Chapter 8 – Tree-Based Methods

Wenjing Liao

School of Mathematics
Georgia Institute of Technology

Math 4803
Fall 2019

Outline

Predict the log salary of a baseball player

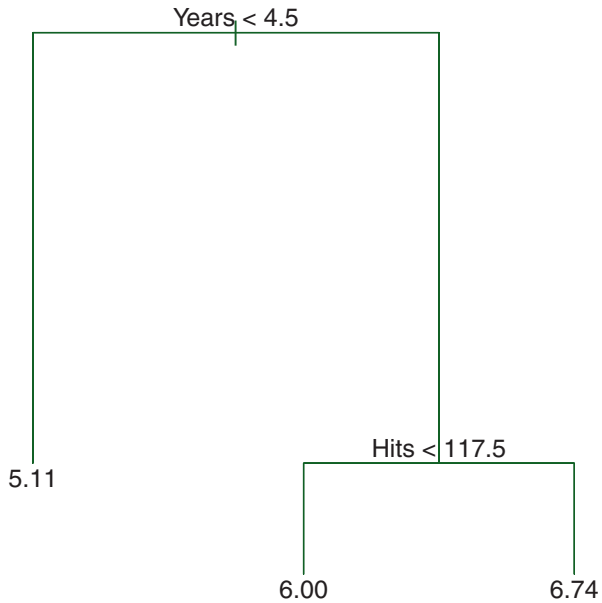

FIGURE 8.1. *For the* `Hitters` *data, a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year. At a given internal node, the label (of the form* $X_j < t_k$*) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to* $X_j \geq t_k$*. For instance, the split at the top of the tree results in two large branches. The left-hand branch corresponds to* `Years<4.5`*, and the right-hand branch corresponds to* `Years>=4.5`*. The tree has two internal nodes and three terminal nodes, or leaves. The number in each leaf is the mean of the response for the observations that fall there.*

Regions of partition

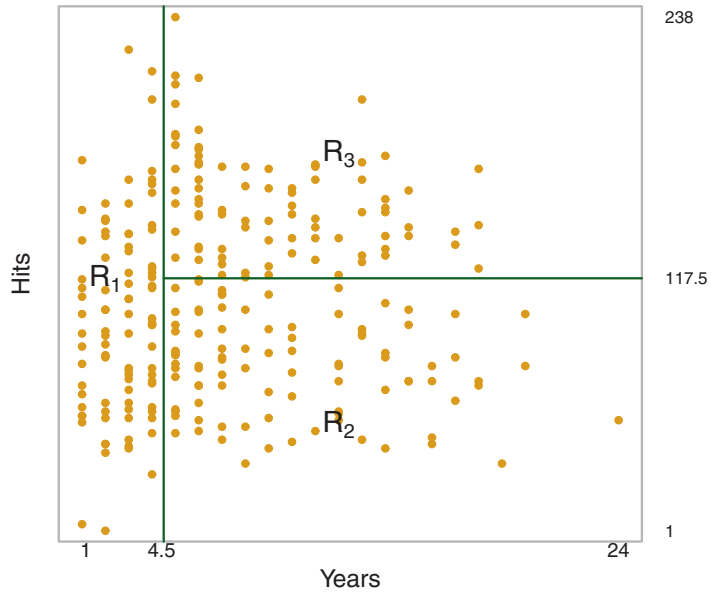

FIGURE 8.2. *The three-region partition for the* `Hitters` *data set from the regression tree illustrated in Figure 8.1.*

$$R_1 = \{X|\text{Years} < 4.5\},\ \hat{Y} = \$1,000 \times e^{5.107} = 165,174$$
$$R_2 = \{X|\text{Years} \geq 4.5,\ \text{Hits} < 117.5\},\ \hat{Y} = \$1,000 \times e^{5.999} = 402,834$$
$$R_3 = \{X|\text{Years} \geq 4.5,\ \text{Hits} \geq 117.5\},\ \hat{Y} = \$1,000 \times e^{6.740} = 845,346$$

Prediction via stratification of the feature space

1. We divide the predictor space—that is, the set of possible values for $X_1, X_2, \ldots, X_p$—into J distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.

2. For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j.

How to construct the regions $R_1, \ldots, R_J$?

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

where $\hat{y}_{R_j}$ is the mean response for the training data within the jth box.

- The optimization above is computationally infeasible.
- Instead we take a *top-down, greedy approach* that known as *recursive binary splitting*.

