Ch9 Tree-based methods for classification and regression

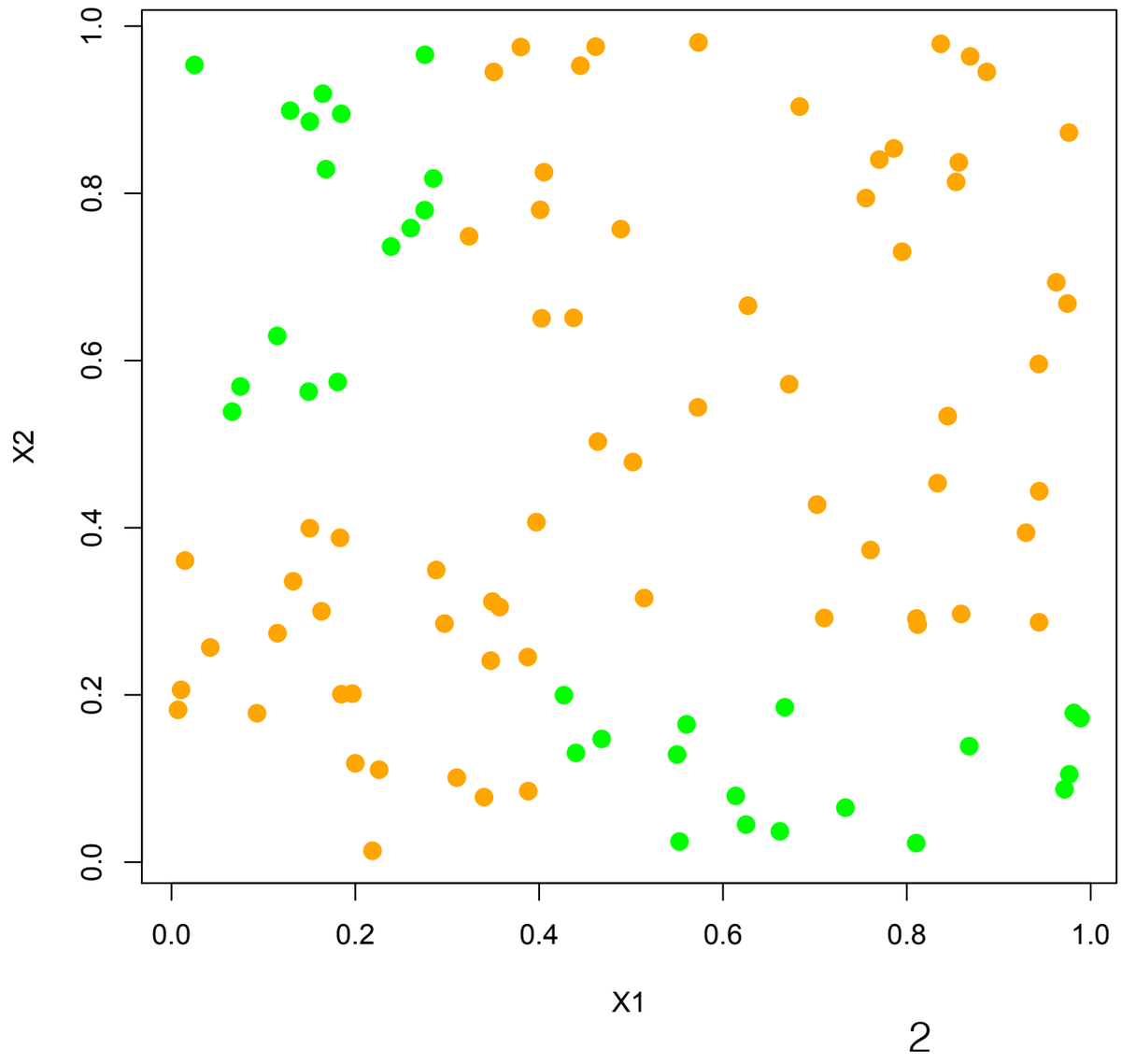
MATH 6312
Department of mathematics, UTA

ESL 9.2, 9.2.1, 9.2.2, 9.2.3, 9.2.5

Optional reading: ESL 9.2.4

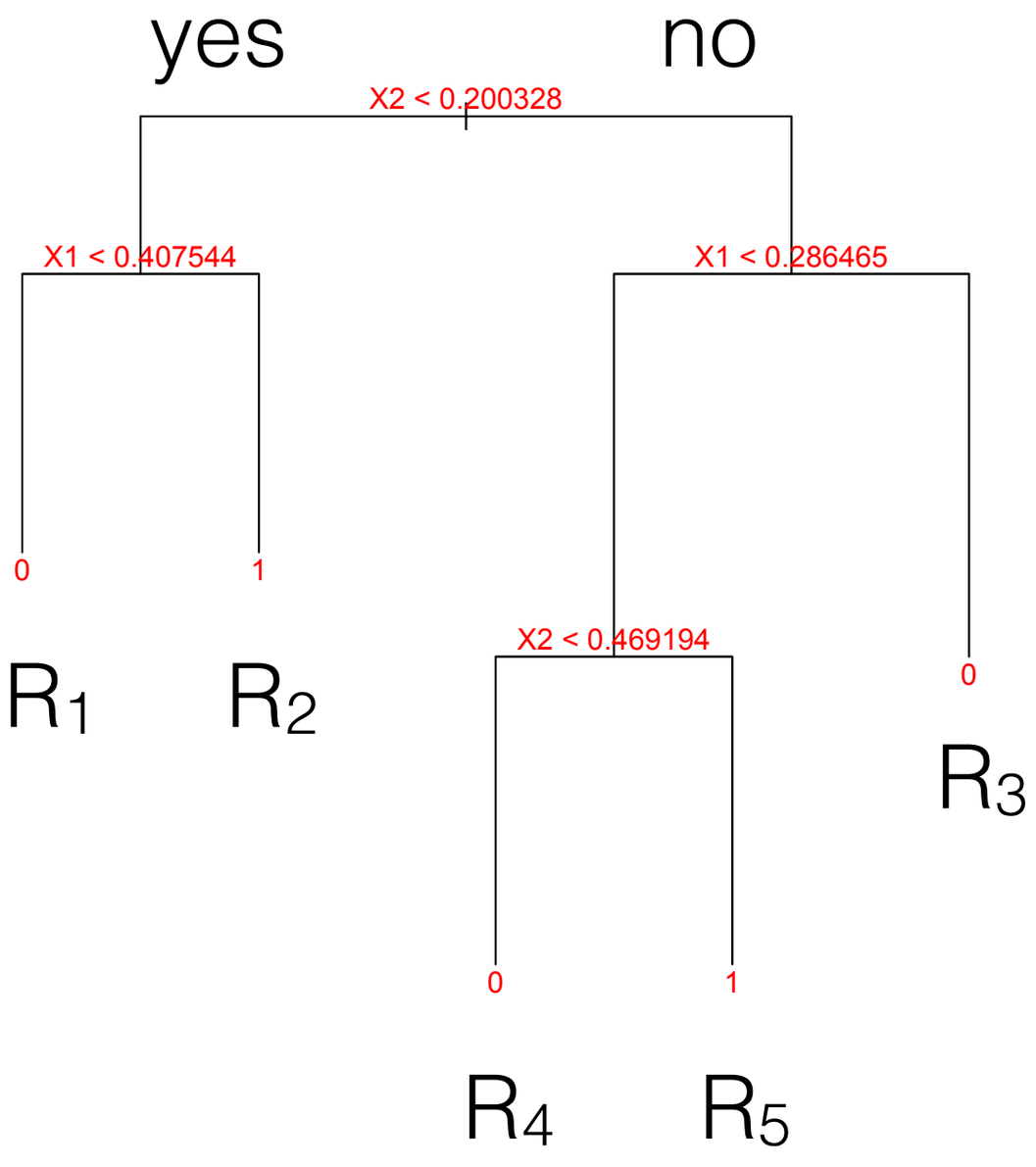
Toy example: classification trees

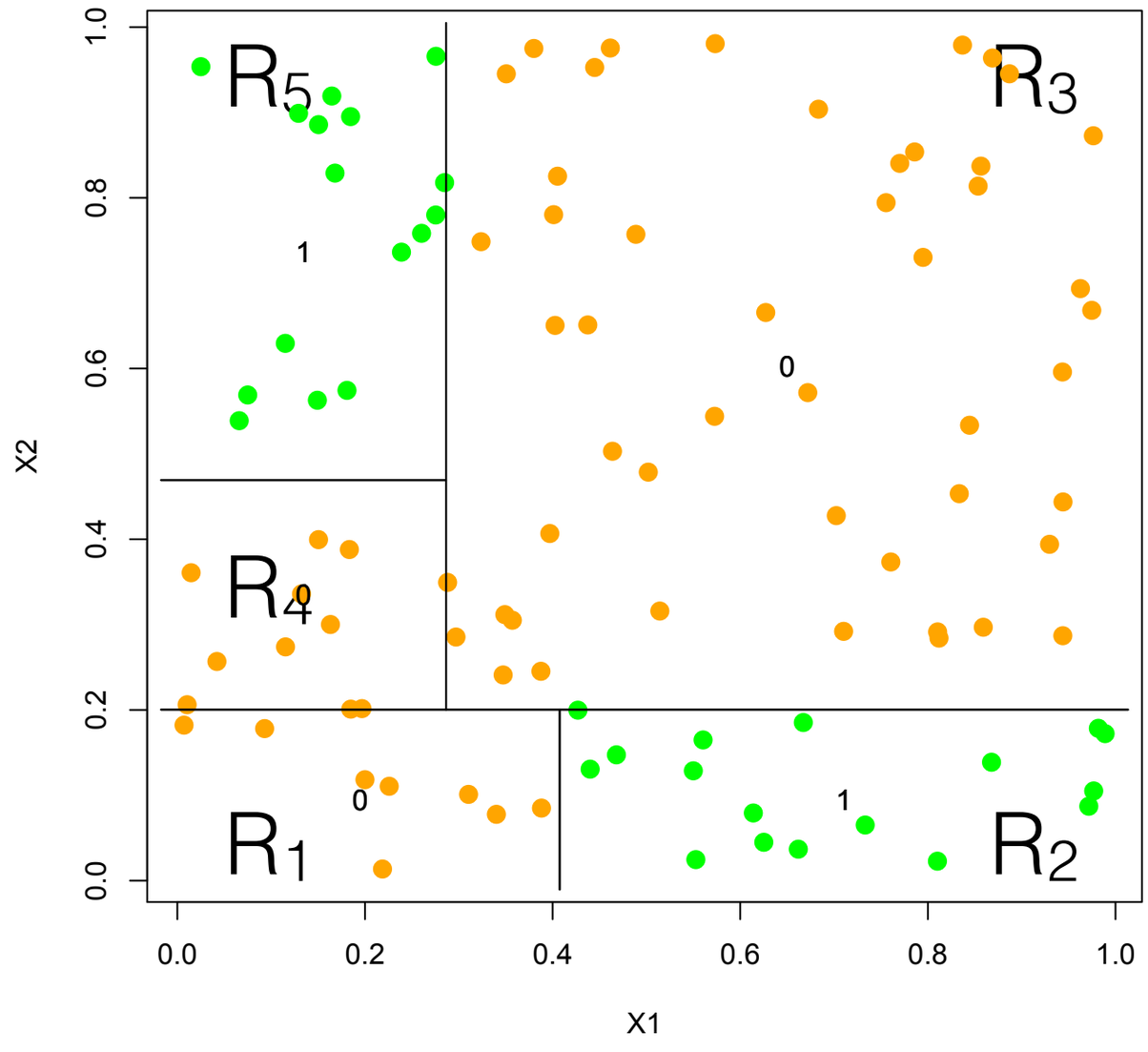
$y \in \{0,1\}$ and two dimensional feature space



Linear decision boundary does not separate points well!

Simple classification tree





Tree-based methods

- Tree-based based methods for predicting y from a feature vector $\mathbf{x} \in \mathbb{R}^p$ divide up the feature space into rectangles, and then fit a very simple model in each rectangle.

