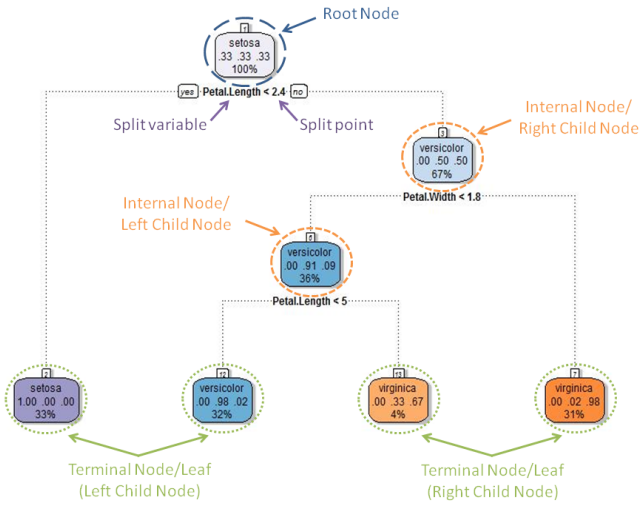
Einführung in das statistische Lernen

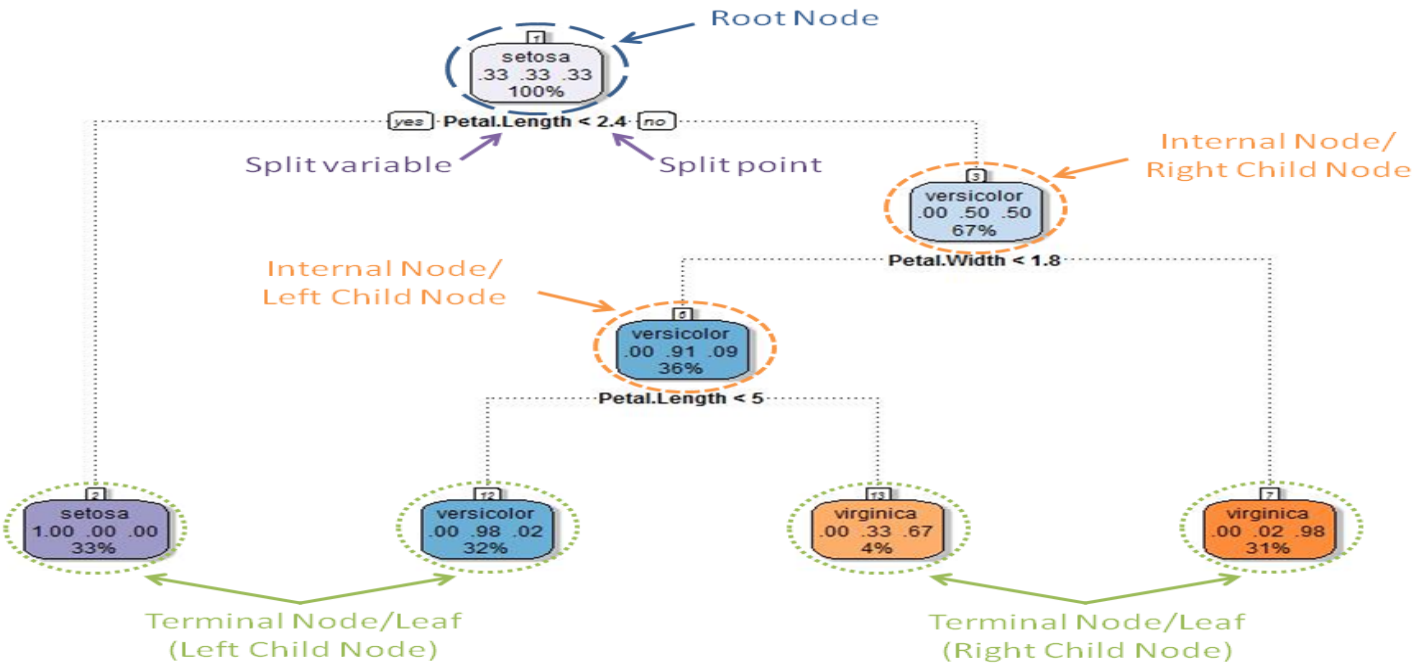
Classification and Regression Trees (CART): Basics



Learning goals

- Understand the basic structure of a tree model
- Understand that the basic idea of a tree model is the same for classification and regression
- Know how the label of a new observation is predicted via CART
- Know hypothesis space of CART

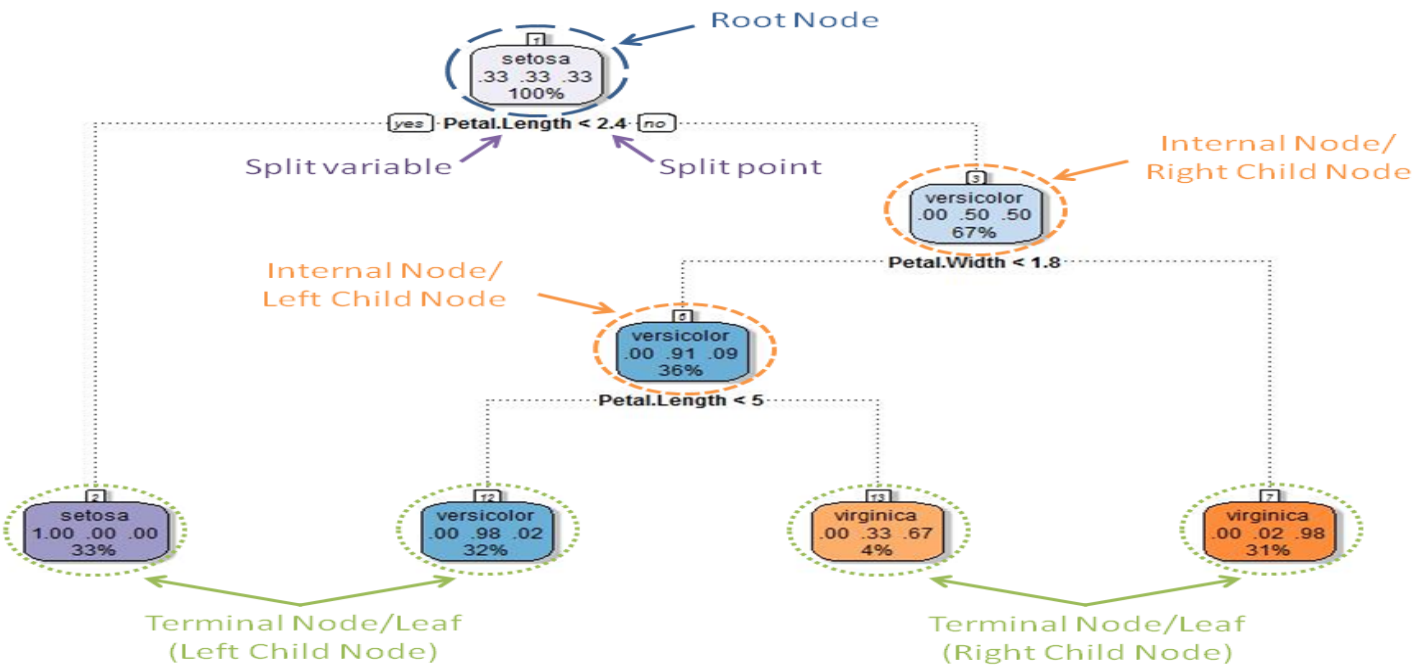
TREE MODEL AND PREDICTION



- Classification and Regression Trees, introduced by Breiman
- Binary splits are constructed top-down
- Constant prediction in each terminal node (leaf): either a numerical value, a class label, or a probability vector over class labels.

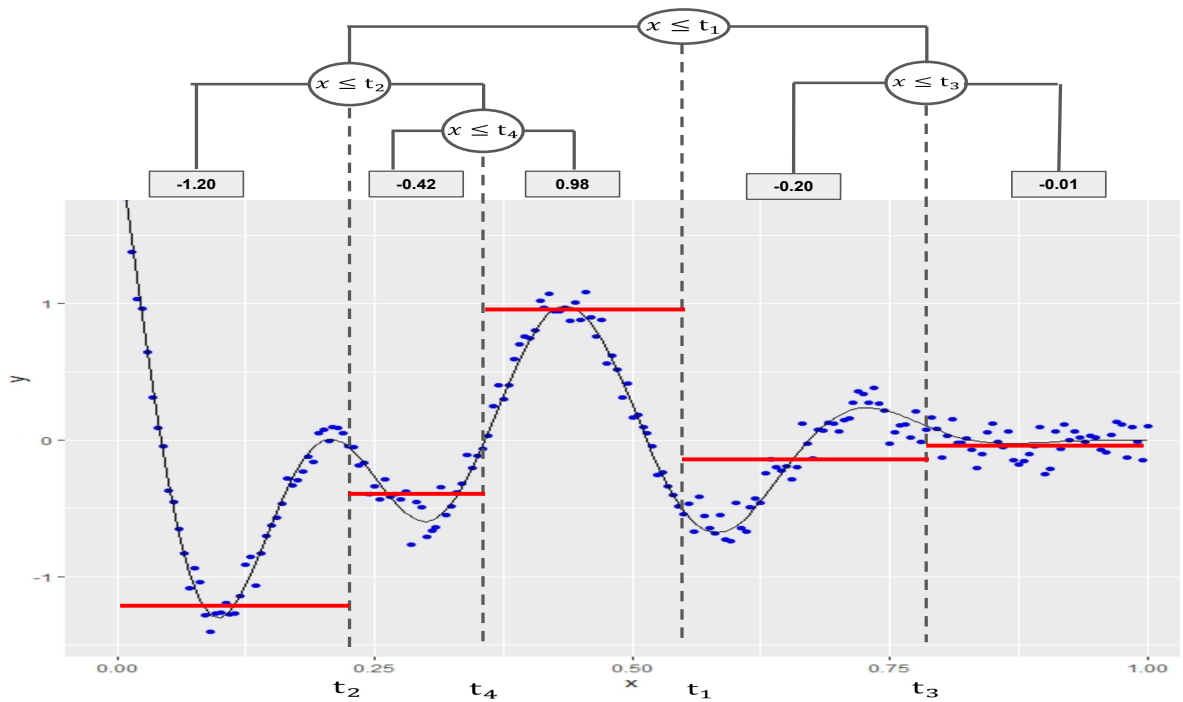
TREE MODEL AND PREDICTION

- For predictions, observations are passed down the tree, according to the splitting rules in each node
- An observation will end up in exactly one leaf node
- All observations in a leaf node are assigned the same prediction for the target