Recursive binary splitting

1. Define the pair of half planes

$$R_1(j, s) = \{X | X_j < s\} \text{ and } R_2(j, s) = \{X | X_j \geq s\}$$

2. Seek the value of j and s that minimizes the equation

$$\sum_{i:\, x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2,$$

where $\hat{y}_{R_1}$ is the mean response for the training data in $R_1(j, s)$, and $\hat{y}_{R_2}$ is the mean response for the training data in $R_2(j, s)$

Repeat this process, looking for the best predictor and best cutpoint in order to split the data further so as to minimize the RSS within each of the resulting regions.

Example

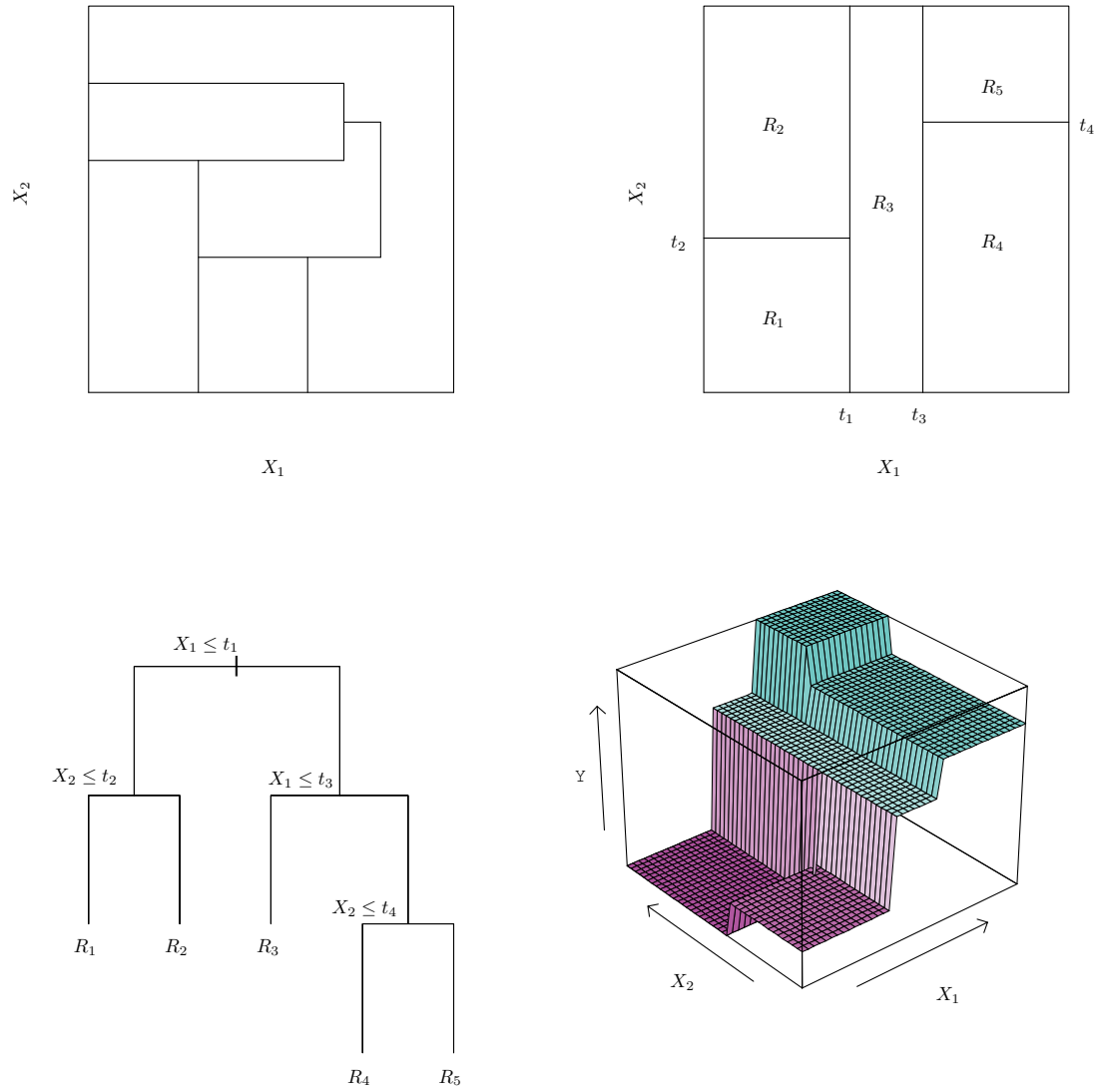

FIGURE 8.3. Top Left: *A partition of two-dimensional feature space that could not result from recursive binary splitting.* Top Right: *The output of recursive binary splitting on a two-dimensional example.* Bottom Left: *A tree corresponding to the partition in the top right panel.* Bottom Right: *A perspective plot of the prediction surface corresponding to that tree.*

Tree pruning

- Pruning is to avoid overfitting to the training data.
- How to prune the tree? – select a subtree that leads to the lowest test error.

