Decision Trees and Random Forests

Yudong Chen
School of ORIE, Cornell University
ORIE 4740 Lec 20-21

Recap: Linear vs. Nonlinear

Linear techniques:

- Linear and logistic regression
- k-means; PCA

Simple extensions of linear techniques:

- Adding high-order and interaction terms
- Converting to dummy variables

Nonlinear techniques:

- KNN
- Basis functions, splines and GAM

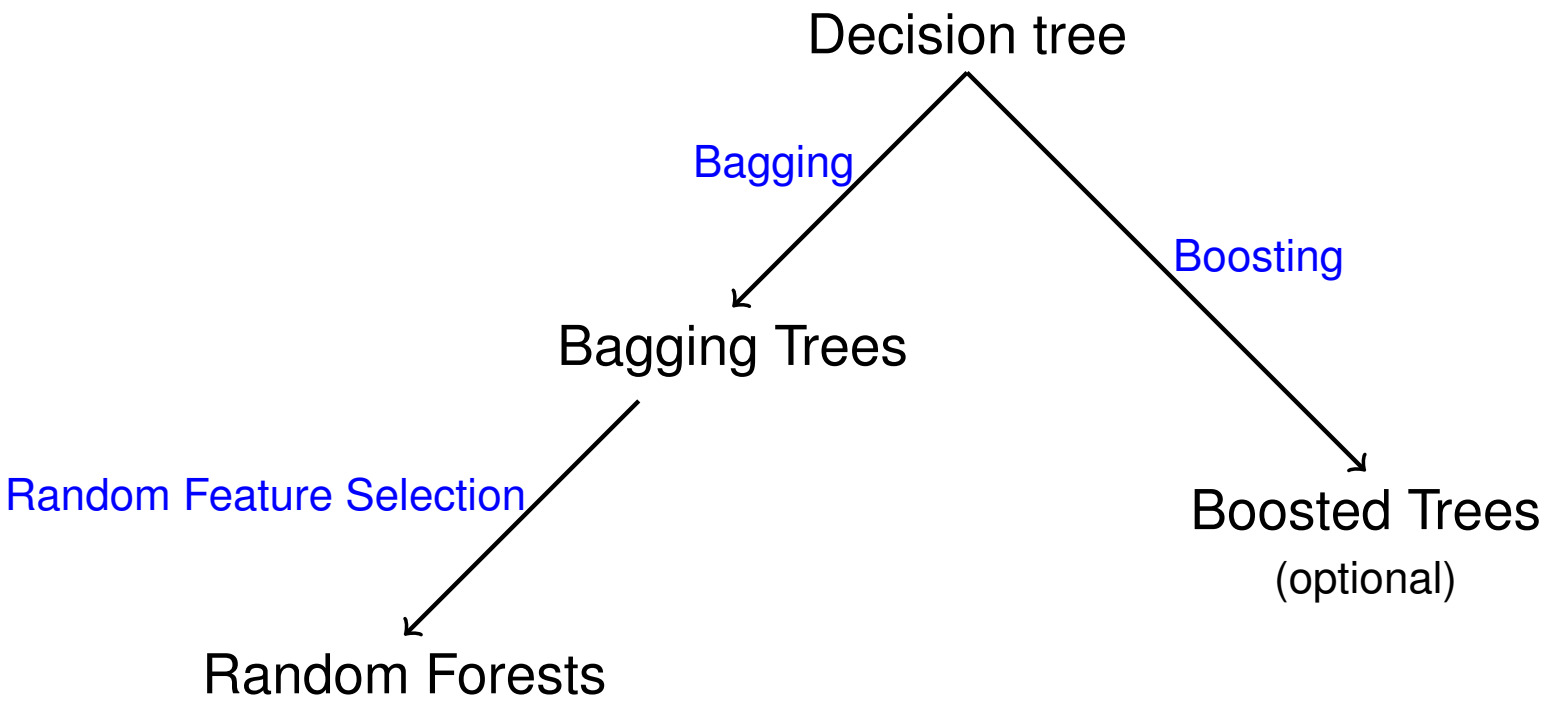
Next:

- Decision Trees & Random Forests

Key issues:

- What determines model flexibility (#variables, λ, k, DoF)
- How to choose it (by CV, p-values, R^2, ANOVA.....)

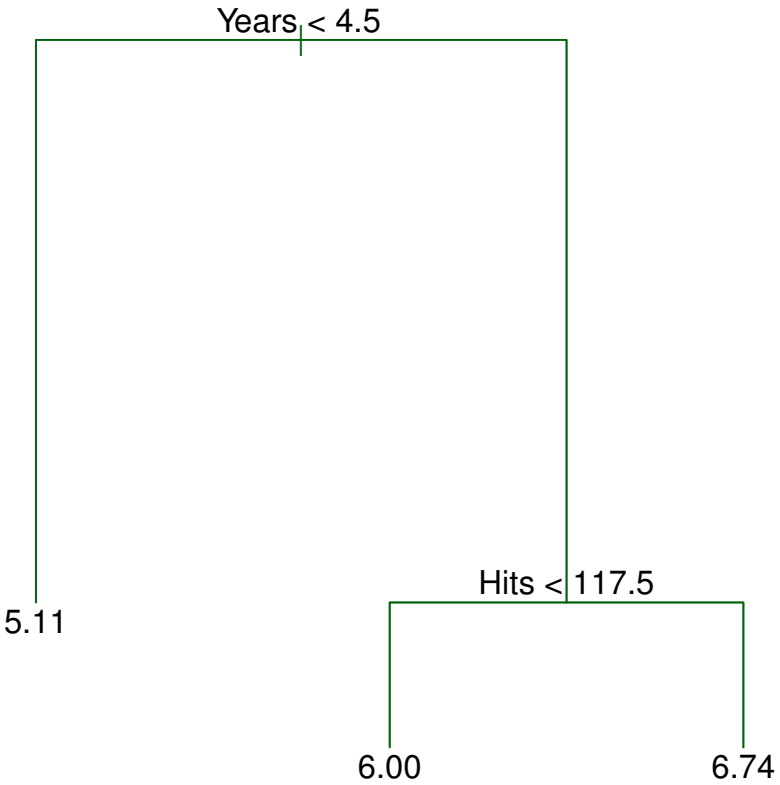
Tree-based Methods

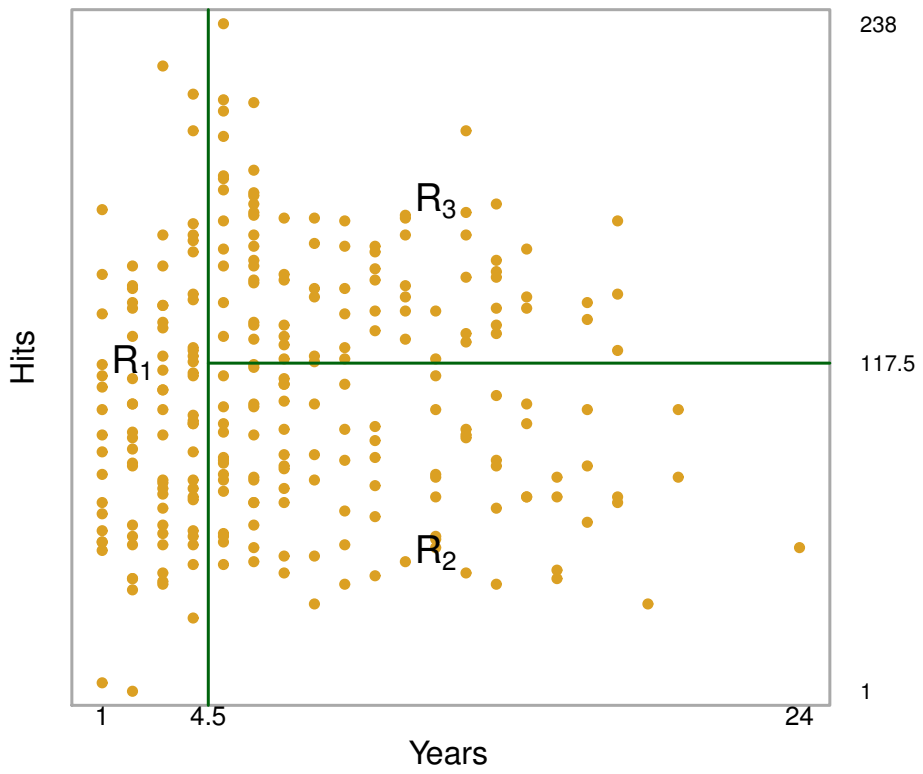


- Popular for classification, but works for regression as well
- The model is easy to explain, but the fitting procedure is harder

Regression Trees (ISLR 8.1.1)

