# Chapter 8: Tree-Based Methods

Yonghyun Kwon

Department of Mathematics, Korea Military Academy

# The Basics of Decision Trees

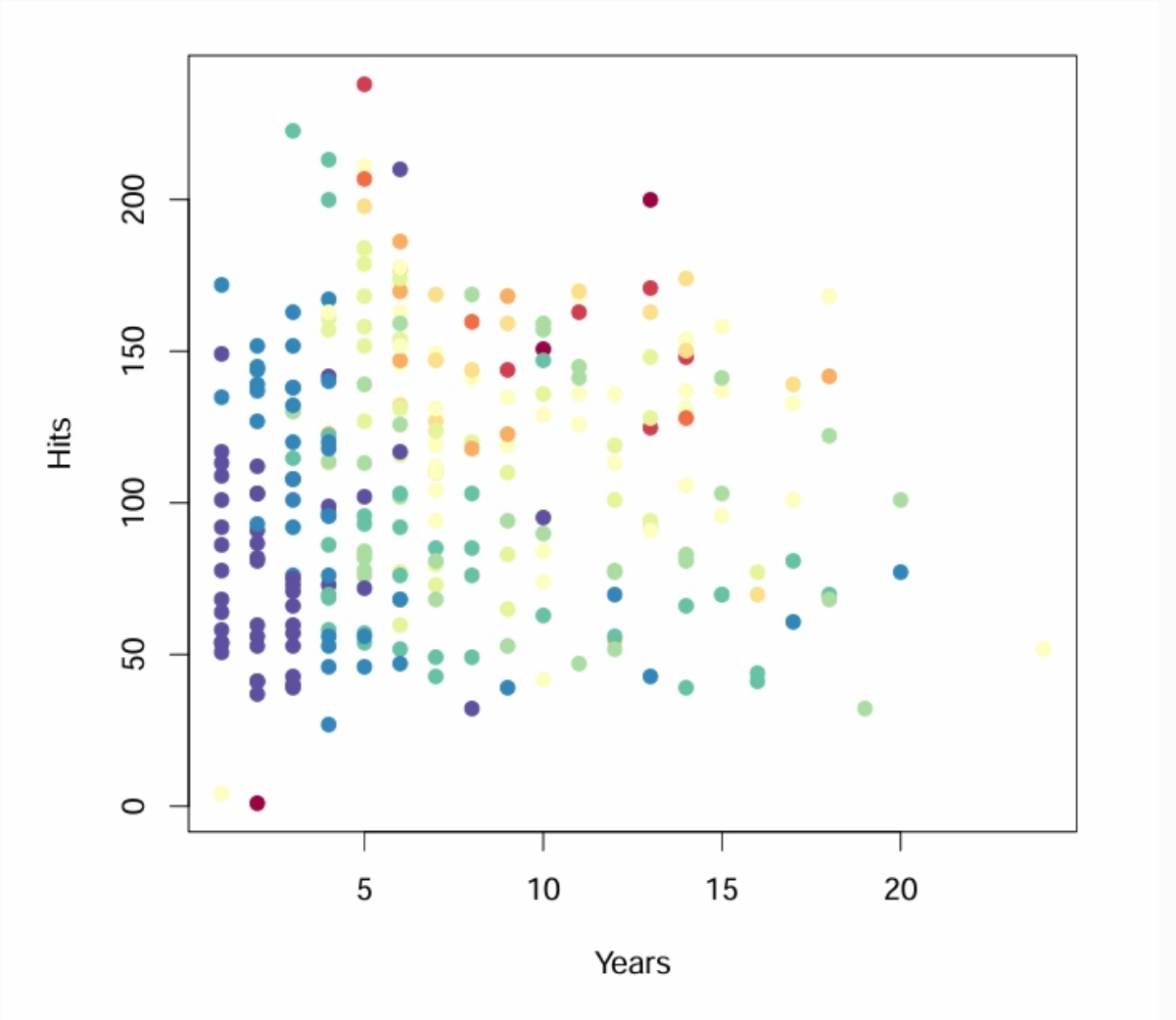
# Tree-based Methods

- Tree-based methods for regression and classification.
- Involve stratifying or segmenting the predictor space into simple regions.
- Since the splitting rules can be summarized in a tree, these are called *decision-tree methods*.

# Pros and Cons

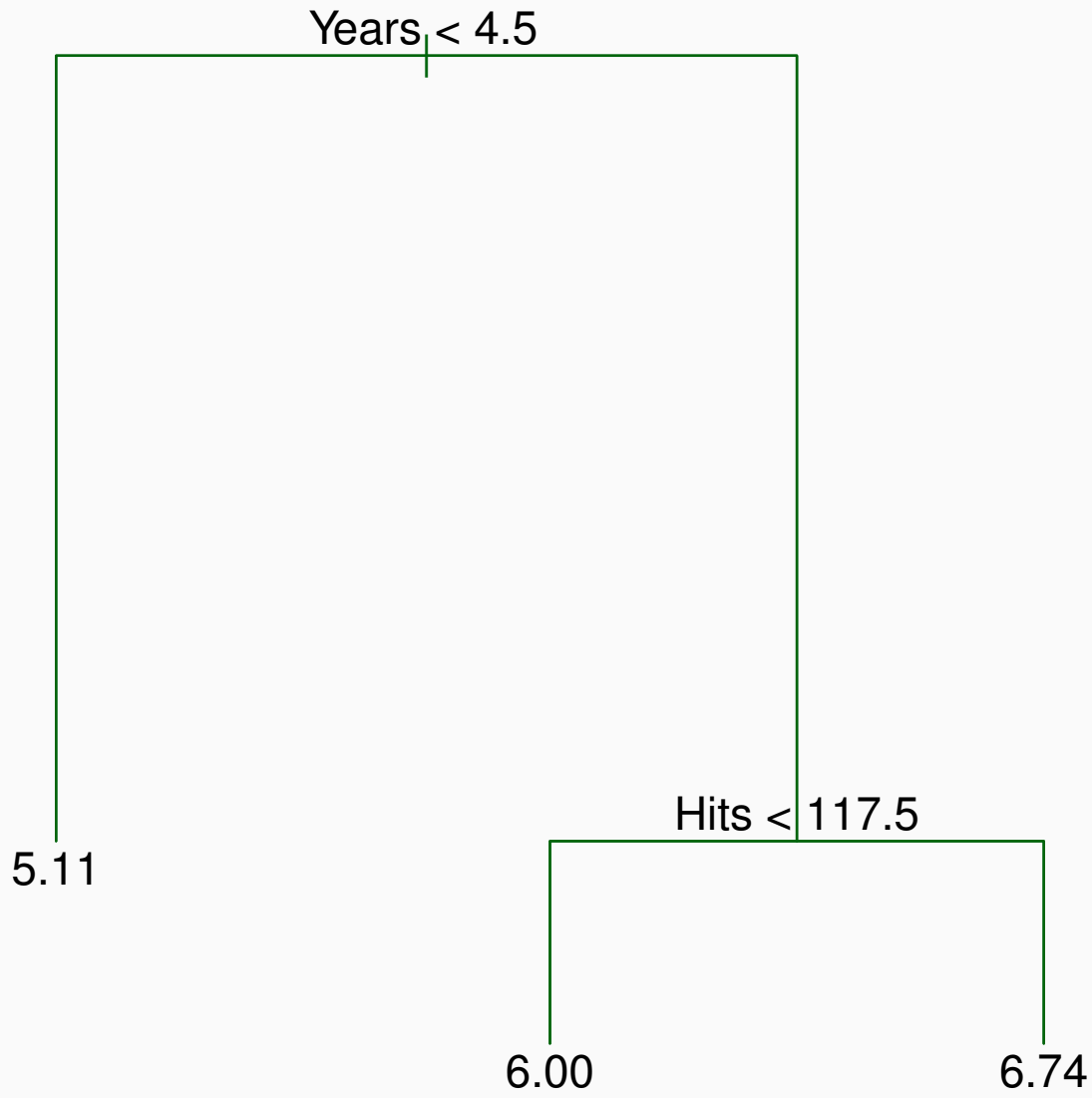
- Tree-based methods are simple and **interpretable**.
- A single tree is typically not competitive with the best supervised learning approaches in terms of **prediction accuracy**.
- *Bagging*, *random forests*, and *boosting* grow multiple trees and combine them.
- Combining many trees often improves prediction accuracy, but at the expense of interpretability.
- We first consider regression problems, then move on to classification.

# Baseball Salary Data




Salary is color-coded from low (blue/green) to high (yellow/red).