Cost complexity pruning: For each tuning parameter $\alpha > 0$, find a subtree $T \subset T_0$ such that

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

is as small as possible. Here $|T|$ indicates the number of terminal nodes in T, and R_m is the rectangle corresponding to the mth terminal node.

- When $\alpha = 0$, $T = T_0$;
- As α increases, the minimizer tend to be a smaller subtree.

Building a regression tree

Algorithm 8.1 *Building a Regression Tree*

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 choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:

   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

   Average the results for each value of α, and pick α to minimize the average error.

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

Example: the unpruned tree for the Hitters data

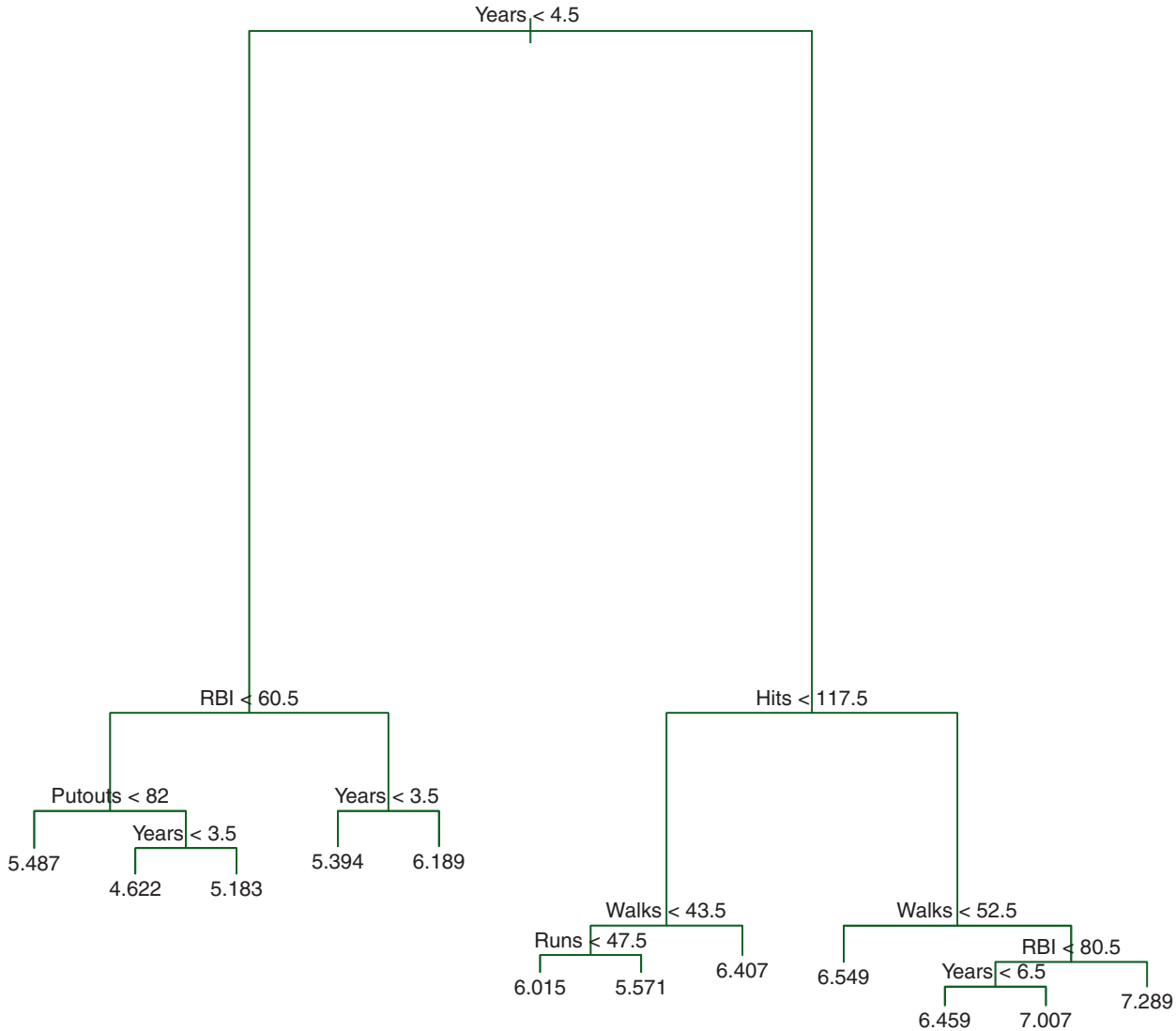

FIGURE 8.4. *Regression tree analysis for the* `Hitters` *data. The unpruned tree that results from top-down greedy splitting on the training data is shown.*

Pruning

- 132 observations for training and 131 observations for test;
- Build a large tree on training data and prune the tree with varied α;
- Perform six-fold cross validation as a function of α.

