehat{R}_m$

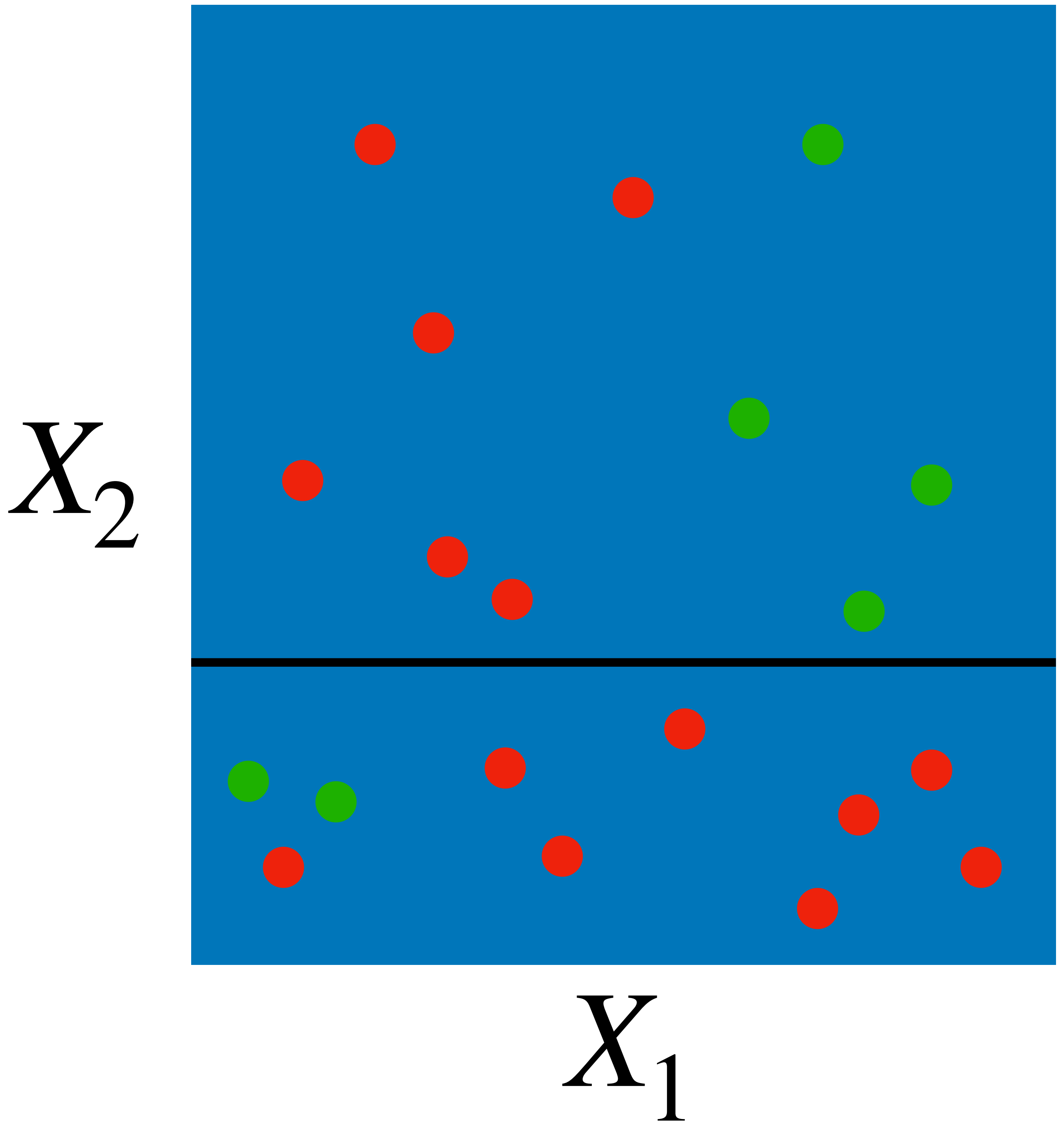
We use a very similar greedy recursive splitting algorithm.

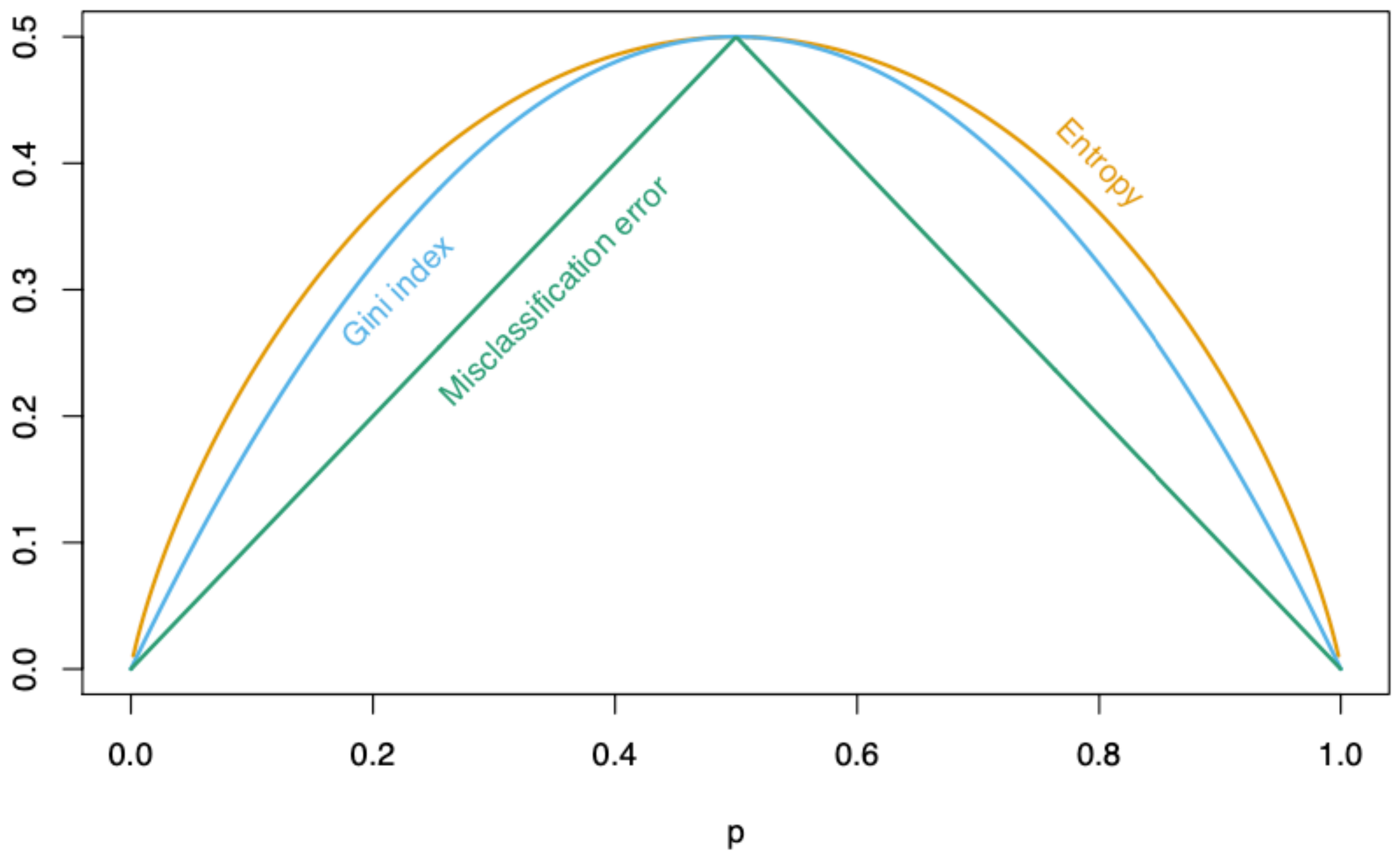
Let $\widehat{p}_m$ be proportion of class 1 in region m. The misclassification error in that region is $\min(\widehat{p}_m, 1 - \widehat{p}_m)$.