TREE MODEL AND PREDICTION

- For predictions, observations are passed down the tree, according to the splitting rules in each node
- An observation will end up in exactly one leaf node
- All observations in a leaf node are assigned the same prediction for the target

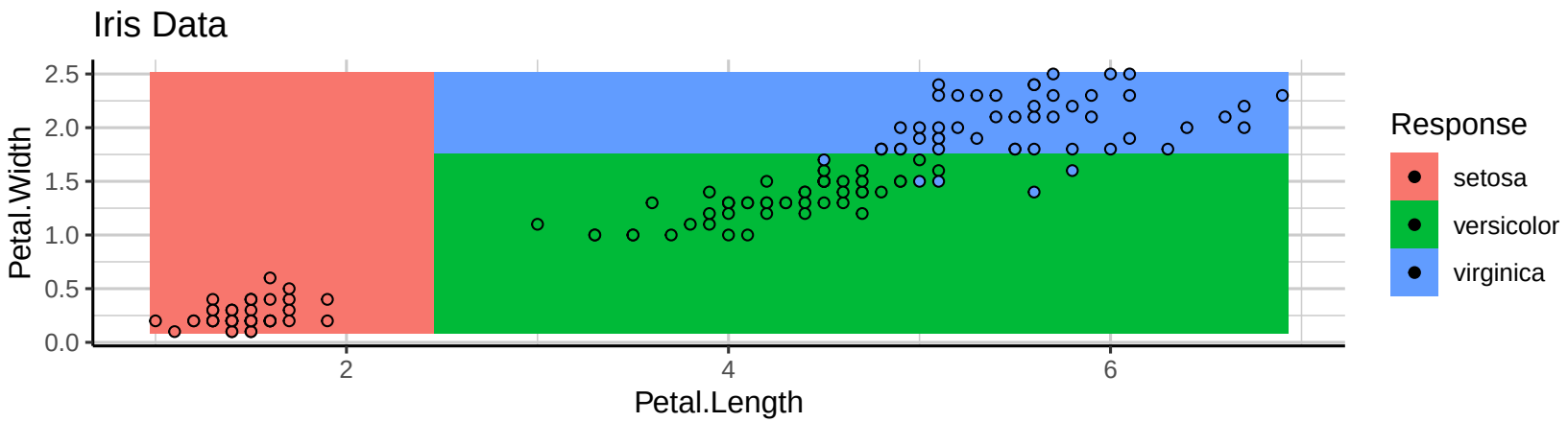


TREE AS AN ADDITIVE MODEL

Each point in $\mathcal{X}$ is assigned to exactly one leaf, and each leaf has a set of input points leading to it through axis-parallel splits.
Hence, trees divide the feature space $\mathcal{X}$ into **rectangular regions**:

$$f(\mathbf{x}) = \sum_{m=1}^{M} c_m \mathbb{I}(\mathbf{x} \in Q_m),$$

where a tree with M leaf nodes defines M "rectangles" Q_m.
c_m is the predicted numerical response, class label or class distribution in the respective leaf node.

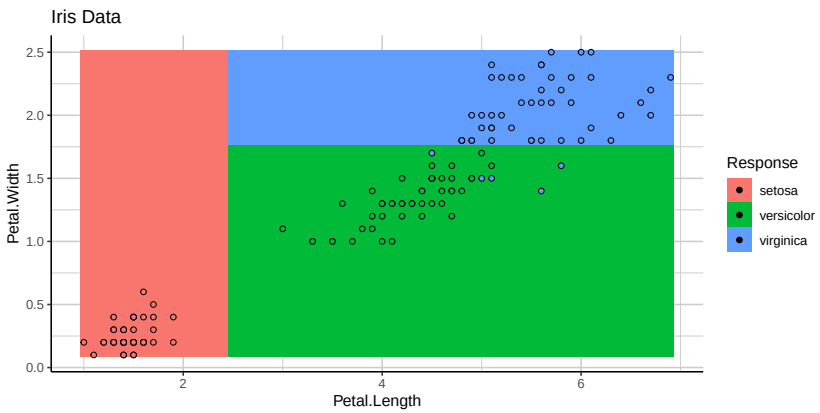


TREES

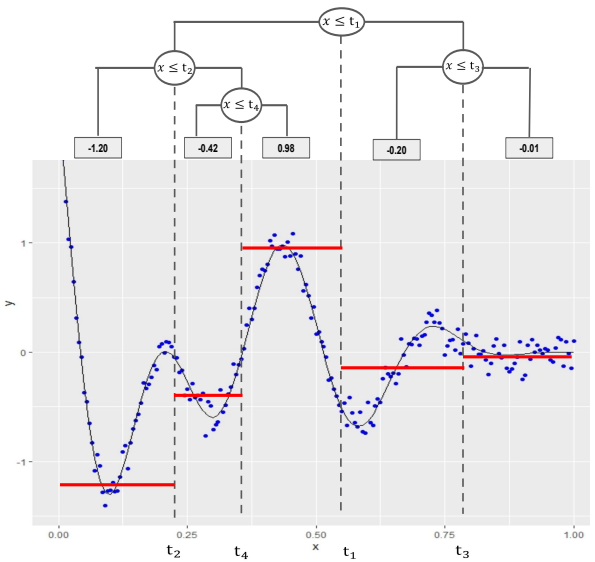
The hypothesis space of a CART is the set of all step functions over rectangular partitions of $\mathcal{X}$:

$$f(\mathbf{x}) = \sum_{m=1}^{M} c_m \mathbb{I}(\mathbf{x} \in Q_m)$$

Classification:

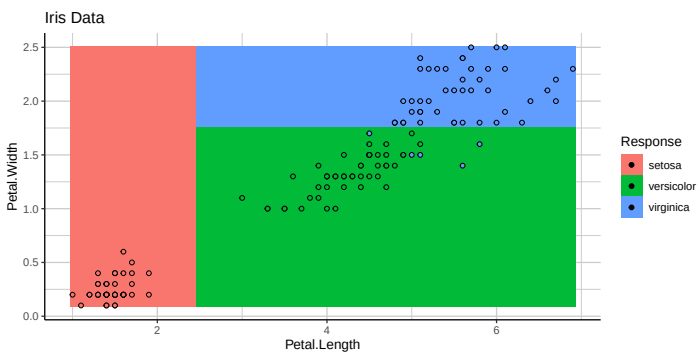


Regression:



Einführung in das statistische Lernen

CART: Splitting Criteria

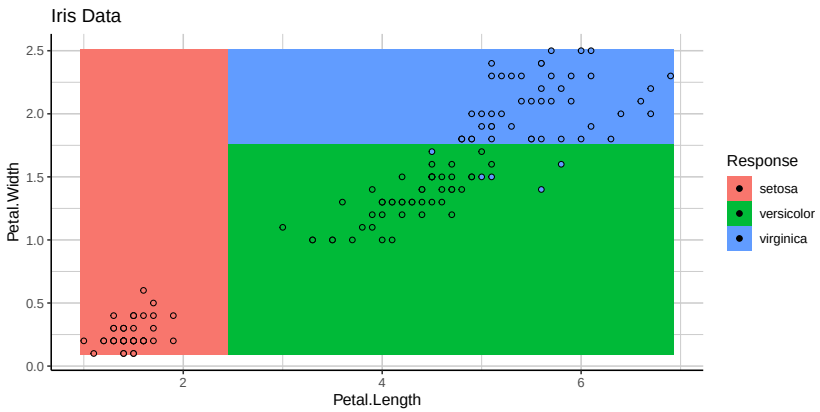


Learning goals

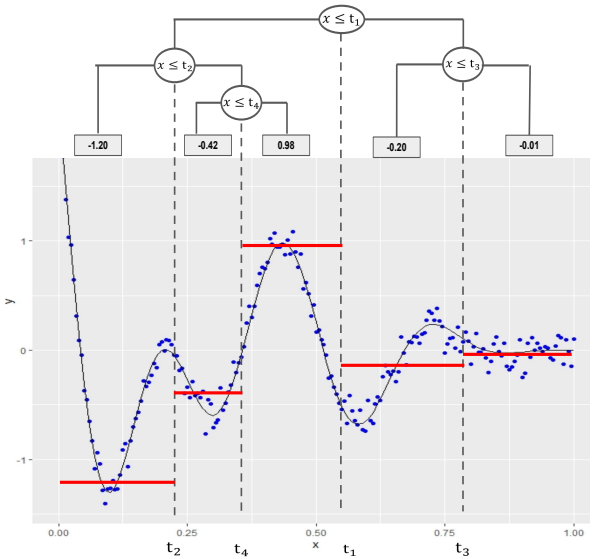
- Know definition of a tree's node
- Understand how split points are derived via empirical risk minimization
- Know which losses can be applied for regression and classification
- Know the connections between empirical risk minimization and impurity minimization