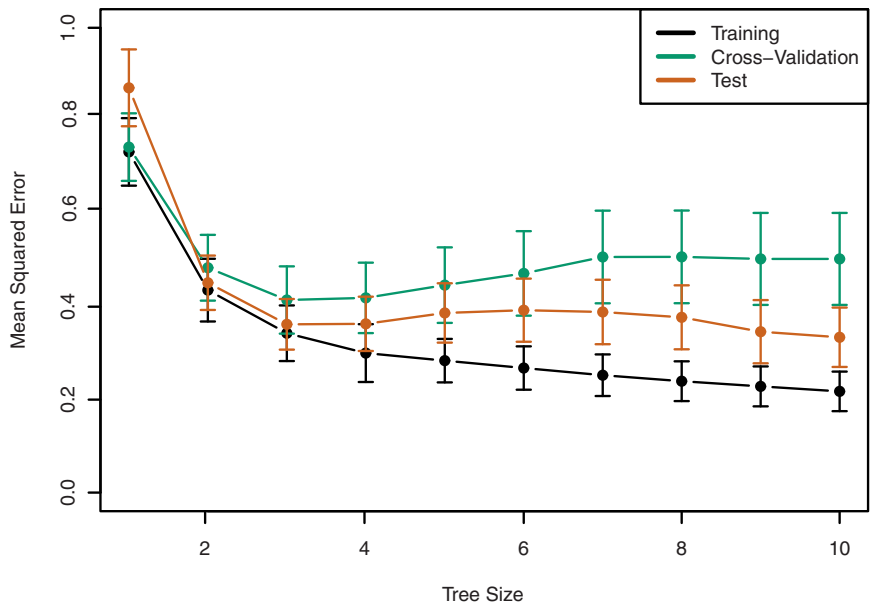

FIGURE 8.5. *Regression tree analysis for the* `Hitters` *data. The training, cross-validation, and test MSE are shown as a function of the number of terminal nodes in the pruned tree. Standard error bands are displayed. The minimum cross-validation error occurs at a tree size of three.*

Outline

How to predict?

Prediction rule: Each observation belongs to the **most commonly occurring class** of training observation in the region to which it belongs.

Classification error rate: the fraction of the training observations in that region that do not belong to the most common class

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

where $\hat{p}_{mk}$ represents the proportion of training observations in the mth region that are from the kth class.

- The classification error is not sufficiently sensitive for tree-growing.

Preferable criterion for the error

Gini index:

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

- It measures the total variance across the K classes.
- The Gini index is small if all of the $\hat{p}_{mk}$'s are close to 0 or 1 – a small value indicates that a node contains predominantly observations from a single class.

Entropy:

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

- $0 \leq \hat{p}_{mk} \leq 1 \Longrightarrow -\hat{p}_{mk} \log \hat{p}_{mk} \geq 0$
- $-\hat{p}_{mk} \log \hat{p}_{mk}$ is small if $\hat{p}_{mk}$ is close to 0 or 1 – the mth node is pure.

Build a classification tree

Training:

- Binary splitting
- Evaluate the quality of a particular split by the Gini index or entropy.
- Pruning the tree with the Gini index or entropy.

Prediction accuracy on test data: use the classification error rate

An example: Heart data

- a binary outcome "HD" for 303 patients
- 13 predictors including "age", "sex", "chol"