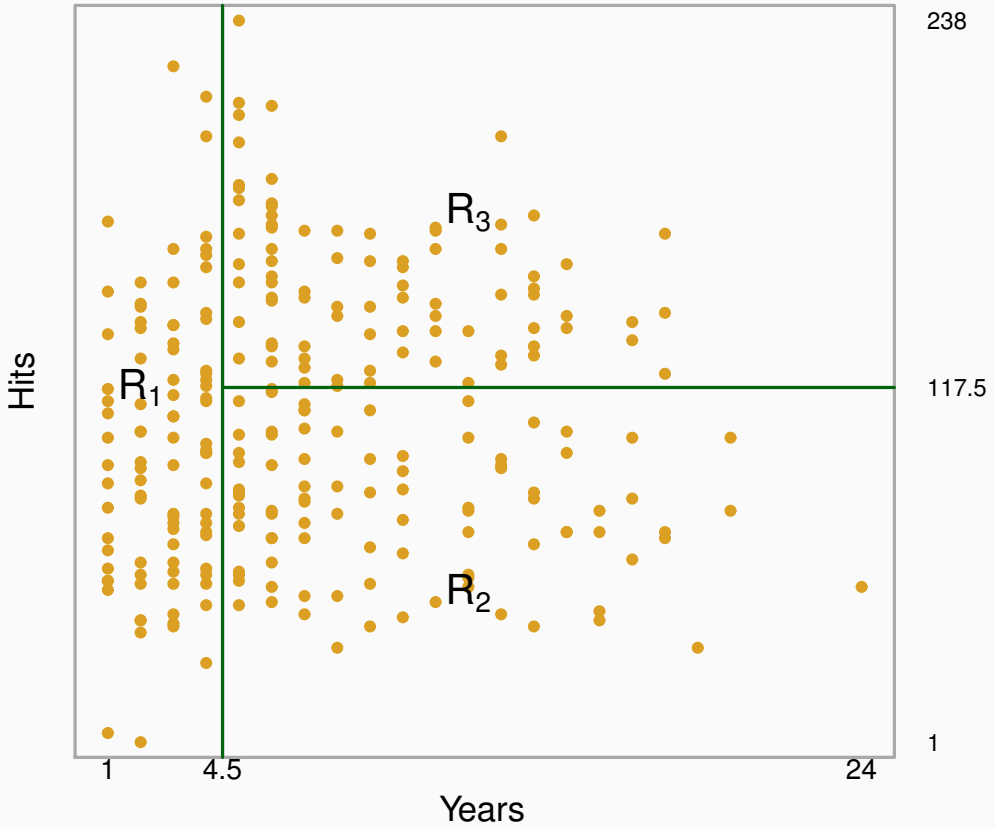
# Decision Tree for These Data



# Details of the Previous Figure

- Regression tree for predicting log `salary` of a baseball player, based on:
  - `Years` played in the major leagues
  - `Hits` in the previous year
- At each *internal node*: split of the form $X_j < t_k$.
- *Leaves(terminal node)* contain the mean response for observations that fall there.
- This tree has 2 internal nodes and 3 terminal nodes.

# Results




$$R_1 = \{X : \text{Years} < 4.5\},$$

$$R_2 = \{X : \text{Years} \geq 4.5,\ \text{Hits} < 117.5\},$$

$$R_3 = \{X : \text{Years} \geq 4.5,\ \text{Hits} \geq 117.5\}$$

# Terminology for Trees

- Regions $R_1, R_2, R_3$ are called *terminal nodes*.
- Decision trees are drawn upside down (leaves at the bottom).
- Split points are called *internal nodes*.
- In the hitters tree: the two internal nodes are "Years $< 4.5$" and "Hits $< 117.5$".

# Interpretation of Results

- `Years` is the most important factor: less experienced players earn lower salaries.
- Among less experienced players, the number of `Hits` matters little.
- Among experienced players ($\geq$ 5 years), more `Hits` $\Rightarrow$ higher `salaries`.
- Simplification, but easier to display, interpret, and explain than regression models.

# Tree-building Process

1. Divide predictor space $(X_1, X_2, \ldots, X_p)$ into $J$ non-overlapping regions $R_1, \ldots, R_J$.
2. For every observation in $R_j$, predict the mean response of training observations in $R_j$.

- For simplicity and interpretability, we use high-dimensional **rectangles (boxes)**.
- The goal: find boxes $R_1, \ldots, R_J$ that minimize

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

where $\hat{y}_{R_j}$ is the mean response for region $R_j$.

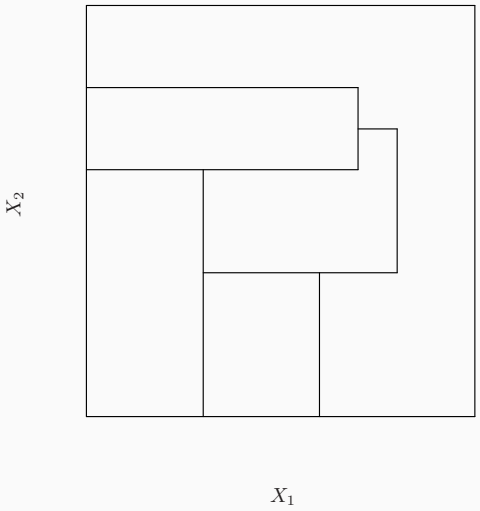
# Recursive Binary Splitting

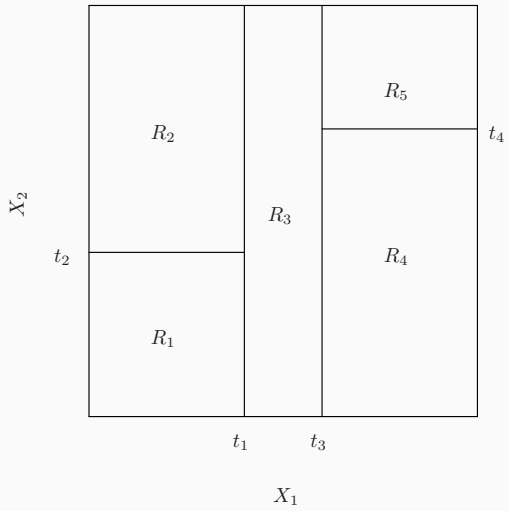
- Considering every possible partition is infeasible.
- Use a *top-down, greedy approach*:
  - Start at the top of the tree and successively split predictor space.
  - At each step, choose the best split at that step (not looking ahead).
  - That is, select predictor $X_j$ and cutpoint $s$ to split:

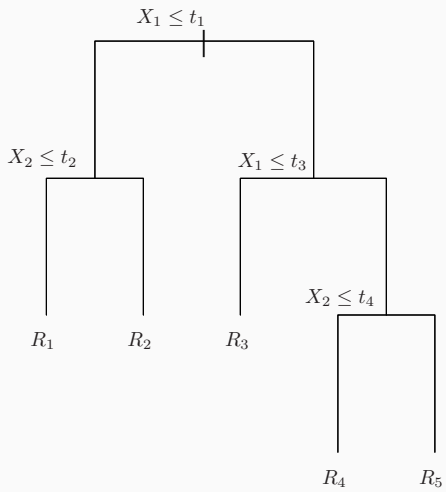
$$\{X : X_j < s\}, \quad \{X : X_j \geq s\}$$

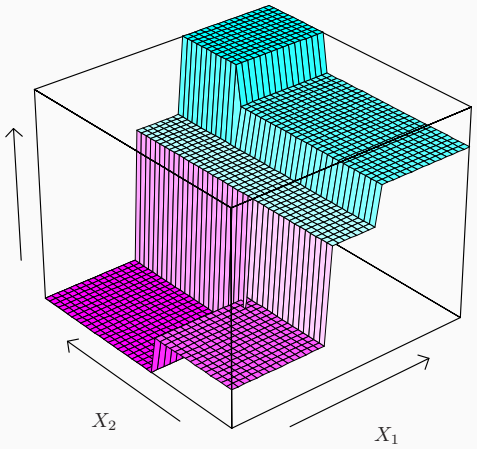
    maximizing reduction in *RSS*.
  - Repeat until stopping rule is met (e.g., no region has more than 5 observations).
- Known as *recursive binary splitting*.

# Predictions from Trees



# Details of the Example Partition

- Top Left: partition not from recursive binary splitting.
- Top Right: partition from recursive binary splitting.
- Bottom Left: corresponding decision tree.
- Bottom Right: prediction surface of that tree.

# Overfitting and Pruning

- Large trees fit training data well, but risk *overfitting*.
- A smaller tree is more interpretable and **lowers variance**, but **may increase bias**.

# Cost Complexity Pruning

- Strategy: grow a large tree $T_0$, then *prune* back.
- *Cost complexity pruning* (a.k.a. weakest link pruning).
- For *tuning parameter* $\alpha \geq 0$, choose subtree $T \subset T_0$ minimizing

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|,$$

where $|T| =$ number of terminal nodes.