TREES

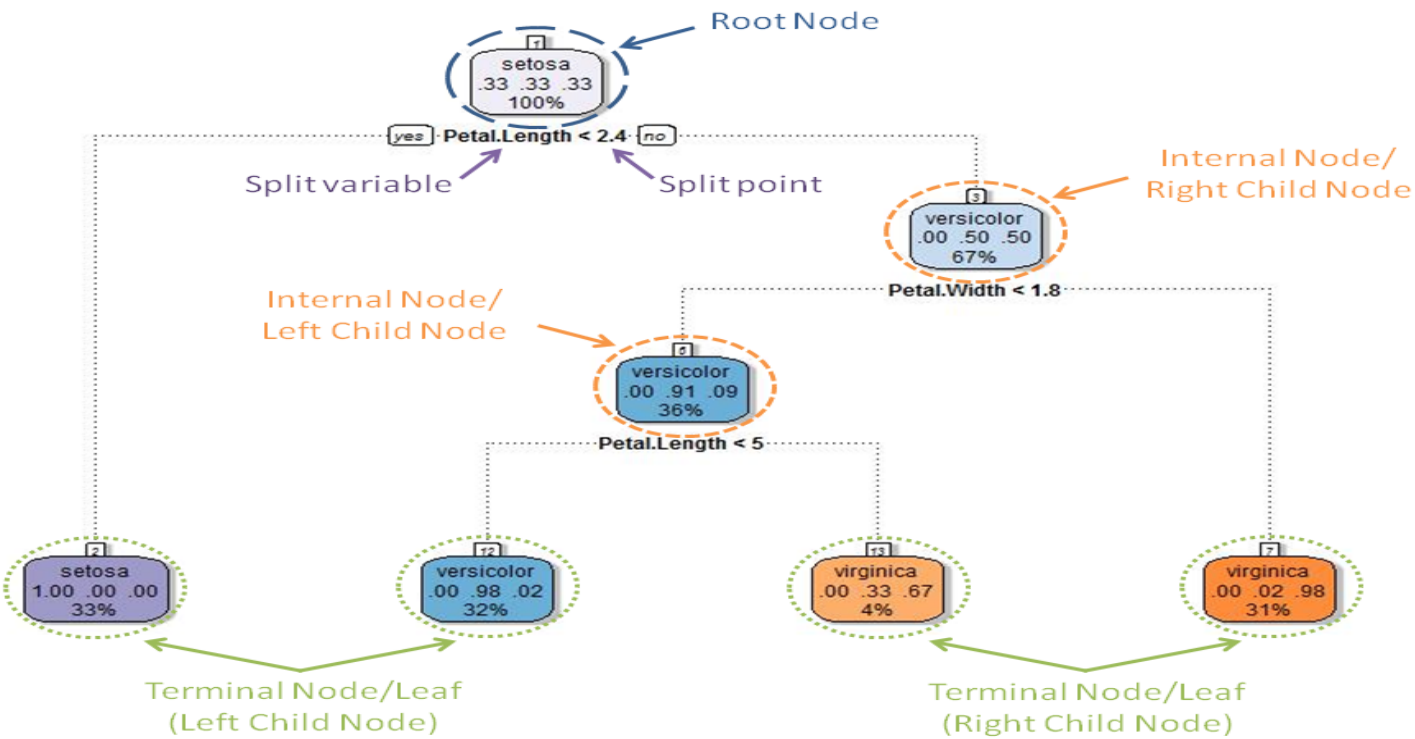
Classification Tree:



Regression Tree:



SPLITTING CRITERIA



How to find good splitting rules to define the tree?

$\Longrightarrow$ **empirical risk minimization**

SPLITTING CRITERIA: FORMALIZATION

- Let $\mathcal{N} \subseteq \mathcal{D}$ be the data that is assigned to a terminal node $\mathcal{N}$ of a tree.
- Let c be the predicted constant value for the data assigned to $\mathcal{N}$: $\hat{y} \equiv c$ for all $(\mathbf{x}, y) \in \mathcal{N}$.
- Then the risk $\mathcal{R}(\mathcal{N})$ for a leaf is simply the average loss for the data assigned to that leaf under a given loss function L:

$$\mathcal{R}(\mathcal{N}) = \frac{1}{|\mathcal{N}|} \sum_{(\mathbf{x},y) \in \mathcal{N}} L(y, c)$$

- The prediction is given by the optimal constant $c = \arg\min_c \mathcal{R}(\mathcal{N})$

SPLITTING CRITERIA: FORMALIZATION

- A split w.r.t. **feature** x_j **at split point** t divides a parent node $\mathcal{N}$ into

$$\mathcal{N}_1 = \{(\mathbf{x}, y) \in \mathcal{N} : x_j \leq t\} \text{ and } \mathcal{N}_2 = \{(\mathbf{x}, y) \in \mathcal{N} : x_j > t\}.$$

- In order to evaluate how good a split is, we compute the empirical risks in both child nodes and sum them up:

$$\begin{aligned} \mathcal{R}(\mathcal{N}, j, t) &= \frac{|\mathcal{N}_1|}{|\mathcal{N}|} \mathcal{R}(\mathcal{N}_1) + \frac{|\mathcal{N}_2|}{|\mathcal{N}|} \mathcal{R}(\mathcal{N}_2) \\ &= \frac{1}{|\mathcal{N}|} \left(\sum_{(\mathbf{x},y) \in \mathcal{N}_1} L(y, c_1) + \sum_{(\mathbf{x},y) \in \mathcal{N}_2} L(y, c_2) \right) \end{aligned}$$

- Finding the best way to split $\mathcal{N}$ into $\mathcal{N}_1, \mathcal{N}_2$ means solving

$$\arg\min_{j,t} \mathcal{R}(\mathcal{N}, j, t)$$

SPLITTING CRITERIA: REGRESSION

- For regression trees, we usually use L_2 loss:

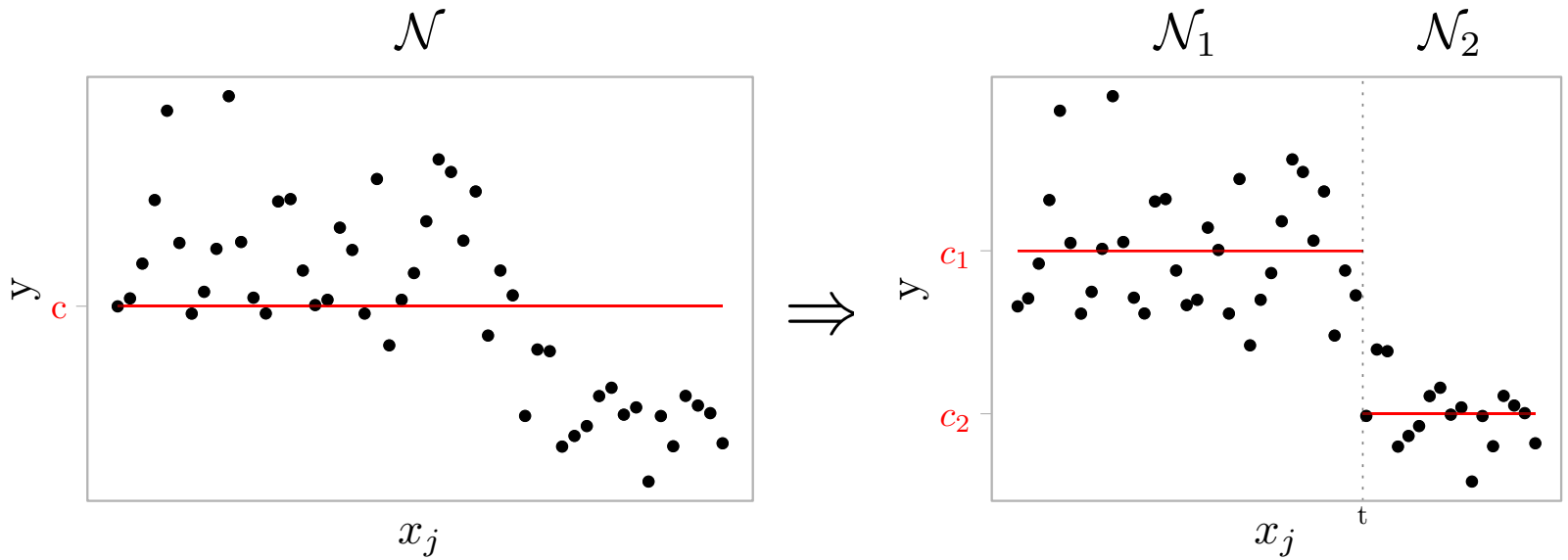
$$\mathcal{R}(\mathcal{N}) = \frac{1}{|\mathcal{N}|} \sum_{(\mathbf{x},y)\in\mathcal{N}} (y - c)^2$$

- The best constant prediction under L_2 is the mean:

$$c = \bar{y}_{\mathcal{N}} = \frac{1}{|\mathcal{N}|} \sum_{(\mathbf{x},y)\in\mathcal{N}} y$$

SPLITTING CRITERIA: REGRESSION

- This means the best split is the one that minimizes the (pooled) variance of the target distribution in the child nodes $\mathcal{N}_1$ and $\mathcal{N}_2$:



We can also interpret this as a way of measuring the impurity of the target distribution, i.e., how much it diverges from a constant in each of the child nodes.

