Unit 5

Regression and classification trees

Tree-based methods

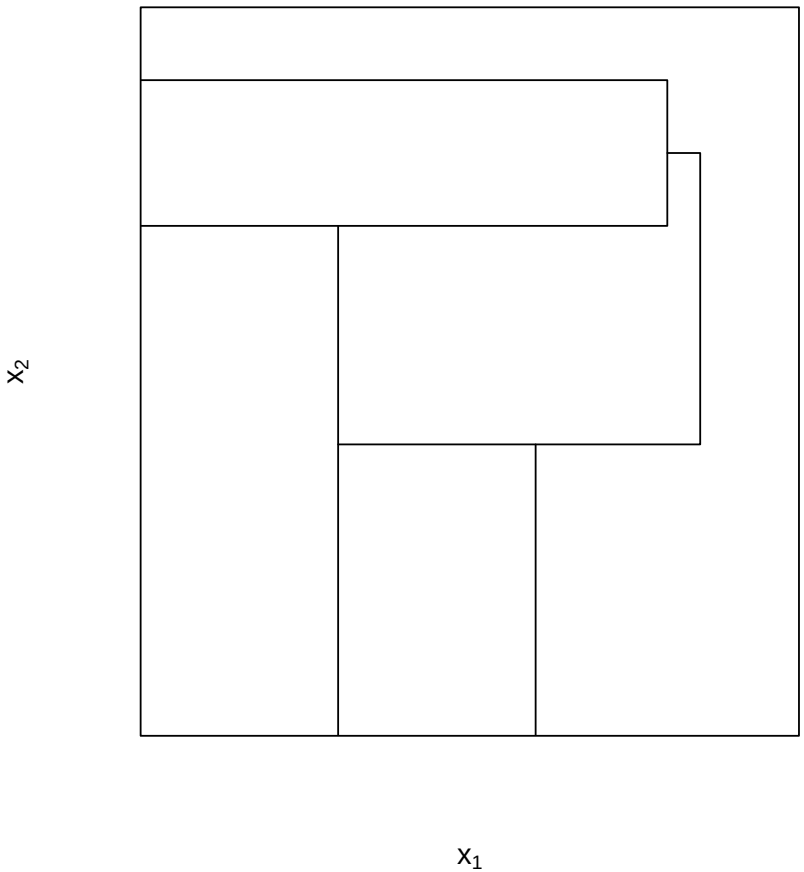
- Tree-based methods partition the feature space into a set of rectangles and fit a simple model (e.g., a constant) in each one.
- In general attention is restricted to recursive binary partitions of the feature space.
- In this case the regression model can be represented by a binary tree.
- Different algorithms have been proposed to construct trees:
  - CART: classification and regression trees
  - C4.5
  - CHAID
  - Conditional inference trees
  - . . .