- Decision trees for regression
- Leaf, internal node, branch, split

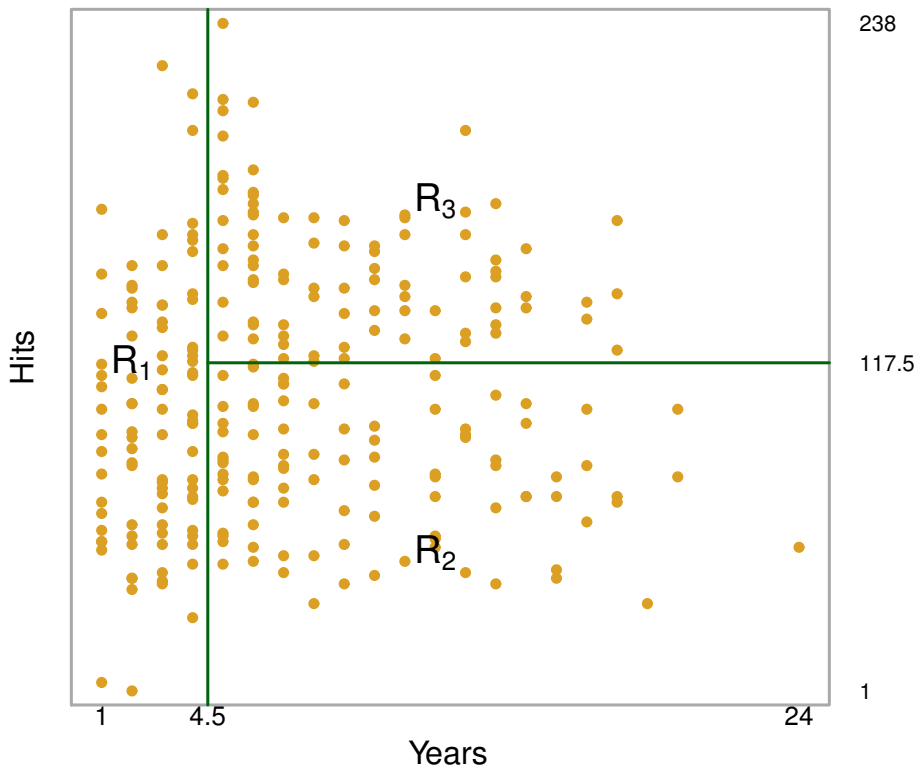




Regression Tree: Prediction

- Predictor space partitioned into J non-overlapping regions $R_1, R_2, \ldots, R_J$
- If an observation $(x_1, \ldots, x_p)$ falls into R_j:

$$\hat{y} = \hat{y}_{R_j} \triangleq \text{mean of } y_i\text{'s in of observations in } R_j$$

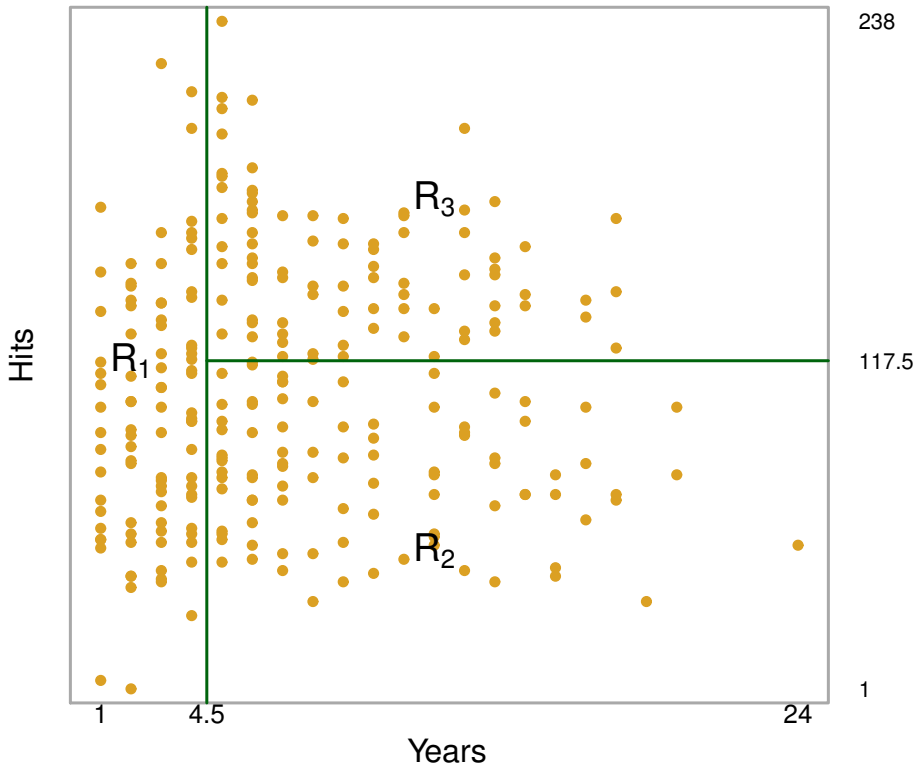


Regression Tree: Fitting

- $R_1, R_2, \ldots, R_J$ are rectangles/boxes
- Ideally, want to minimize

$$\text{RSS} = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

- Instead, we fit a tree greedily
  - First grow a large tree
  - Then prune the tree



Growing a Regression Tree

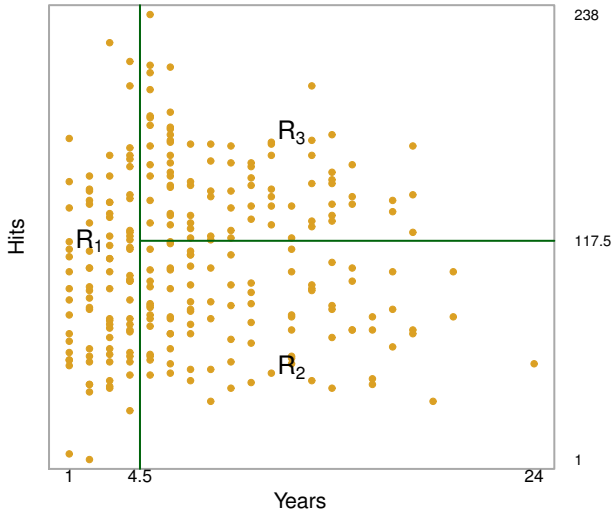
Growing: split the predictor space greedily

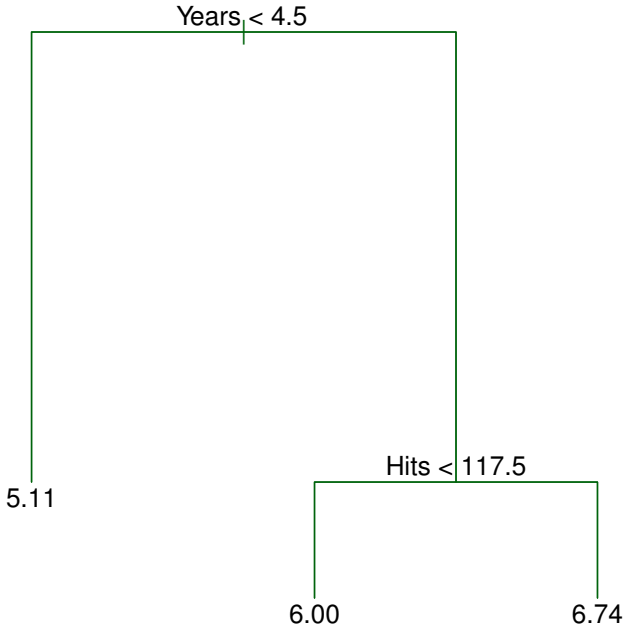
Recursive binary splitting:

1. Find a predictor X_j and a cutpoint s that gives the best RSS by splitting the predictor space into

$$\{X|X_j < s\} \text{ and } \{X|X_j \geq s\}$$

2. Repeat: find the best region to split, and the best way to split it





Pruning a Regression Tree

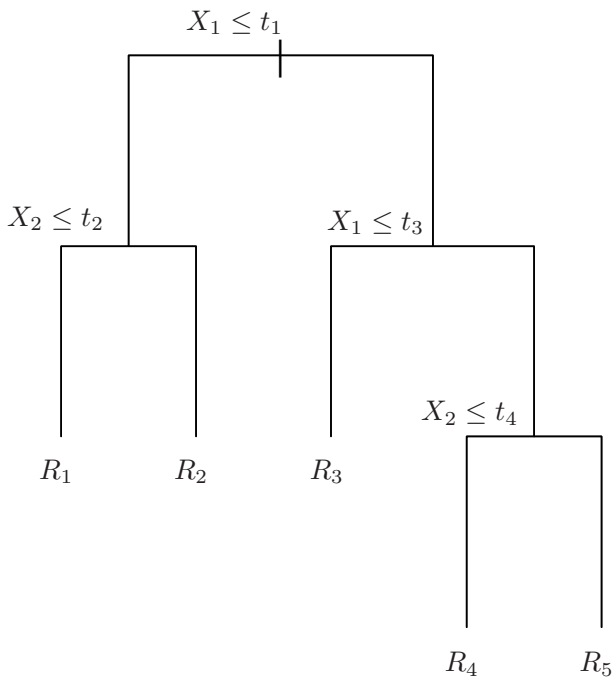
Pruning: Find a subtree that minimizes the CV error