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

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

Find split that minimizes the total impurity of the regions.





Training a classification tree

Finding the rectangles $\widehat{R}_m$

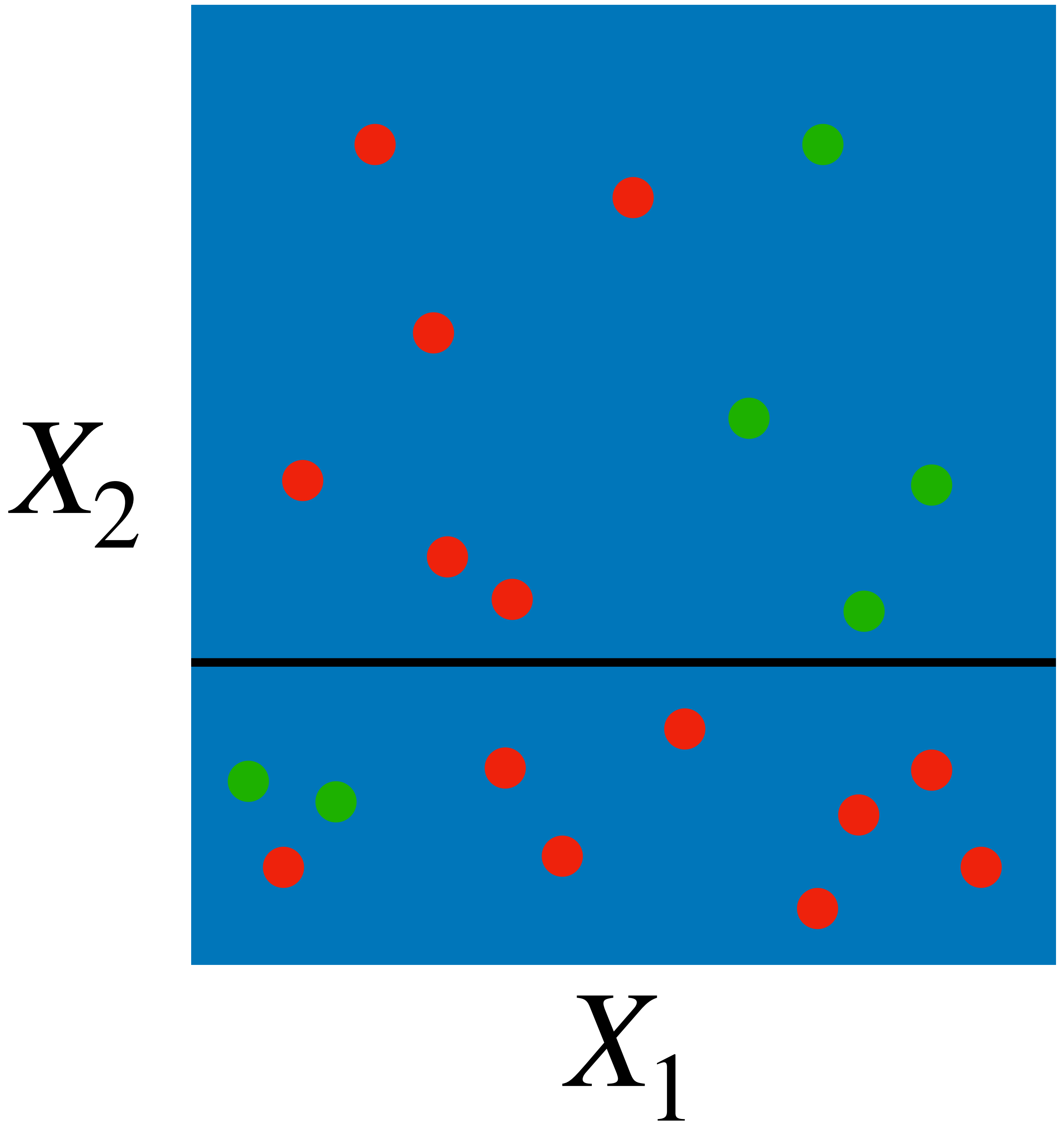
We use a very similar greedy recursive splitting algorithm.