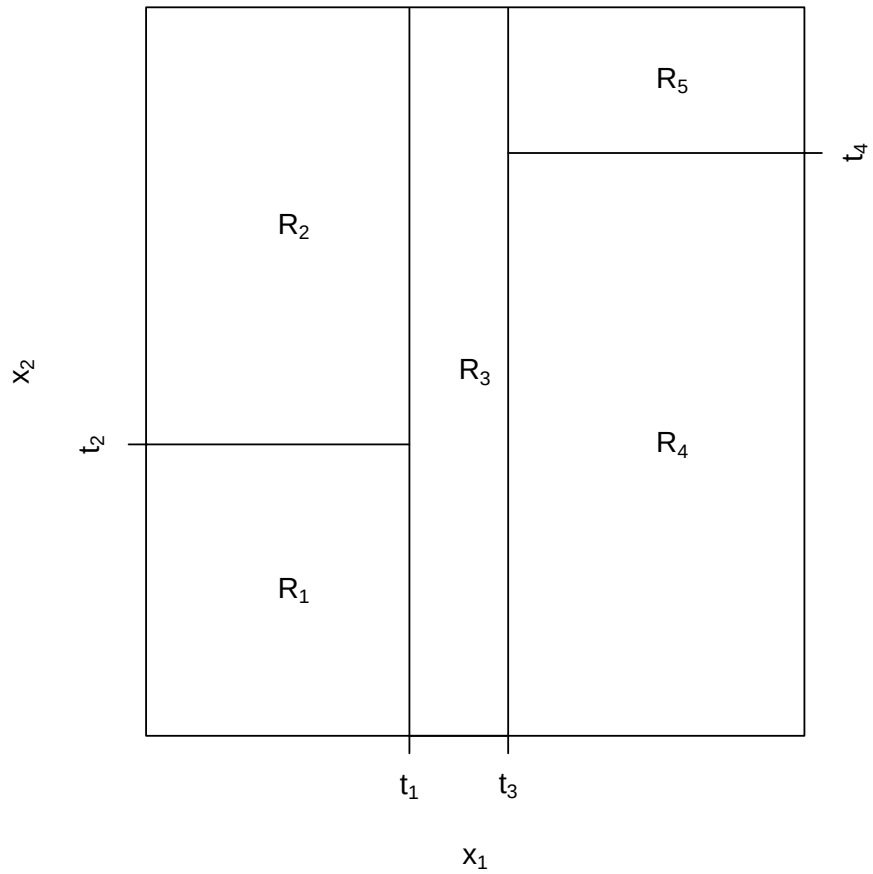
Tree-based methods / 2

- Assume a regression problem with continuous response Y and inputs X_1 and X_2 each taking values in the unit interval.
- Different partitions of the input space into rectangles are possible depending on if they are just limited by lines parallel to the coordinate axes or result from binary recursive partitions.
- The response Y is modeled by the mean in each region.
- The regression model is given by:

$$\hat{f}(X) = \sum_{m=1}^{M} c_m I\{(X_1, X_2) \in R_m\}$$

Tree-based methods / 3





Tree-based methods / 4

- The regions R_m in keeping with the tree analogy are referred to as *terminal nodes* or *leaves* of the tree.
- The points along the tree where the predictor space is split are referred to as *internal nodes*.
- The segments that connect the nodes are referred to as *branches*.

Advantages

- Fitted model is easy to understand.
- Interpretability also due to the visualization.
- Interactions between predictors can be well captured.
- The tree structure reflects stepwise decision making.
- Works for unknown, non-linear functions.
- Predictors can be correlated.
- Also applicable for small sample sizes and a large number of predictors.
- The models are invariant under transformations in the predictor space.

Disadvantages

- High instability / variance.
  - Often only small changes in the data can lead to a very different series of splits.
  - This feature makes interpretation rather precarious.
  - The hierarchical nature of the process is crucial for this instability: An error on a higher level propagates to a lower level.
- Lack of smoothness, but step function is fitted.
- Difficulty in capturing additive structure:
  - Linear relationships need to be captured by a series of splits.
- Simple trees usually do not have a lot of predictive power.
- It can be hard to assess uncertainty in inference about trees.

Regression trees

- The data consist of p inputs and a response, for each of N observations: (x_i, y_i), for $i = 1, \ldots, N$ with $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$.
- The algorithm needs to decide on:
  - the splitting variable;
  - the split point;
  - the topology of the tree;
  - the tree size;
  - the prediction of the response in the terminal nodes.
- Suppose we have a partition in M regions $R_1, R_2, \ldots, R_M$ and we model the response as a constant c_m in each region:

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m).$$

Regression trees / 2

- If the optimization criterion is minimization of the sum of squares

$$\sum_{i=1}^{N}(y_i - f(x_i))^2,$$

  the best $\hat{c}_m$ is the average of the y_i in the region R_m:

$$\hat{c}_m = \text{ave}(y_i | x_i \in R_m).$$

- Finding the best binary partition in terms of minimum sum of squares is generally computationally infeasible. A greedy algorithm is thus pursued.
- Starting with all of the data, consider a splitting variable j and split point s, and define the pair of half-planes:

$$R_1(j, s) = \{X | X_j \leq s\} \qquad \text{and} \qquad R_2(j, s) = \{X | X_j > s\}.$$

Regression trees / 3

- Then we seek the splitting variable j and split point s that solve:

$$\min_{j,s} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right].$$

- For any choice of j and s, the inner minimization is solved by

$$\hat{c}_1 = \text{ave}(y_i | x_i \in R_1(j,s)),$$
$$\hat{c}_2 = \text{ave}(y_i | x_i \in R_2(j,s)).$$

- For each splitting variable, the determination of the split point s can be done rather quickly.
- By scanning through all of the inputs, the best pair (j, s) can be determined.