- This works both when y is discrete and continuous, i.e., both for classification and regression

- CART: classification and regression tree

- CART is based on binary recursive partitioning of the feature space into regions and fitting a tree model.

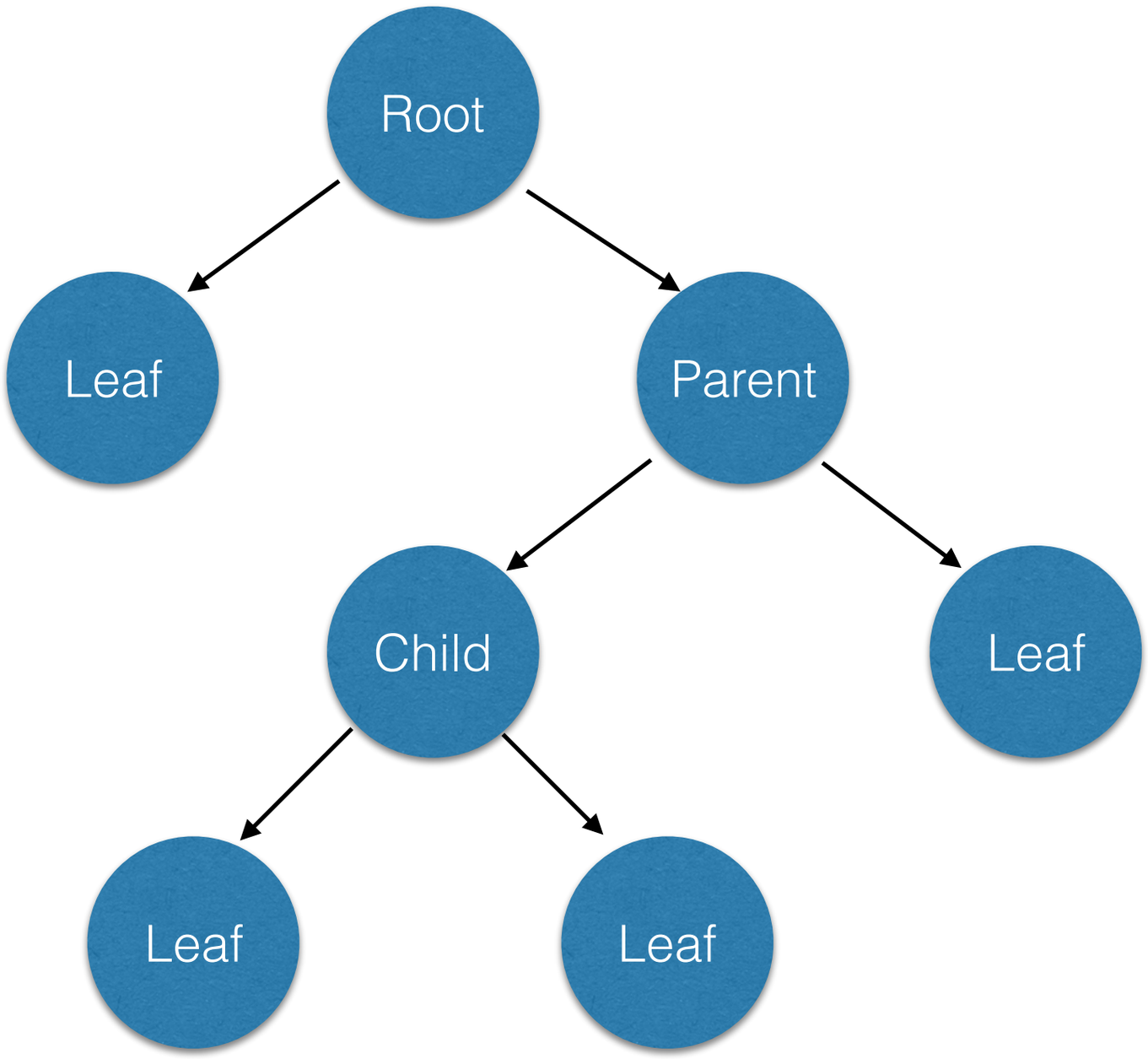
- Some characteristics

  - No distribution assumptions on the variables

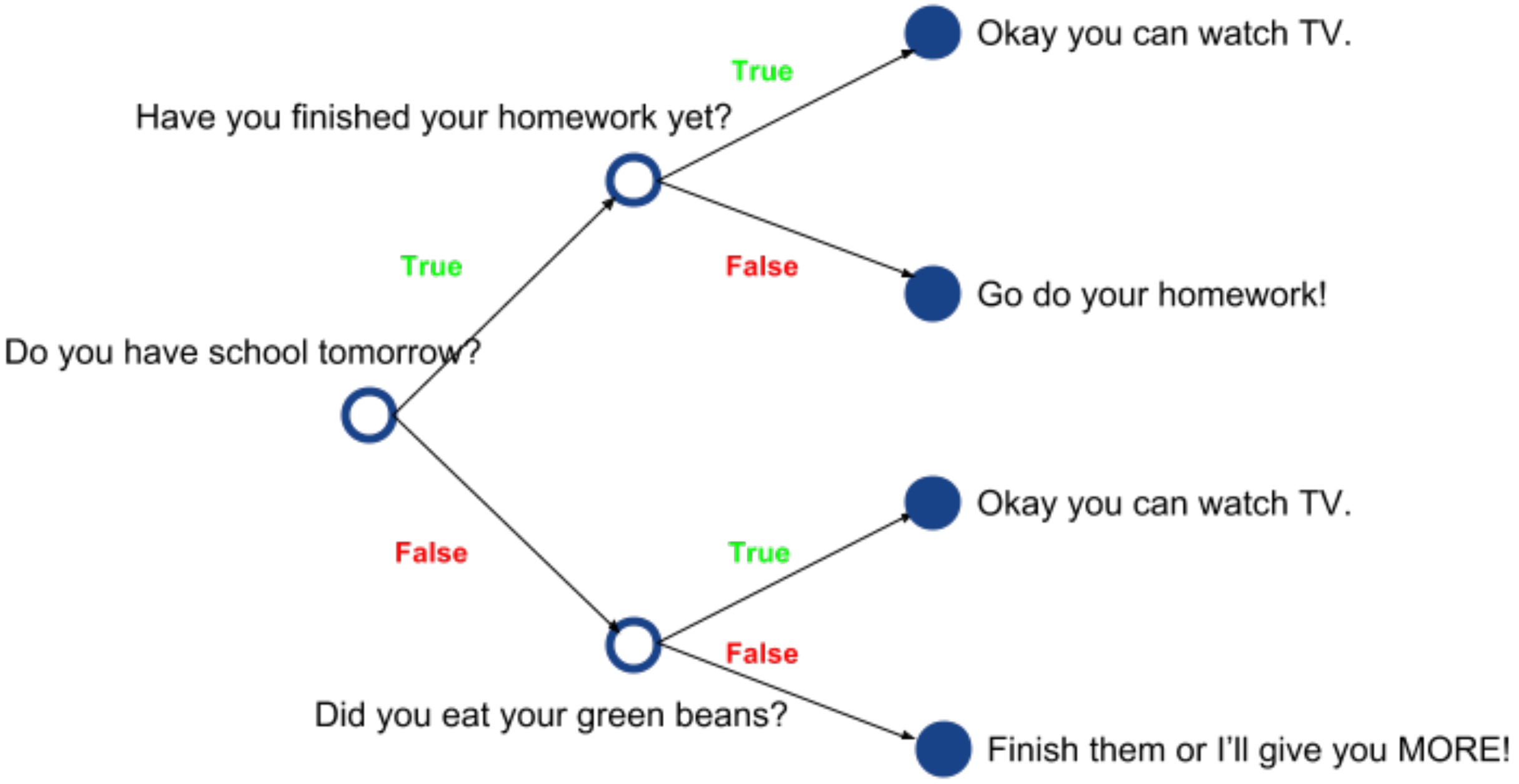
  - The feature space can be fully represented by a single tree

  - Interpretable

Tree structure



- Root node
- Parent node
- Child node
- Internal node
- Terminal node (leaf)


Okay you can watch TV.
True
Have you finished your homework yet?
True
False
Go do your homework!
Do you have school tomorrow?
False
True
Okay you can watch TV.
False
Did you eat your green beans?
Finish them or I'll give you MORE!

- CART is highly interpretable, and they mimic the way that we make decisions.

- Tree-based methods involve segmenting the predictor space into a number of simple regions.

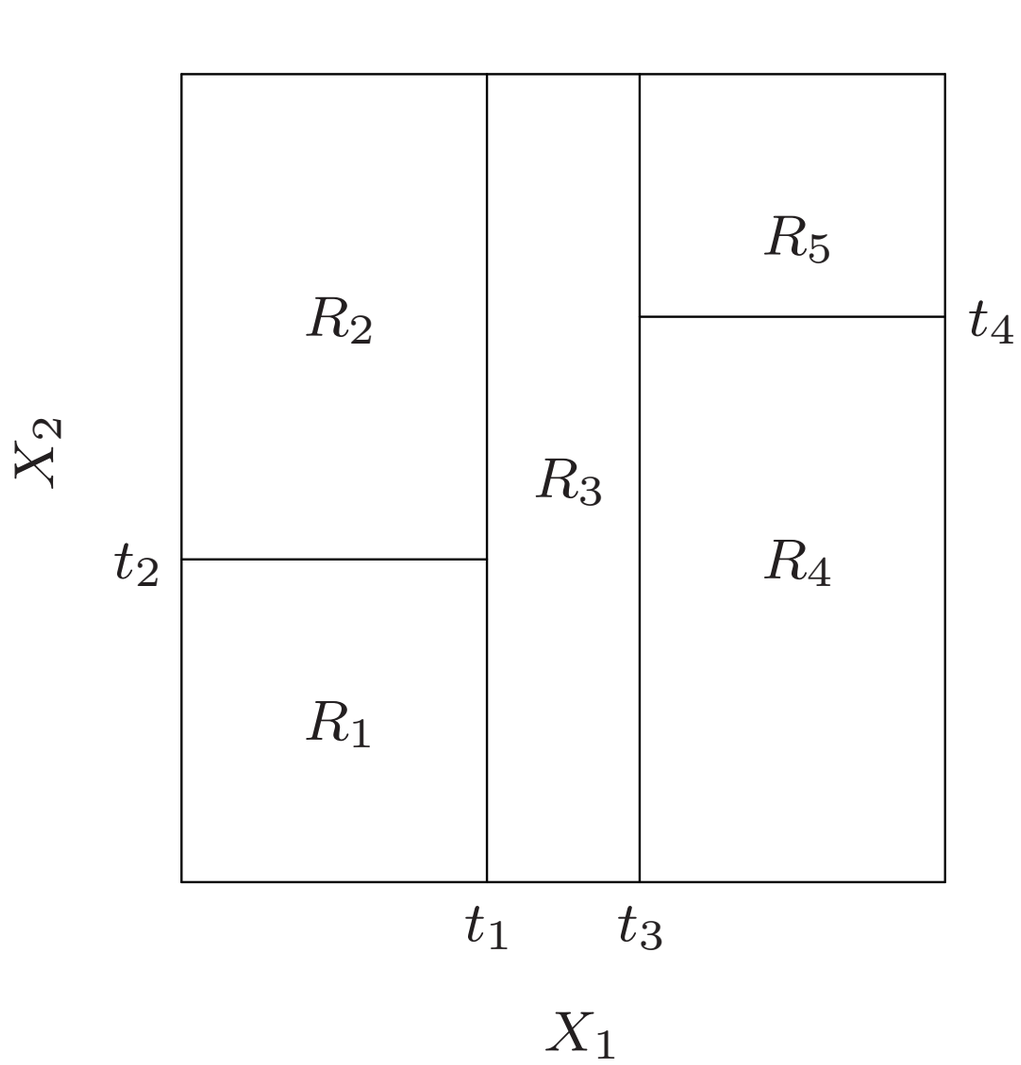
- Regions can be achieved by making successive binary splits on the predictors variables $X_1,\ldots,X_p$.