# Choosing the Best Subtree

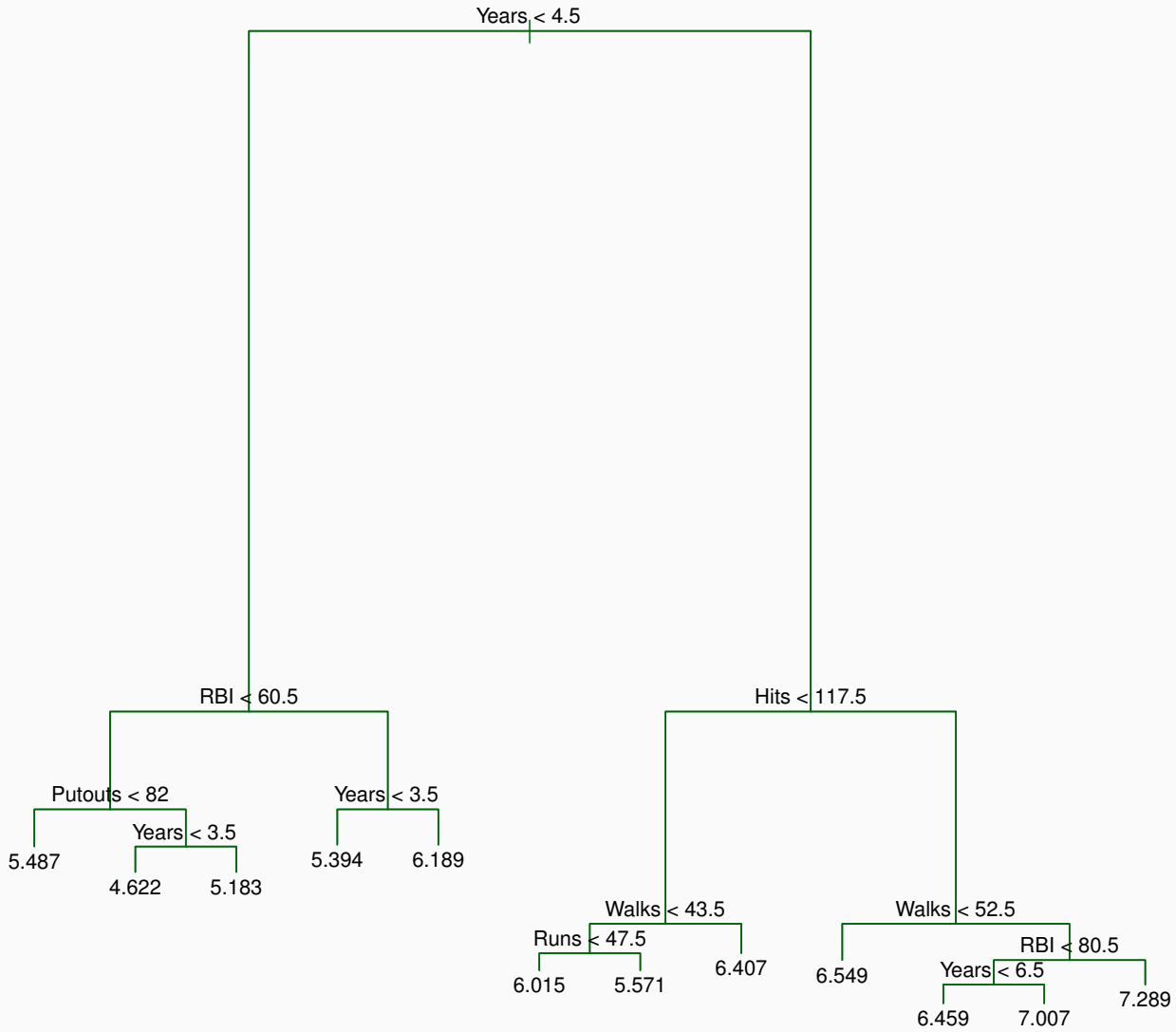
- Tuning parameter $\alpha$ controls trade-off:
  - small $\alpha$: larger trees, $\downarrow$ **bias,** $\uparrow$ **variance.**
  - large $\alpha$: smaller trees, $\downarrow$ **variance,** $\uparrow$ **bias.**
- Select optimal $\hat{\alpha}$ using *cross-validation*.
- Refit subtree on full data corresponding to $\hat{\alpha}$.

# Summary: Tree Algorithm

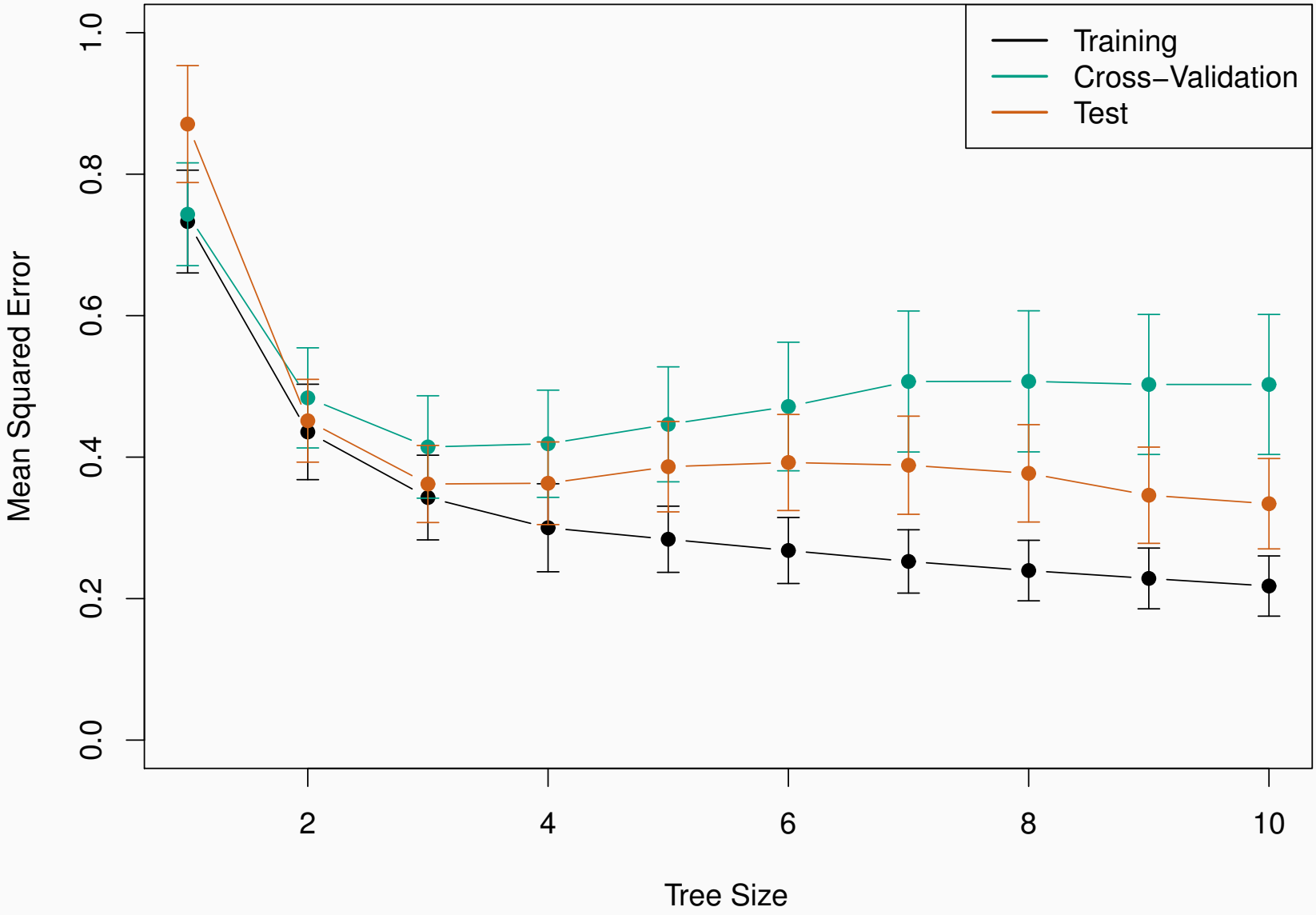
1. Grow large tree with training data using recursive binary splitting.
2. Apply cost complexity pruning to obtain sequence of subtrees.
3. Use $K$-fold cross-validation to choose $\alpha$. For each $k = 1, \cdots, K$:
   - Repeat Steps 1 and 2 on all but $k$th fold.
   - Evaluate the mean squared prediction error on the data in the left-out $k$th fold.

   Pick $\alpha$ to minimize the average error.
4. Return to Step 2 using the chosen value of $\alpha$.

# Baseball Example Continued



# Baseball Example Continued



# Classification Trees

- Similar to regression trees, but for a **qualitative response**.
- Predict that each observation belongs to the most commonly occurring class in its region.
- *Classification and regression tree* is often referred to as *CART*.

# Classification Trees: Details

- Built with recursive binary splitting, like regression trees.
- Cannot use RSS; instead consider classification measures.
- *Classification error rate*:

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

where $\hat{p}_{mk}$ is proportion of training obs. in region $m$ from class $k$.

# Gini Index and Deviance

- *Gini index*:

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

  — $G$ measures total variance across classes.