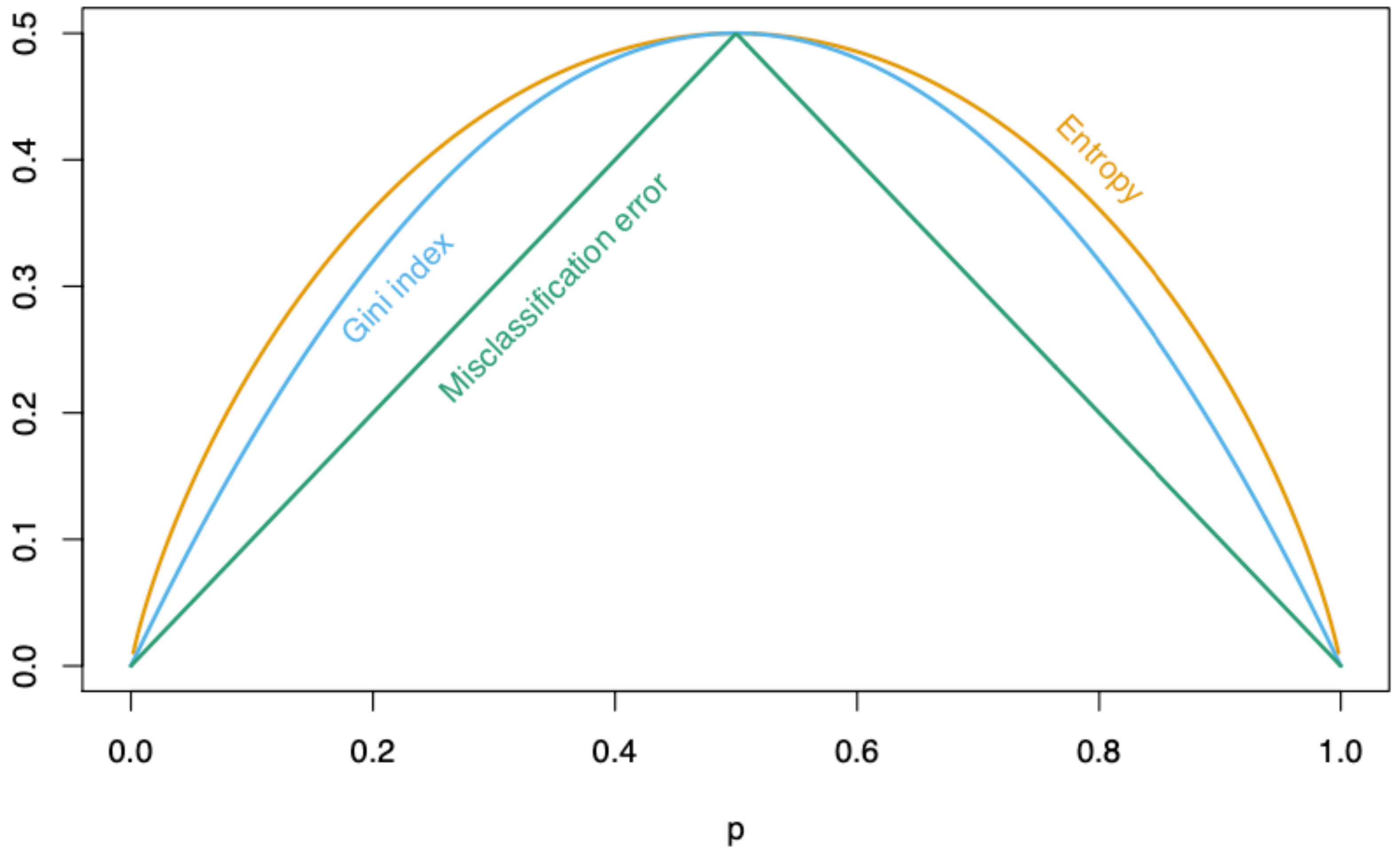
Let $\hat{p}_m$ be proportion of class 1 in region m. The misclassification error in that region is $\min(\hat{p}_m, 1 - \hat{p}_m)$.

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

Finding the rectangles $\widehat{R}_m$

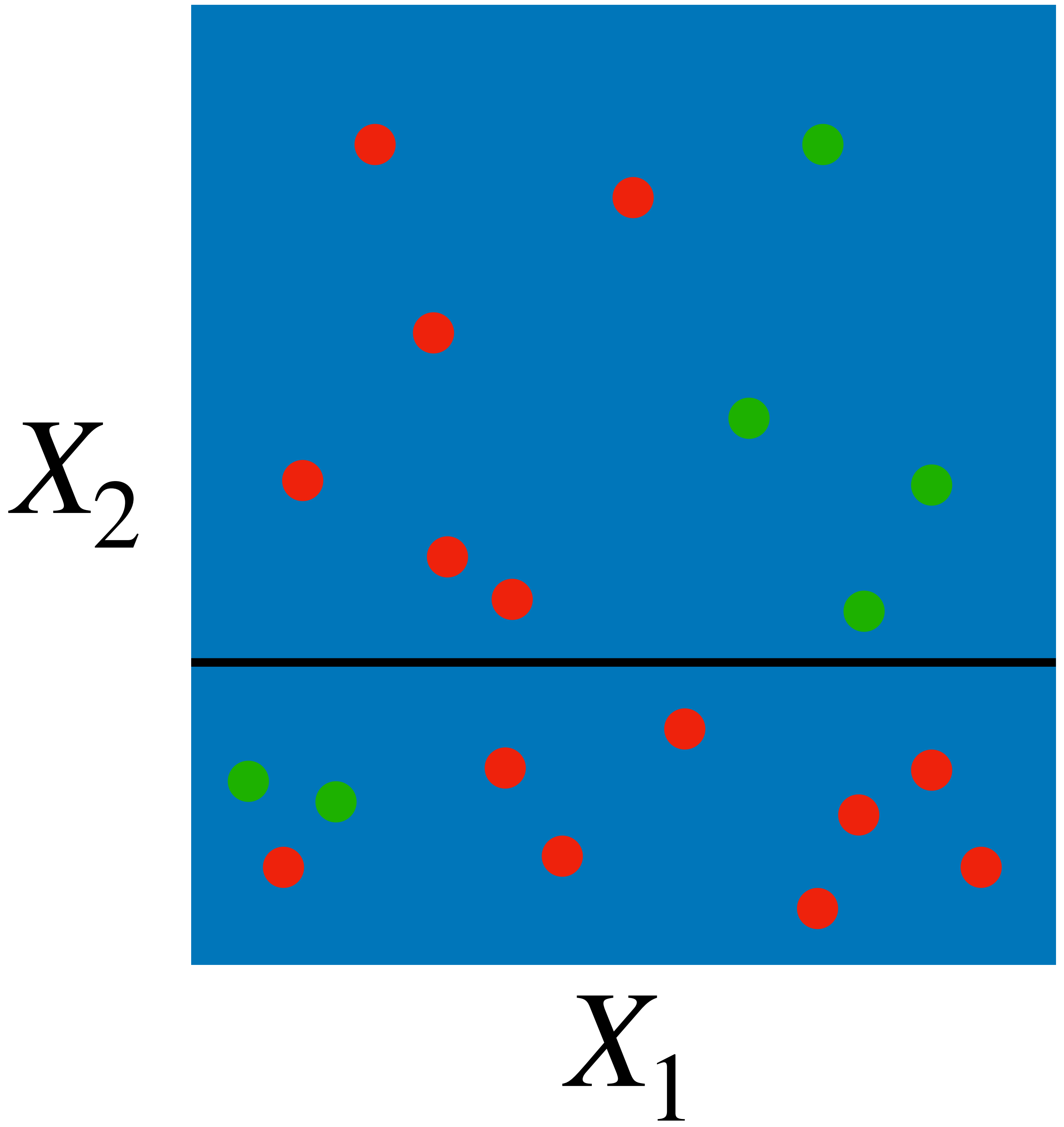
We use a very similar greedy recursive splitting algorithm.