Regression trees / 4

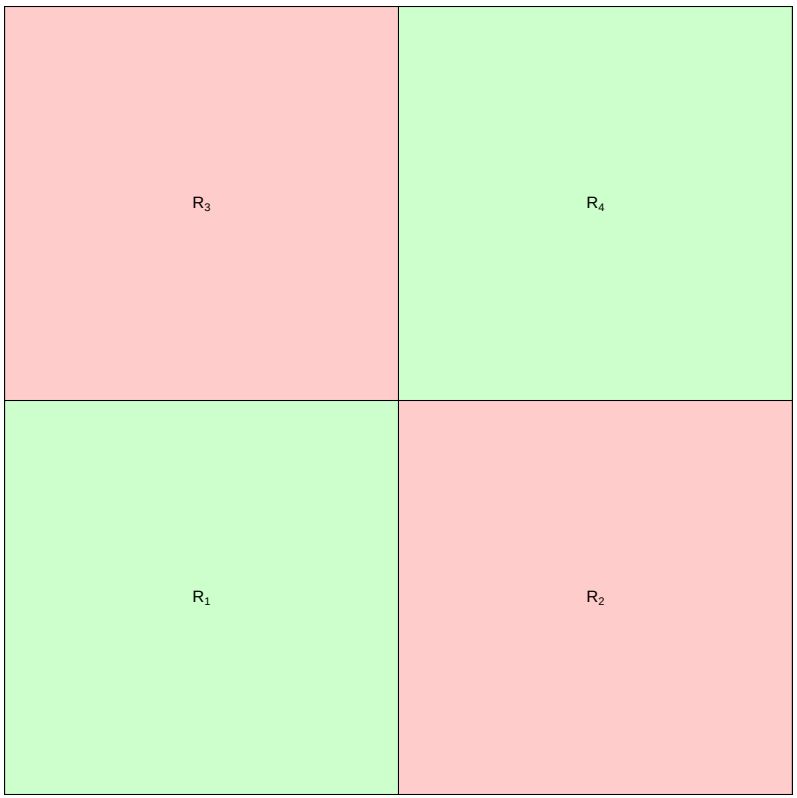
- Given the best split, the data is partitioned into the two resulting regions.
- The splitting process is repeated on each of the two regions.
- Then this process is repeated on all of the resulting regions.

Regression trees: Determining the size

- The tree size is a tuning parameter which governs the model's complexity and thus potential over- or underfitting.
- The optimal tree size should be adaptively chosen from the data.
- There are two options:
  - Splits are only performed as long as the decrease in sum of squares due to a split exceeds a given threshold.
    $\rightarrow$ Problem: Sometimes the combination of several splits is necessary to obtain a good partition.
  - Grow a large tree T_0, stopping the splitting process only when some minimum node size (say 5) is reached. Then prune this tree using *cost-complexity pruning*.

Regression trees: Determining the size / 2

- Assume the following true model:
  - Splitting only once leads to no improvement.
  - Two splits are necessary for a perfect fit.



Cost-complexity pruning

- Define a subtree $T \subset T_0$ to be any tree that can be obtained by pruning T_0.
- Pruning refers to collapsing any number of its internal (non-terminal) nodes.
- Terminal nodes are indexed by m, with node m representing region R_m.
- Let $|T|$ denote the number of terminal nodes in T.
- Let

$$N_m = \#\{x_i \in R_m\},$$

$$\hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i,$$

$$Q_m(T) = \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2.$$

Cost-complexity pruning / 2

- The cost-complexity criterion is given by:

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T|.$$

- The parameter $\alpha \geq 0$ governs the tradeoff between tree size and its goodness of fit to the data.
- Large values of α result in smaller trees T_α, and conversely for smaller values of α.
- For $\alpha = 0$ the solution is the full tree T_0.
- For any other α the subtree $T_\alpha \subseteq T_0$ which minimizes $C_\alpha(T)$ needs to be determined.
- One can show that for each α there exists a unique smallest subtree T_α.

Cost-complexity pruning / 3

- *Weakest link pruning* can be used to find T_α:
  - Successively collapse the internal node that produces the smallest per-node increase in

$$\sum_{m=1}^{M} N_m Q_m(T).$$

  - Continue until only the single-node (root) tree is obtained.

  This finite sequence of subtrees can be shown to contain T_α.
- Estimation of α is achieved by five- or tenfold cross-validation: The value $\hat{\alpha}$ is selected that minimizes the cross-validated sum of squares.
- The final tree is $T_{\hat{\alpha}}$.

Classification trees

- If the target is a classification outcome taking values $1, 2, \ldots, K$, the algorithm needs to be modified w.r.t. the criteria for splitting and pruning the tree.
- In a node m, representing a region R_m with N_m observations, let

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k),$$

  be the proportion of class k observations in node m.