- *Cross-entropy* (Deviance):

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

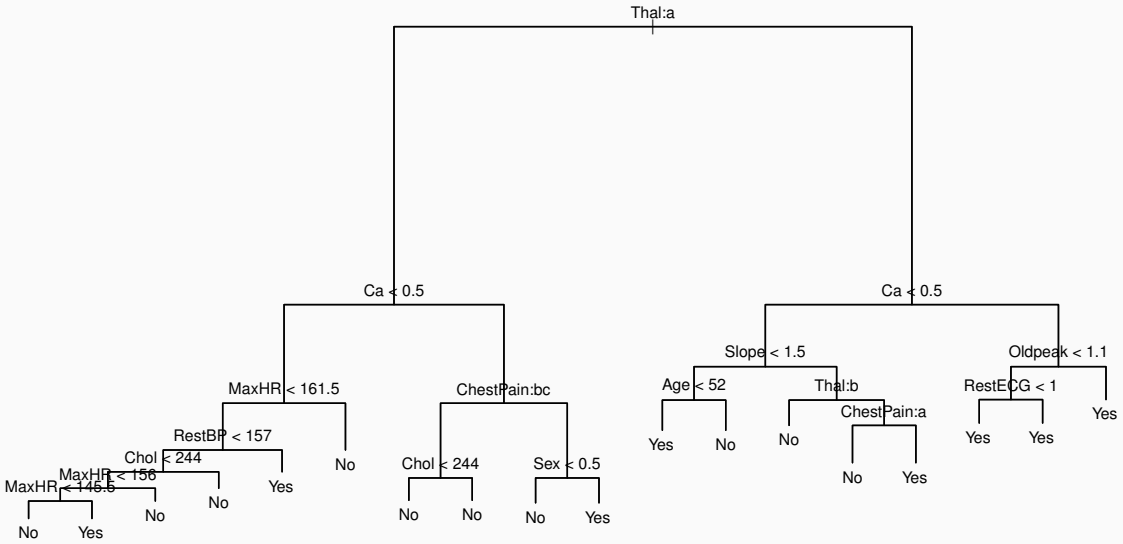
  — $D$ measures the uncertainty of the class distribution.

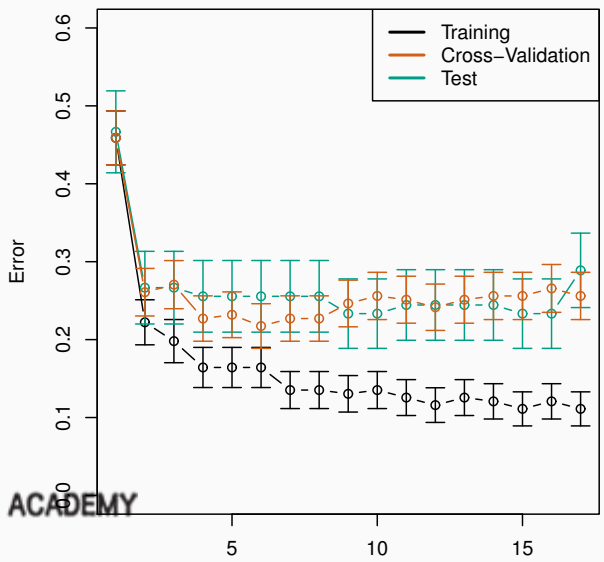
Small $E$, $G$, or $D$ means node is "pure."

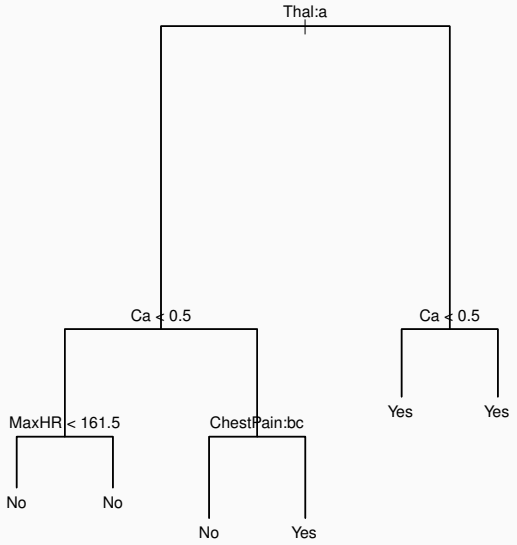
# Example: Heart Data

- 303 patients with chest pain.
- Response: presence of heart disease(`HD`) (`Yes`/`No`).
- Predictors: 13 variables (`Age`, `Sex`, `Chol`, etc.).
- Cross-validation yields tree with six terminal nodes.

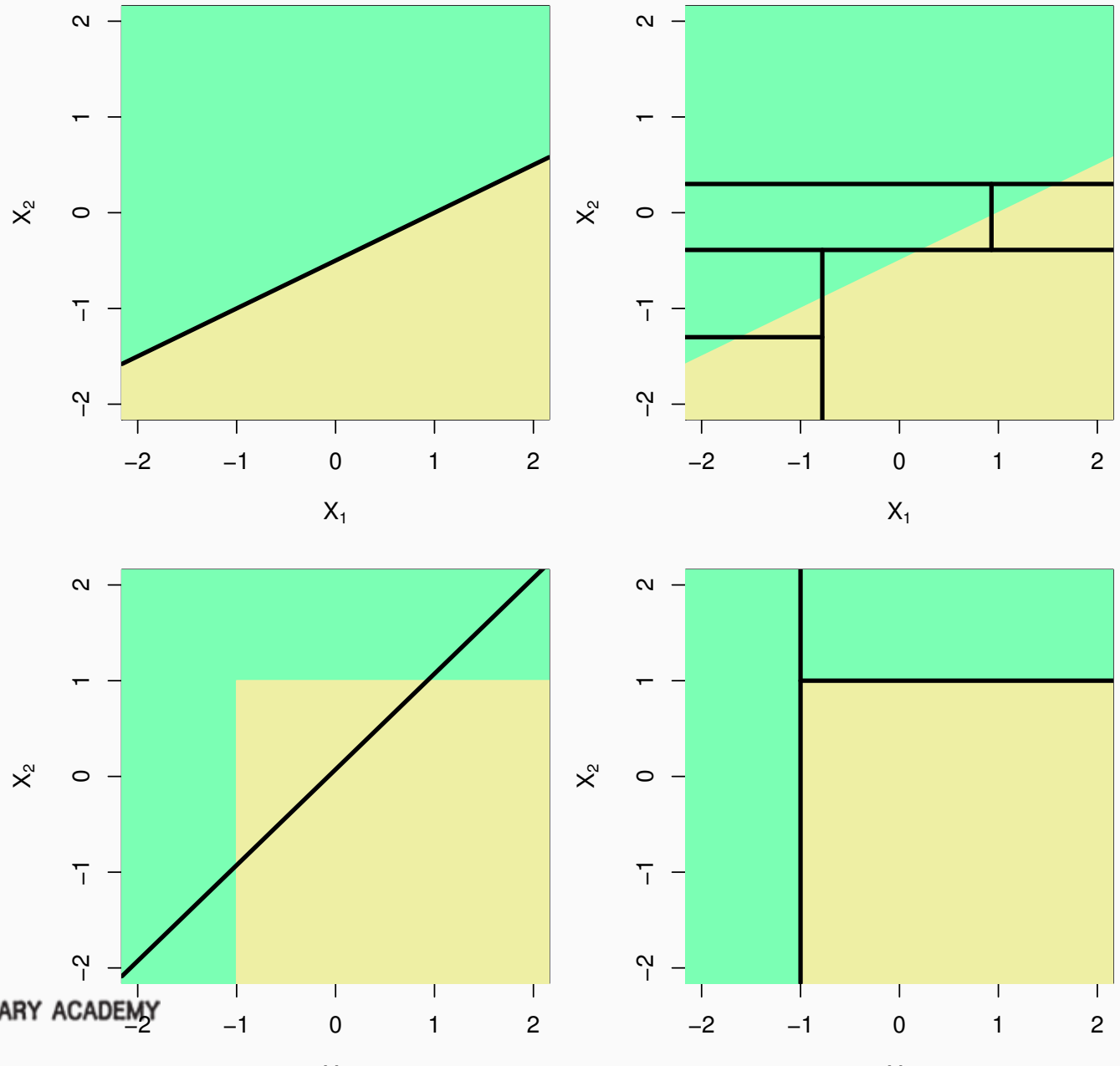
# Heart Data Example



# Trees versus Linear Models



# Details

- Top row: True linear boundary
- Bottom row: True non-linear boundary
- Left column: linear model
- Right column: tree-based model

# Advantages and Disadvantages of Trees

**Advantages**

- Easy to explain.
- Reflect human decision-making more naturally.
- Can be displayed graphically.

**Disadvantages**

- Generally lower predictive accuracy compared to other methods.
  $\Rightarrow$ Aggregation (bagging, random forests, boosting) improves performance.