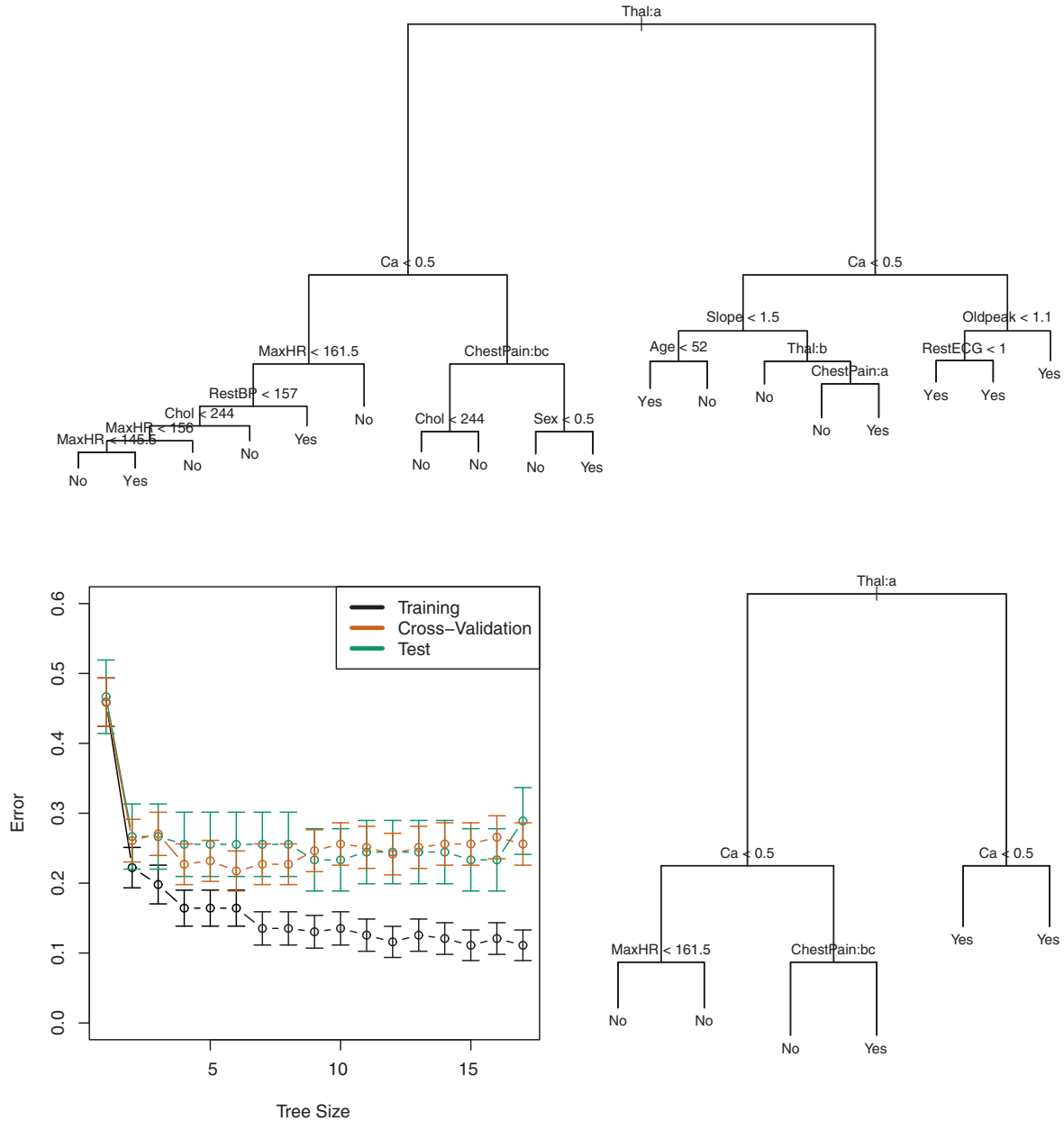

FIGURE 8.6. `Heart` *data.* Top: *The unpruned tree.* Bottom Left: *Cross -validation error, training, and test error, for different sizes of the pruned tree.* Bottom Right: *The pruned tree corresponding to the minimal cross-validation error.*

8.1.3 - Tree versus linear models

Linear regression:

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j,$$

- Linear regression works well if the relation between the response and the predictors are linear.

Regression tree:

$$f(X) = \sum_{m=1}^{M} c_m \cdot 1_{(X \in R_m)}$$

Example

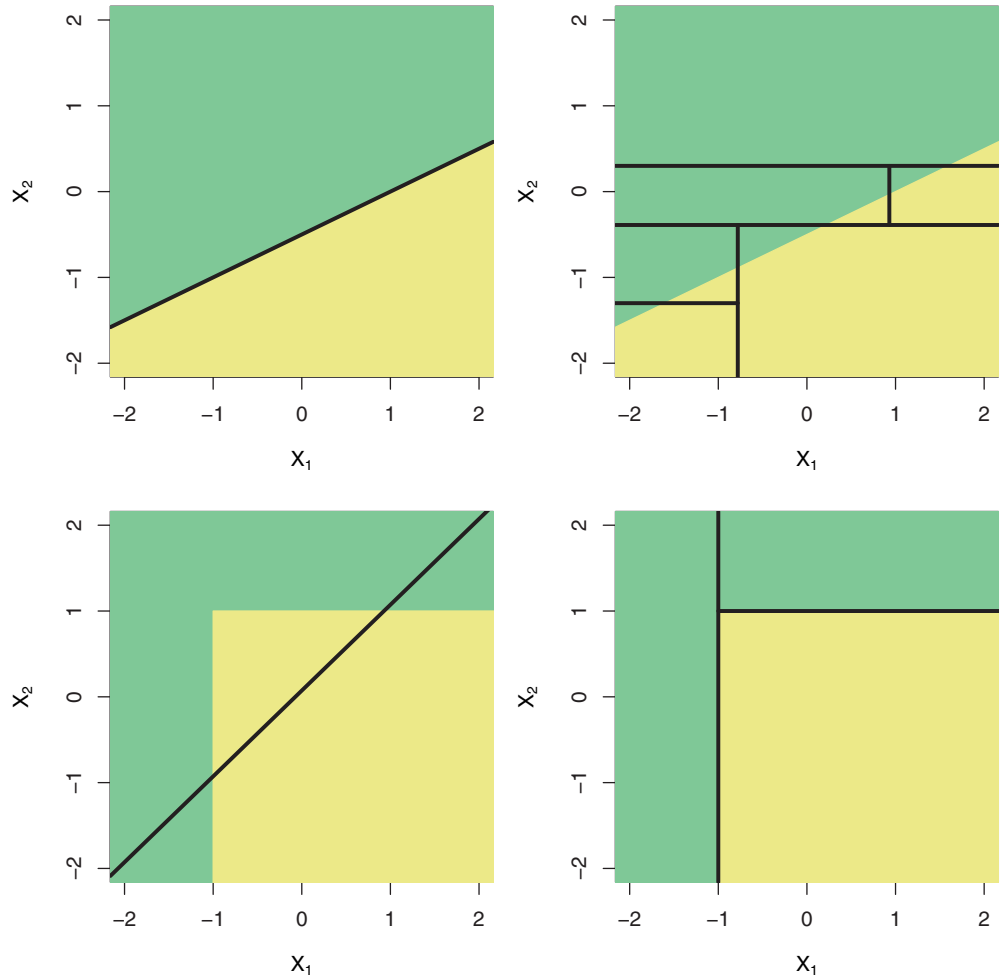

FIGURE 8.7. Top Row: *A two-dimensional classification example in which the true decision boundary is linear, and is indicated by the shaded regions. A classical approach that assumes a linear boundary (left) will outperform a decision tree that performs splits parallel to the axes (right).* Bottom Row: *Here the true decision boundary is non-linear. Here a linear model is unable to capture the true decision boundary (left), whereas a decision tree is successful (right).*

Outline

Basics

High variance: if we split the training data into two parts at random and fit a tree to both halves, the results could be quite different.

Average: average multiple training results

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

where $\hat{f}^{1}(x), \ldots, \hat{f}^{B}(x)$ are calculated via B separate training sets.

Bagging:

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

where each $\hat{f}^{*b}(x)$ is built on a subset of the training data.

Bagging for classification problems: take a majority vote – the overall prediction is the most commonly occurring class among the B predictors.