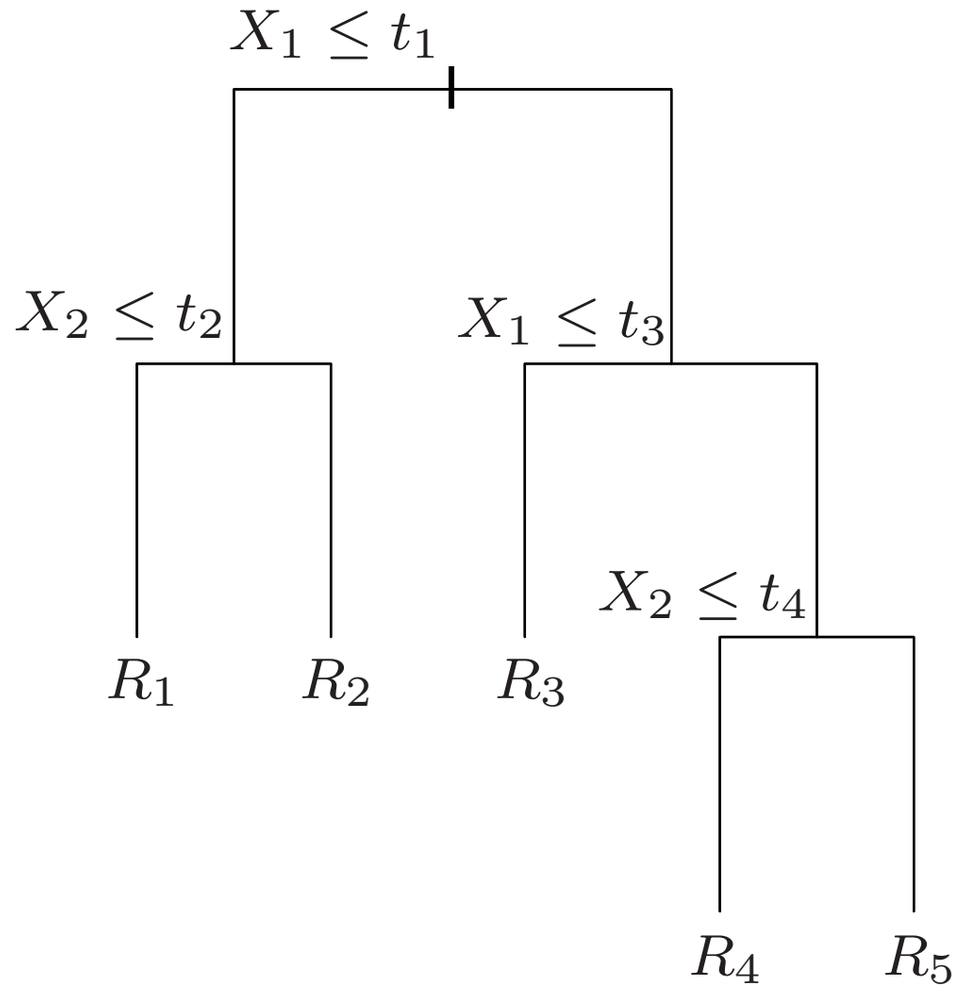
  - We choose a splitting variable X_j, and then divide the feature space into two parts ($X_j \leq c$ vs $X_j > c$).

  - We continue to make binary splits on each half.

Regression trees

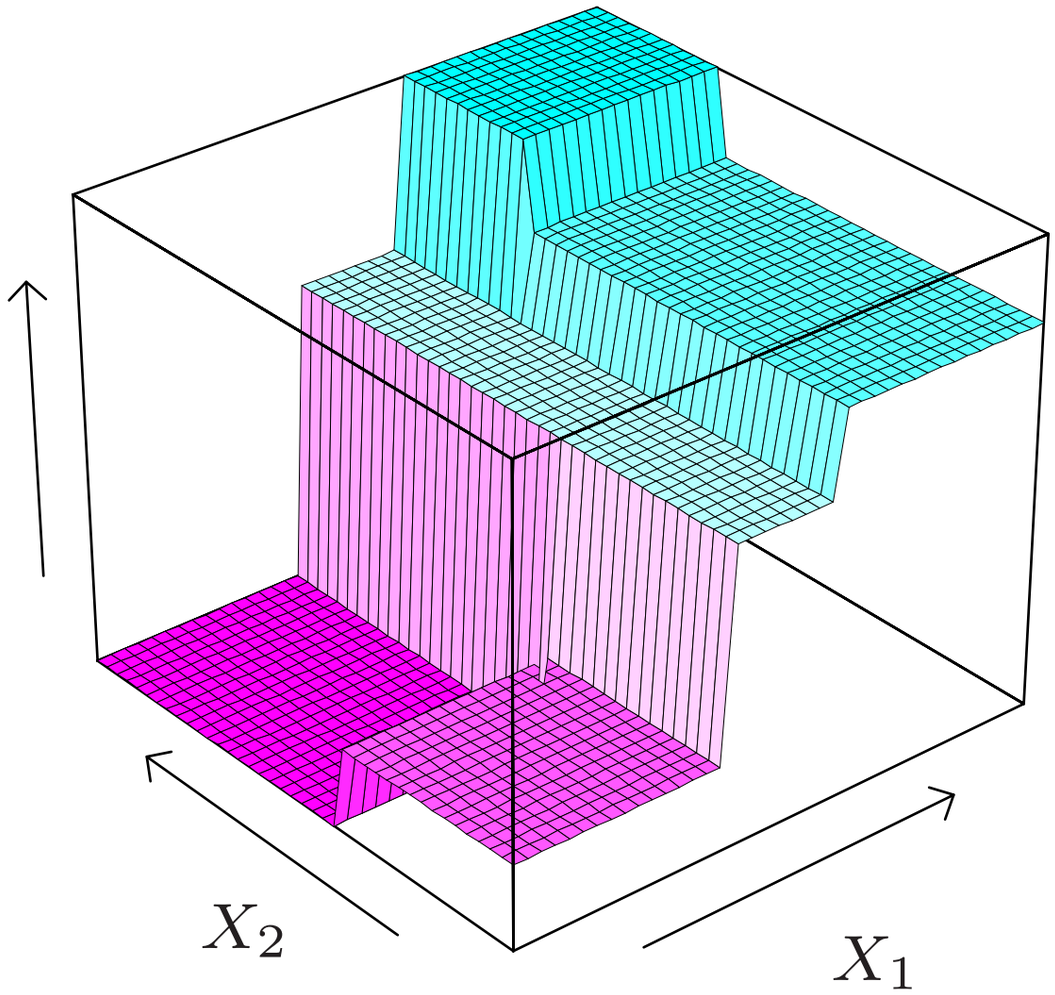
Continuous y and two dimensional feature space

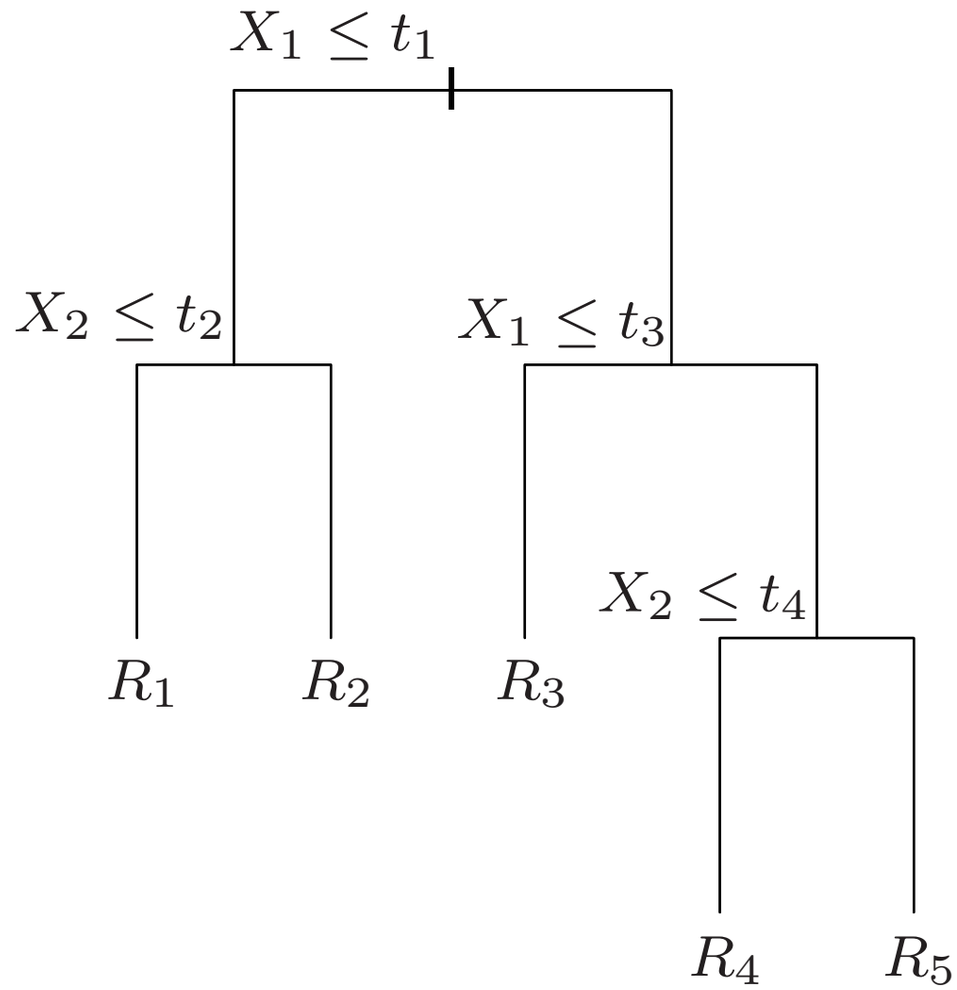




Regression trees

Average values of y on each rectangle are used for the prediction





$\hat{f}^{tree}(x) = \sum_{j=1}^{m} c_j \mathbb{I}(x \in R_j)$, c_j: avg values of y on R_j

Impurity

- Suppose the feature space is partitioned into M regions $\{R_1, R_2,\ldots,R_m\}$ and the response in each region R_j is represented as a constant c_j, then the regression model can be represented as :

$$\hat{f}^{tree}(x) = \sum_{j=1}^{m} c_j \, \mathbb{I}(x \in R_j), \; c_j\text{: avg values of y on } R_j$$

- Node impurity is computed using squared error:

$$\sum_{i \in R_j} (y_i - c_j)^2$$

- Our aim is to reduce the node impurity.

Regression tree building process

- Can we choose a tree model that minimizes the training error (residual sum of square)?

$$RSS = \sum_{j=1}^{M} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2, \hat{y}_{R_j} = c_j : \text{avg of y values on rectangle } R_j$$

- It is computationally infeasible to consider every possible partition of the feature space into M rectangles.

Basic idea of tree building

- For this reason, we take a top-down, greedy approach that is known as recursive binary splitting.