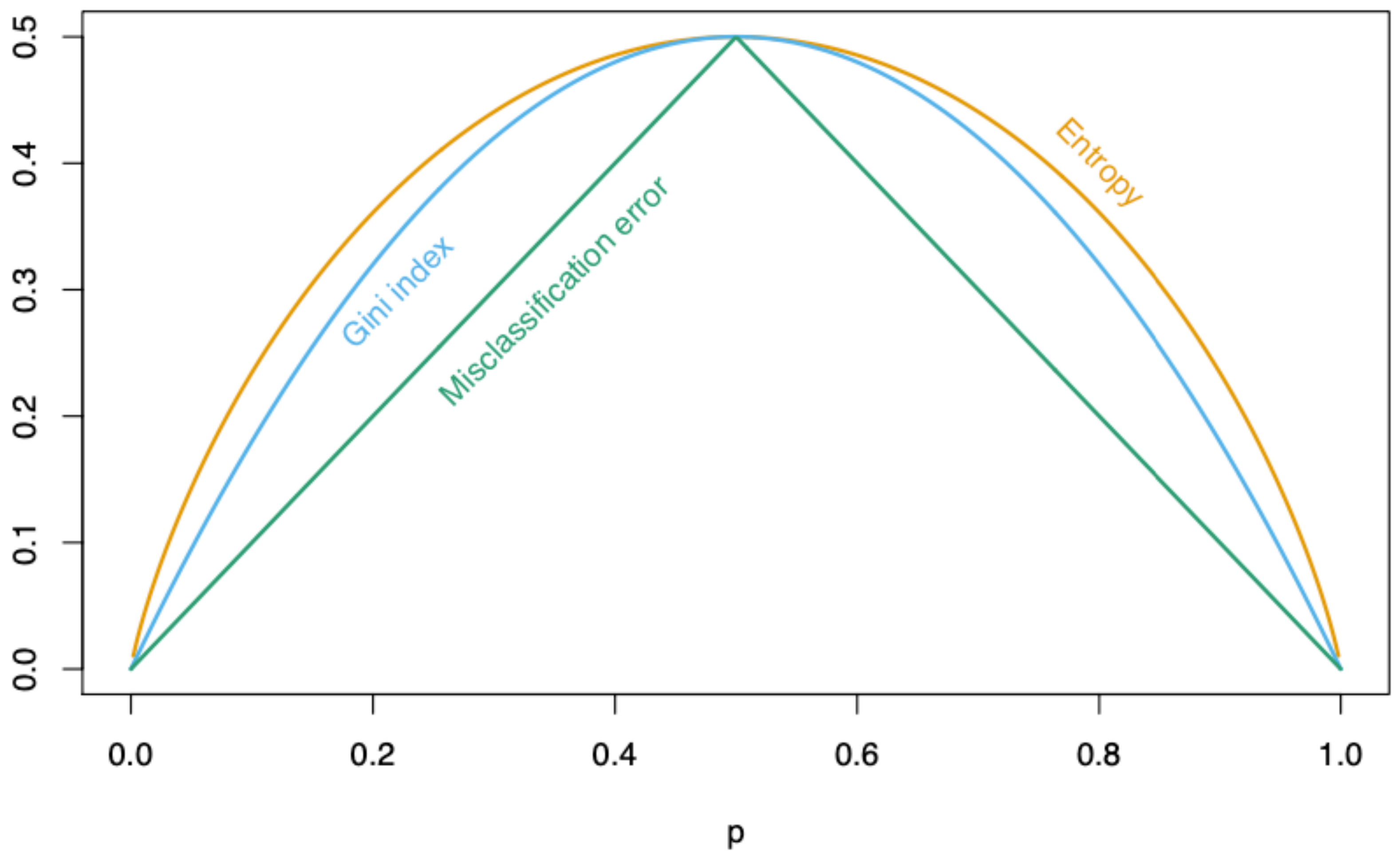
Let $\widehat{p}_m$ be proportion of class 1 in region m. The misclassification error in that region is $\min(\widehat{p}_m, 1 - \widehat{p}_m)$.

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

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

Find split that minimizes the total impurity of the regions.





Training a classification tree

Finding the rectangles $\widehat{R}_m$

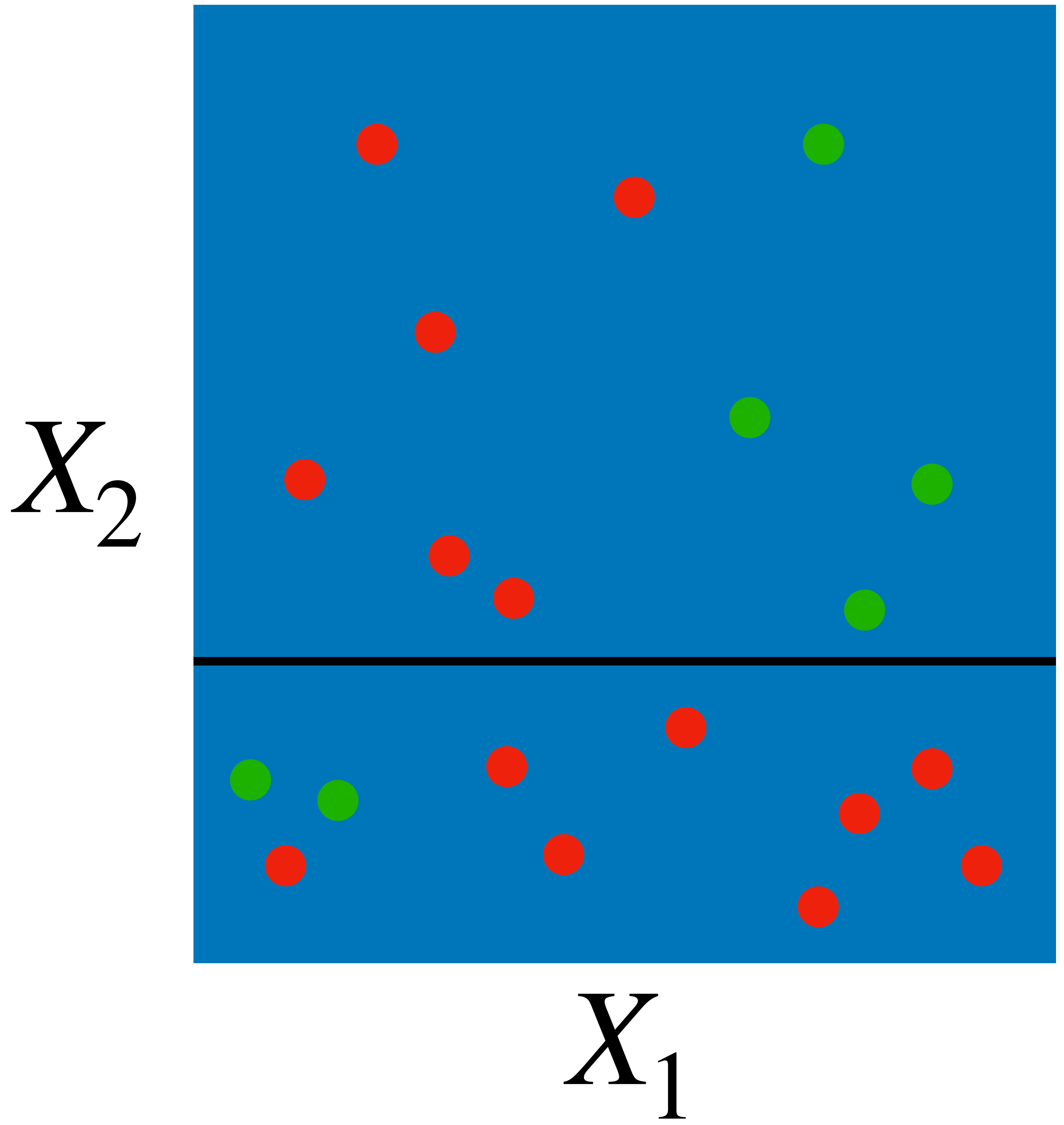
We use a very similar greedy recursive splitting algorithm.