Classification trees / 2

- Standard measures for the impurity $Q_m(T)$ are:
  - Misclassification error:

$$\frac{1}{N_m} \sum_{x_i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)}.$$

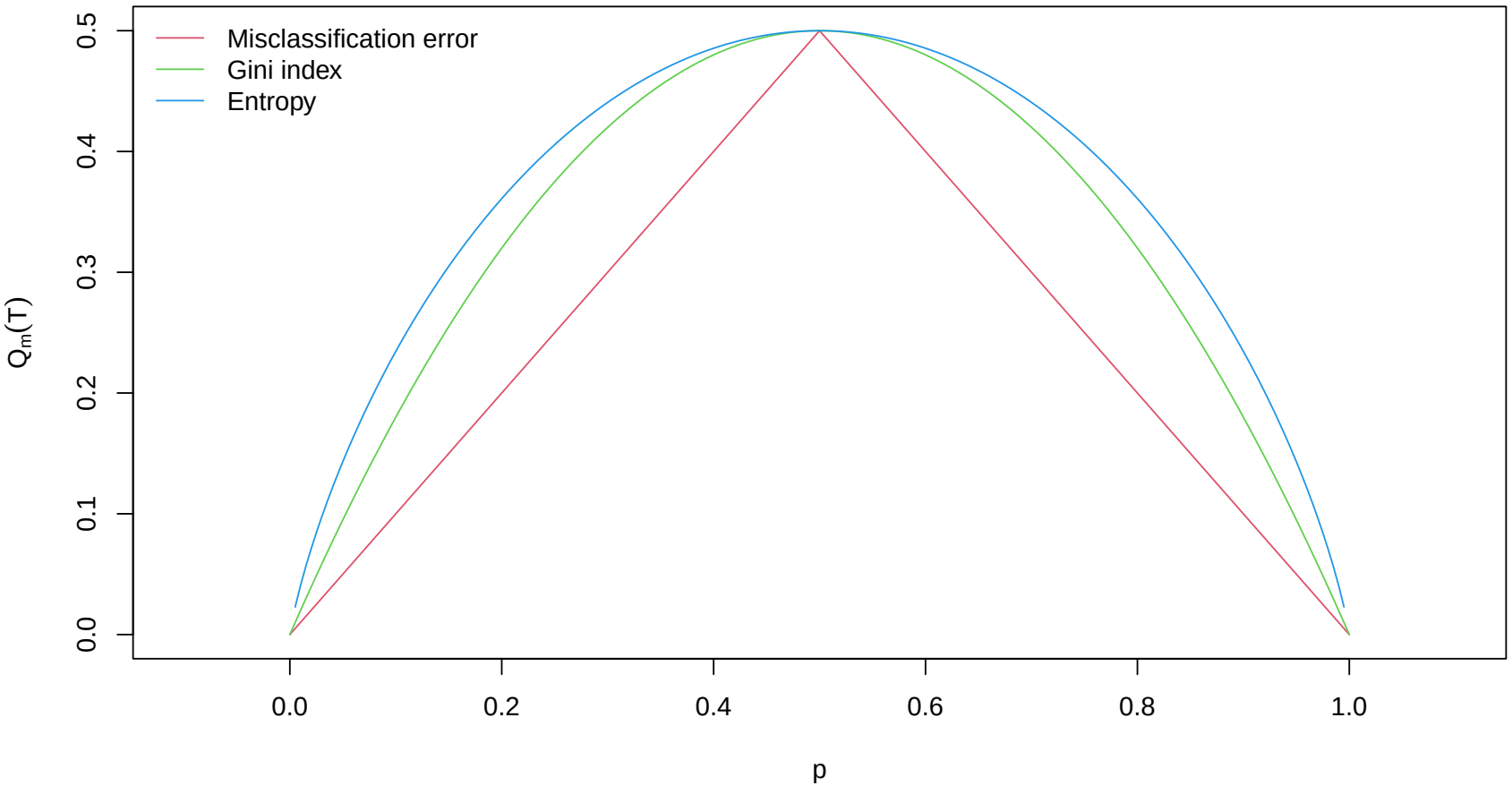
  - Gini index:

$$\sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}).$$

  - Cross-entropy/deviance:

$$-\sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk}).$$

Classification trees / 3



Categorical predictors

- For a predictor with q unordered values there are $2^{q-1} - 1$ possible partitions of the q values into two groups.
- An enumerative computation might be prohibitive for large q.
- The computation can be simplified for $0/1$ output: The predictor classes can be ordered by proportion falling in outcome class 1 and then the predictor treated like an ordinal/ordered one.
  - This gives the optimal split for cross-entropy or the Gini index.
  - The same holds true for a quantitative outcome with squared error loss.
- The partitioning algorithm tends to favor categorical predictors with many levels q: The number of possible partitions grows exponentially in q and the more likely it is to spuriously obtain a good partition.