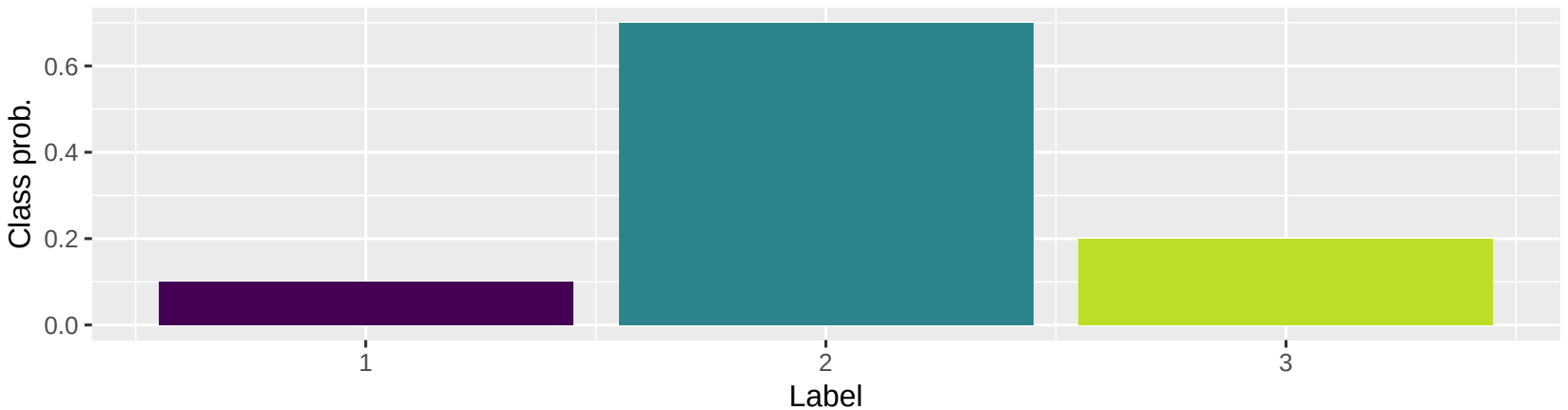
- For L_1 loss, c is the median of $y \in \mathcal{N}$.

SPLITTING CRITERIA: CLASSIFICATION

- Typically uses either Brier score (so: L_2 loss on probabilities) or Bernoulli loss (as in logistic regression) as loss functions
- Predicted probabilities in node $\mathcal{N}$ are simply the class proportions in the node:

$$\hat{\pi}_k^{(\mathcal{N})} = \frac{1}{|\mathcal{N}|} \sum_{(\mathbf{x},y) \in \mathcal{N}} \mathbb{I}(y = k)$$

This is the optimal prediction under both the logistic / Bernoulli loss and the Brier loss.



SPLITTING CRITERIA: COMMENTS

- Splitting criteria for trees are usually defined in terms of "impurity reduction". Instead of minimizing empirical risk in the child nodes over all possible splits, a measure of “impurity” of the distribution of the target y in the child nodes is minimized.
- For regression trees, the “impurity” of a node is usually defined as the variance of the $y^{(i)}$ in the node. Minimizing this “variance impurity” is equivalent to minimizing the squared error loss for a predicted constant in the nodes.

SPLITTING CRITERIA: COMMENTS

- Minimizing the Brier score is equivalent to minimizing the Gini impurity

$$I(\mathcal{N}) = \sum_{k=1}^{g} \hat{\pi}_k^{(\mathcal{N})} \left(1 - \hat{\pi}_k^{(\mathcal{N})}\right)$$

- Minimizing the Bernoulli loss is equivalent to minimizing entropy impurity

$$I(\mathcal{N}) = -\sum_{k=1}^{g} \hat{\pi}_k^{(\mathcal{N})} \log \hat{\pi}_k^{(\mathcal{N})}$$

- The approach based on loss functions instead of impurity measures is simpler and more straightforward, mathematically equivalent and shows that growing a tree can be understood in terms of empirical risk minimization.

SPLITTING WITH MISCLASSIFICATION LOSS

- Why don't we use the misclassification loss for classification trees? I.e., always predict the majority class in each child node and count how many errors we make.
- In many other cases, we are interested in minimizing this kind of error but have to approximate it by some other criterion instead, since the misclassification loss does not have derivatives that we can use for optimization.
  We don't need derivatives when we optimize the tree, so we could go for it!
- This is possible, but Brier score and Bernoulli loss are more sensitive to changes in the node probabilities, and therefore often preferred

SPLITTING WITH MISCLASSIFICATION LOSS

Example: two-class problem with 400 obs in each class and two possible splits:

Split 1:

	class 0	class 1
$\mathcal{N}_1$	300	100
$\mathcal{N}_2$	100	300

Split 2:

	class 0	class 1
$\mathcal{N}_1$	400	200
$\mathcal{N}_2$	0	200

- Both splits are equivalent in terms of misclassification error, they each misclassify 200 observations.
- But: Split 2 produces one pure node and is probably preferable.
- Brier loss (Gini impurity) and Bernoulli loss (entropy impurity) prefer the second split

SPLITTING WITH MISCLASSIFICATION LOSS

- Calculation for Gini:

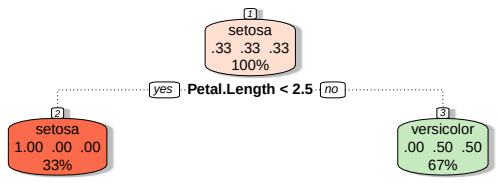
$$\text{Formula}: \frac{|\mathcal{N}_1|}{|\mathcal{N}|} \cdot 2 \cdot \hat{\pi}_0^{(\mathcal{N}_1)} \hat{\pi}_1^{(\mathcal{N}_1)} + \frac{|\mathcal{N}_2|}{|\mathcal{N}|} \cdot 2 \cdot \hat{\pi}_0^{(\mathcal{N}_2)} \hat{\pi}_1^{(\mathcal{N}_2)} =$$

$$\text{Split 1}: \quad \frac{1}{2} \cdot 2 \cdot \frac{3}{4} \cdot \frac{1}{4} + \quad \frac{1}{2} \cdot 2 \cdot \frac{1}{4} \cdot \frac{3}{4} = \frac{3}{8}$$

$$\text{Split 2}: \quad \frac{3}{4} \cdot 2 \cdot \frac{2}{3} \cdot \frac{1}{3} + \quad \frac{1}{4} \cdot 2 \cdot 0 \cdot 1 = \frac{1}{3}$$

Einführung in das statistische Lernen

CART: Growing a Tree



Learning goals

- Understand how a tree is grown by an exhaustive search over all possible features and split points
- Know where exactly the split point is set if several yield the same empirical risk