# Aggregation of trees

# Bagging

- *Bootstrap aggregation (bagging)* reduces **variance** of a method.
- Recall: given $n$ independent observations $X_1, \ldots, X_n$ with variance $\sigma^2$, the sample mean $\bar{X}$ has variance $\sigma^2/n$.
- Averaging reduces variance — but we typically have only one training set.
- Idea: *bootstrap* samples to mimic multiple training sets.
  - It samples repeated observations *with replacement*.

# Bagging: Procedure

- Generate $B$ bootstrap samples from training data.
- Fit a model (e.g., decision tree) to each sample, obtaining $\hat{f}_b^*(x)$.
- Average predictions:

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

- For classification: take *majority vote* among predictions.
  - the overall prediction is the most commonly occurring majority class.

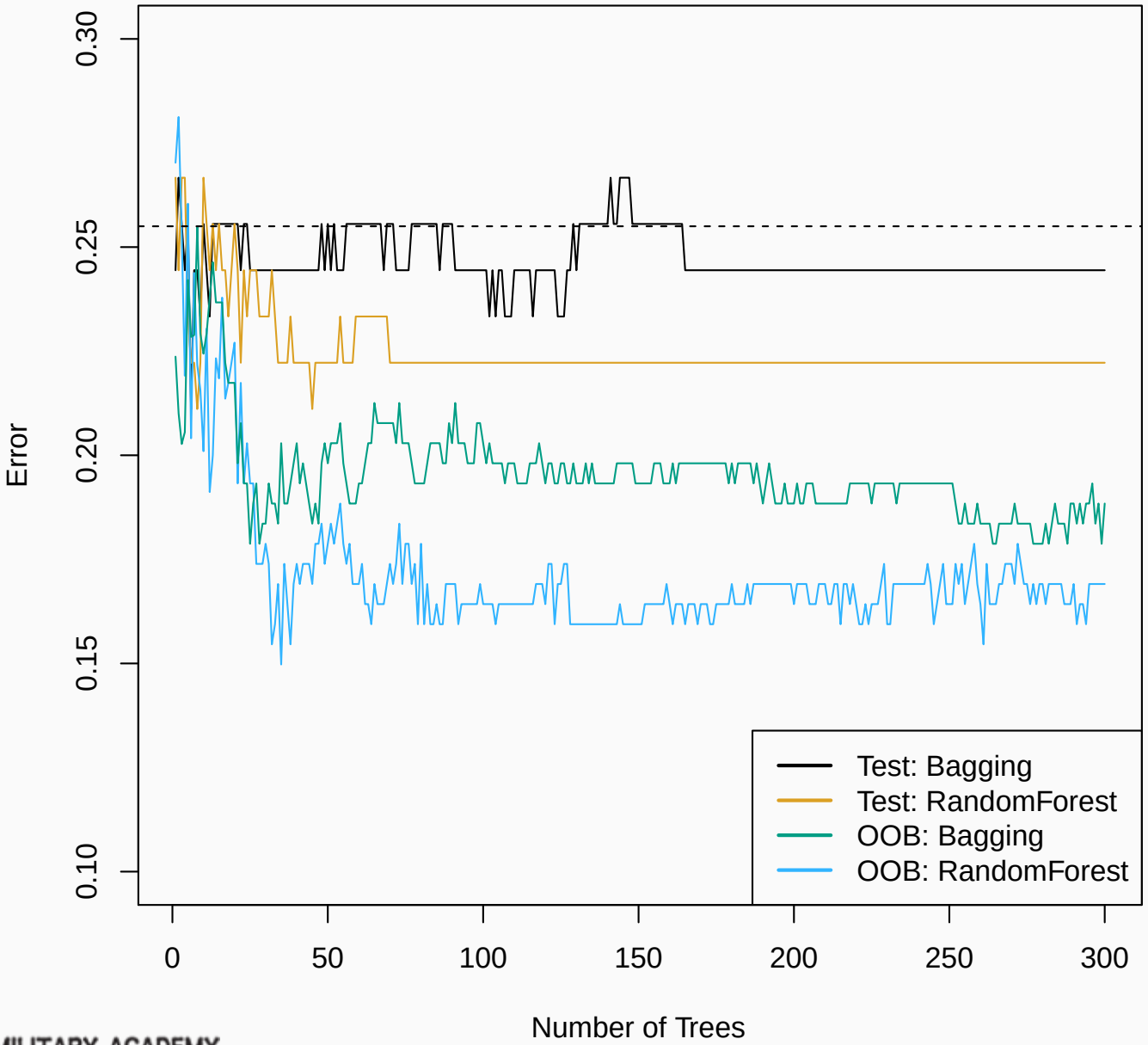
# Out-of-Bag (OOB) Error Estimation

- Each bootstrap sample uses about 2/3 of the training data.
- The remaining 1/3 are the *out-of-bag (OOB)* observations.
- For a training observation $i$:
  - Collect predictions for $i$ from all trees where $i$ was not used in training (OOB).
  - Average these predictions (regression) or take majority vote (classification).
- The OOB error is the average prediction error across all training observations.
- This provides an internal estimate of test error, similar to *leave-one-out CV*.

# Random Forests

- *Random forest* improves on bagging by decorrelating trees.
- Procedure:
  - Build many trees on bootstrap samples.
  - At each split: choose **a random subset of $m$ predictors (out of $p$).**
  - Split only among those $m$ predictors.
- Typical choice: $m \approx \sqrt{p}$.

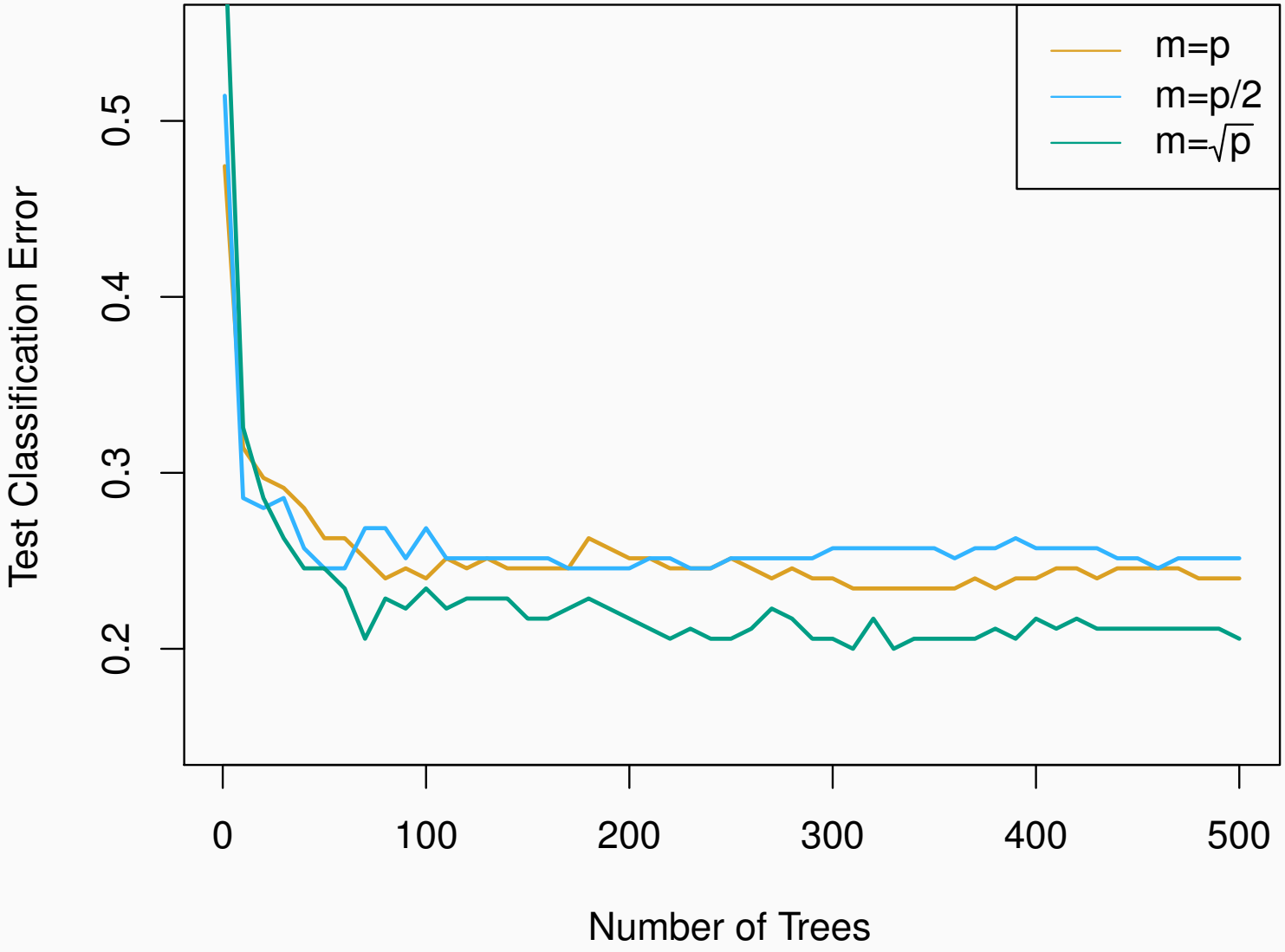
# Bagging the Heart Data



# Example: Gene Expression Data

- High-dimensional dataset: 4,718 genes measured on 349 patients.
- Response: 15-class cancer type (normal or 14 cancer types).
- Used the 500 genes with largest variance in training set.
- Train/test split. Random forests applied with different $m$ values.