- The approach is top-down because it begins at the top of the tree and then successively splits the feature space

- It is greedy because at each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

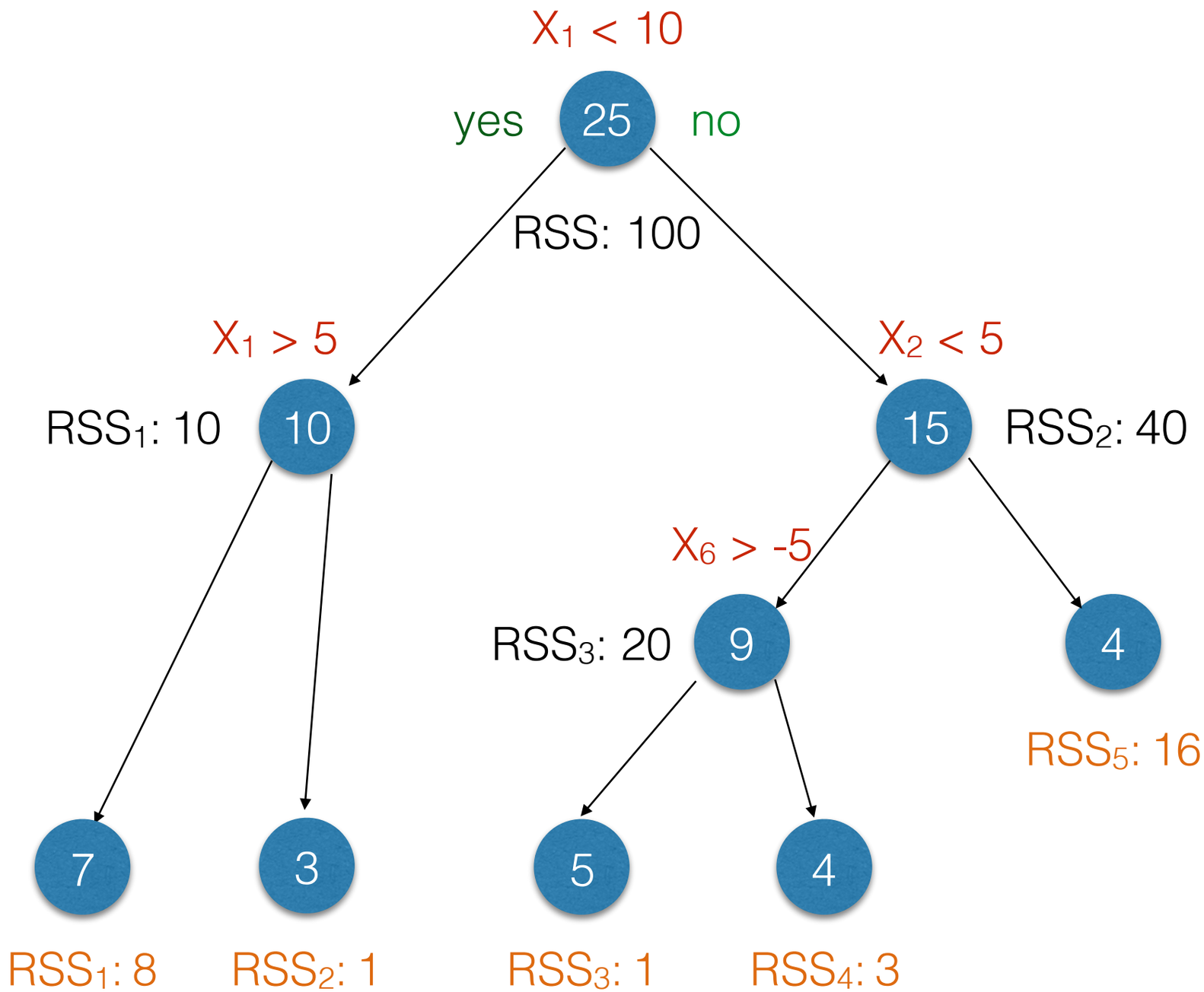
- First, we scan through all predictors (X_j) and cut-point c to find the best binary split $\{X_j < c\}$ that leads to the greatest possible reduction in RSS. Split the feature space into two regions.

- We repeat the process, looking for the best binary split within each of the resulting regions.

- However, this time, instead of splitting the entire feature space, we split one of the two previously identified regions (recursive binary splitting). We now have three regions.

- The process continues until a stopping criterion is reached; for instance, we may continue until no region contains more than five observations.

stopping rule: no splits if nodes have < 6 samples



How to control overfitting?

- The tree building process may produce good predictions on the training set, but is likely to overfit the data, leading to poor test set performance.

- Bias-variance tradeoff: A smaller tree with fewer splits might lead to lower variance and better interpretation at the cost of a little bias.

Grow a small tree?

- One possible alternative to the pruning is to grow the tree only so long as the decrease in the RSS due to each split exceeds some (high) threshold.

- This strategy will result in smaller trees, but is too short-sighted: a seemingly worthless split early on in the tree might be followed by a very good split — that is, a split that leads to a large reduction in RSS later on.

Prune a large tree

- A better strategy is to grow a very large tree T_0, and then prune it back in order to obtain a subtree (smaller tree).

- Weakest link pruning or cost complexity pruning can be used to do this.

Weakest link pruning

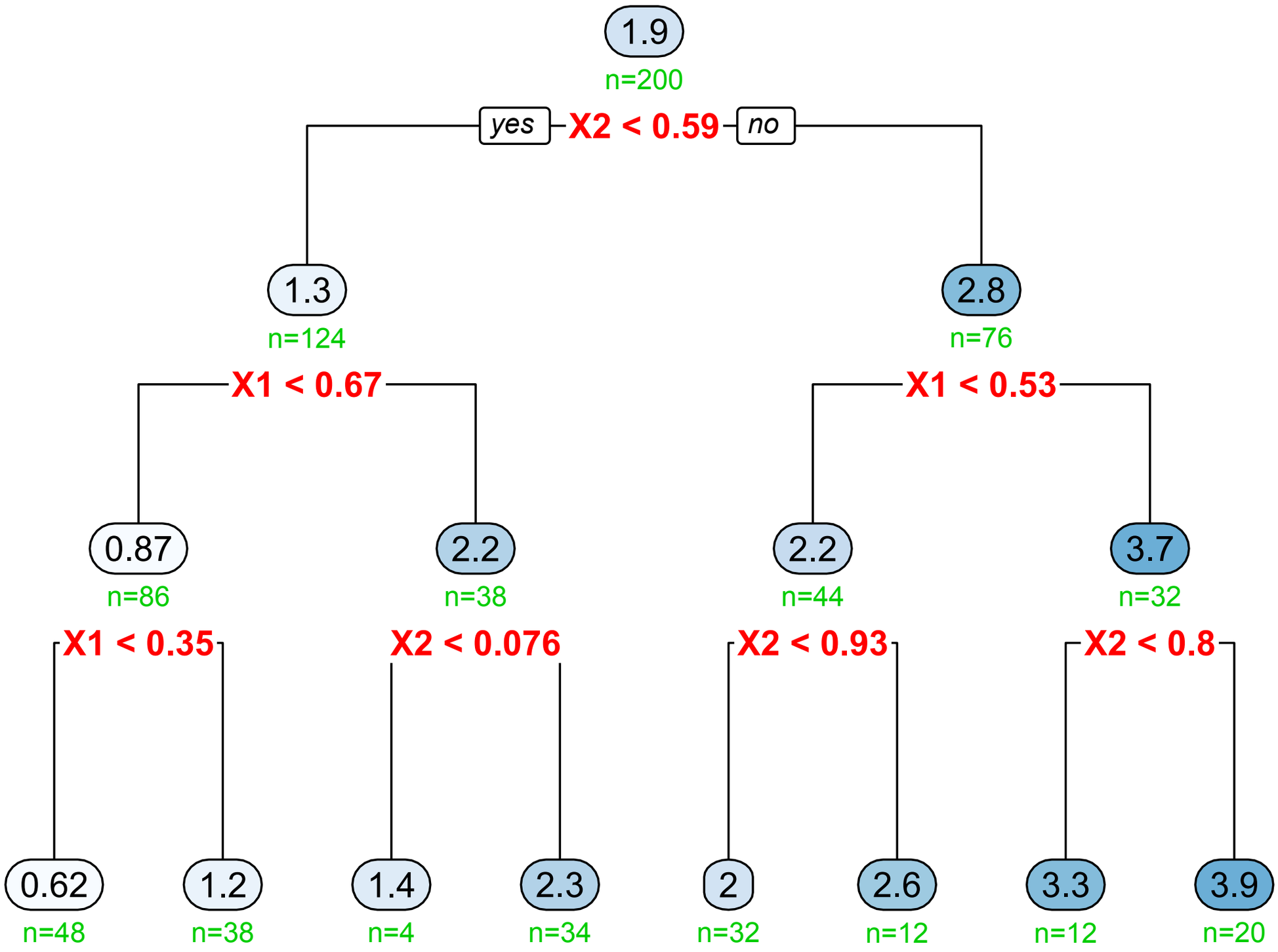
- Starting with with the initial full tree T_0, replace a subtree with a leaf node to obtain a new tree T_1. Select subtree to prune by minimizing:

$$\frac{RSS(T_0) - RSS(T_1)}{|T_0| - |T_1|}$$

- Iterate this pruning to obtain a sequence T_0, T_1, T_2,.., T_m where T_m is the tree with a single leaf node.

- Select the optimal tree T_i by cross validation.

Suppose this is a full tree T_0.
Discuss how the weakest link pruning works.



- Weakest link pruning is similar to the backward stepwise selection in GLM.

- In the backward stepwise selection,

- We iteratively choose a term to drop by looking at R^2.

- Then, select a single best model from among models with different number of predictors, $M_0,\ldots,M_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

Cost complexity pruning

- Focus is to balance RSS against a measure of complexity

- we consider a sequence of trees indexed by a nonnegative tuning parameter α. For each value of α, there exists a subtree $T \subset T_0$ such that

$$\sum_{j=1}^{|T|} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 + \alpha |T|$$

loss penalty

is as small as possible. |T| indicates the number of terminal nodes of the tree T.

$$\sum_{j=1}^{|T|} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 + \alpha |T|$$

penalty

loss

- Let T_α be the pruned tree minimizing the above.
- α (complexity parameter) determines the trade-off between the overall RSS and the model complexity
  - when α is small, the penalty for having a larger tree is small so T_α is large.
  - when α increases, $|T_\alpha|$ decreases

Summary: tree algorithm

- 1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.

- 2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.

- 3. Use K-fold cross-validation to estimate the mean squared prediction error, and choose α using the one standard error rule.

- 4. Return the subtree from Step 2 that corresponds to the chosen value of α.

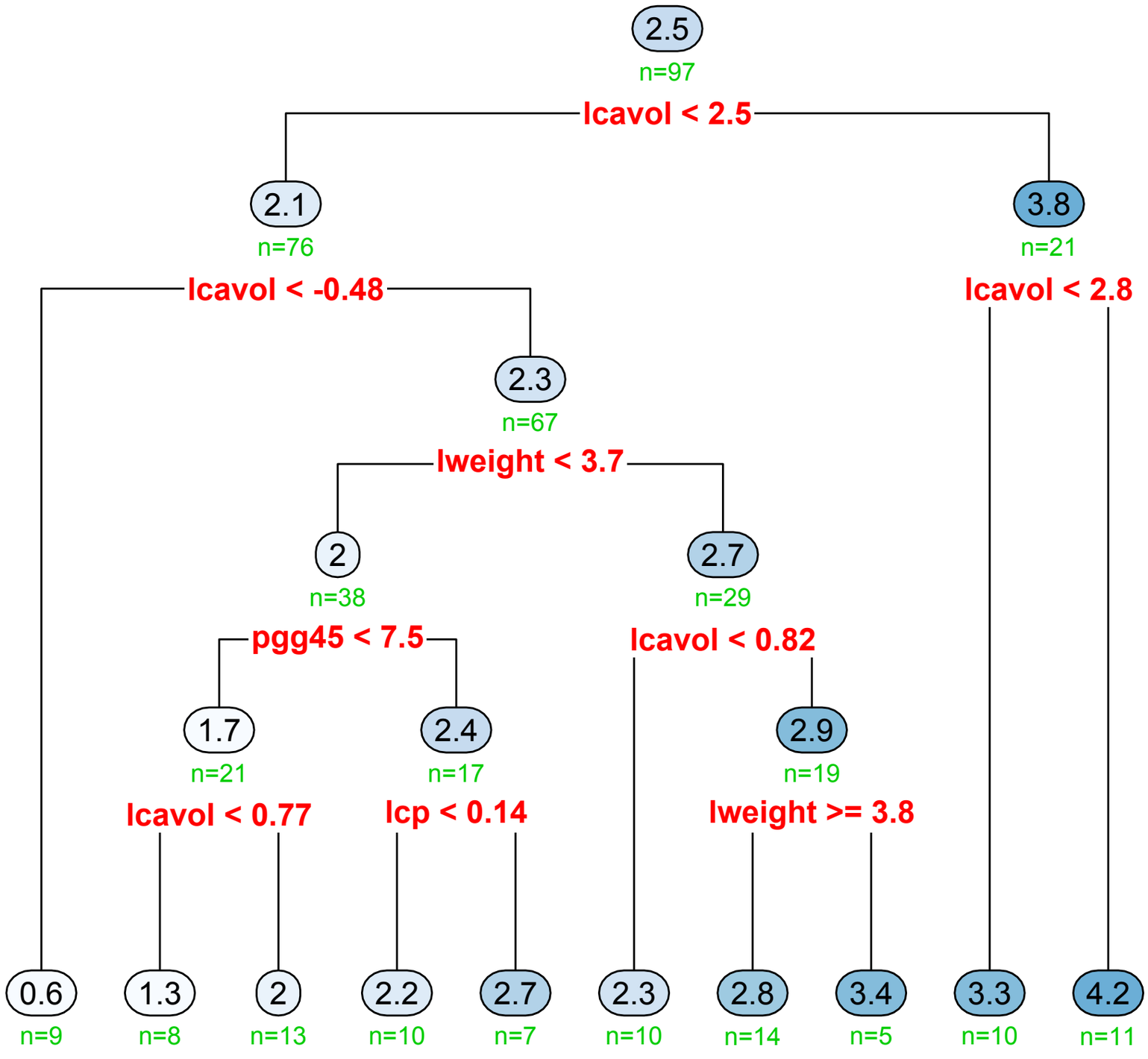
Example: Prostate cancer

- The goal is to predict the log of PSA (lpsa)
- Predictors
  - log cancer volume (lcavol), log prostate weight (lweight), age, log of benign prostatic hyperplasia amount (lbph), seminal vesicle invasion (svi), log of capsular penetration (lcp), Gleason score (gleason), and percent of Gleason scores 4 or 5 (pgg45).