"Cost complexity pruning":

1. For each α, find the subtree T that minimizes

$$\text{RSS}(T) + \alpha|T|$$

   where $|T| = \#\text{leaves} = \#\text{regions}$
2. As α increases, branches are cut off sequentially
3. Find the best α by CV

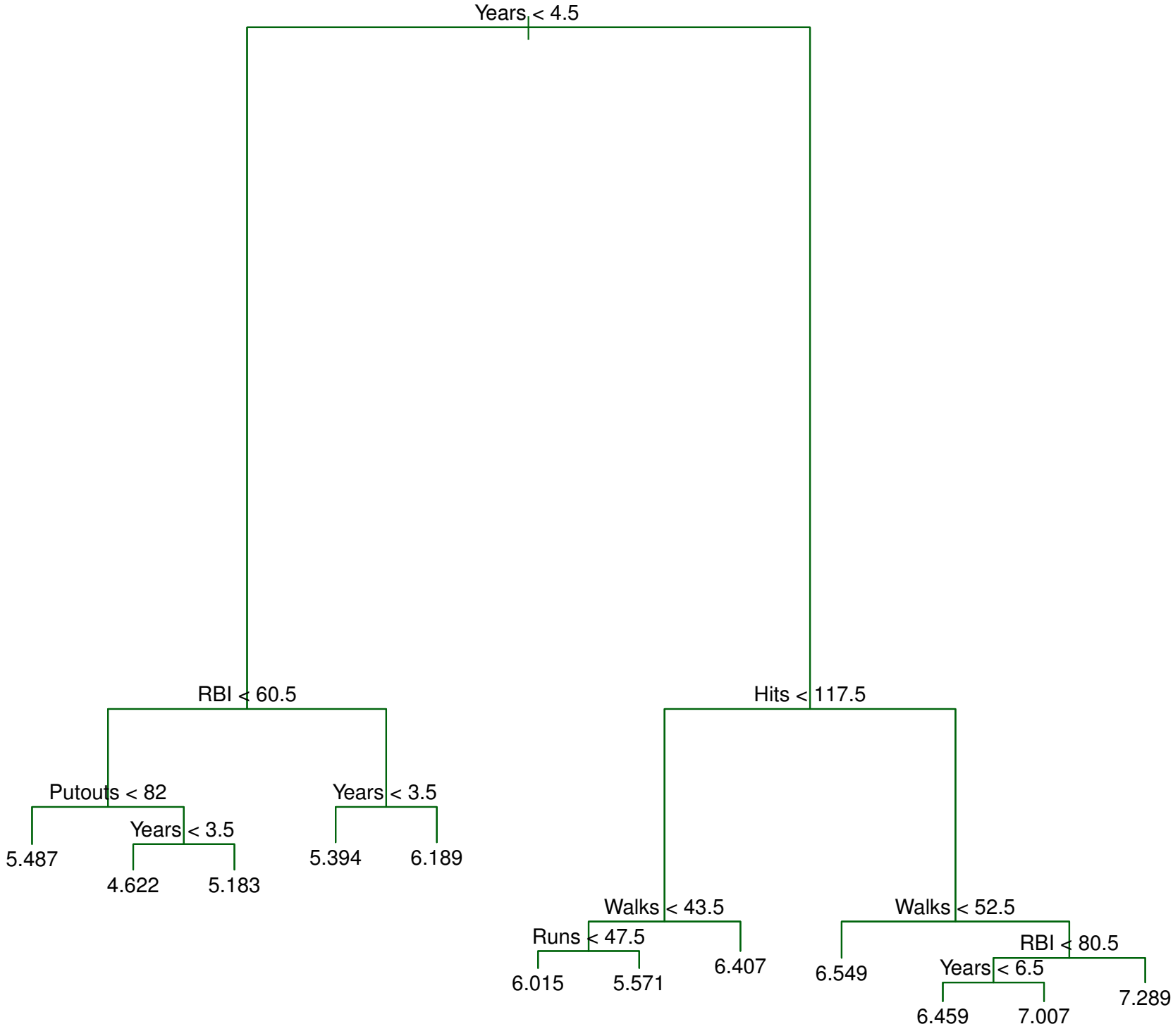


Regression Tree Fitting Algorithm

1. Grow a large tree by recursive binary splitting
   Stop when a leaf/region has too few observations

2. For each value of α, obtain a subtree by cost complexity pruning

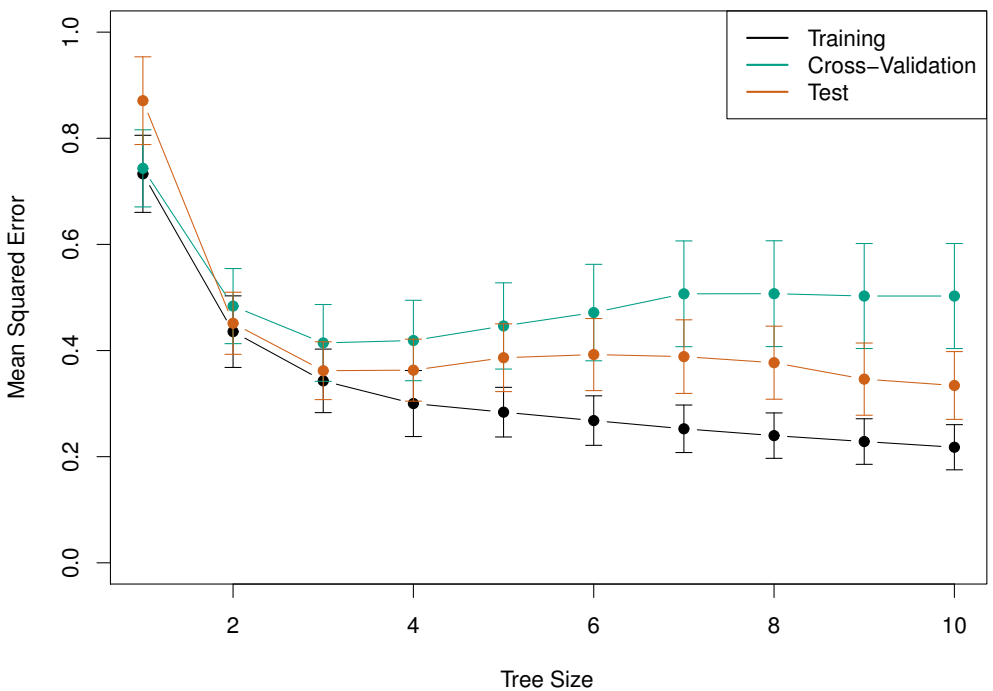
3. Compute the CV error of the subtree corresponding to each α.
   Pick and output the best subtree


Years < 4.5
RBI < 60.5
Hits < 117.5
Putouts < 82
Years < 3.5
Years < 3.5
Walks < 43.5
Walks < 52.5
Runs < 47.5
RBI < 80.5
Years < 6.5
5.487
4.622
5.183
5.394
6.189
6.015
5.571
6.407
6.549
6.459
7.007
7.289