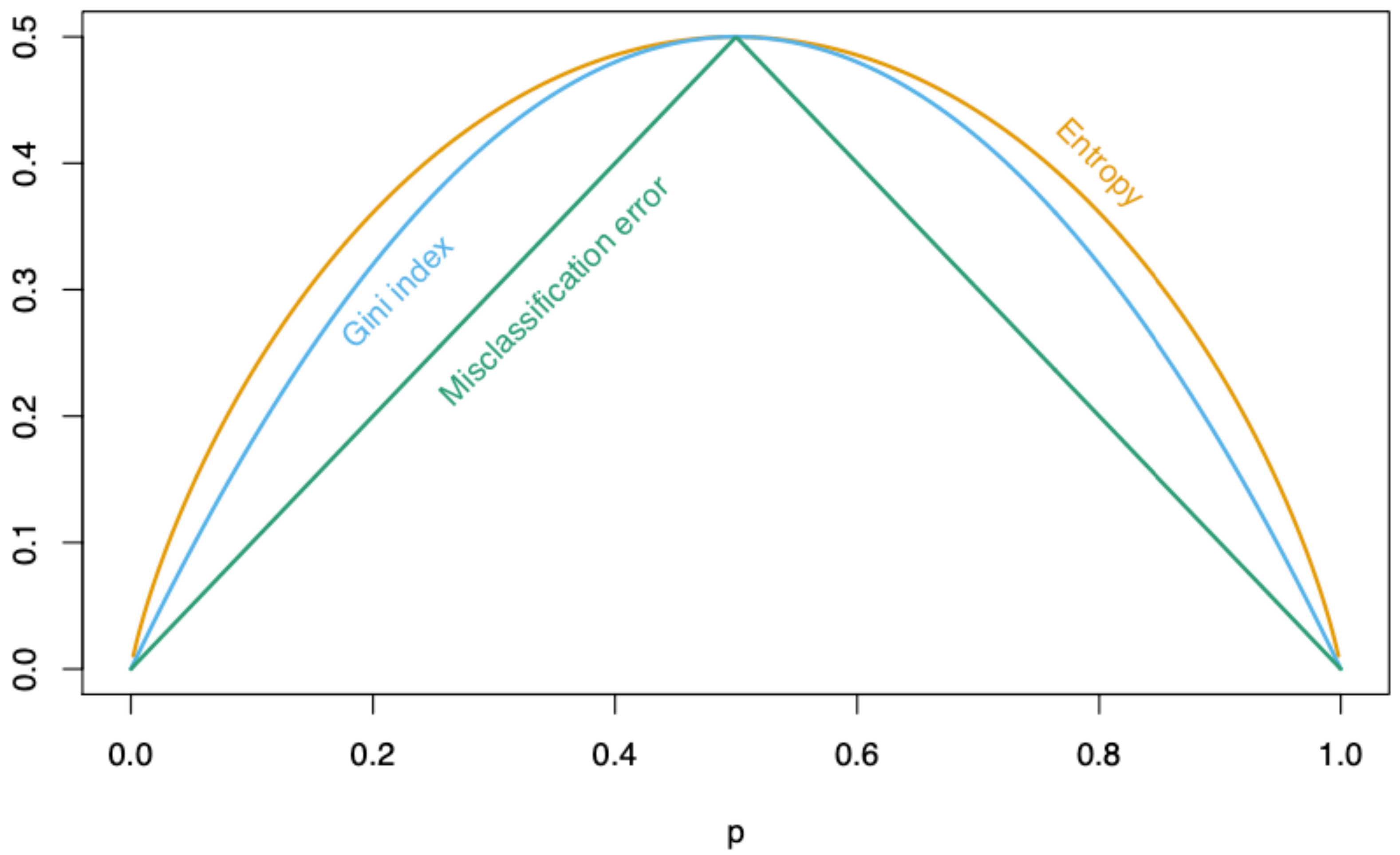
Let $\hat{p}_m$ be proportion of class 1 in region m. The misclassification error in that region is $\min(\hat{p}_m, 1 - \hat{p}_m)$.

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

Finding the rectangles $\widehat{R}_m$

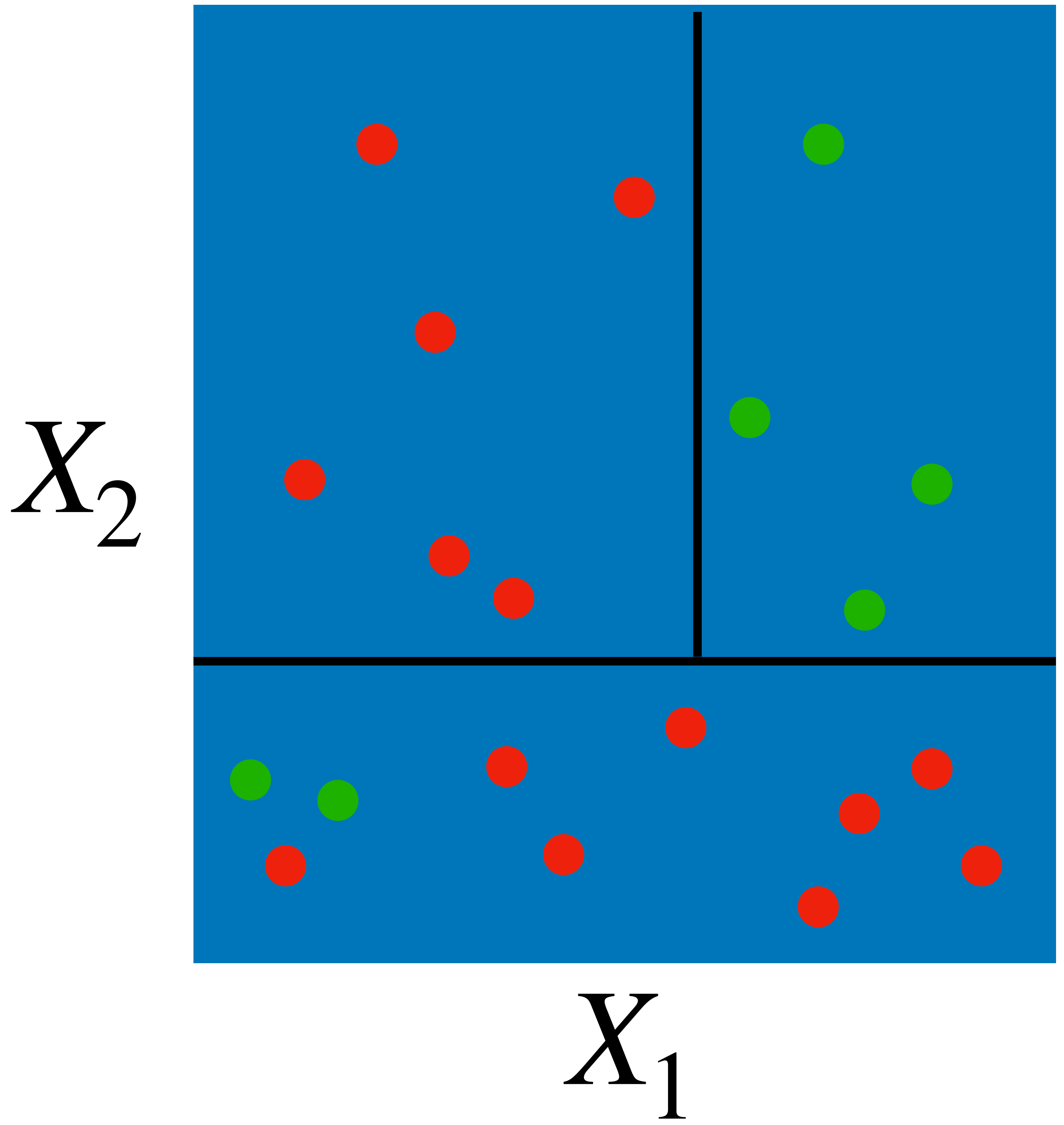
We use a very similar greedy recursive splitting algorithm.