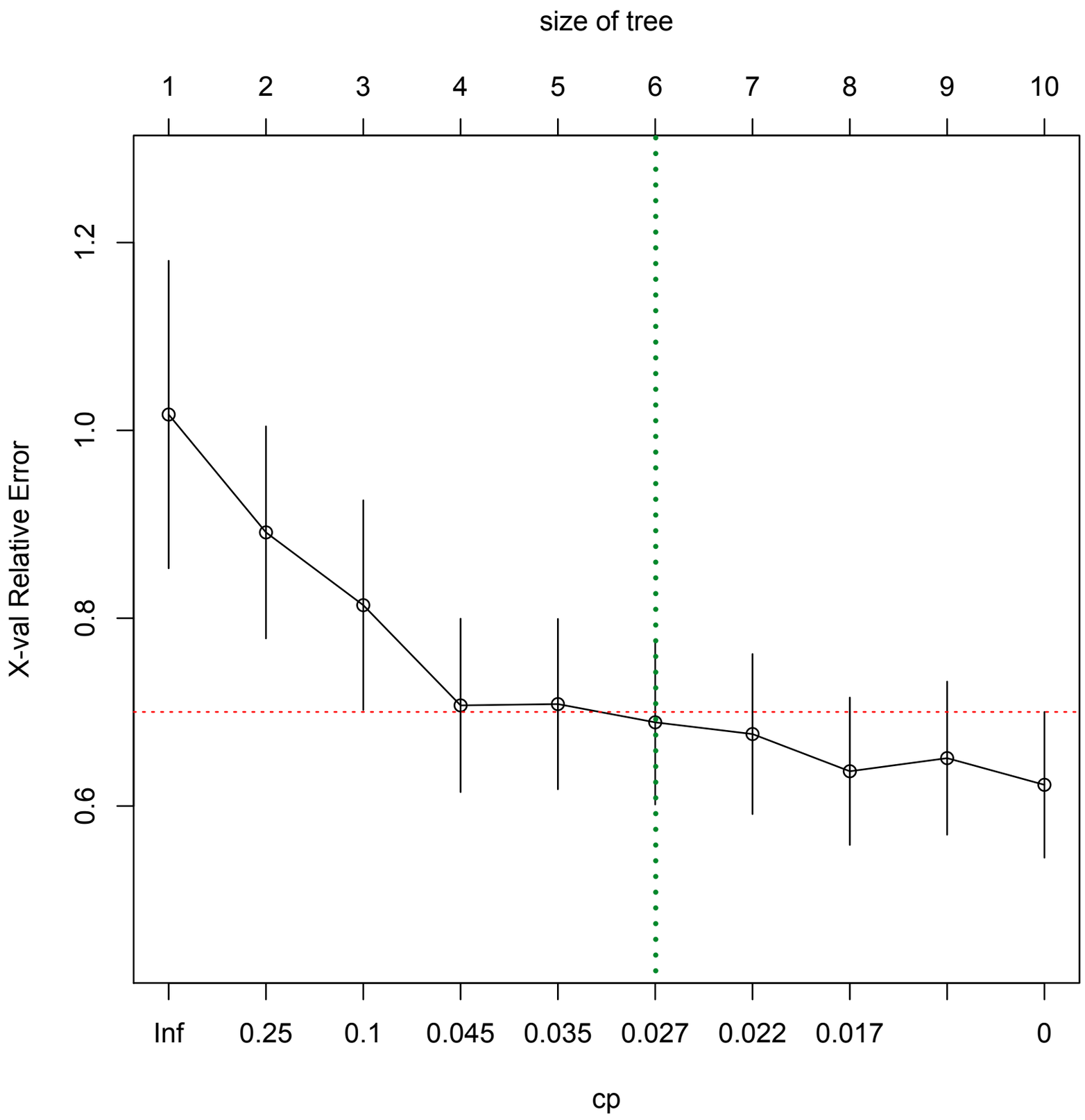
1. We grow a large tree

Stopping rule:

All leaves should have at least 5 samples



2. Perform the CV to determine α(=cp).

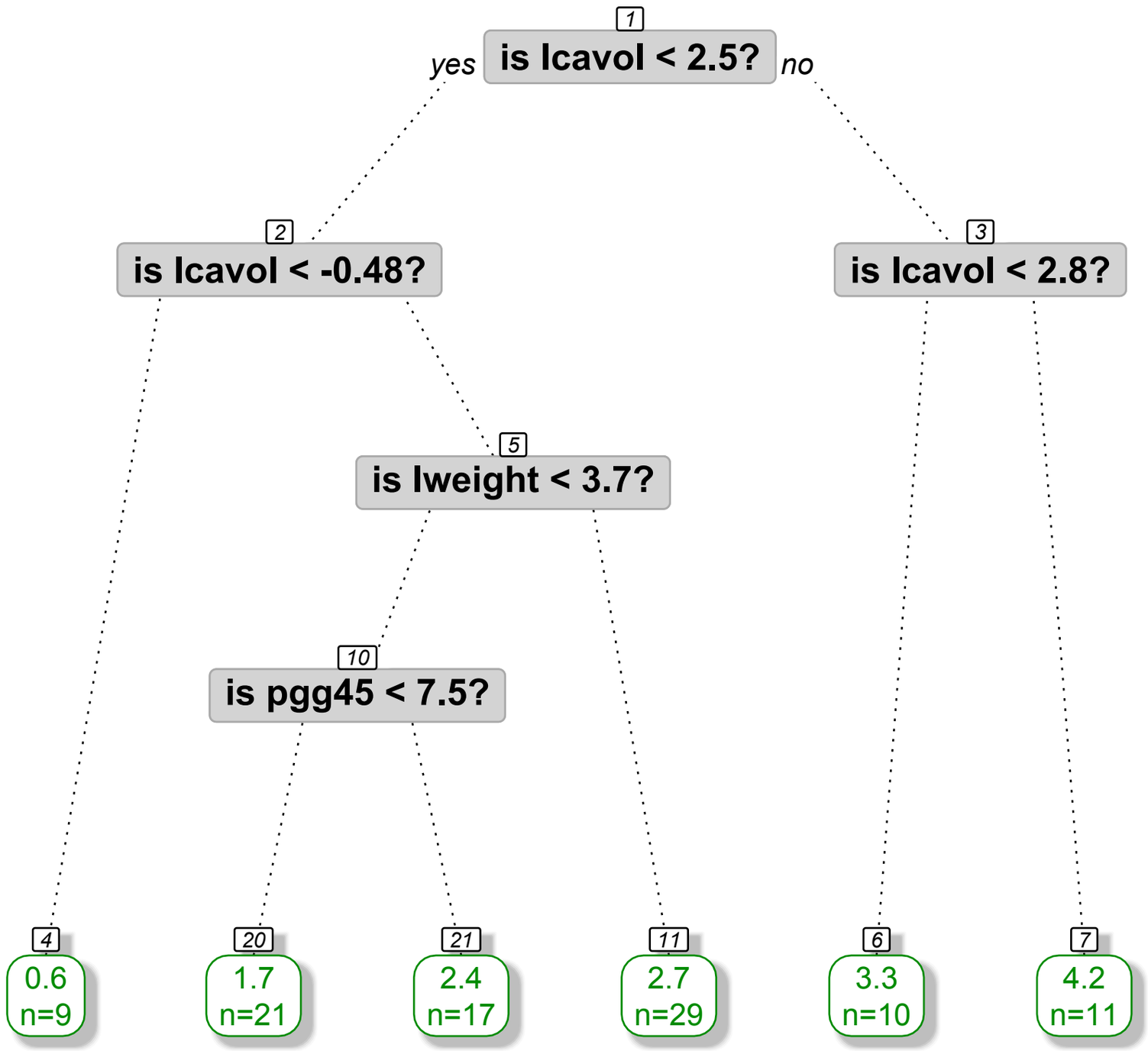


cp table for fitted tree model

	CP	nsplit	rel error	xerror	xstd
1	0.3471083	0	1	1.01687	0.163742
2	0.1846474	1	0.65289	0.89137	0.112926
3	0.0593158	2	0.46824	0.81387	0.11169
4	0.0347563	3	0.40893	0.70716	0.092324
5	0.034609	4	0.37417	0.70854	0.090713
6	0.0215637	5	0.33956	0.68911	0.087449
7	0.0214699	6	0.318	0.67664	0.08522
8	0.0131943	7	0.29653	0.63702	0.078493
9	0.0072435	8	0.28334	0.65098	0.081525
10	0	9	0.27609	0.62259	0.07761

- We choose cp = 0.027
- In fact, any cp value between 0.034609 and 0.0215637 will result in the same tree.
  - These two values can be found in the cp table.
- What is the size of chosen tree (# of leaves)?
- How many splits do you need to have the chosen tree?

Pruned tree

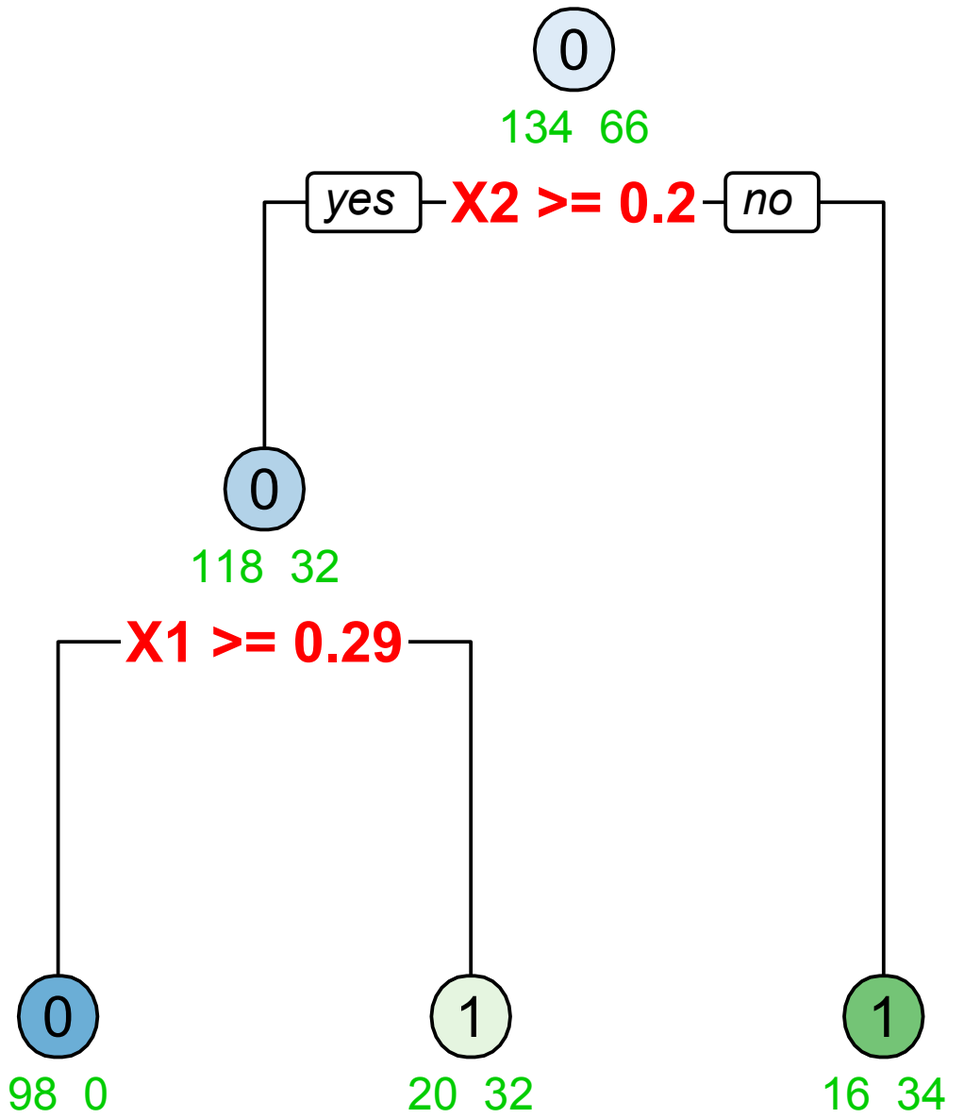

1
is lcavol < 2.5?
yes
no
2
is lcavol < -0.48?
3
is lcavol < 2.8?
5
is lweight < 3.7?
10
is pgg45 < 7.5?
4
0.6
n=9
20
1.7
n=21
21
2.4
n=17
11
2.7
n=29
6
3.3
n=10
7
4.2
n=11