Example: Hitters dataset

Response: Salary
Predictors: Year, Hits, RBI, Putouts, Walks, Runs

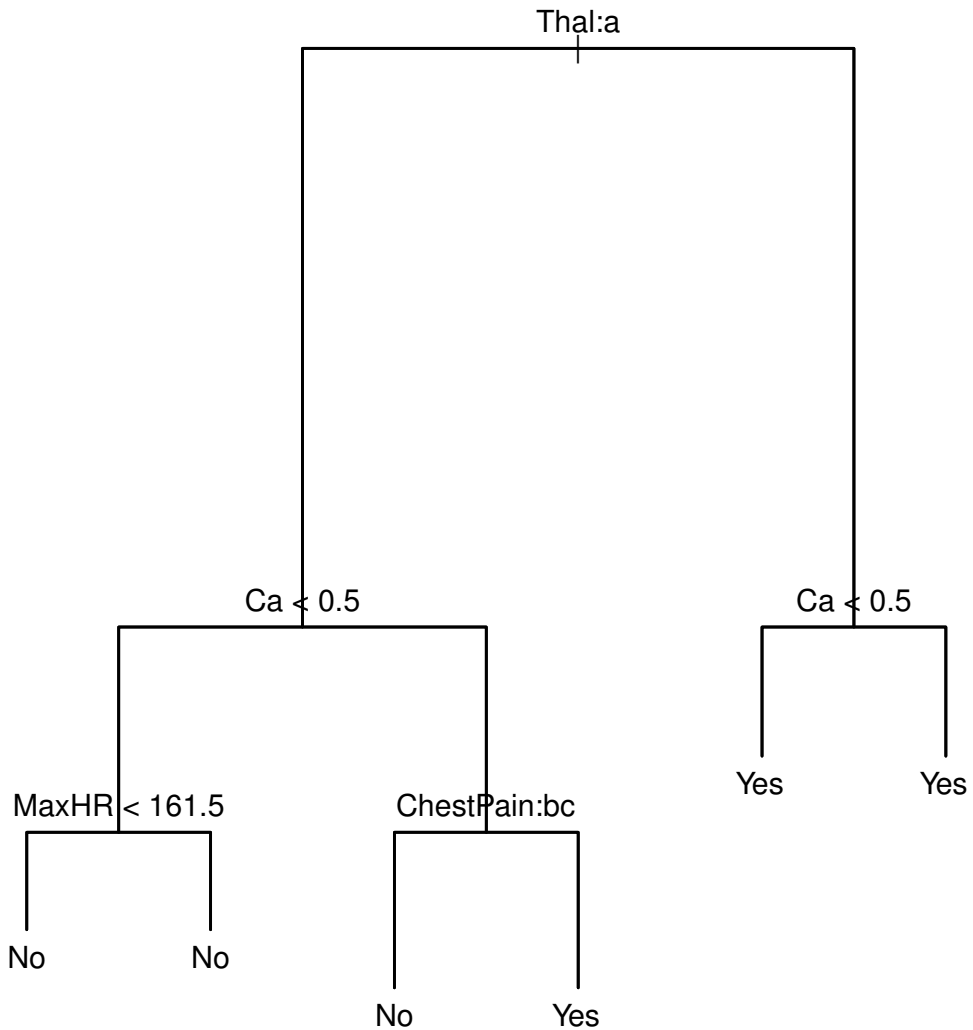
- Grow a large tree
- Prune the tree and select α by CV



- $\alpha \Rightarrow$ tree size = #leaves = #regions $\Rightarrow$ model flexibility

Classification Trees (ISLR 8.1.2)

Heart dataset
Response: Have heart disease or not
13 Predictors (numeric and categorical): Age, Sex, ChestPain, MaxHR, Thal, Ca . . .



Classification Tree: Prediction

- Predictor space partitioned into J non-overlapping regions $R_1, R_2, \ldots, R_J$
- If an observation $(x_1, \ldots, x_p)$ falls into R_j:

$$\hat{y} = \text{most common class of observations in } R_j$$

Classification Tree: Fitting

Algorithm:

1. Grow a large tree by recursive binary splitting
   Stop when a leaf/region has too few observations
2. For each value of α, obtain a subtree by cost complexity pruning
3. Compute the CV error of the subtree corresponding to each α.
   Pick and output the best subtree

- Need a different metric than RSS
- Natural choice: the classification error rate at each region R_m:

$$E = 1 - \max_k(\hat{p}_{mk})$$

- Another choice: the Gini index, measuring the *purity* of a region R_m:

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

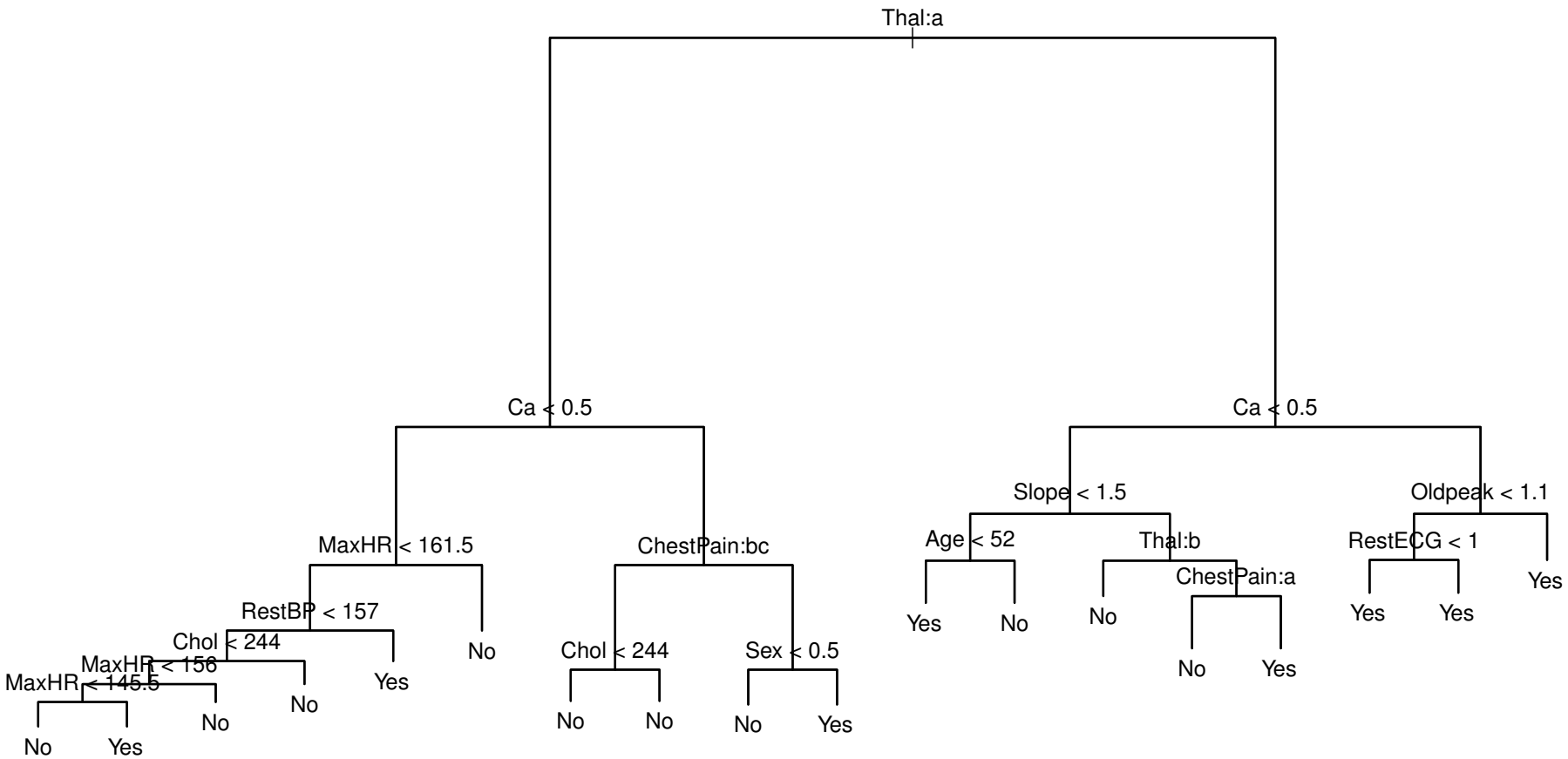
- Third choice: cross-entropy (similar to Gini index)

Classification Tree: Fitting

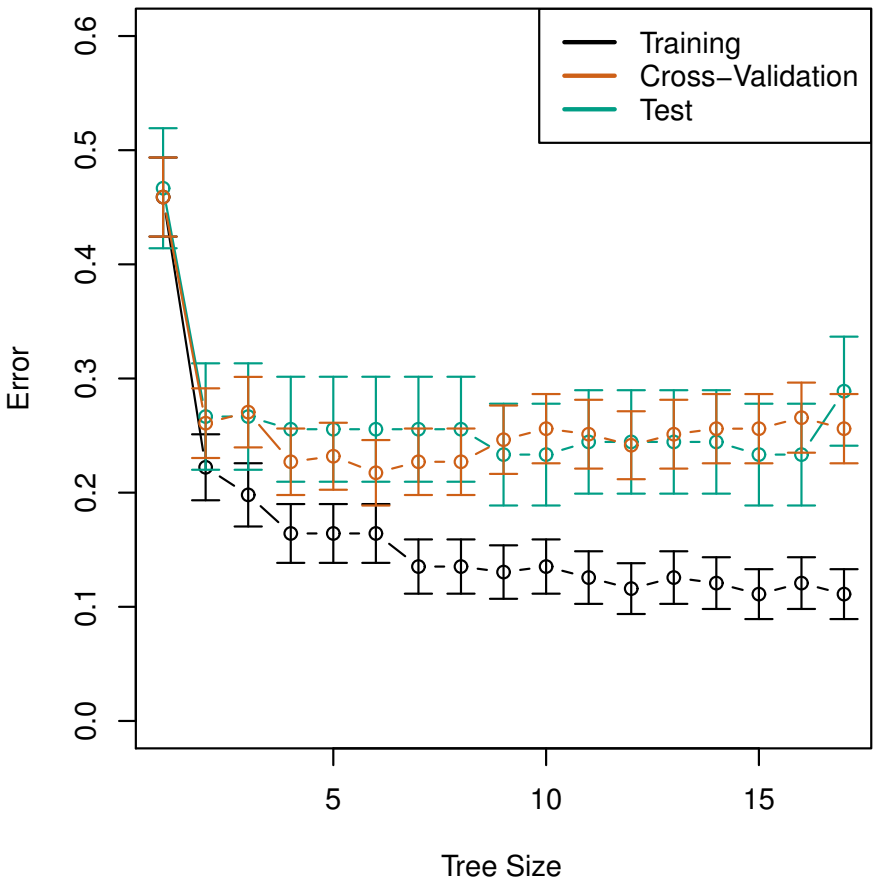
Algorithm:

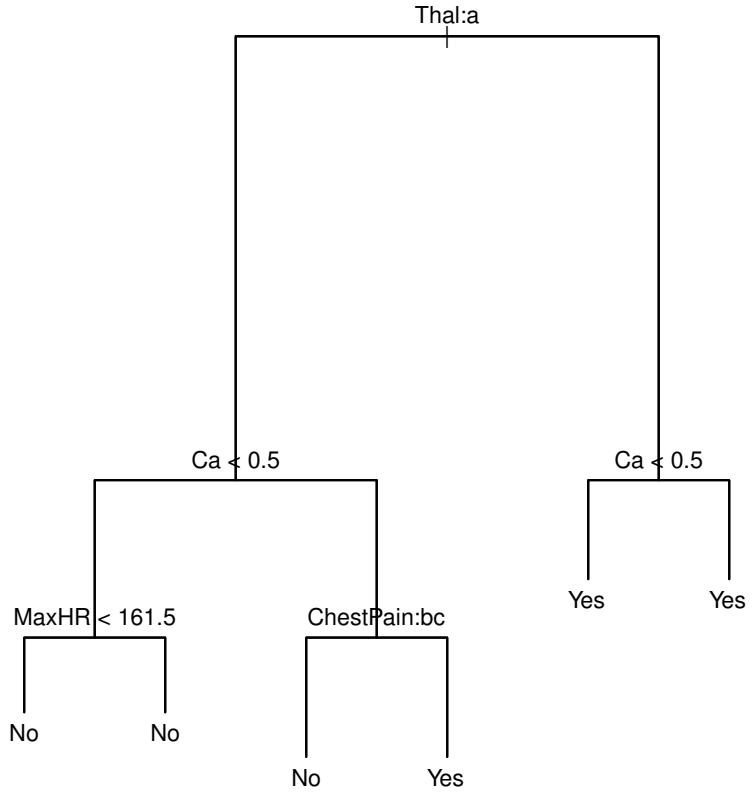
1. Grow a large tree by recursive binary splitting using Gini index.
   Stop when a leaf/region has too few observations
2. For each value of α, obtain a subtree by cost complexity pruning using classification error
3. Compute the CV error of the subtree corresponding to each α.
   Pick and output the best subtree

Heart data: Growing a classification tree



Heart data: Pruning a classification tree





Trees vs. Linear Models

Consider classification

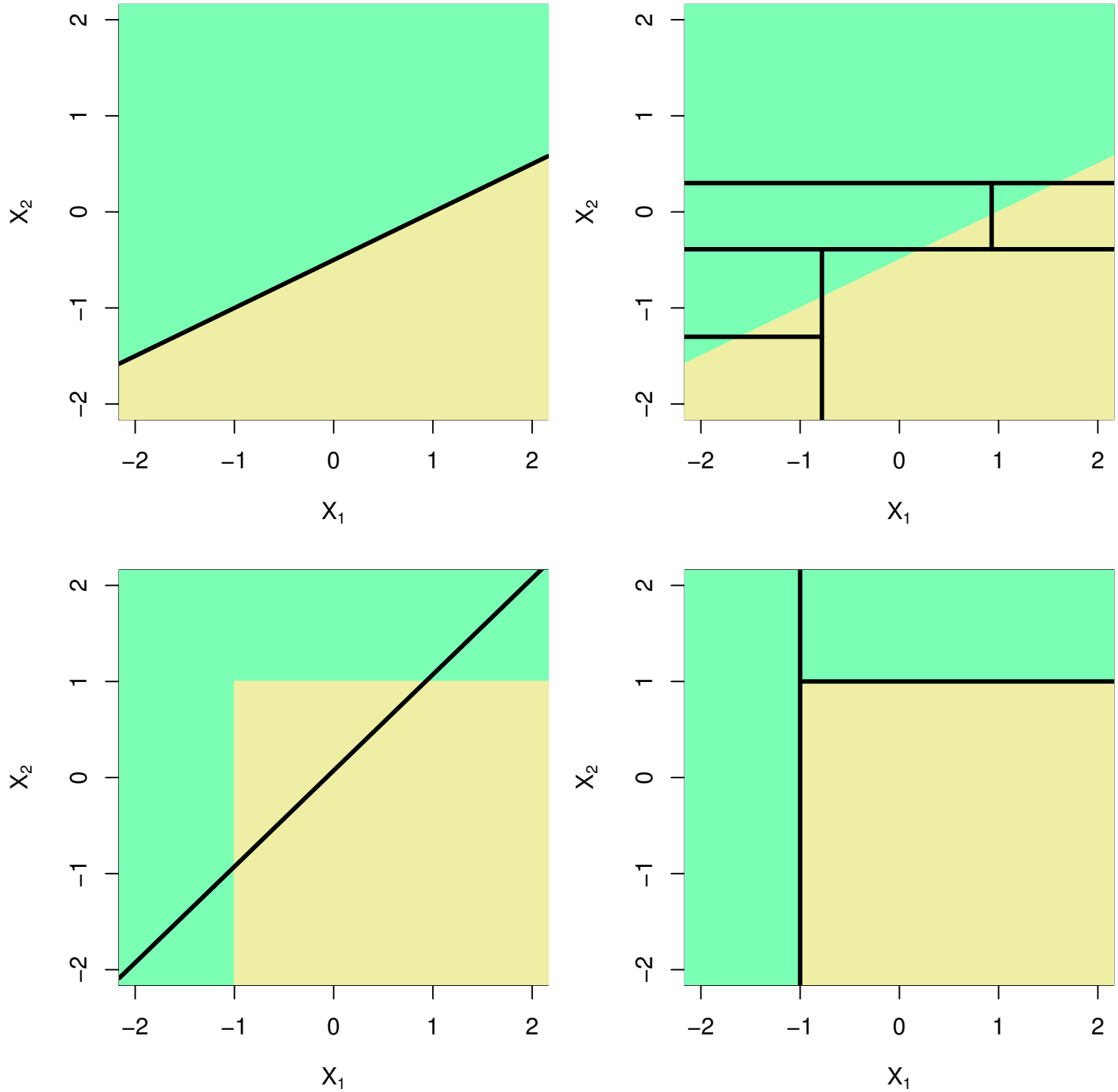
- A linear model (logistic regression) partitions the predictor space by a linear boundary

$$\log\left(\frac{\hat{\Pr}(Y=1)}{1-\hat{\Pr}(Y=0)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

- A decision tree partitions the predictor space into boxes

They work well under different scenarios

Trees are easy to interpret and visualize


X2
X1
−2
−1
0
1
2

Decision Trees: Pros and Cons

- Easy to visualize
- Easy to interpret and explain
- Easily handles categorical predictors
- Some believe that decision trees are similar to decision-making by human

- A single decision tree is often not very accurate

Bagging

A single tree often has high variance

To reduce variance: average many decision trees

- Ideally: Fit trees to many training sets
- Alternatively: Divide a training set into B (non-overlapping) subsets

Bagging: Bootstrap Aggregating ISLR 8.2.1

Bagging: Sample B overlapping sets with replacement

- Grow a large regression tree $\hat{f}^{*b}(\cdot)$ based on the b-th subset
- Then average