Variance important measures

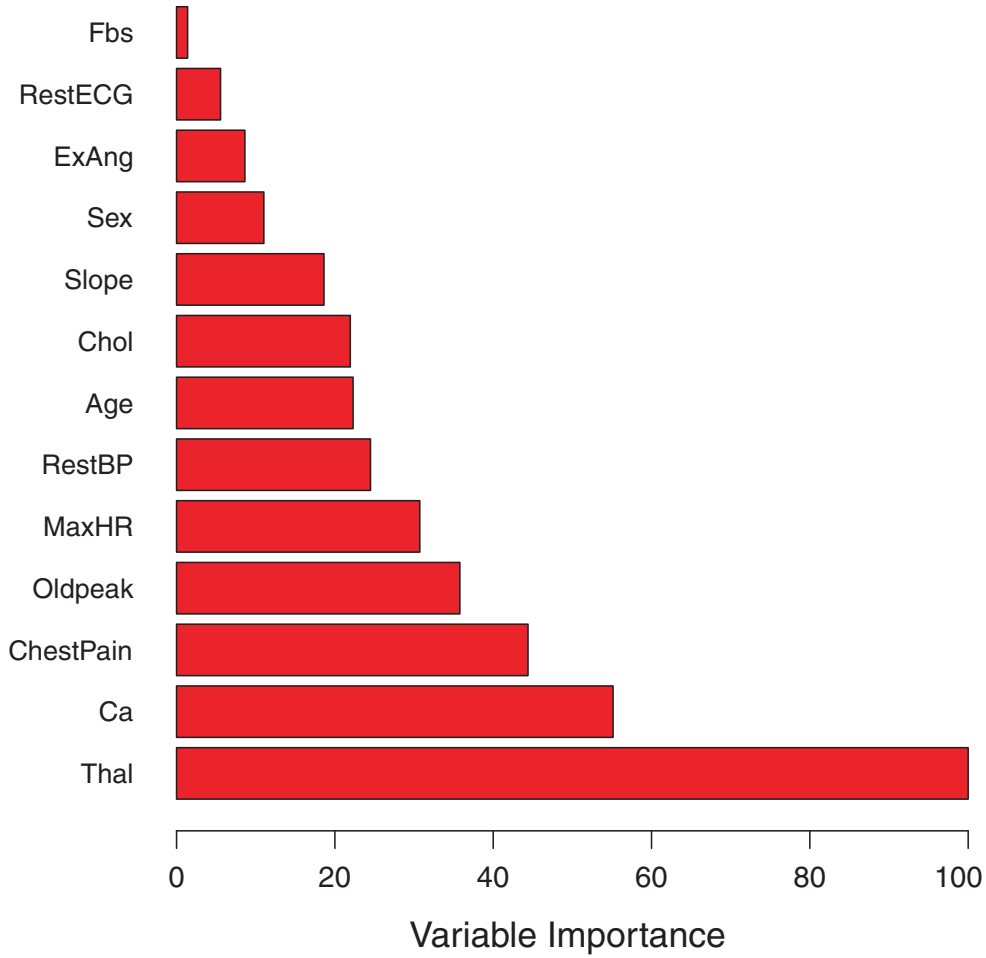

FIGURE 8.9. *A variable importance plot for the* `Heart` *data. Variable importance is computed using the mean decrease in Gini index, and expressed relative to the maximum.*

Random forests

Splitting: Each time a random sample of m predictors is chosen as split candidates from the full set of p predictors.

$$m \approx \sqrt{p}$$

Improvements:

- If there is a strong predictor, all bagged trees will use this strong predictor at the top split, making all bagged trees high correlated.
- Random forests consider m predictors at a time, and the other $p - m$ ones are not used at all.