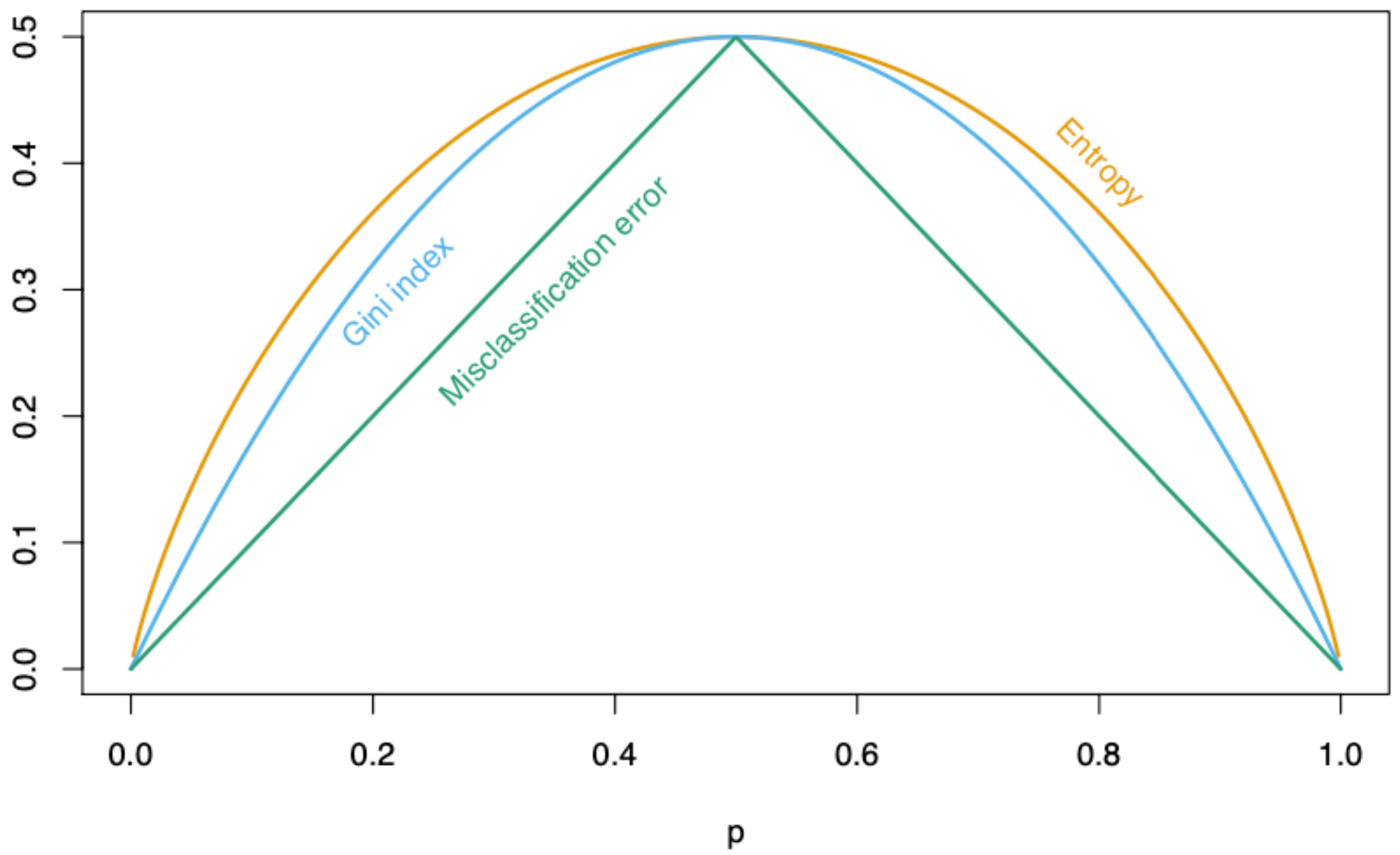
Let $\widehat{p}_m$ be proportion of class 1 in region m. The misclassification error in that region is $\min(\widehat{p}_m, 1 - \widehat{p}_m)$.

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

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

Find split that minimizes the total impurity of the regions.