# Results: Gene Expression Data



# Boosting

- Like bagging, *boosting* can be applied to regression or classification.
- Key difference: trees are grown **sequentially**, not independently.
- Each new tree uses information from the previously grown trees.
- Idea: improve the model slowly by focusing on areas where it performs poorly.

# Boosting Algorithm for Regression Trees

1. Initialize $\hat{f}(x) = 0$ and residuals $r_i = y_i$.
2. For $b = 1, 2, \ldots, B$:
   2.1 Fit a tree $\hat{f}_b$ with $d$ splits to the training data $(X, r)$.
   2.2 Update the model:

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

   2.3 Update residuals:

$$r_i \leftarrow r_i - \lambda \hat{f}_b(x_i)$$

3. Final boosted model:

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}_b(x)$$

# Boosting and Stagewise regression

*Note:* *Consider a linear model*

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \varepsilon_i.$$

*Stagewise regression fits the model by the following steps:*

1. *Regress $y$ on $x_1$:*

   $$y_i = \alpha_0 + \alpha_1 x_{i1} + \varepsilon_i$$

   *and get the residual $e_i = y_i - \hat{\alpha}_0 - \hat{\alpha}_1 x_{i1}$.*
2. *Regress the residual $e$ on $x_{i2}$:*

   $$e_i = \beta_0 + \beta_2 x_{i2} + \varepsilon_i$$

   *and get the fitted value $\hat{e}_i = \hat{\beta}_0 + \hat{\beta}_2 x_{i2}$.*
3. *The stagewise regression estimator is*

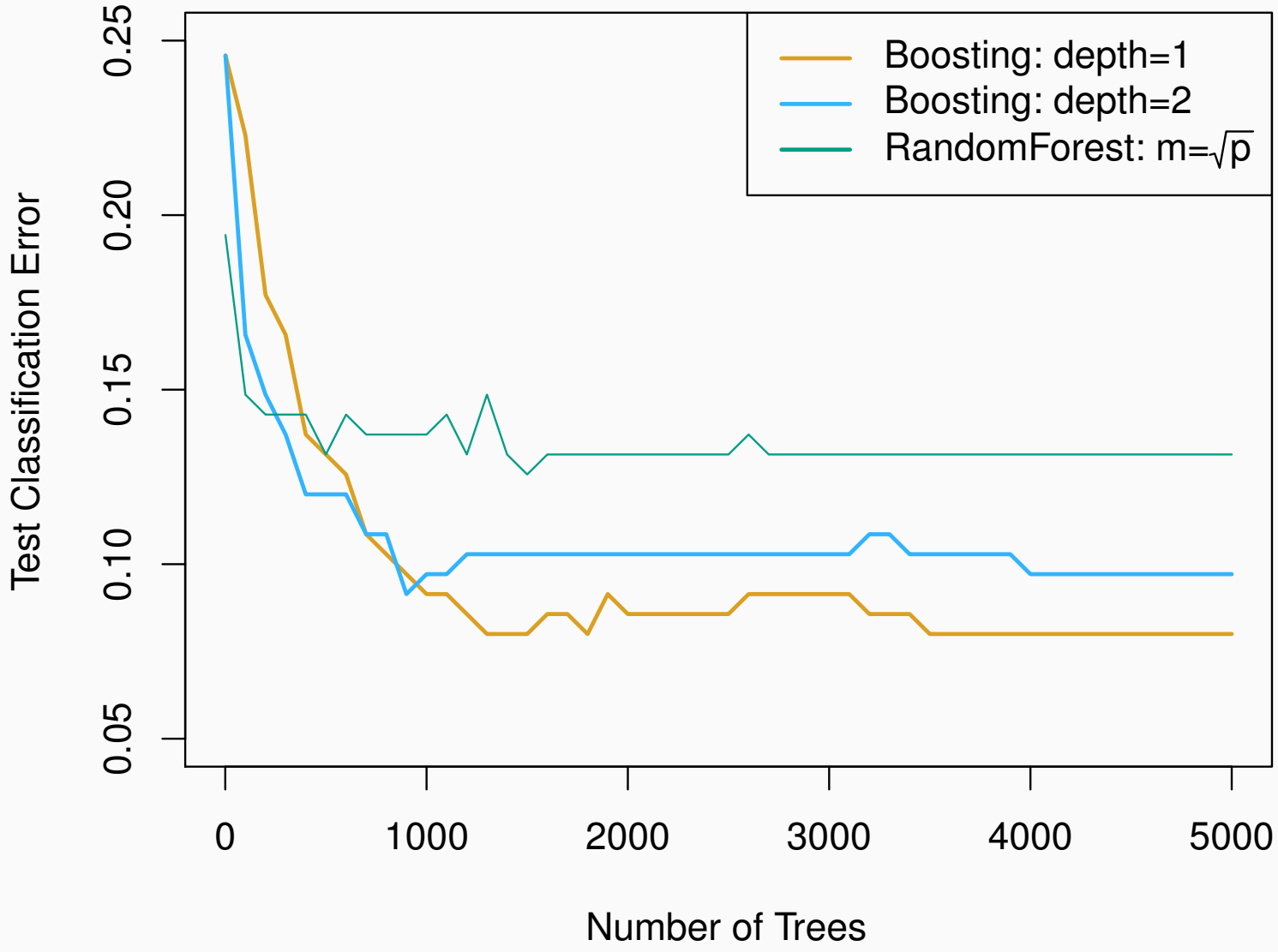
   $$\begin{aligned} \hat{y}^{\text{stage}} &= \hat{\alpha}_0 + \hat{\alpha}_1 x_{i1} + \hat{e}_i \\ &= (\hat{\alpha}_0 + \hat{\beta}_0) + \hat{\alpha}_1 x_{i1} + \hat{\beta}_2 x_{i2} \end{aligned}$$

*Despite bias, $\hat{y}^{stage}$ can have lower MSPE (mean-squared prediction error) than OLS.*

# Idea Behind Boosting for Regression Trees

- A single large tree risks **overfitting**.
- Boosting fits **small trees** (stumps or shallow trees).
- Each tree is fit to the **residuals** of the current model.
- By adding trees sequentially, the model improves in regions where errors are large.
- The **shrinkage parameter** $\lambda$ slows learning, giving better generalization.

# Gene Expression Data: Boosting vs RF



# Tuning Parameters for Boosting

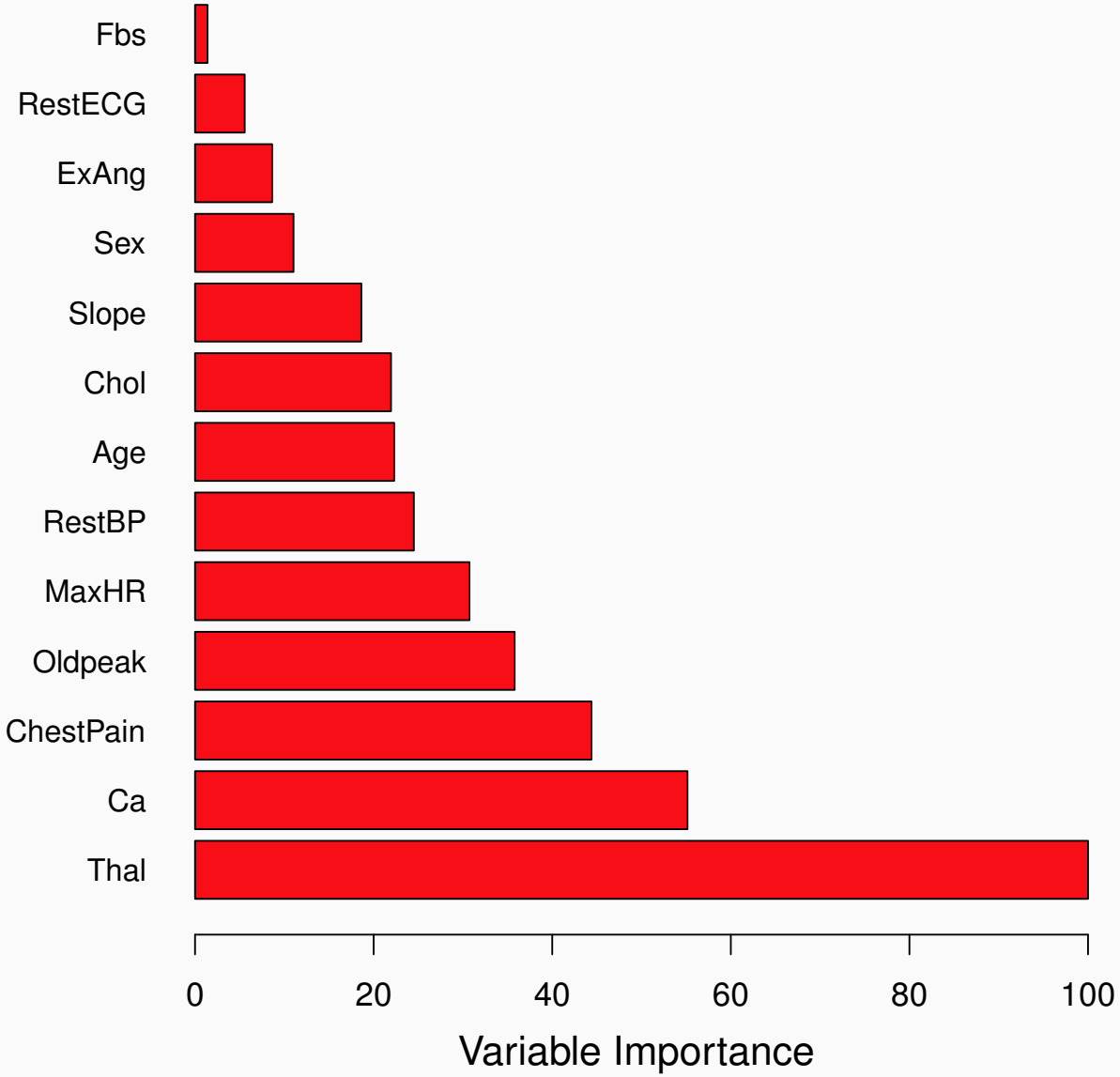
1. Number of trees $B$
   - Too large $B$ can cause overfitting (slowly).
   - Use cross-validation to select $B$.
2. Shrinkage parameter $\lambda$
   - Small positive number (e.g., 0.01 or 0.001).
   - Controls learning rate; smaller $\lambda$ requires larger $B$.
3. Number of splits $d$ in each tree
   - $d = 1$: tree stumps $\Rightarrow$ often works well.