Classification tree

- Very similar to a regression tree, except that it is used to predict a qualitative response rather than quantitative.

- Key differences:
  - The outcome variables are categorical (class)
  - The calculation of predicted value: it is done by majority class in a node
  - The criteria for splitting and pruning: don’t know yet, but it’s not mean squared error

Prediction

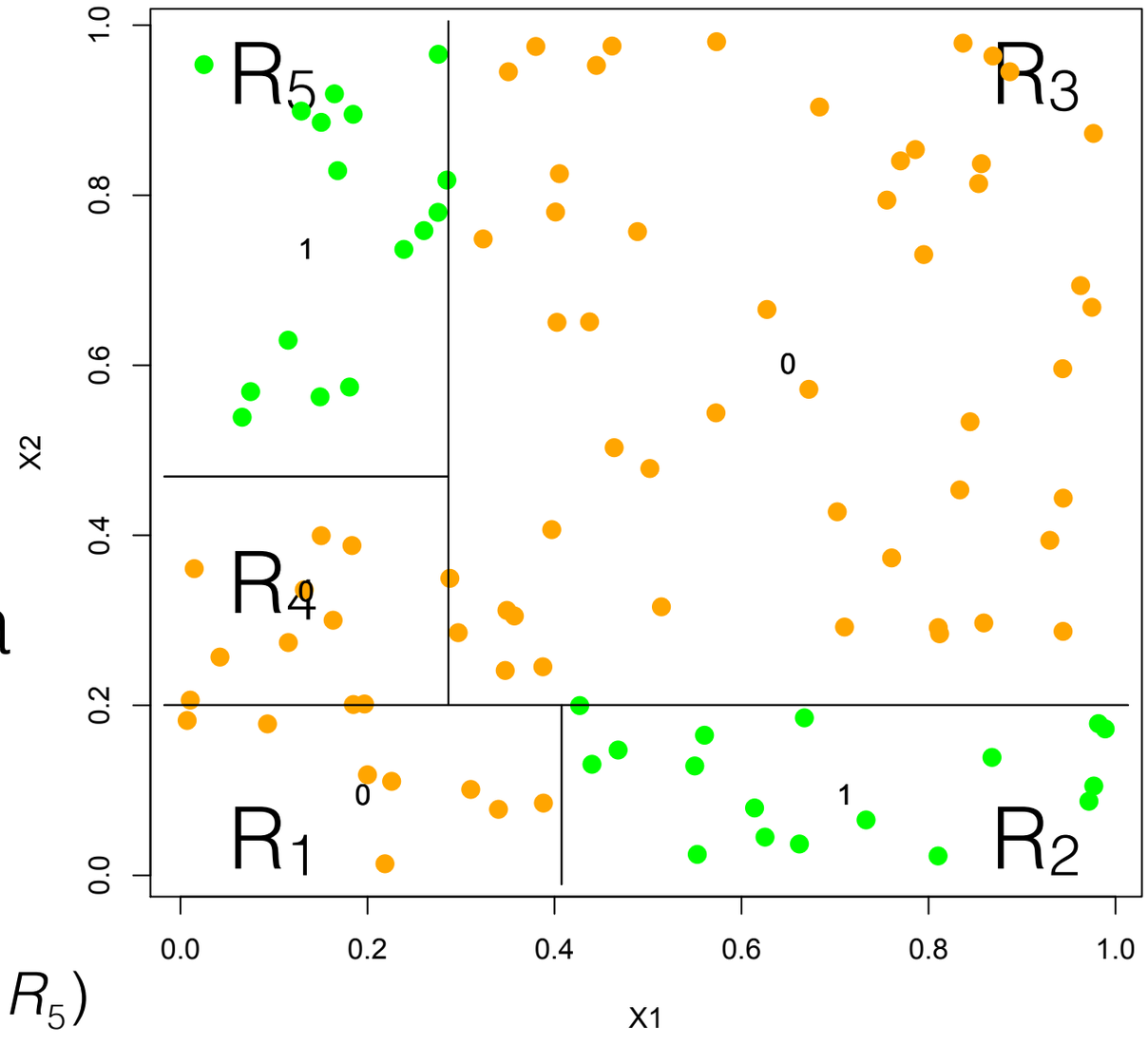


Classification tree

- Indicator function I(·)
  - I(statement) = 1 if the statement is true; otherwise 0.
- Using the I(·), we classify a point x by

$$\hat{f}^{tree}(x) = \sum_{j=1}^{m} c_j \, \mathrm{I}(x \in R_j)$$
$$= 0 \cdot \mathrm{I}(x \in R_1) + 1 \cdot \mathrm{I}(x \in R_2) + \ldots + 1 \cdot \mathrm{I}(x \in R_5)$$

- $c_j = \mathrm{argmax}_k \, \hat{p}_k(R_j), \; \hat{p}_k(R_j) = \frac{1}{N_j} \#\{y_i = k \text{ for i} \in R_j\}$



Classification rule

- Our aim is to reduce the node impurity (misclassification cost). i.e., make all the samples in a node belongs to one class

- This can be seen as reducing the node impurity

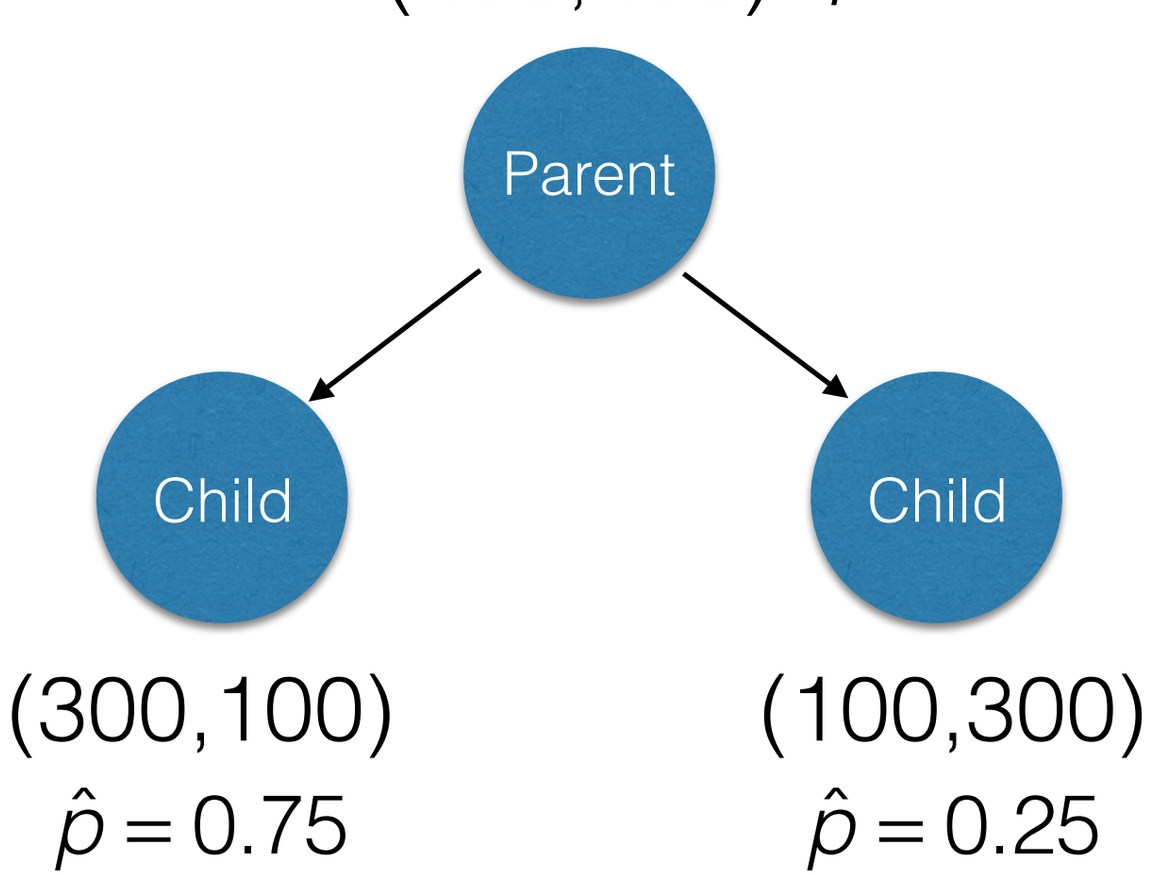
  - Gini index: $\sum_{i=1}^{K} \hat{p}_k (1 - \hat{p}_k)$

  - Cross-entropy (deviance): $-\sum_{i=1}^{K} \hat{p}_k \log \hat{p}_k$

  - Misclassification error: $1 - \max_k \hat{p}_k$

Three measures: binary classification

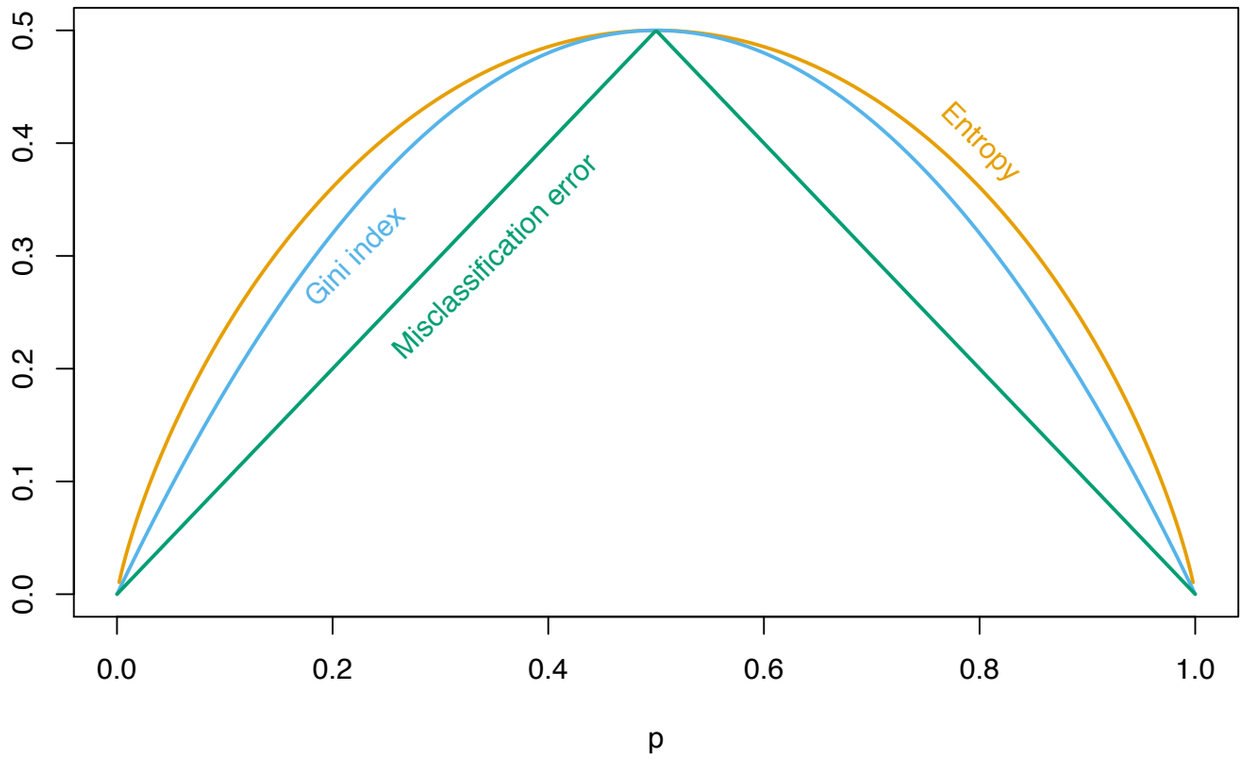
- The binary split that reduces the largest amount of the node impurity is chosen.