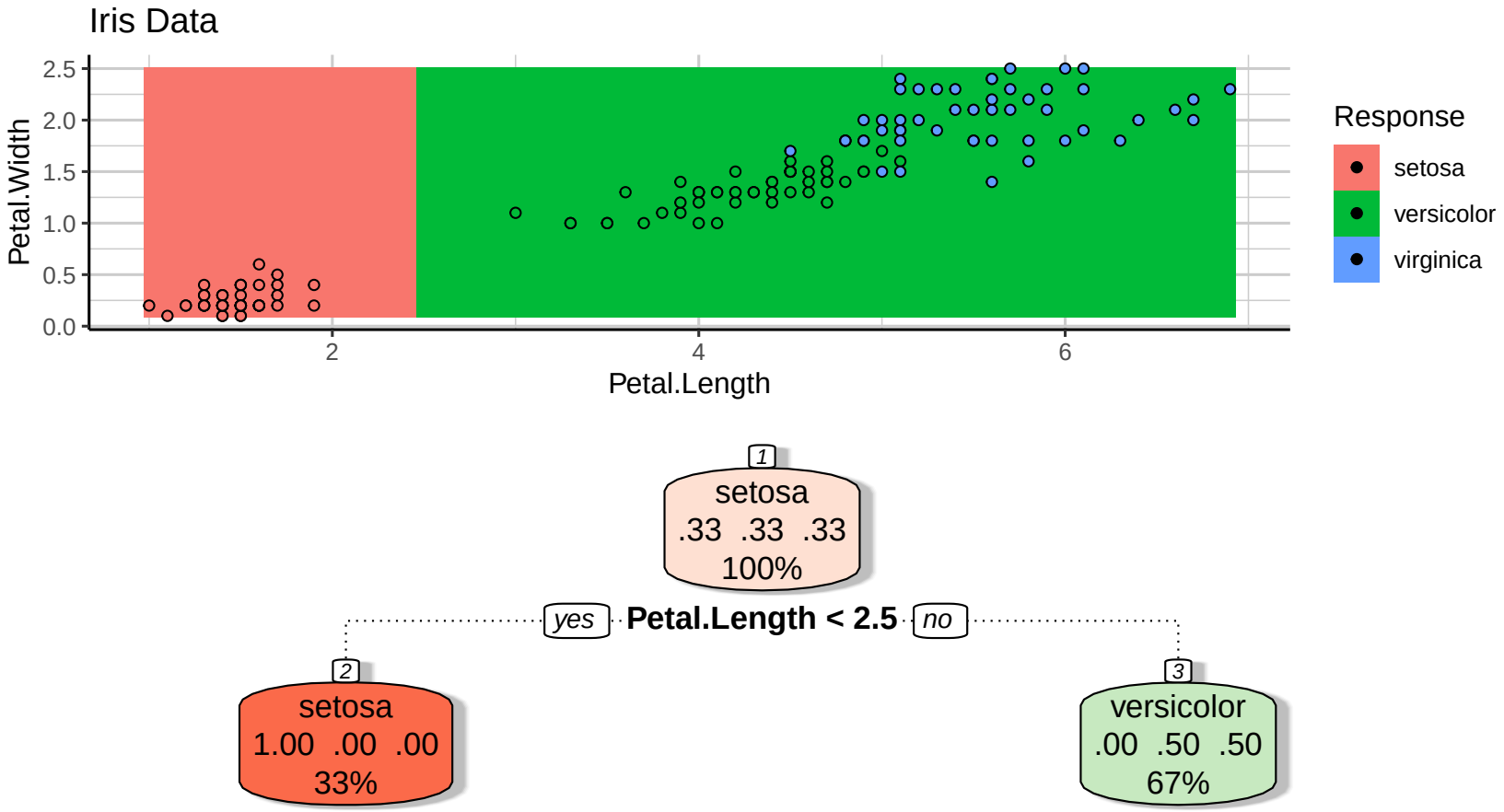
TREE GROWING

We start with an empty tree, a root node that contains all the data. Trees are then grown by recursively applying *greedy* optimization to each node $\mathcal{N}$.

Greedy means we do an **exhaustive search**: All possible splits of $\mathcal{N}$ on all possible points t for all features x_j are compared in terms of their empirical risk $\mathcal{R}(\mathcal{N}, j, t)$.
The training data is then distributed to child nodes according to the optimal split and the procedure is repeated in the child nodes.

TREE GROWING

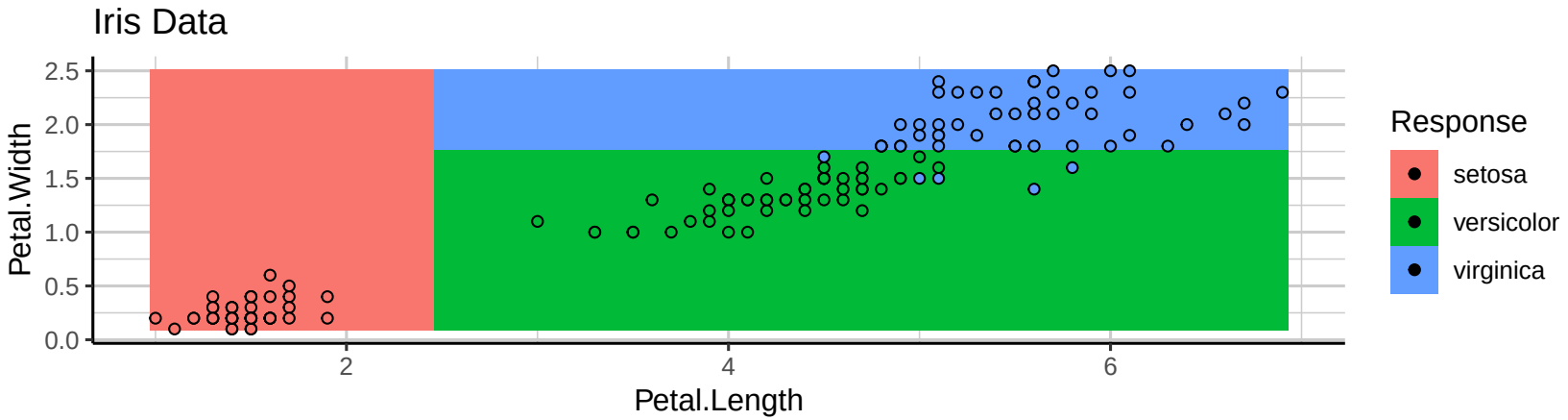
Start with a root node of all data, then search for a feature and split point that minimizes the empirical risk in the child nodes.

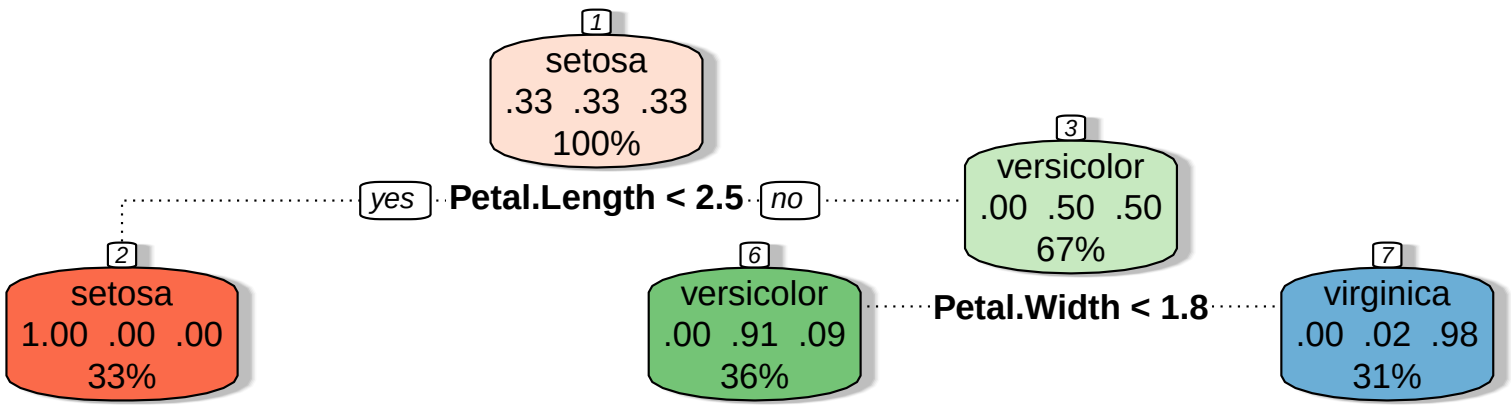


Nodes display their current label distribution here for illustration.

TREE GROWING

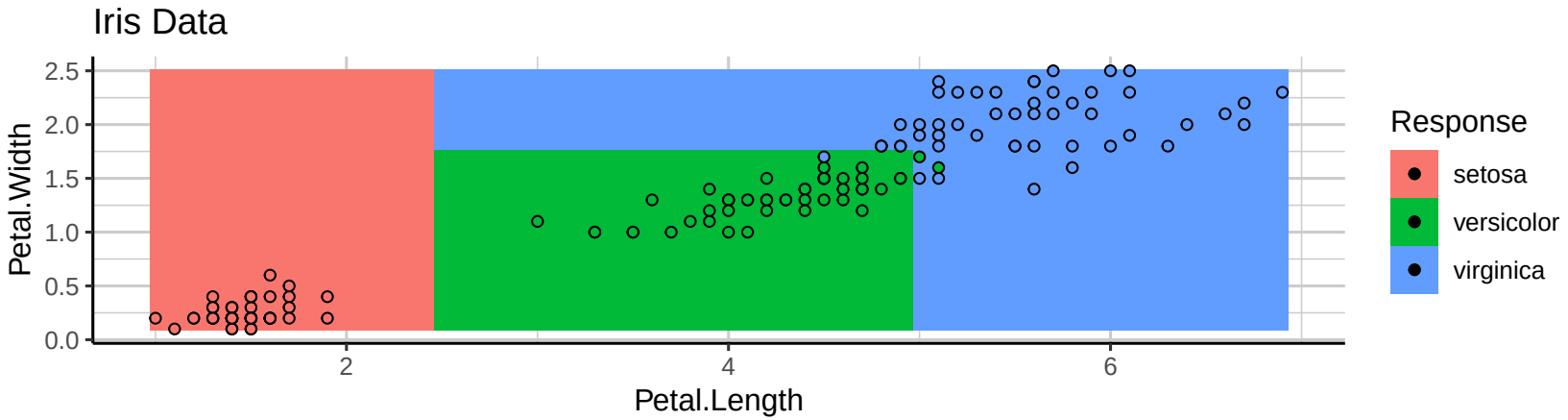
We then proceed recursively for each child node: Iterate over all features, and for each feature over all possible split points. Select the best split and divide data in parent node into left and right child nodes:

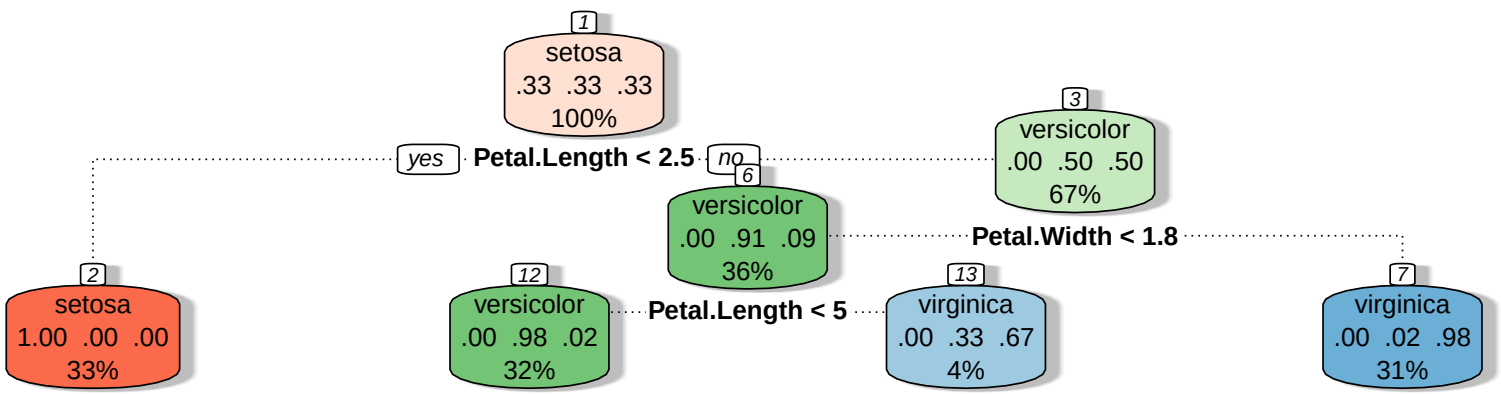




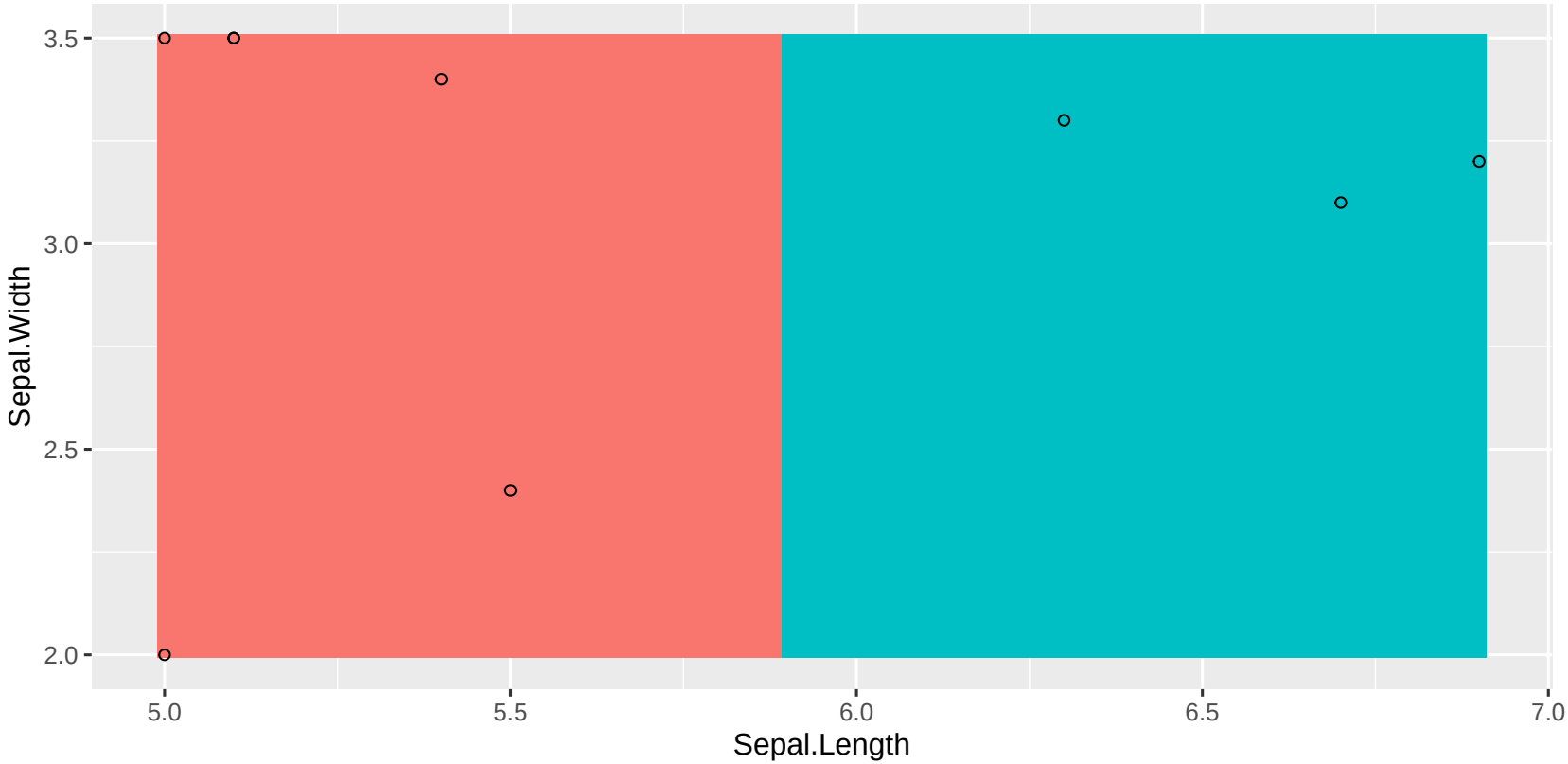
TREE GROWING

We then proceed recursively for each child node: Iterate over all features, and for each feature over all possible split points. Select the best split and divide data in parent node into left and right child nodes:





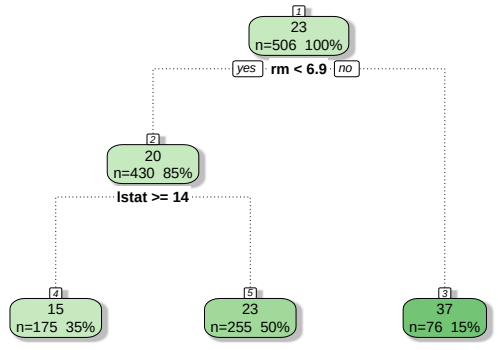
SPLIT PLACEMENT



Splits are usually placed at the mid-point of the observations they split: the large margin to the next closest observations makes better generalization on new, unseen data more likely.

Einführung in das statistische Lernen

CART: Stopping Criteria & Pruning



Pruning with complexity parameter = 0.072.

Learning goals

- Understand which problems arise when growing the tree until the end
- Know different stopping criteria
- Understand the idea of pruning

OVERFITTING TREES

The **recursive partitioning** procedure used to grow a CART would run until every leaf only contains a single observation.

- Problem 1: This would take a very long time, as the amount of splits we have to try *grows exponentially* with the number of leaves in the trees.
- Problem 2: At some point before that we should stop splitting nodes into ever smaller child nodes: very complex trees with lots of branches and leaves will *overfit the training data.*
- Problem 3: However, it is very hard to tell where we should stop while we're growing the tree: Before we have actually tried all possible additional splits further down a branch, we can't know whether any one of them will be able to reduce the risk by a lot (*horizon effect*).

STOPPING CRITERIA

Problems 1 and 2 can be “solved” by defining different **stopping criteria**:

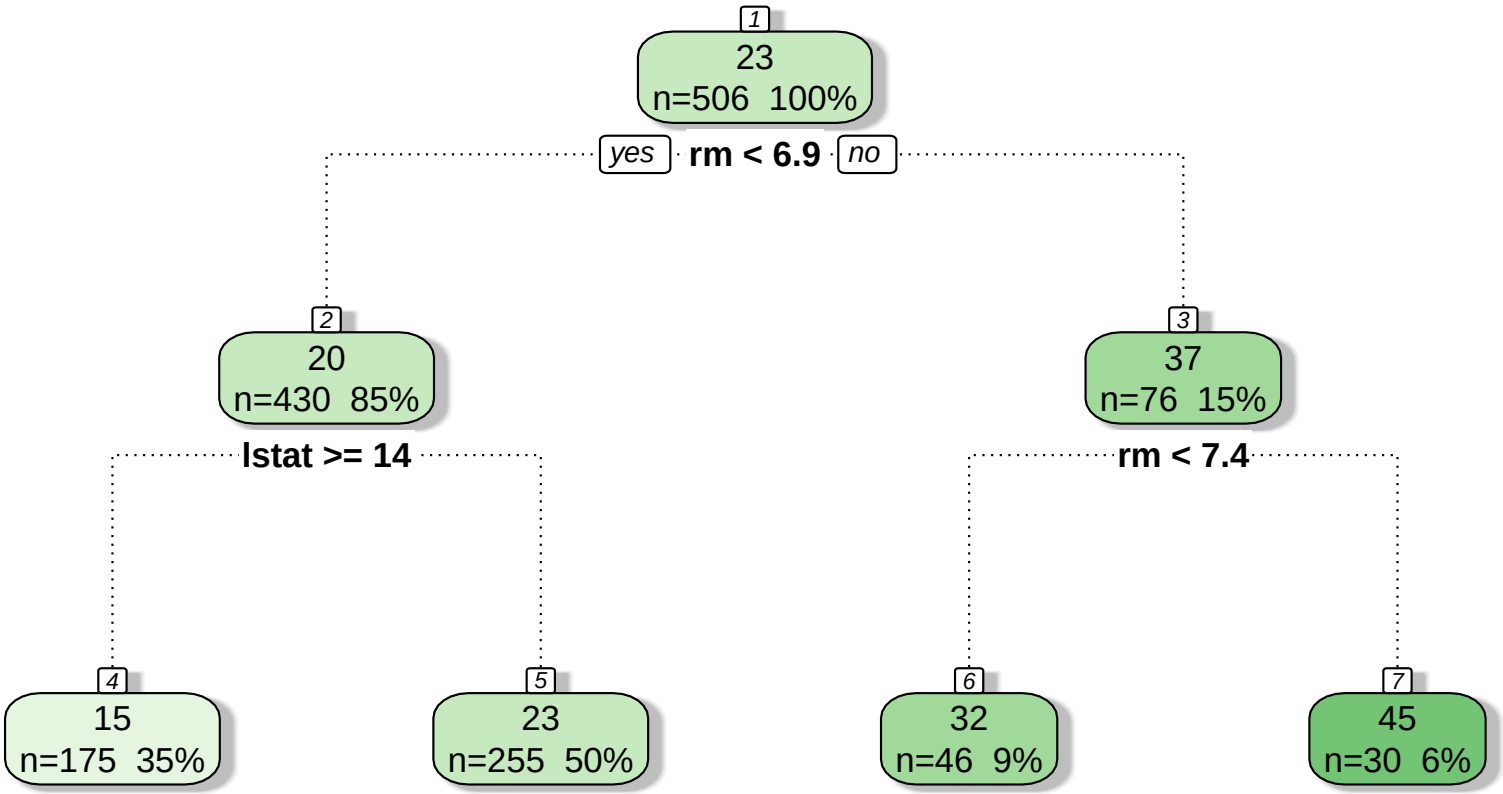
- Stop once the tree has reached a certain number of leaves.
- Don’t try to split a node further if it contains too few observations.
- Don’t perform a split that results in child nodes with too few observations.
- Don’t perform a split unless it achieves a certain minimal improvement of the empirical risk in the child nodes, compared to the empirical risk in the parent node.
- Obviously: Stop once all observations in a node have the same target value (**pure node**) or identical values for all features.

PRUNING

We try to solve problem 3 by **pruning**:

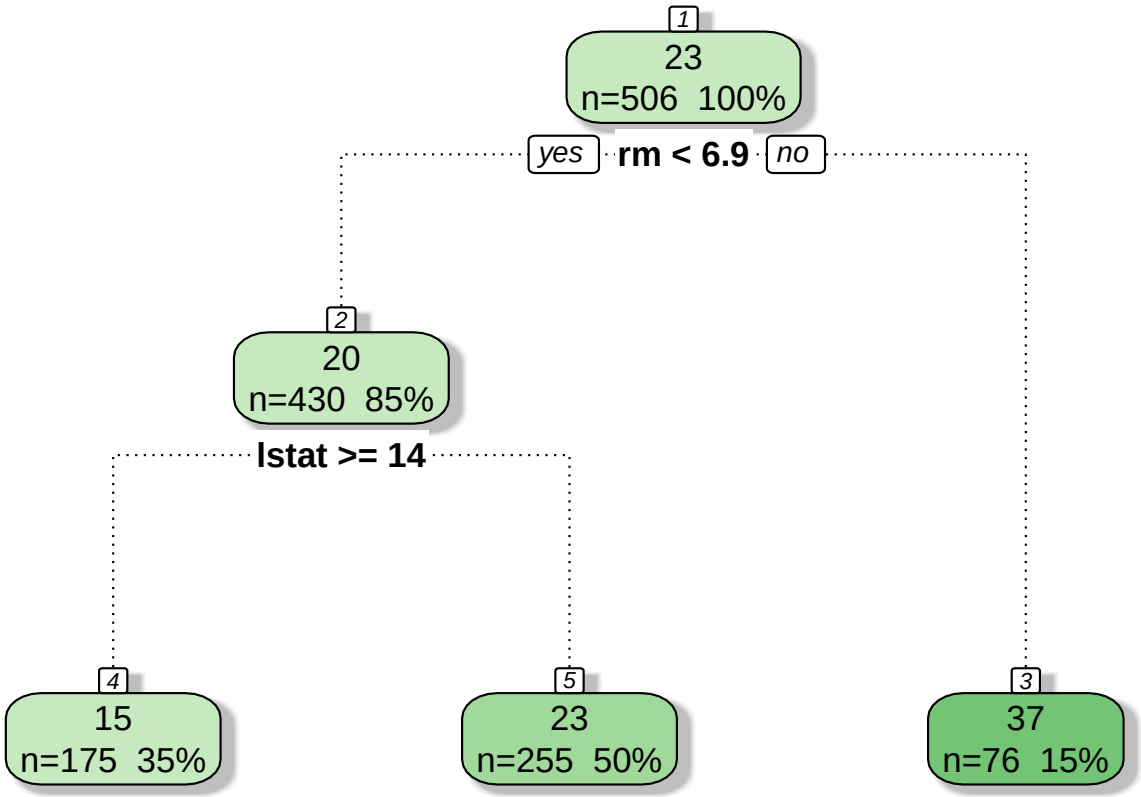
- A method to select the optimal size of a tree
- Finding a combination of suitable strict stopping criteria ("pre-pruning") is a hard problem: there are many different stopping criteria and it's hard to find the best combination (see chapter on **tuning**)
- Better: Grow a large tree, then remove branches so that the resulting smaller tree has optimal cross-validation risk
- Feasible without cross-validation: Grow a large tree, then remove branches so that the resulting smaller tree has a good balance between training set performance (risk) and complexity (i.e., number of terminal nodes). The trade-off between complexity and accuracy is governed by a **complexity parameter**.

PRUNING



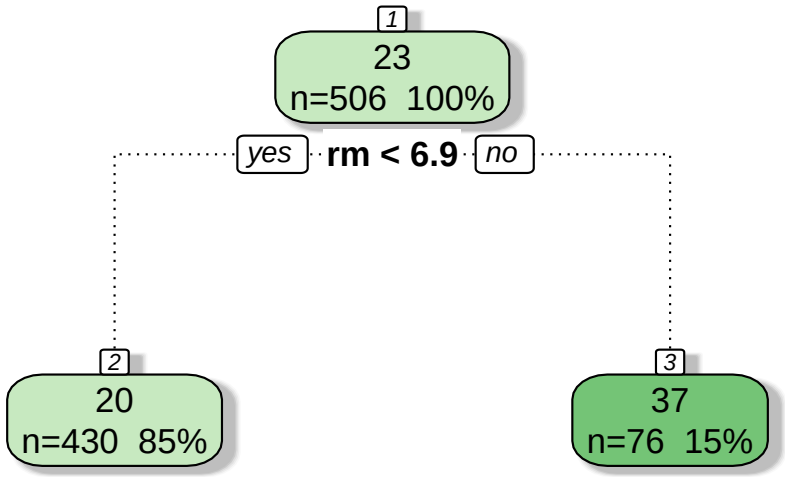
Full tree

PRUNING



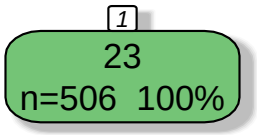
Pruning with complexity parameter = 0.072.

PRUNING



Pruning with complexity parameter = 0.171.

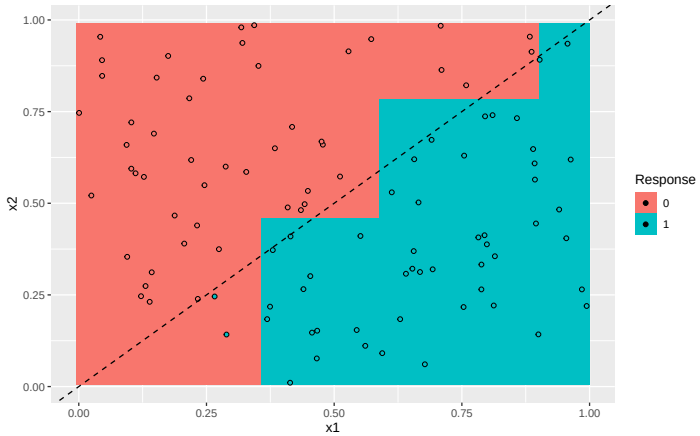
PRUNING



Pruning with complexity parameter = 0.453.