# Variable Importance Measures

For bagged or random forest trees, *variable importance* is measured by:

- the total decrease in **RSS** (regression) or in **Gini index** (classification) due to splits on a given predictor,
- averaged over all $B$ trees.
- A larger value $\Rightarrow$ a more important predictor.

# Variable importance plot for the Heart data.



# Summary of Tree-Based Methods

- **Decision trees**: simple, interpretable for regression and classification, but often not competitive in prediction accuracy.
- **Bagging, random forests, and boosting**: improve prediction accuracy by aggregating many trees.

# Bayesian Additive Regression Trees (BART)

- Ensemble method using decision trees as building blocks.
- Comparison:
  - Bagging / Random Forests: average predictions from many trees, each fit separately.
  - Boosting: weighted sum of trees, each fit sequentially to residuals.
- *BART(Bayesian Additive Regression Tree)* combines both ideas:
  - Trees constructed randomly (like bagging/RF).
  - Each new tree captures signal not yet explained (like boosting).

# BART: Notation

- Let $K$ = number of regression trees.
- Let $B$ = number of iterations of the algorithm.
- Prediction at $x$ for $k$th tree in $b$th iteration:

$$\hat{f}_k^{(b)}(x).$$

- At iteration $b$, total prediction is

$$\hat{f}^{(b)}(x) = \sum_{k=1}^{K} \hat{f}_k^{(b)}(x), \quad b = 1, \ldots, B.$$

# BART Iterations

- First iteration:

$$\hat{f}_k^{(1)}(x) = \frac{1}{nK}\sum_{i=1}^{n} y_i, \quad \hat{f}^{(1)}(x) = \frac{1}{n}\sum_{i=1}^{n} y_i$$

  (mean of $y$ distributed across $K$ trees).
- Subsequent iterations:
  - Update each tree one at a time.
  - Compute *partial residual* for observation $i$:

$$r_i = y_i - \sum_{k'<k} \hat{f}_{k'}^{(b)}(x_i) - \sum_{k'>k} \hat{f}_{k'}^{(b-1)}(x_i).$$

  - Fit/perturb the $k$th tree to explain $r_i$.

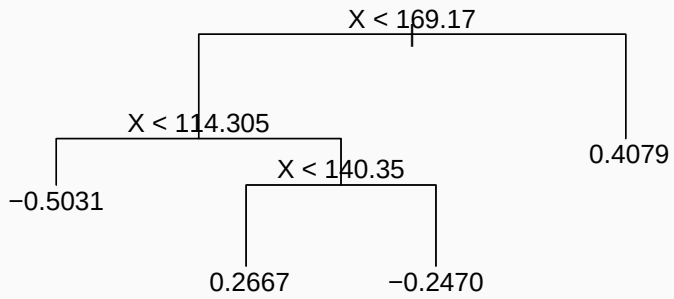
# BART Iterations and Tree Updates

- At iteration $b$, update trees sequentially:
  - For observation $i$, compute partial residual

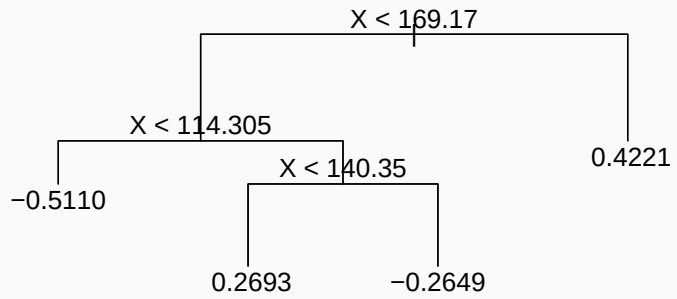
$$r_i = y_i - \sum_{k'<k} \hat{f}_{k'}^{(b)}(x_i) - \sum_{k'>k} \hat{f}_{k'}^{(b-1)}(x_i).$$

  - Instead of fitting a fresh tree, randomly *perturb* the $k$th tree to fit $r_i$.
- Perturbations may involve:
  1. Changing structure (add/prune splits).
  2. Updating terminal node predictions.
- Perturbations that improve the fit to residuals are favored.

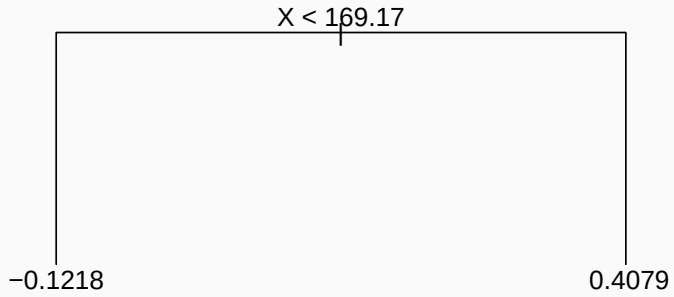
# Tree Perturbations in BART



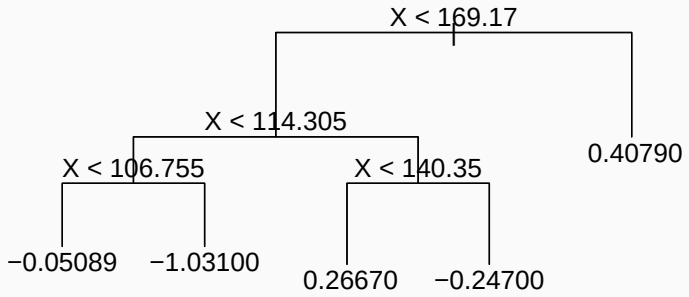

(a): $\hat{f}_k^{(b-1)(X)}$




(b): Possibility #1 for $\hat{f}_k^{(b)(X)}$




(c): Possibility #2 for $\hat{f}_k^{(b)(X)}$




(d): Possibility #3 for $\hat{f}_k^{(b)(X)}$

# BART Predictions

- After $B$ iterations, we obtain $B$ prediction models:

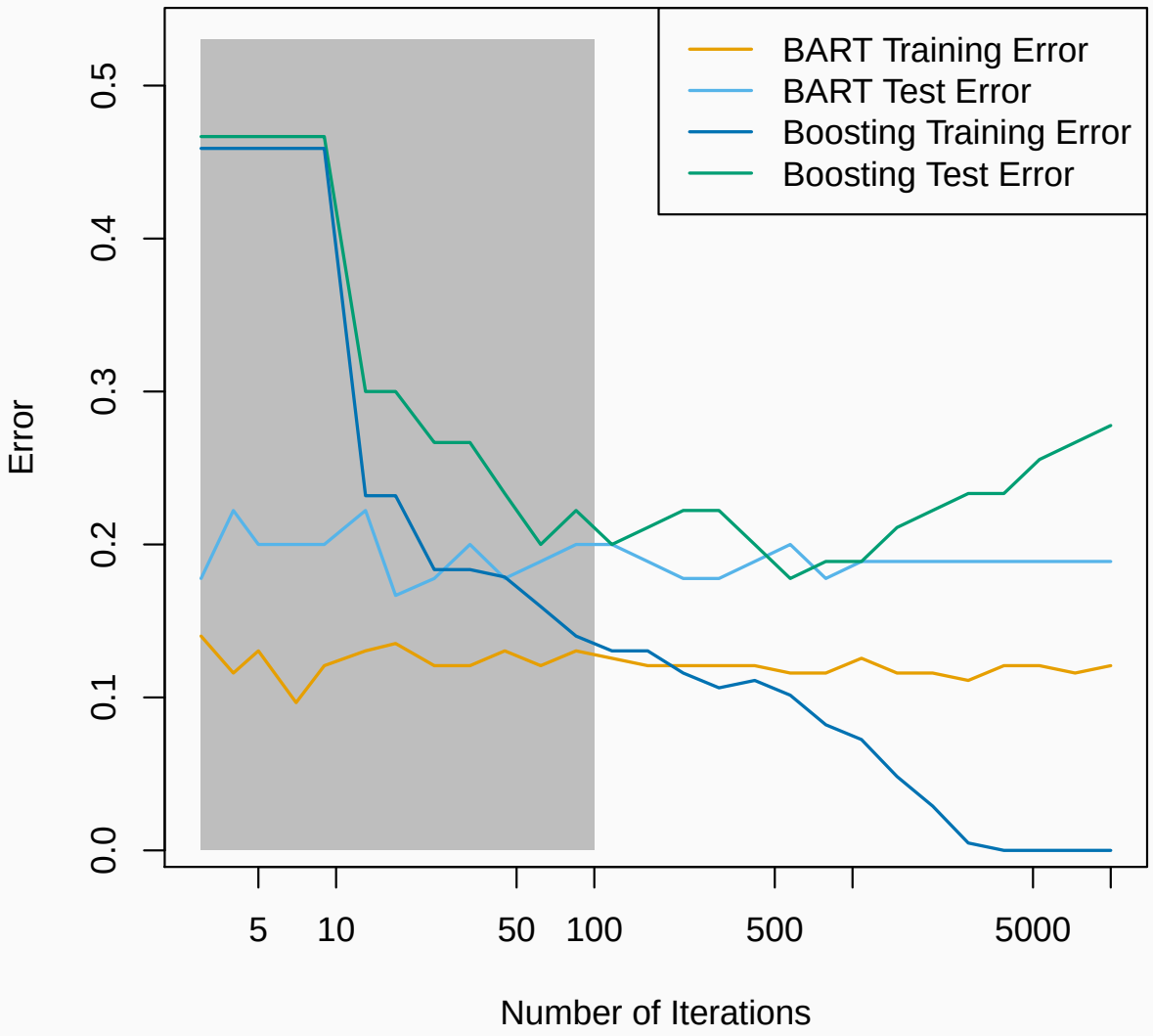
$$\hat{f}^{(b)}(x) = \sum_{k=1}^{K} \hat{f}_k^{(b)}(x), \quad b = 1, \ldots, B.$$

- Final prediction after *burn-in* $L$:

$$\hat{f}(x) = \frac{1}{B-L} \sum_{b=L+1}^{B} \hat{f}^{(b)}(x).$$

- Perturbations limit overfitting by preventing overly complex fits at each step.
- Beyond mean prediction, BART provides *uncertainty* estimates.

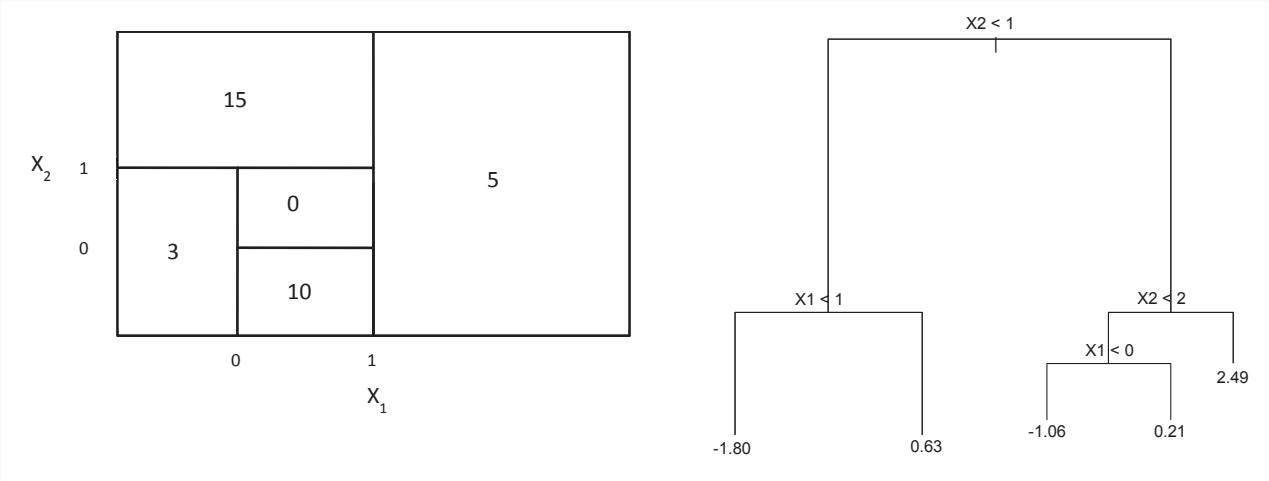
# BART Applied to Heart Data



# BART as a Bayesian Method

- BART = Bayesian ensemble of trees.
- Random perturbations $\Rightarrow$ draw new trees from *posterior* distribution.
- Algorithm can be viewed as a *Markov chain Monte Carlo (MCMC)* procedure.
- Typical choices: $K = 200$, $B = 1000$, burn-in $L = 100$.
- Strong out-of-the-box performance with minimal tuning.

# Exercises 1




(a) Sketch the tree for the partition in the left panel. The numbers show the mean of $Y$ in each region.

(b) Create a partition diagram (like the left panel) for the tree in the right panel, labeling the means.

# Exercises 2

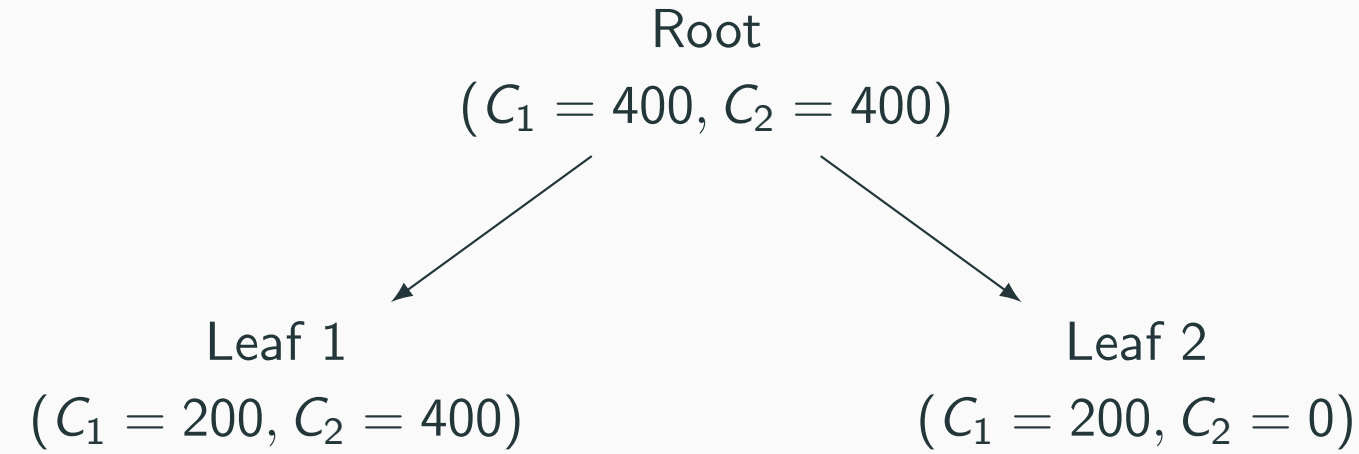
Suppose we generate ten bootstrap samples from a data set with red and green classes. For a given $X$, the classification trees yield these estimates of $P(\text{Red} \mid X)$:

$$0.1,\ 0.15,\ 0.2,\ 0.2,\ 0.55,\ 0.6,\ 0.6,\ 0.65,\ 0.7,\ 0.75.$$

- What is the final classification using **majority vote**?
- What is the final classification using the **average probability**?

# Exercises 3

Consider a dataset with 800 observations, with 400 observations from class $C_1$ and 400 observations from class $C_2$. A decision tree partitions them as follows:




Compute the following quantities of this tree for Leaf 1 and Leaf 2:

1. The **classification error rate**: $E = 1 - \max_k(\hat{p}_{mk})$,
2. The **cross-entropy**: $D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$,
3. The **Gini index**: $G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$.

where $\hat{p}_{mk}$ is proportion of observation in leaf $m$ from class $k$.