Decorrelating the trees

Choice of m

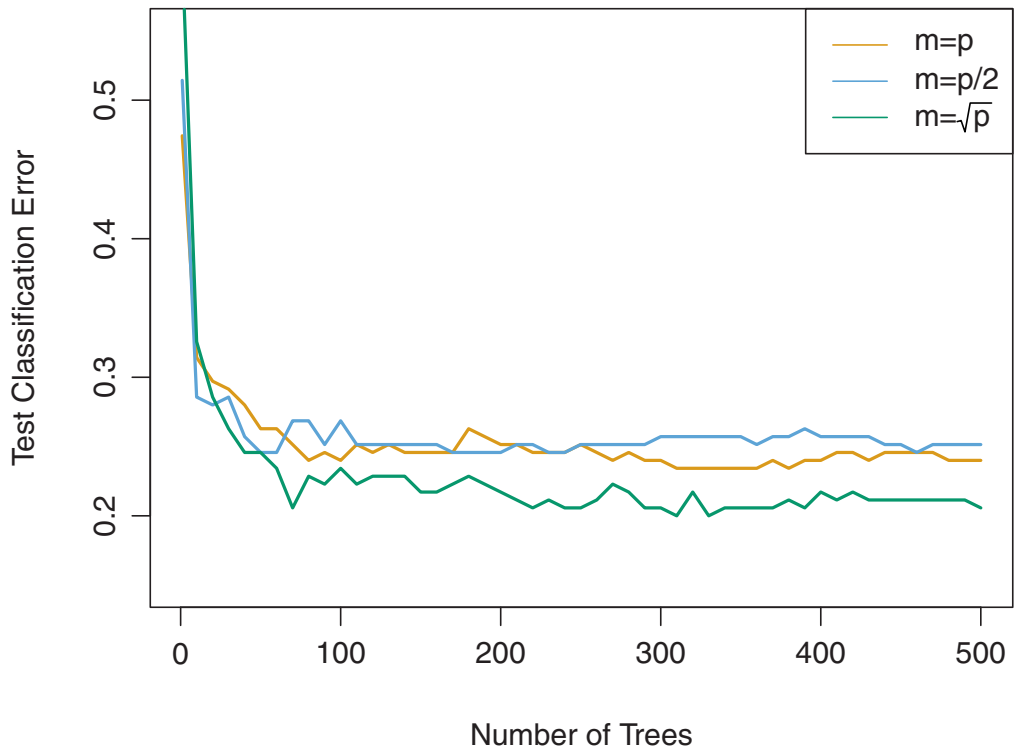

FIGURE 8.10. *Results from random forests for the 15-class gene expression data set with $p = 500$ predictors. The test error is displayed as a function of the number of trees. Each colored line corresponds to a different value of m, the number of predictors available for splitting at each interior tree node. Random forests ($m < p$) lead to a slight improvement over bagging ($m = p$). A single classification tree has an error rate of 45.7%.*

How many trees?

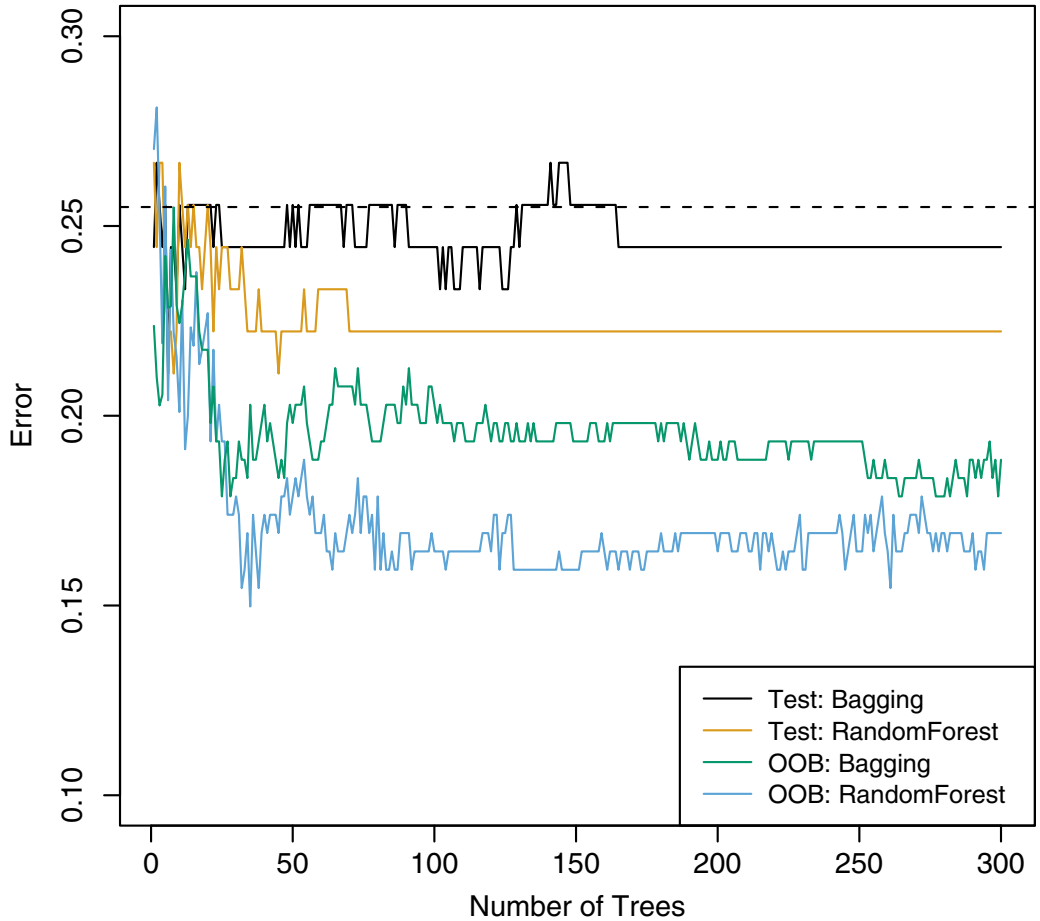

FIGURE 8.8. *Bagging and random forest results for the* `Heart` *data. The test error (black and orange) is shown as a function of* B*, the number of bootstrapped training sets used. Random forests were applied with* $m = \sqrt{p}$*. The dashed line indicates the test error resulting from a single classification tree. The green and blue traces show the OOB error, which in this case is considerably lower.*

Reference

Chapter 8: James, Gareth, Daniela Witten, Trevor Hastie and Robert Tibshirani, An introduction to statistical learning. Vol. 112, New York: Springer, 2013.

Wikipedia:
`https://en.wikipedia.org/wiki/Decision_tree_learning`