Einführung in das statistische Lernen

CART: Advantages & Disadvantages



Learning goals

- Understand advantages and disadvantages of CART
- Know how CART can be expressed in terms of hypothesis space, risk and optimization

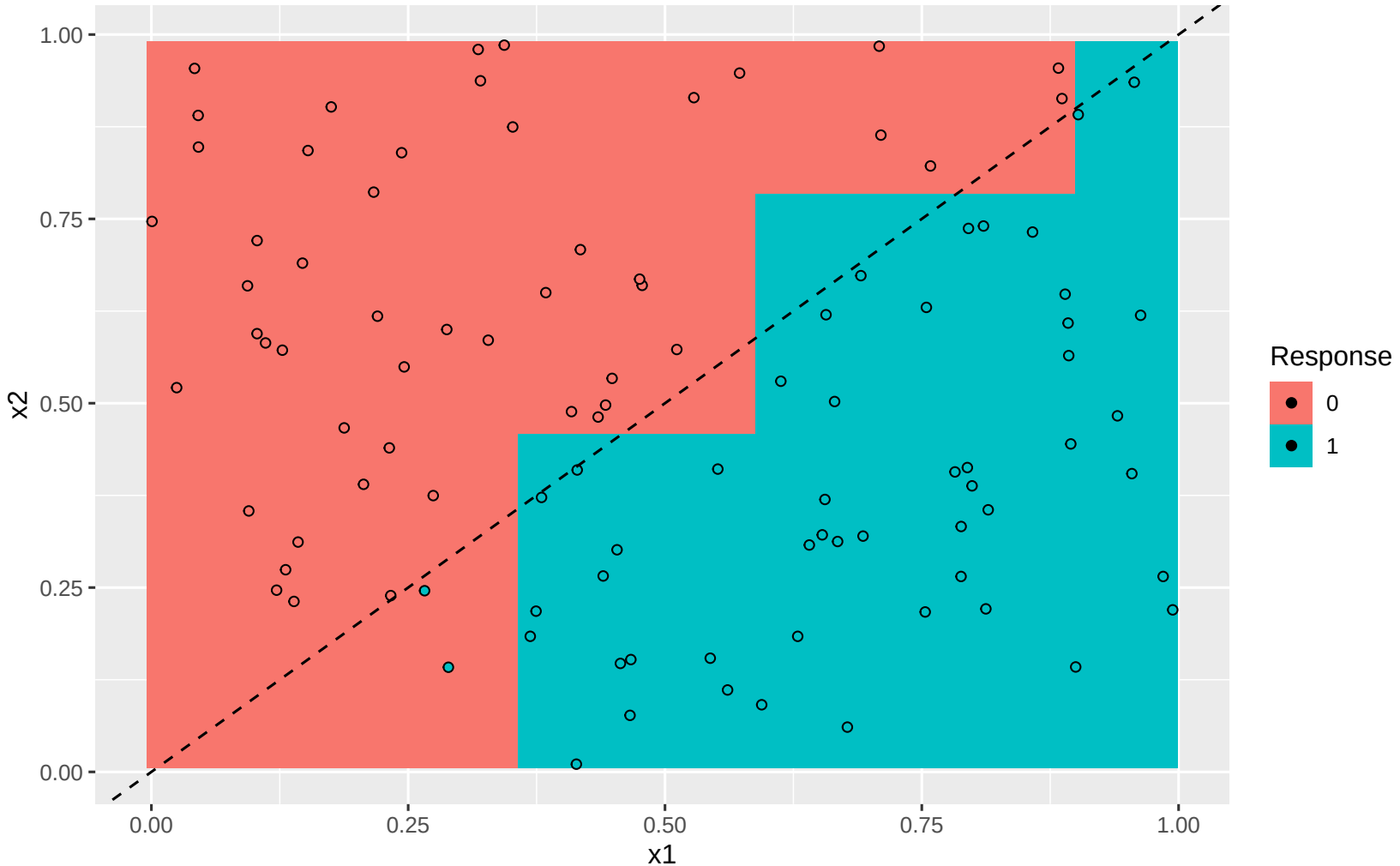
ADVANTAGES

- Fairly easy to understand, interpret and visualize.
- Not much preprocessing required:
  - automatic handling of non-numerical features
  - automatic handling of missing values via surrogate splits
  - no problems with outliers in features
  - monotone transformations of features change nothing so scaling of features is irrelevant
- Interaction effects between features are easily possible, even of higher orders
- Can model discontinuities and non-linearities (but see "disadvantages")

ADVANTAGES

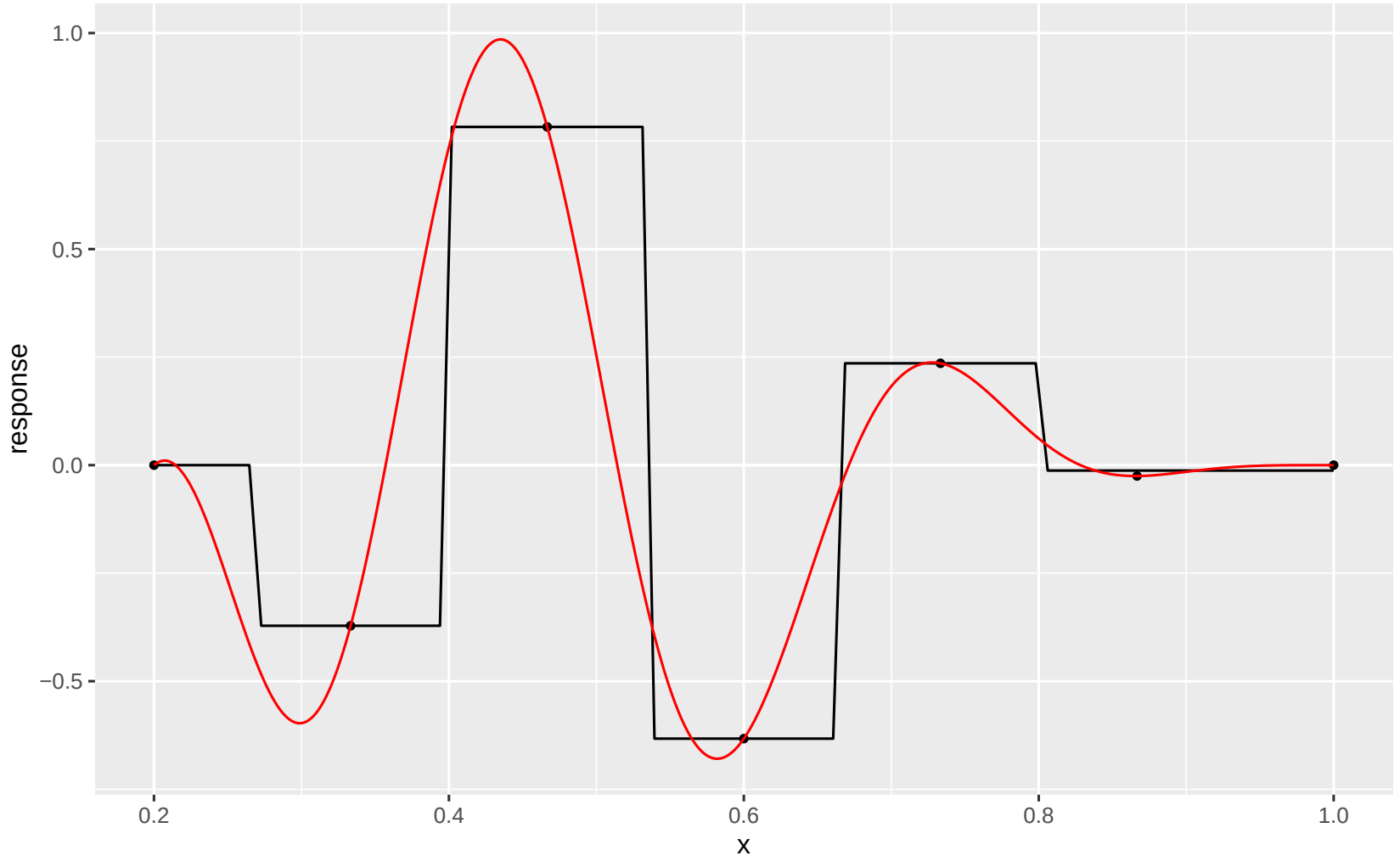
- Performs automatic feature selection
- Quite fast, scales well with larger data
- Flexibility through definition of custom split criteria or leaf-node prediction rules: clustering trees, semi-supervised trees, density estimation, etc.

DISADVANTAGE: LINEAR DEPENDENCIES



Linear dependencies must be modeled over several splits. Logistic regression would model this easily.

DISADVANTAGE: SMOOTH FUNCTIONS



Prediction functions of trees are never smooth as they are always step functions.

DISADVANTAGES

- Empirically not the best predictor: Combine with bagging (forest) or boosting!
- High instability (variance) of the trees. Small changes in the training data can lead to completely different trees. This leads to reduced trust in interpretation and is a reason why prediction errors of trees are usually not the best.
- In regression: Trees define piecewise constant functions, so trees often do not extrapolate well.

FURTHER TREE METHODOLOGIES

- AID (Sonquist and Morgan, 1964)
- CHAID (Kass, 1980)
- CART (Breiman et al., 1984)
- C4.5 (Quinlan, 1993)
- Unbiased Recursive Partitioning (Hothorn et al., 2006)

CART: SYNOPSIS

Hypothesis Space:
CART models are step functions over a rectangular partition of $\mathcal{X}$.
Their maximal complexity is controlled by the stopping criteria and the pruning method.

Risk:
Trees can use any kind of loss function for regression or classification.

Optimization:
Exhaustive search over all possible splits in each node to minimize the empirical risk in the child nodes.
Most literature on CARTs based on "impurity reduction" which is mathematically equivalent to empirical risk minimization:
Gini impurity $\cong$ Brier Score loss,
entropy impurity $\cong$ Bernoulli loss,
variance impurity $\cong$ L2 loss.