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$$

Bagging: Bootstrap Aggregating

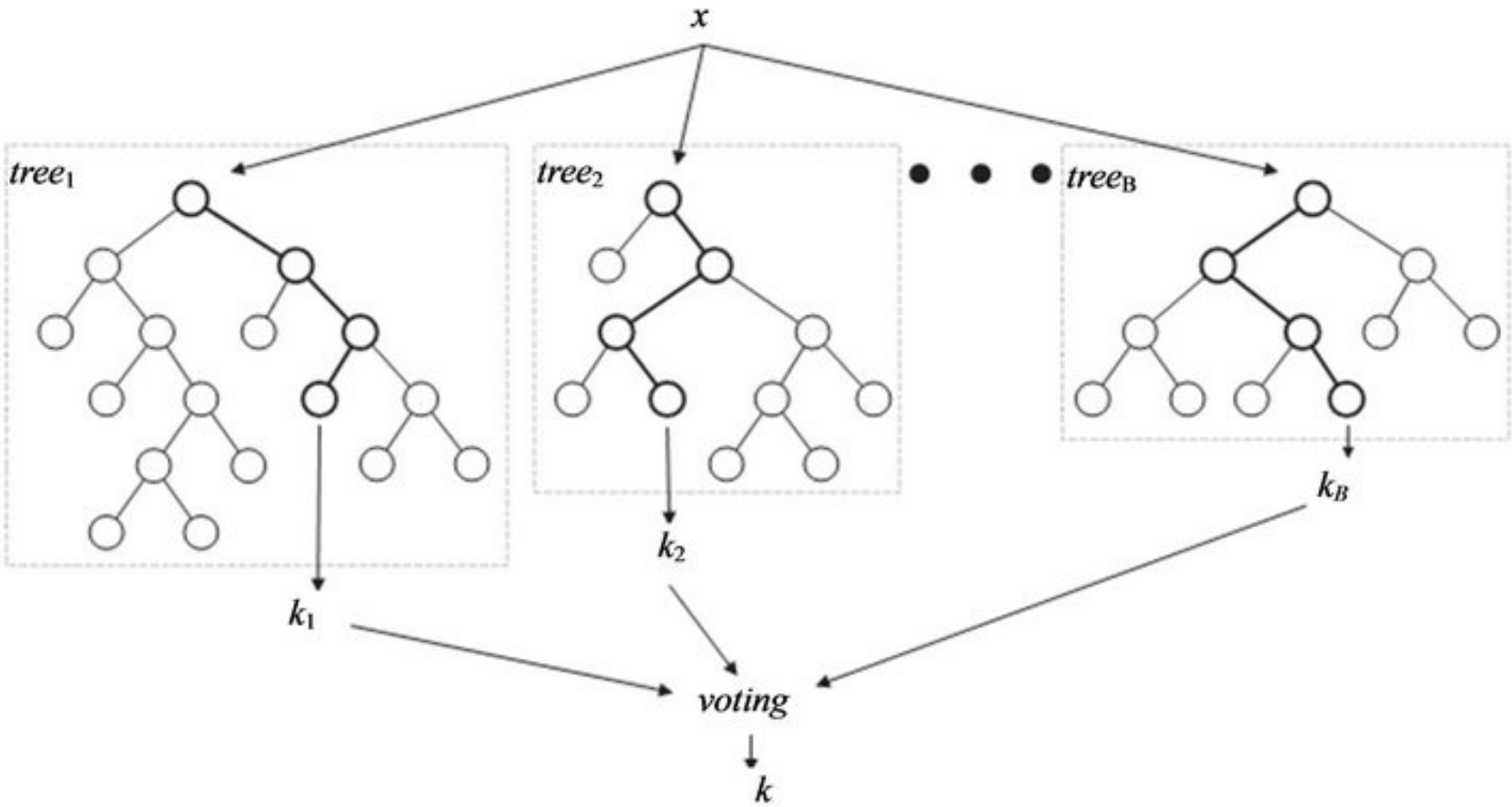
For regression:

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$$

For classification:

$$\hat{f}_{\text{bag}}(x) = \text{majority-vote}\left(\hat{f}^{*1}(x), \hat{f}^{*2}(x), \ldots \hat{f}^{*B}(x)\right)$$

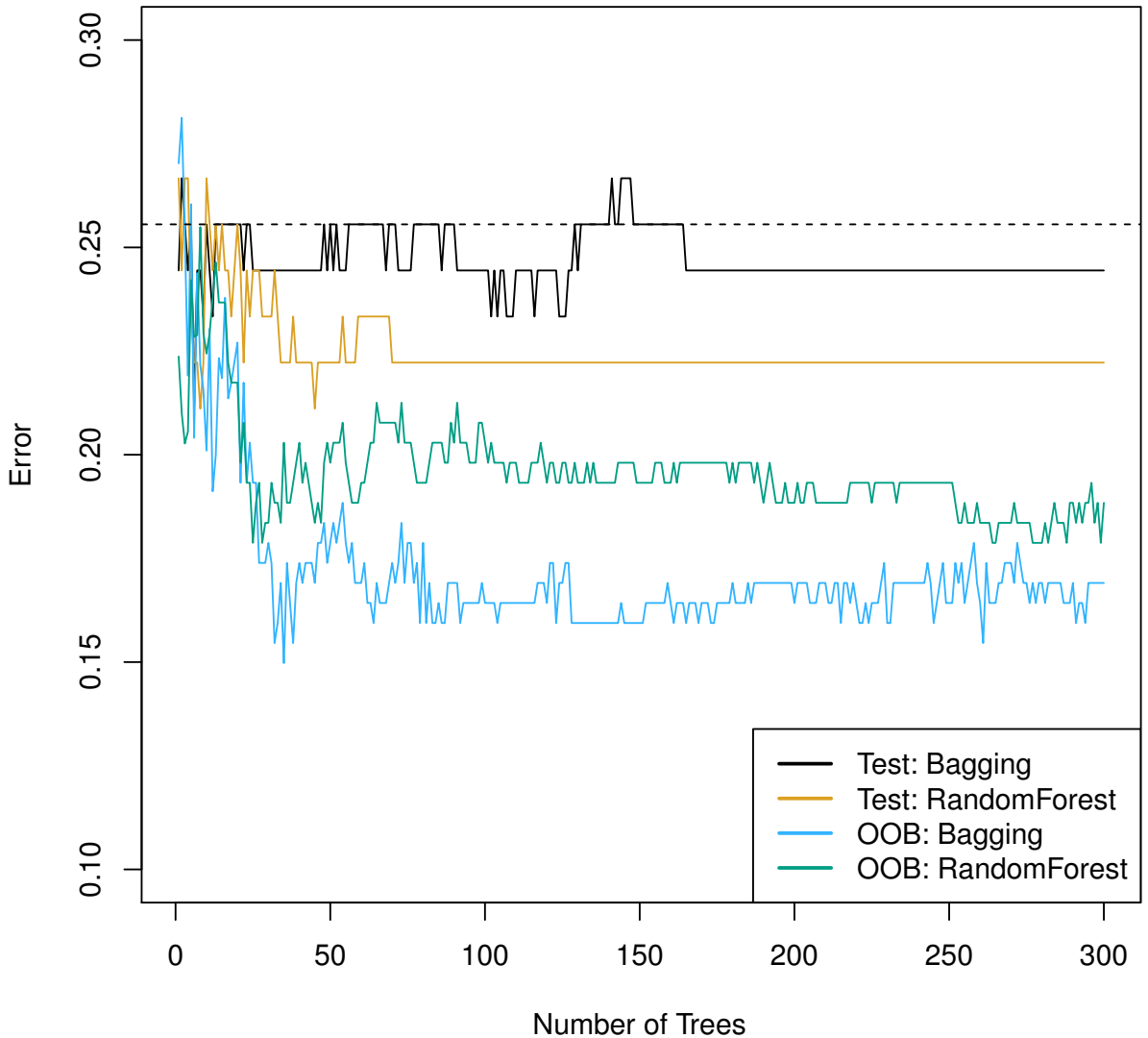
Bagging: Bootstrap Aggregating



(Credit: Cuong Nguyen, Yong Wang, Ha Nam Nguyen 2013)

Heart Dataset

Response: Heart disease or not
13 Predictors: Age, Sex, ChestPain, MaxHR, Thal, Ca . . .



Bagging Tree Algorithm

Given: training data (X, Y)

---

1. For $b = 1, \ldots, B$
   - **a** Sample a set (X^{*b}, Y^{*b}) with replacement from (X, Y)
   - **b** Grow a large tree f^{*b} based on (X^{*b}, Y^{*b})
2. Aggregate the B trees by

   (for regression)

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x)$$

   (for classification)

$$\hat{f}_{\text{bag}}(x) = \text{majority-vote}\left(\hat{f}^{*1}(x), \hat{f}^{*2}(x), \ldots \hat{f}^{*B}(x)\right)$$

- Each tree f^{*b} should be large (no pruning)
- B (# trees) not critical; use large values (hundreds or thousands)