Training a classification tree

Finding the rectangles $\widehat{R}_m$

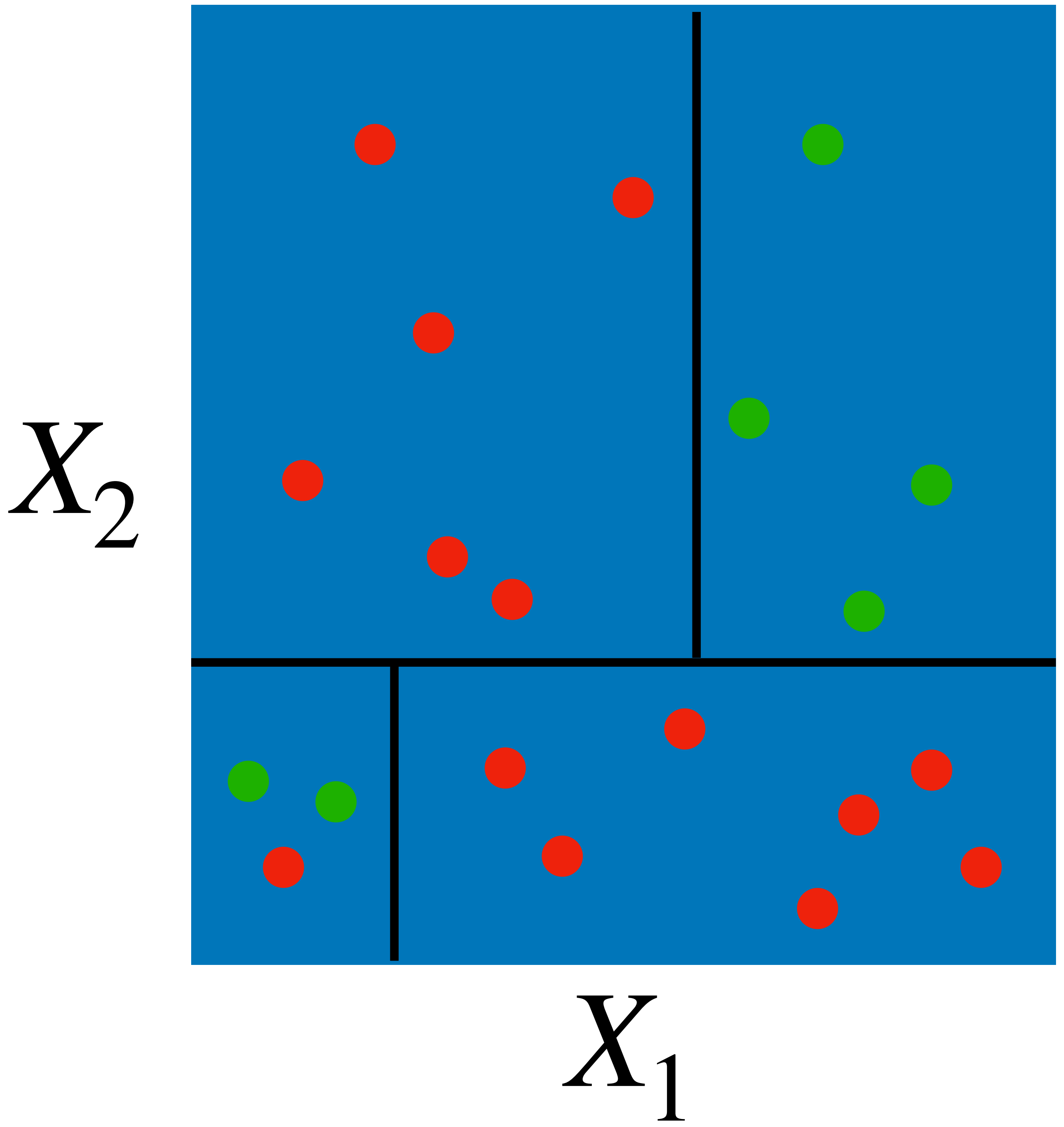
We use a very similar greedy recursive splitting algorithm.

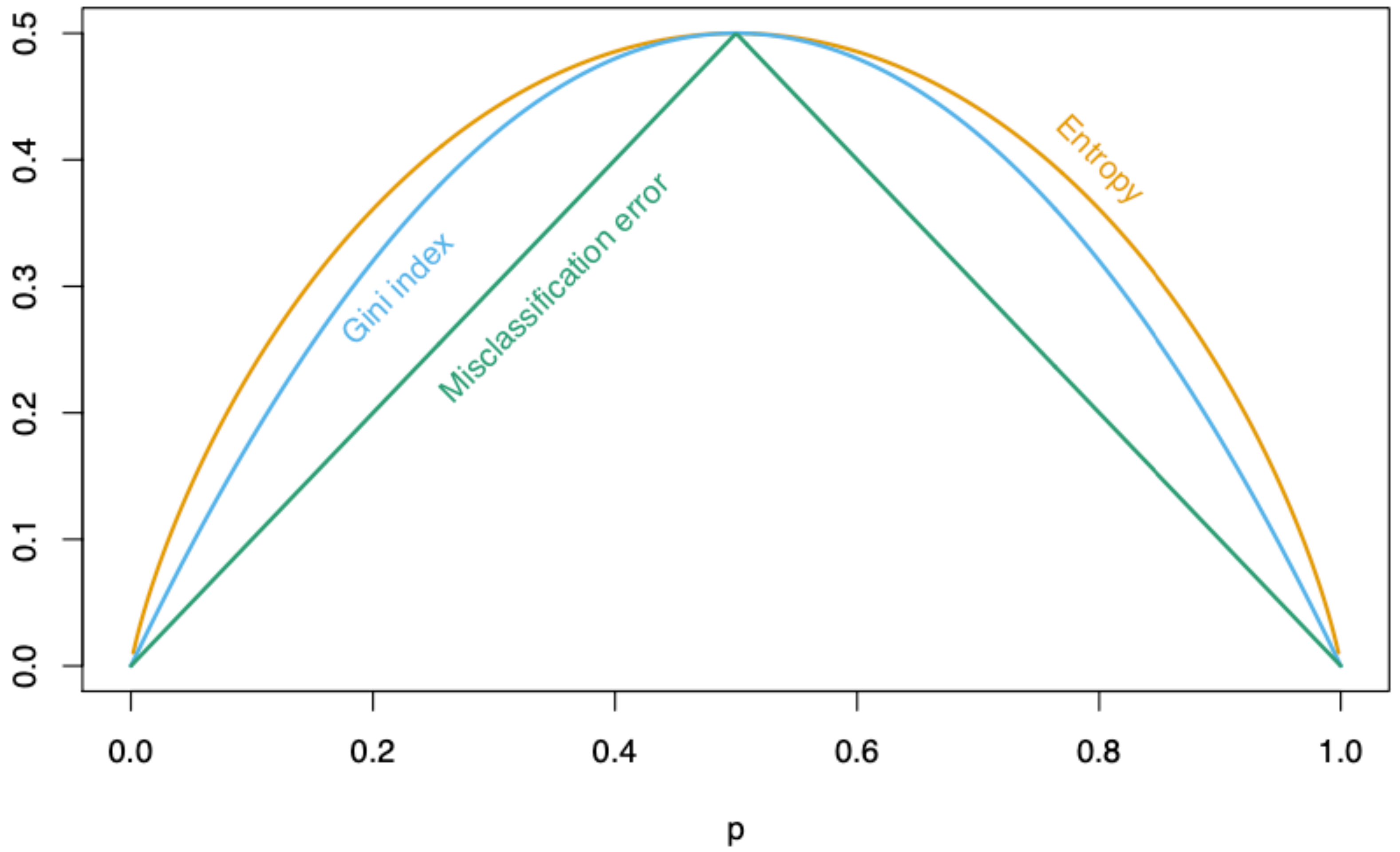
Let $\widehat{p}_m$ be proportion of class 1 in region m. The misclassification error in that region is $\min(\widehat{p}_m, 1 - \widehat{p}_m)$.

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

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

Find split that minimizes the total impurity of the regions.