Gini index: 2·p(1-p)

Deviance: -p·log(p)-(1-p)·log(1-p)

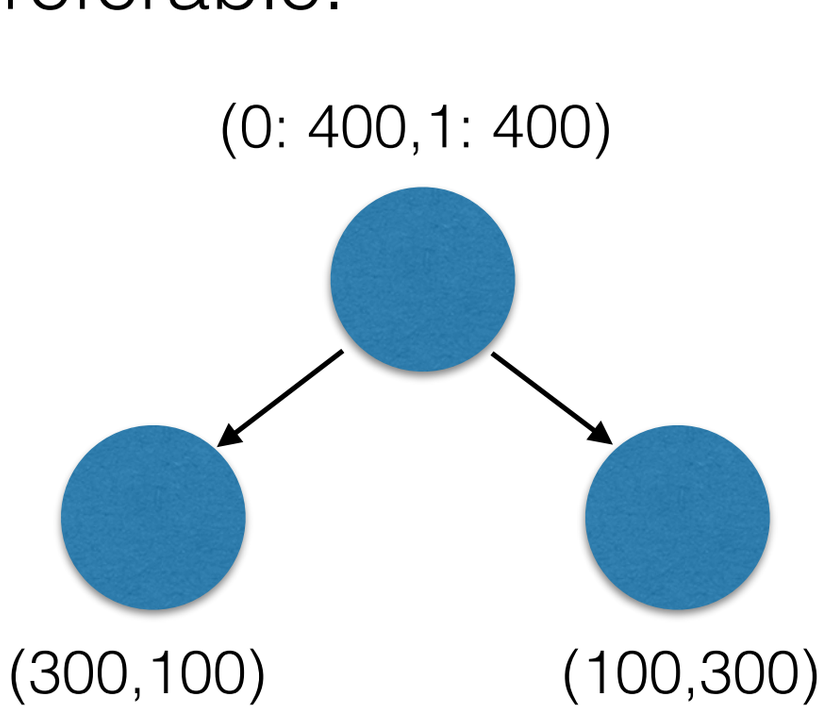
Miss. error: 1-max(p,1-p)

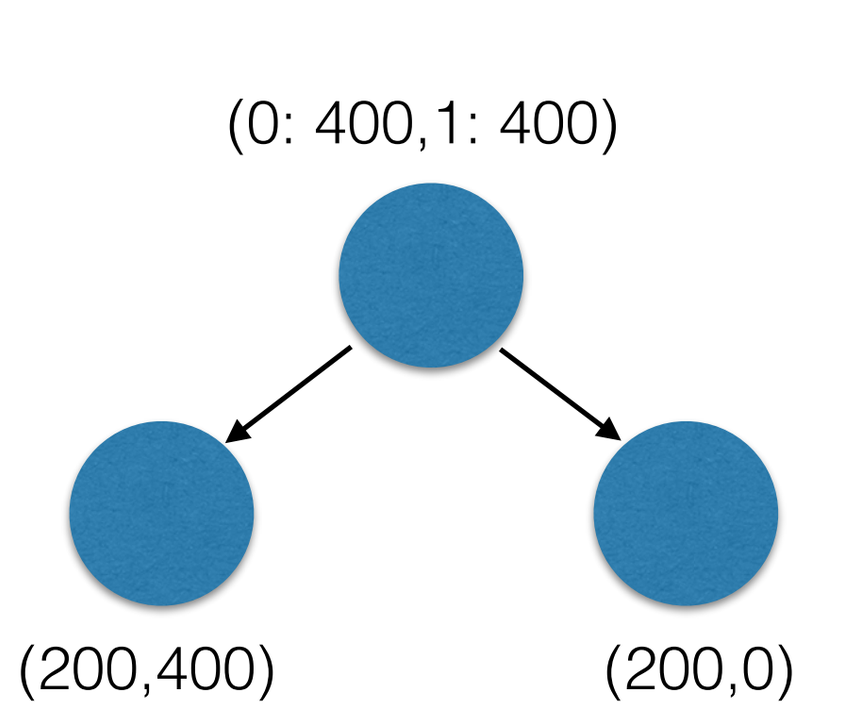


- Encourages the formation of regions in which high proportion of data points assigns to one class

- Gini index and cross entropy are differentiable and more sensitive to change in node proportions

- Which impurity measure is preferred for the recursive binary splitting to grow a classification tree?
  - Misclassification error is not sufficiently sensitive for tree-growing, and in practice two other measures are preferable.
  -





Compute weight averages of node impurities (Gini and misclassification) for two splits

Cost complexity pruning

- For the pruning, any of the three node impurity measures can be used.
  - Often, we choose the misclassification error for the pruning.
- For each value α, we seek a subtree $T \subset T_0$ minimizes

$$\sum_{j=1}^{|T|} \frac{N_j}{N}[1-\max\{\hat{p}(R_j),1-\hat{p}(R_j)\}]+\alpha|T|$$

loss penalty

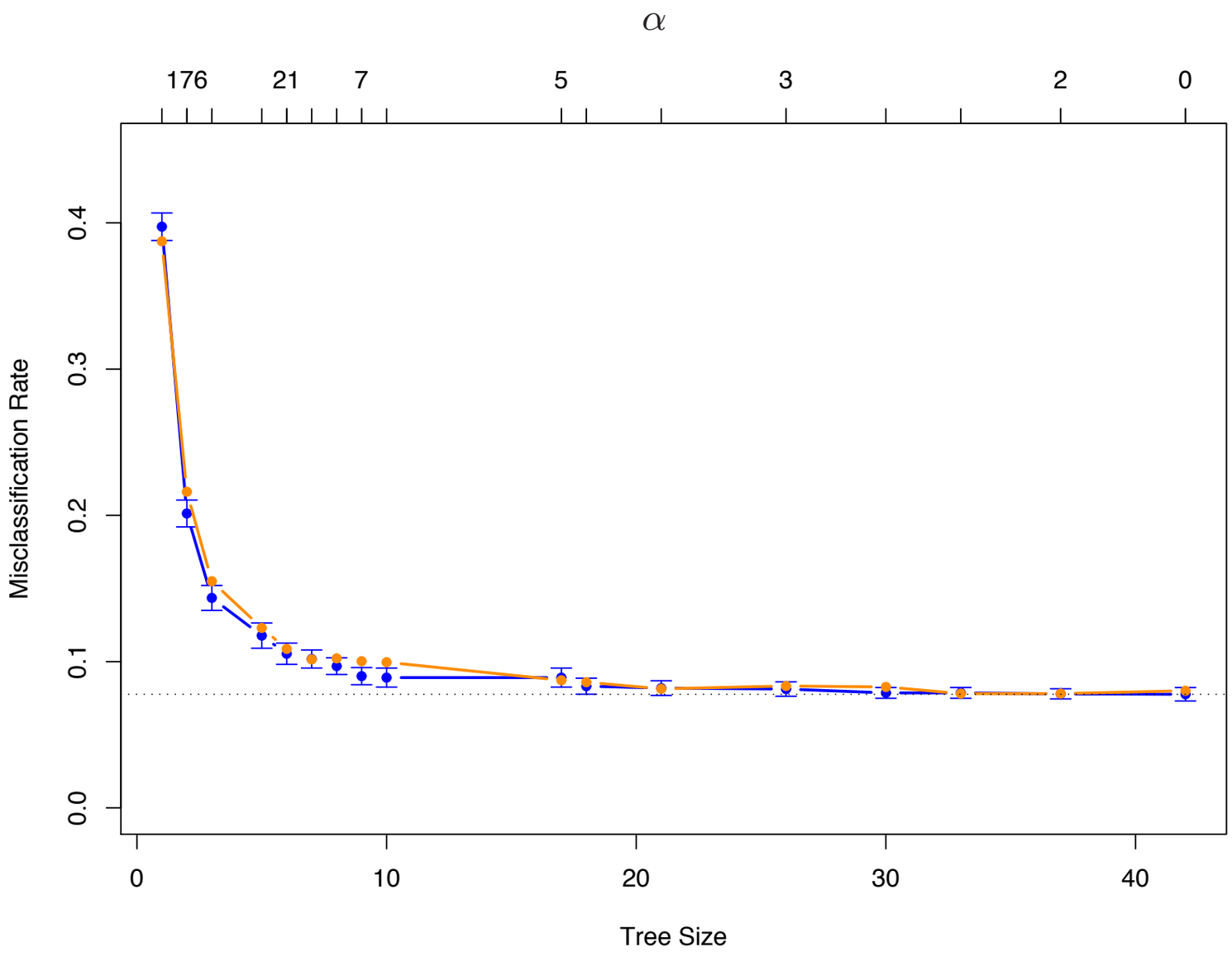
- The cross validation is used to choose the tuning parameter α.

Spam data

- For each email we have p = 58 attributes.
  - The first 54 measure the frequencies of 54 key words or characters (e.g., “free”, “need”, “$”).
  - The last 3 measure the average length of uninterrupted sequences of capitals; the length of the longest uninterrupted sequence of capitals; the sum of lengths of uninterrupted sequences of capitals.
- How would we possibly get thousands of emails labeled as spam or not?