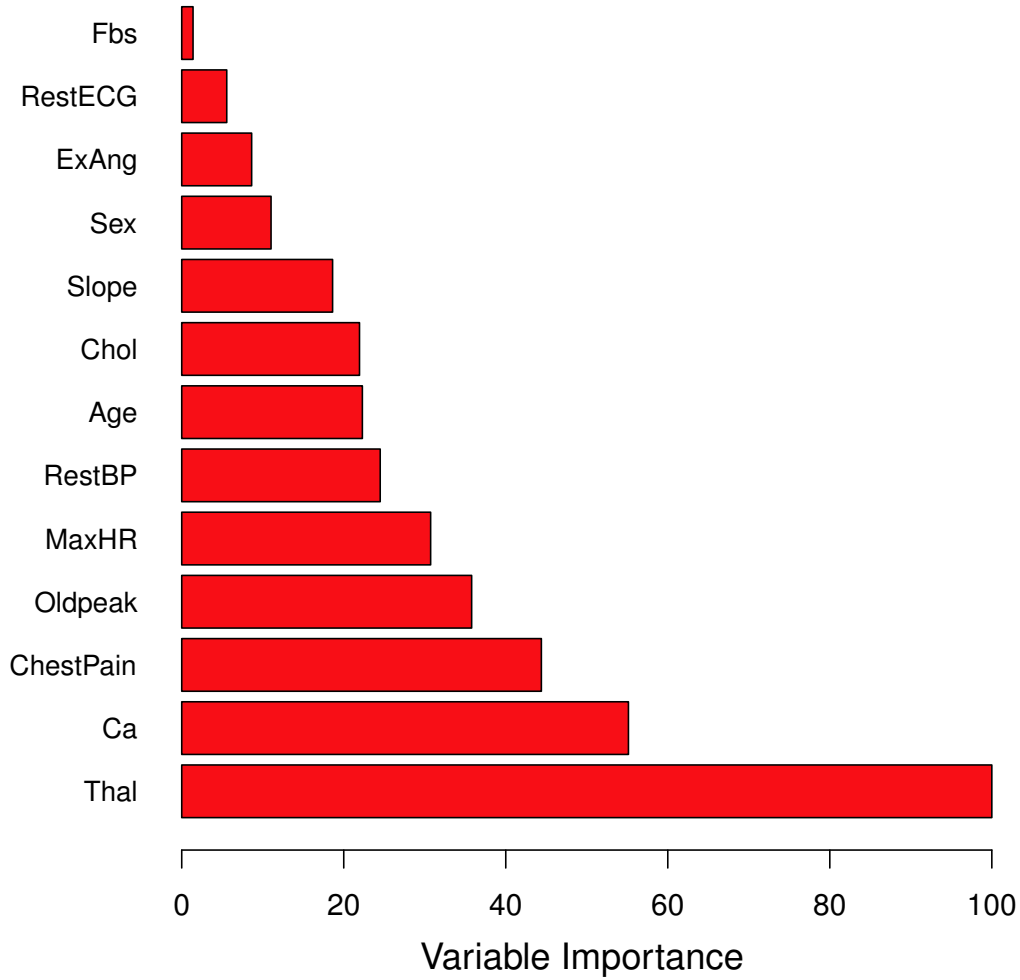
Interpreting Bagging Trees

Bagging

- Improve accuracy over a single tree
- Harder to interpret

Importance of variable *j*:

Reduction of RSS (or Gini index) due to splitting over variable *j*, average over *B* trees



Random Forests

Bagging:

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$$

Averaging reduces variance

Averaging uncorrelated quantities reduces variance!

But the trees $\hat{f}^{*1}, \hat{f}^{*2}, \ldots, \hat{f}^{*B}$ may be very correlated.
Example: A strong predictor/feature

Random Forest: decorrelates trees by random feature selection

Random Forests

Growing a tree $\hat{f}^{*b}$:

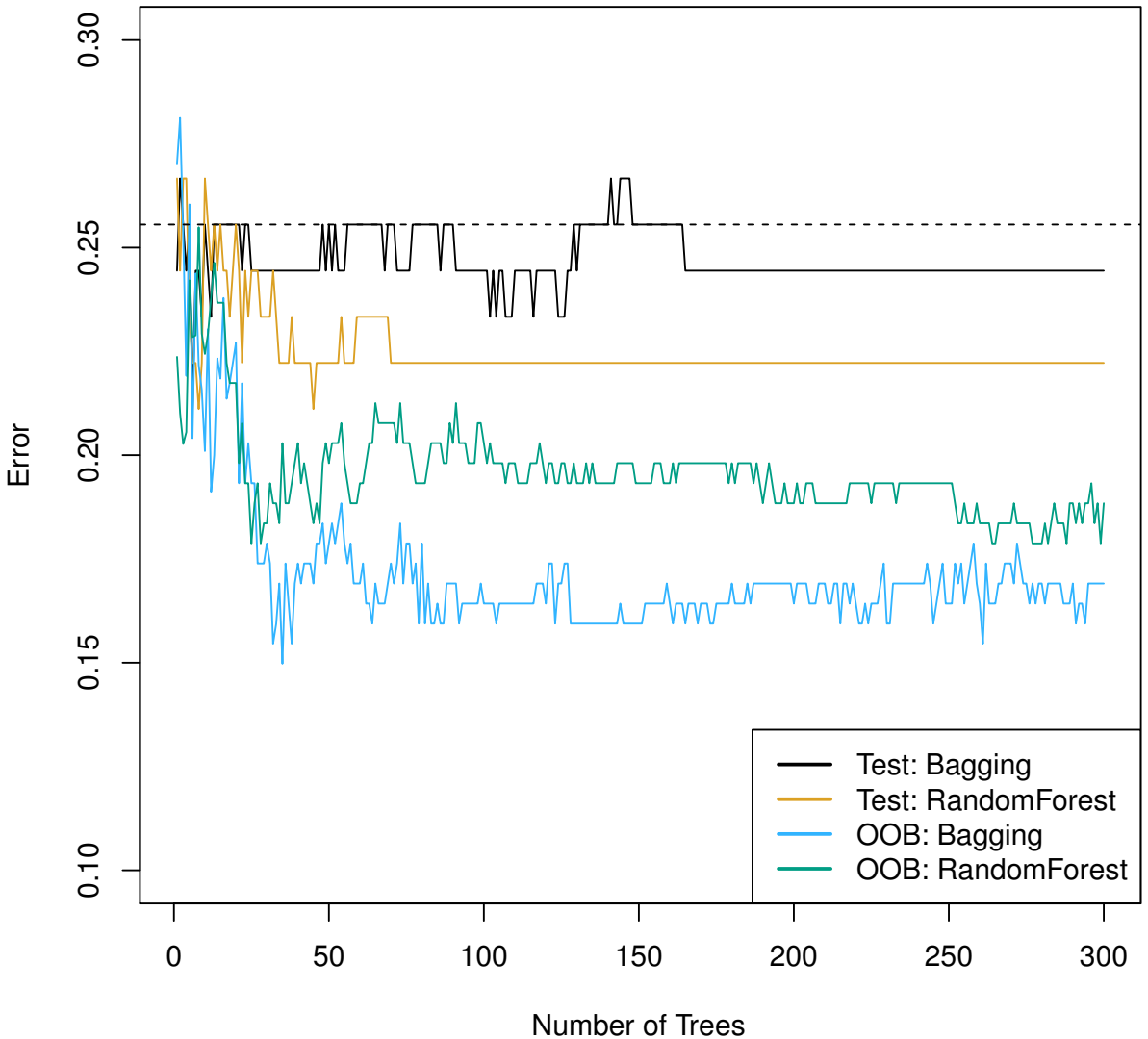
Each time a split is to be identified

- Out of p predictors, randomly sample $m \ll p$ of them as candidates
- Find the best split based on one of these m predictors

- Trees are decorrelated $\Rightarrow$ Reduced variance
- Popular: $m = \sqrt{p}$
- If $m = p$: same as bagging