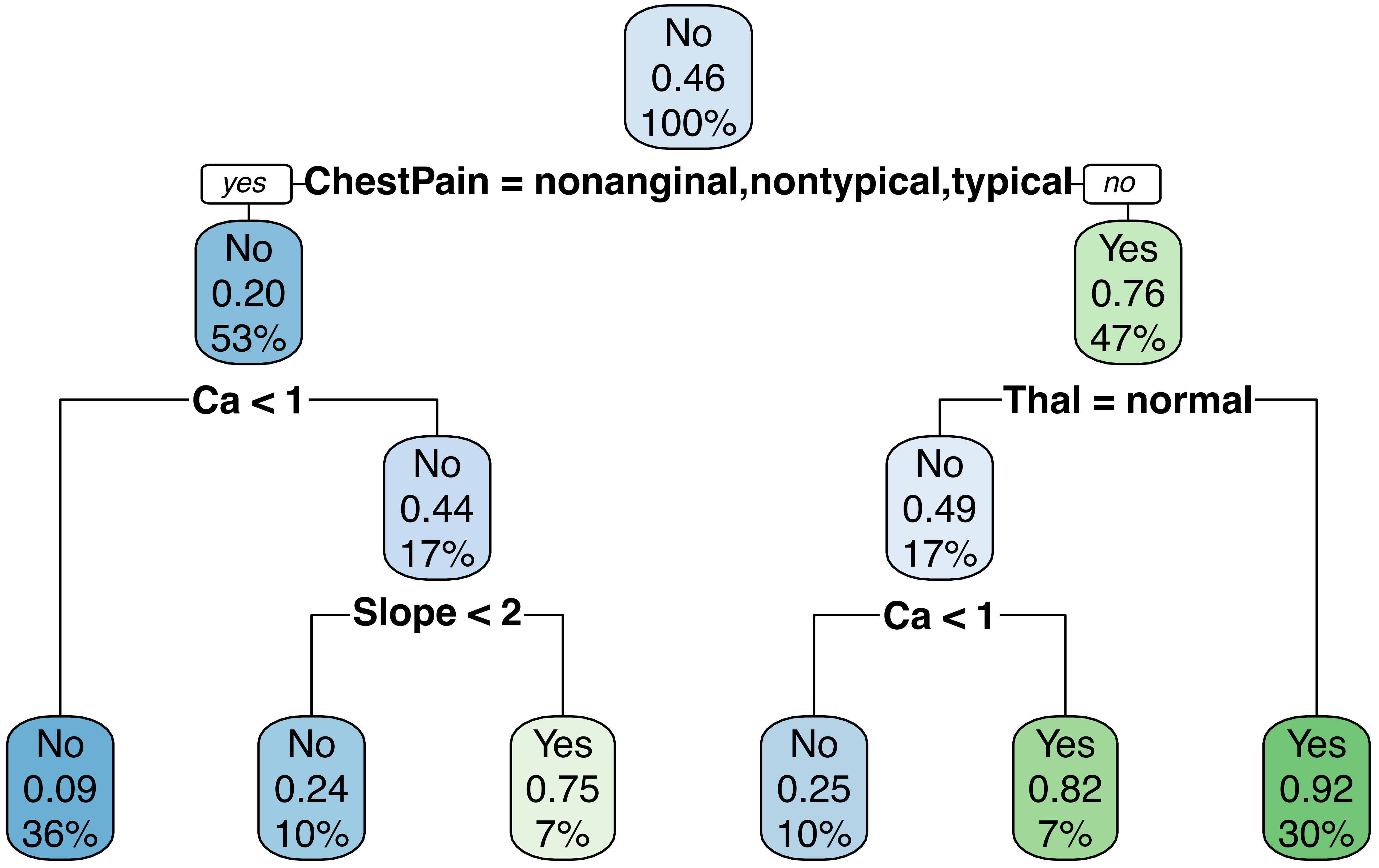


Training a classification tree

Final output

Example: Heart disease data set.

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

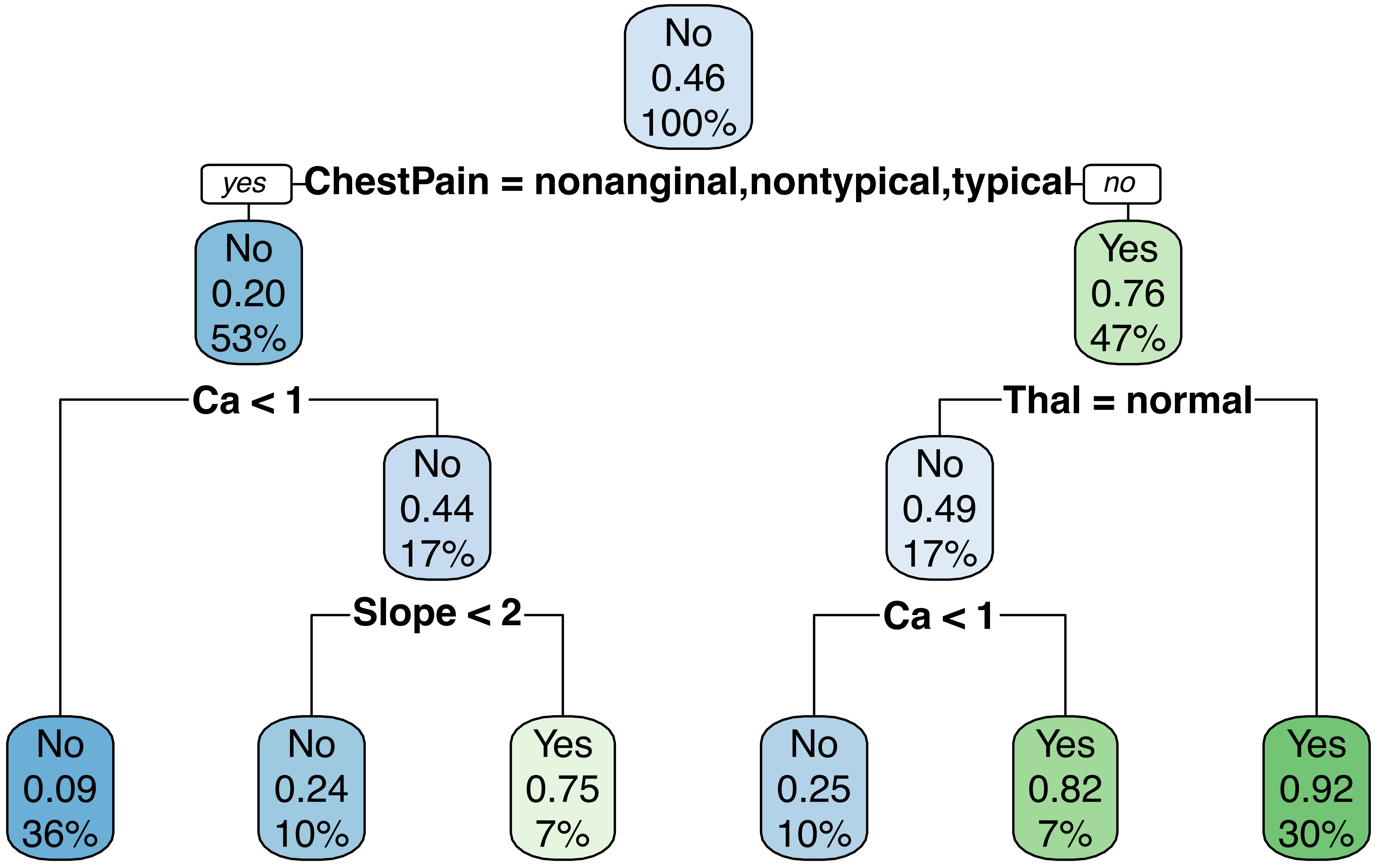


Training a classification tree

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.