The loss matrix

- In a classification setting the consequences of misclassifying observations might be different for different classes.
- This differences in loss associated with misclassifications can be captured using a $K \times K$ loss matrix $\boldsymbol{L}$.
- $L_{kk'}$ is the loss incurred for classifying a class k observation as class k'.
- Typically no loss is incurred for correct classifications, that is $L_{kk} = 0$, for all k.

The loss matrix / 2

- The losses can be incorporated by modifying the Gini index to

$$\sum_{k \neq k'} L_{kk'} \hat{p}_{mk} \hat{p}_{mk'}$$

  for the multiclass case.
- For two classes the observations in class k can be weighted by $L_{kk'}$ (assuming this is constant for $k' \neq k$).
- Observation weighting can also be used with the deviance.
- In a terminal node the final classification is then determined by

$$k(m) = \arg\min_k \sum_l L_{lk} \hat{p}_{ml}.$$

Missing predictor values

- Missing values are in general handled by either removing observations or imputing values.
- Two alternative approaches for tree-based methods exit.
- A new category “missing” can be added for categorical predictor variables.

Missing predictor values / 2

- Surrogate variables can be constructed:
  - For considering a split only the observations where that predictor is not missing are used.
  - Having chosen the best (primary) predictor and split point, a list of surrogate predictors and split points are formed.
  - The first surrogate predictor is the predictor and corresponding split point that best mimics the split of the training data achieved by the primary split. The second surrogate split is the second best, etc.
  - When sending observations down the tree either during training or prediction, the surrogate splits are used in order, if the primary splitting predictor is missing.
  - Surrogate splits exploit correlations between predictors to alleviate the problem of missing data.

Why binary splits?

- Multiway splits possibly partition the data into more than two groups.
- The drawback of multiway splits is that the data is fragmented too quickly, leaving insufficient data at the next level down.
- Multiway splits can be achieved by a series of binary splits.

Other tree-building procedures

- The discussion of tree-based methods so far focused on CART (classification and regression trees).
- Other popular algorithms are:
  - C4.5 (ID3, C5.0)
    - Early versions were limited to categorical predictors.
    - Early versions used a top-down rule without pruning.
    - Newer versions are more similar to CART.
  - CHAID (Chi-square Automatic Interaction Detector):
    - Only for categorical predictors.
  - Condititional inference trees:
    - Statistical association tests used for variable selection.
    - *p*-values serve as stopping criterion.
    - Statistical two-sample tests used for split point selection.

Implementations in R

- **rpart**: implements CART.
- **party** / **partykit**: implements conditional inference trees.
- **RWeka** provides an interface to **Weka** which implements the J4.8 variant of C4.5.

Example: spam

- Binary classification problem.
- 57 potential predictor variables.
- Fit a tree using **rpart**:

```
> data("spam", package = "ElemStatLearn")
> library("rpart")
> tree <- rpart(spam ~ ., data = spam,
+   method = "class", parms = list(split = "gini"),
+   control = list(cp = 0.0001))
> options(digits = 4, width= 60)
> printcp(tree)
```

Example: spam / 2

```
Classification tree:
rpart(formula = spam ~ ., data = spam, method = "class", pa
    control = list(cp = 1e-04))

Variables actually used in tree construction:
 [1] A.12 A.16 A.18 A.19 A.2  A.21 A.22 A.24 A.25 A.27 A.28
[12] A.33 A.36 A.37 A.45 A.46 A.49 A.5  A.50 A.52 A.53 A.55
[23] A.56 A.57 A.6  A.7  A.8

Root node error: 1813/4601 = 0.39

n= 4601

        CP nsplit rel error xerror  xstd
1  0.47656      0      1.00   1.00 0.018
```