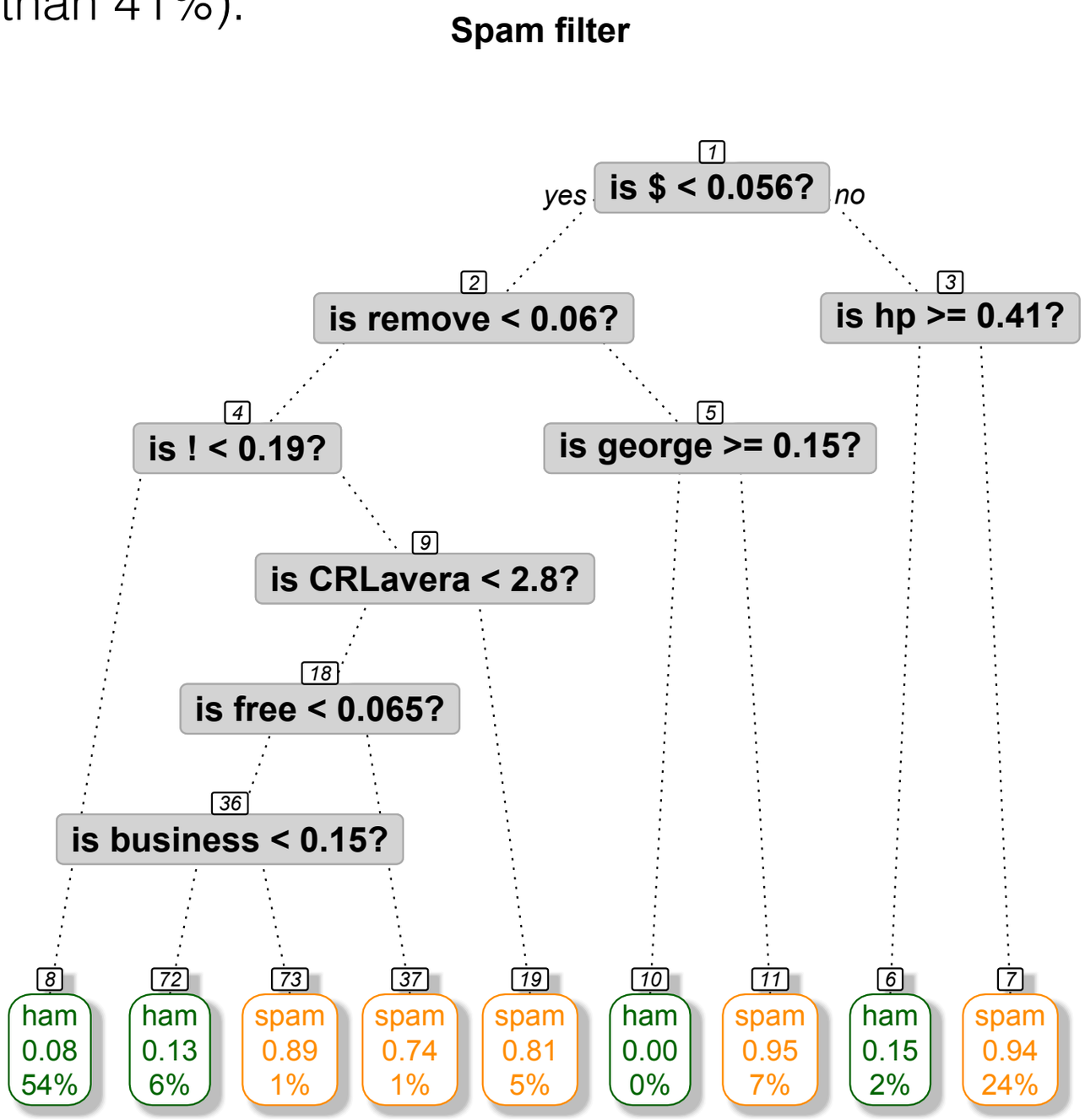
- We use the deviance to grow the tree and misclassification error to prune it.

- CV error curve and test error curve are shown below. (α is chosen by the one standard error rule).



We branch to the right with a spam warning if more than 5.6% of the characters are the $ sign, unless the phrase hp occurs frequently (more than 41%).

CRL = capital run length



Sensitivity vs Specificity

- Sensitivity (true positive rate): probability of predicting disease (spam) given true state is disease (spam).
- Specificity (true negative rate): probability of predicting non-disease (email) given true state is non-disease (email).
-

	Predicted	
True	email	spam
email	57.3%	4.0%
spam	5.3%	33.4%

$$\textit{Sensitivity} = 100 \times \frac{33.4}{33.4 + 5.3} = 86.3\%,$$

$$\textit{Specificity} = 100 \times \frac{57.3}{57.3 + 4.0} = 93.4\%.$$

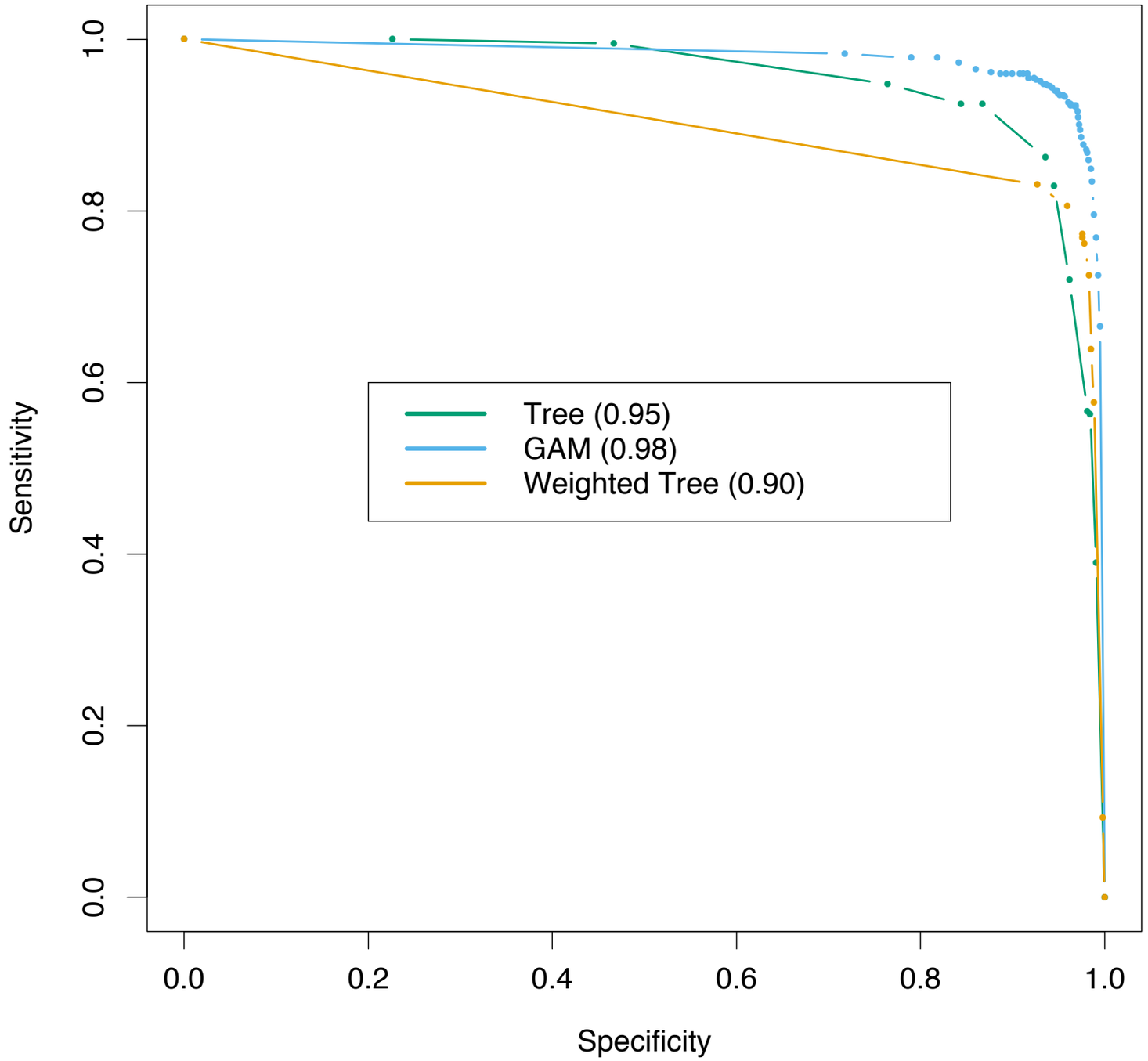
Receiver operating characteristic curve (ROC curve)

- Each terminal node classifies to spam if the proportion of spam is ≥ 0.5.

- We want to avoid marking good email as spam, and thus we want the specificity to be very high.

- Using a value less than 0.5 increases the sensitivity and decreases the specificity.

- ROC curve is a commonly used summary for assessing the tradeoff between sensitivity and specificity.



- We see that in order to achieve a specificity of close to 100%, the sensitivity has to drop to about 50%.
- The area under the curve is a commonly used quantitative summary.

CART and missing values

- 1. New categorical variable for “missing.”
  - We might discover that observations with missing values behave differently than those with nonmissing values.
- 2. Surrogate variables (=rpart= package have this option).
  - We form a list of surrogate splits. The i-th surrogate is the i-th best binary split that mimics the split of the training data by the primary split.
  - We use the surrogate splits in order, if the primary splitting predictor is missing.
  - Surrogate splits exploit correlations between predictors to try and alleviate the effect of missing data.

Summary: advantage of CART

- CART makes no distribution assumptions on the variables and supports both categorical and continuous variables

- Can be used for ranking the variables (by summing up the impurity measures across all nodes in the tree)

- Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!

- Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous lectures.

- Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).

Disadvantage of CART

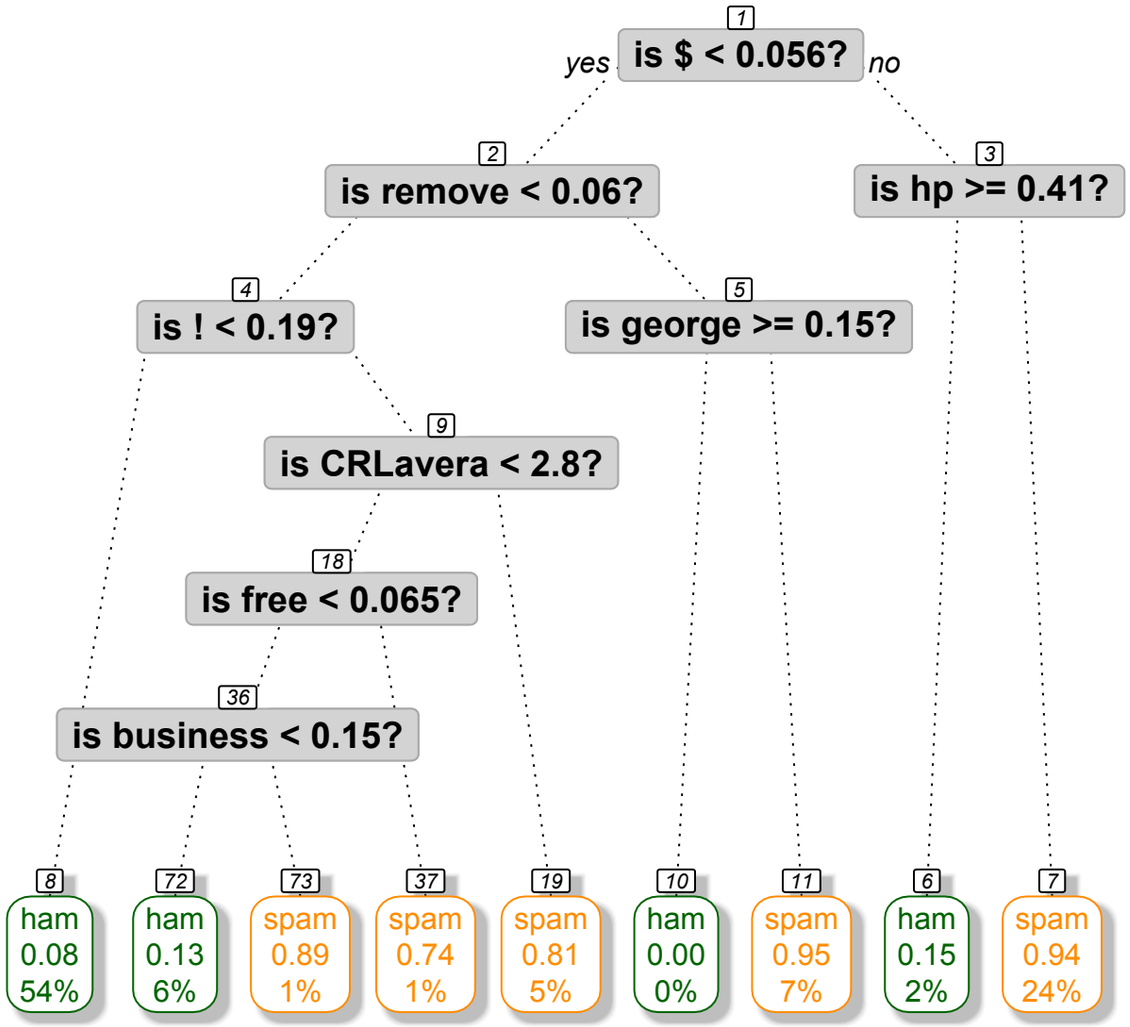
- Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches.

- Trees generally suffer from high variance because they are quite unstable: a smaller change in the observed data can lead to a dramatically different sequence of splits, and hence a different prediction.

- However, by aggregating many decision trees, the predictive performance of trees can be substantially improved. We revisit these concepts later.

Sensitivity to the training set

Training set 1



Training set 2