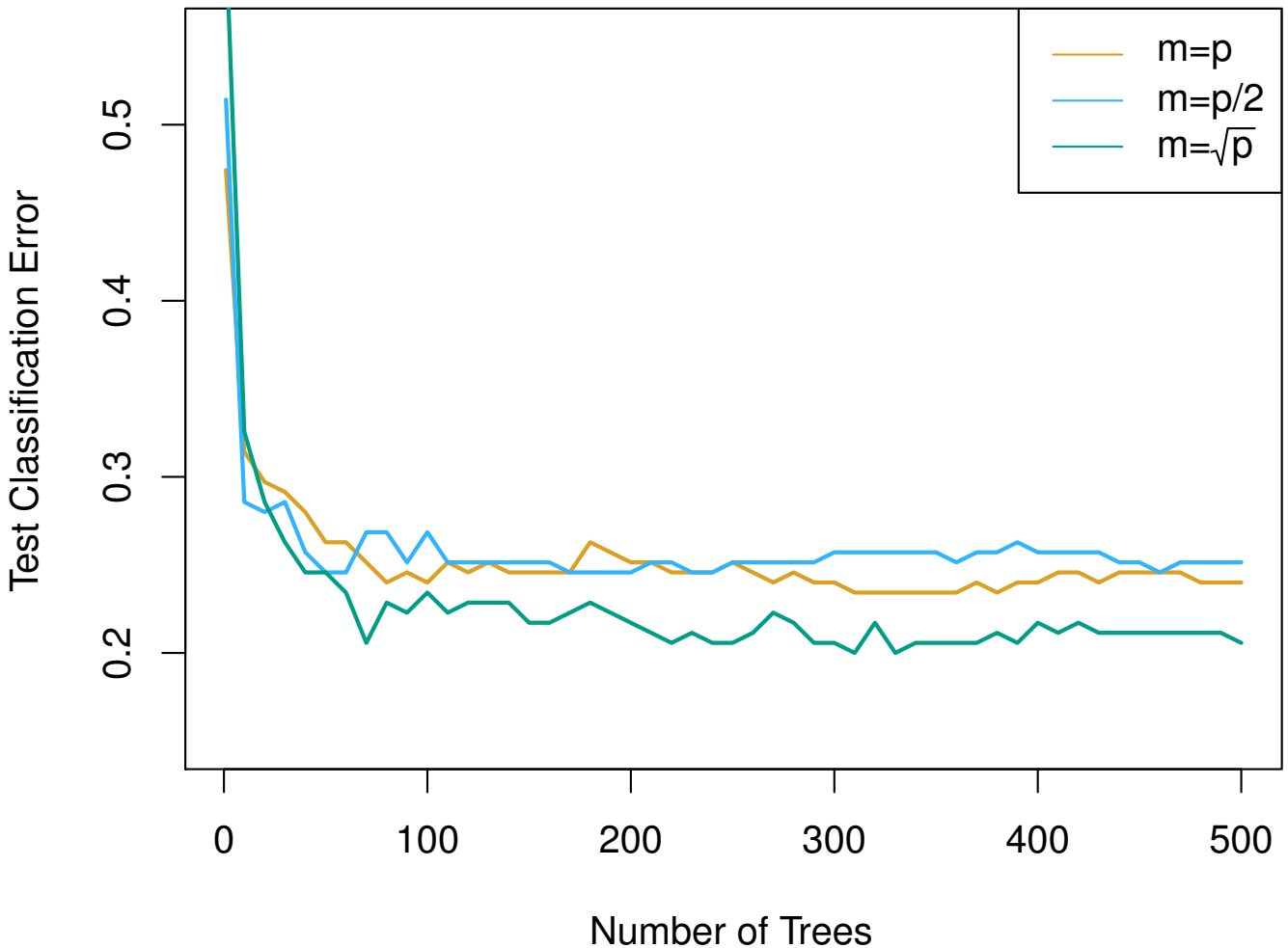
Example 1: Heart Dataset

Response: Heart disease or not
13 Predictors: Age, Sex, ChestPain, MaxHR, Thal, Ca . . .



Example 2: Cancer Dataset

- $n = 349$ observations (patients)
- $p = 500$ predictors (gene measurements)
- Response: Normal, Cancer 1, Cancer 2, . . . , Cancer 14



Boosting and Bagging:

- Combine many trees
- Can be applied to improve other learning methods

Boosting: combine simple trees sequentially

Boosting (Optional) ISLR 8.2.3

Idea: Fit a new tree to the residual of the previous model

Algorithm (for boosting regression trees)

1. Initialization: $\hat{f}(x) \equiv 0$ and $r_i = y_i$, $i = 1, 2, \ldots, n$
2. For $b = 1, 2, \ldots, B$:
   - **a** Fit residual: Fit a new tree $\hat{f}^b$ with $d + 1$ leaves to the data X, r
   - **b** Shrink and combine: Update $\hat{f}$ by adding a shrunken version of $\hat{f}^b$:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$

   - **c** Update residuals:

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i), \quad i = 1, 2, \ldots, n$$

3. Output $\hat{f}$

Key parameters: #trees B, tree complexity/depth d, shrinkage rate λ

Boosting

2 For $b = 1, 2, \ldots, B$:

a Fit residual: Fit a new tree $\hat{f}^b$ with $d + 1$ leaves to the data X, r

b Shrink and combine: Update $\hat{f}$ by adding a shrunken version of $\hat{f}^b$:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$

c Update residuals:

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i), \quad i = 1, 2, \ldots, n$$

Philosophy: Learn slowly

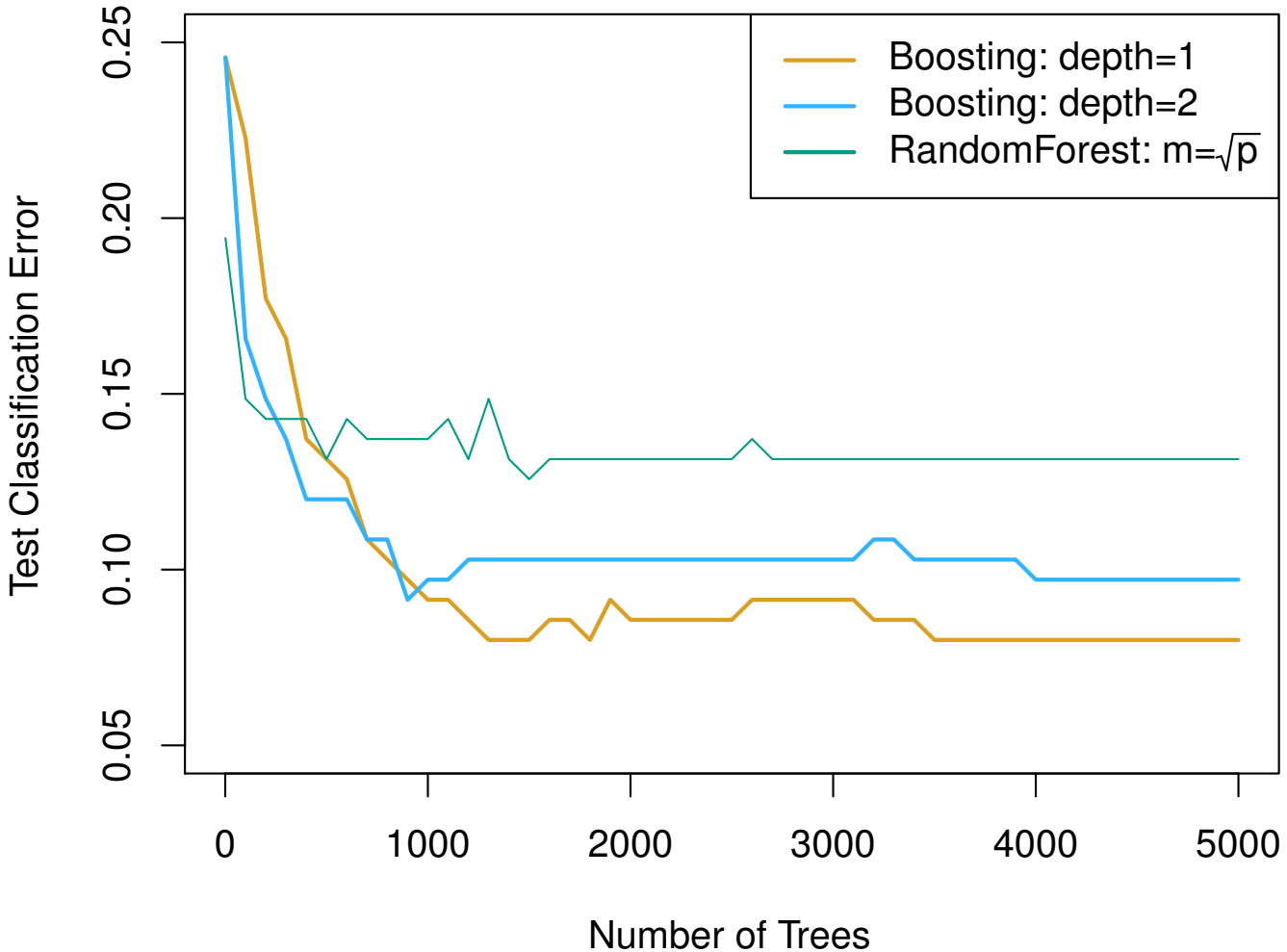
- Each time fit a simple tree (with small d) to the residual
- Combine slowly with a small λ

Choosing parameters:

- #trees B: by CV (large B may overfit)
- Shrinkage rate λ: 0.01 or 0.001
- Depth/complexity d of each tree: $d = 1, 2, 3, 4$

Example: Cancer dataset

- $n = 349$ observations (patients)
- $p = 500$ predictors (gene measurements)
- Response: Normal, Cancer 1, Cancer 2, . . . , Cancer 14



Random Forests vs. Boosting (Optional)

- Combine many trees into one model
- Improve over a single decision tree

- RF: use large trees (grow without pruning)
- Boosting: use small trees (small d)

- RF: train trees on bootstrap samples
- Boosting: train trees on residuals

- RF: combine by averaging/voting
- Boosting: combine by adding trees sequentially

- RF: take B large; won't overfit
- Boosting: choose B by CV; large B may overfit

Random Forests vs. Boosting (Optional)

Two "opposite" ways of improving a single tree

RF: easy to tune; usually don't overfit

Boosting: hard to tune; may overfit
but well-tuned boosting trees often outperform RF