Example: spam / 3

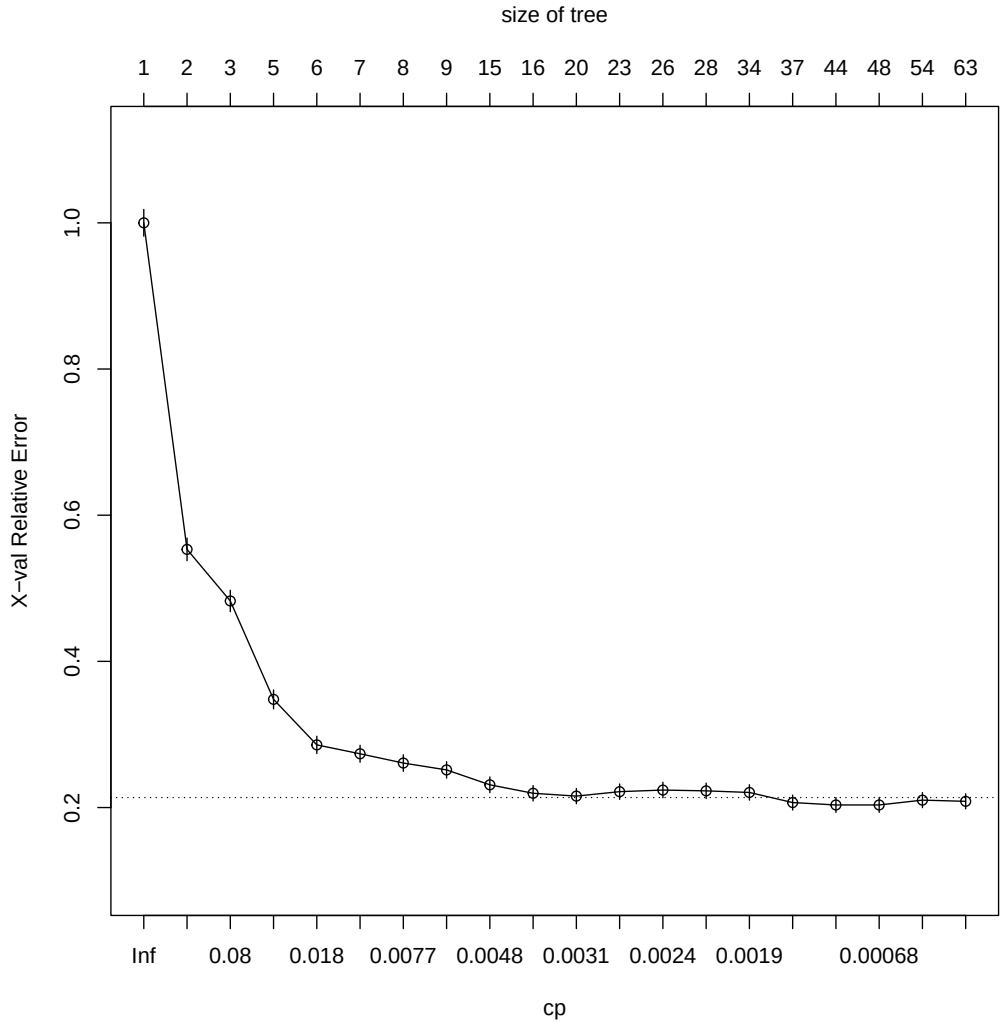
2	0.14892	1	0.52	0.55	0.015
3	0.04302	2	0.37	0.48	0.015
4	0.03089	4	0.29	0.35	0.013
5	0.01048	5	0.26	0.29	0.012
6	0.00827	6	0.25	0.27	0.012
7	0.00717	7	0.24	0.26	0.011
8	0.00530	8	0.23	0.25	0.011
9	0.00441	14	0.20	0.23	0.011
10	0.00359	15	0.19	0.22	0.011
11	0.00276	19	0.18	0.22	0.010
12	0.00257	22	0.17	0.22	0.011
13	0.00221	25	0.16	0.22	0.011
14	0.00211	27	0.16	0.22	0.011
15	0.00165	33	0.14	0.22	0.011
16	0.00110	36	0.14	0.21	0.010

Example: spam / 4

```
17 0.00083     43       0.13   0.20 0.010
18 0.00055     47       0.13   0.20 0.010
19 0.00037     53       0.12   0.21 0.010
20 0.00010     62       0.12   0.21 0.010

> plotcp(tree)
```

Example: spam / 5



Example: spam / 6

```
> imin <- which.min(tree$cptable[, "xerror"])
> select <- which(
+   tree$cptable[, "xerror"] <
+     sum(tree$cptable[imin, c("xerror", "xstd")]))[1]
> ptree <- prune(tree, cp = tree$cptable[select, "CP"])
```

- Select a smaller model for visualization.

```
> ptree <- prune(tree, cp = 0.01)
> ptree
```

- Coerced to **partykit** output for plotting the tree:

```
> library("partykit")
> ptree <- partykit::as.party(ptree)
```

Example: spam / 7