Tree-Based Methods

ddebarr@uw.edu

2017-02-23

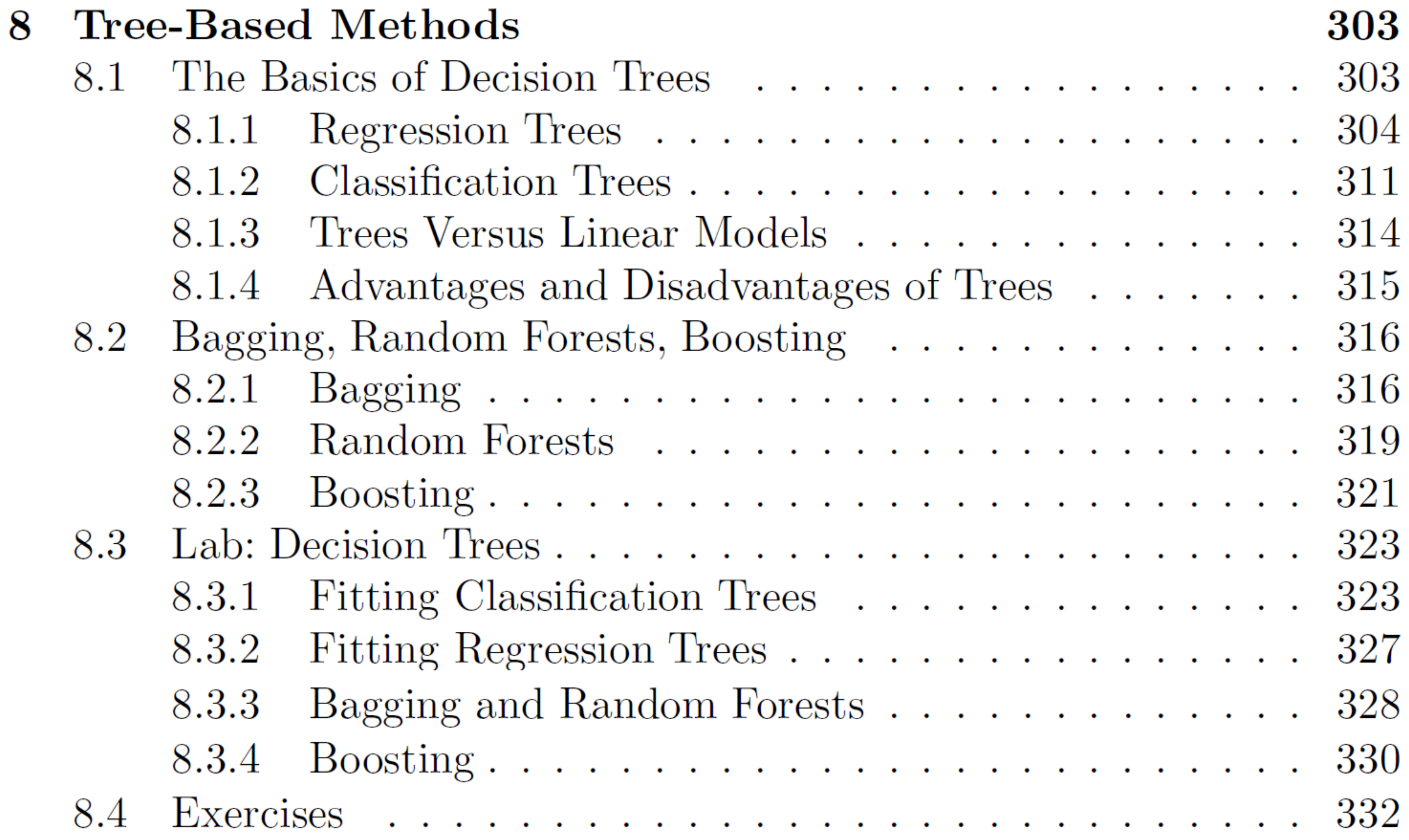
THE #1 deep learning expert EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY model's training"
HEY! GET BACK TO WORK!
training!
OH. CARRY ON.

https://www.reddit.com/r/MachineLearning/comments/4hfift/deep_learning_humor/

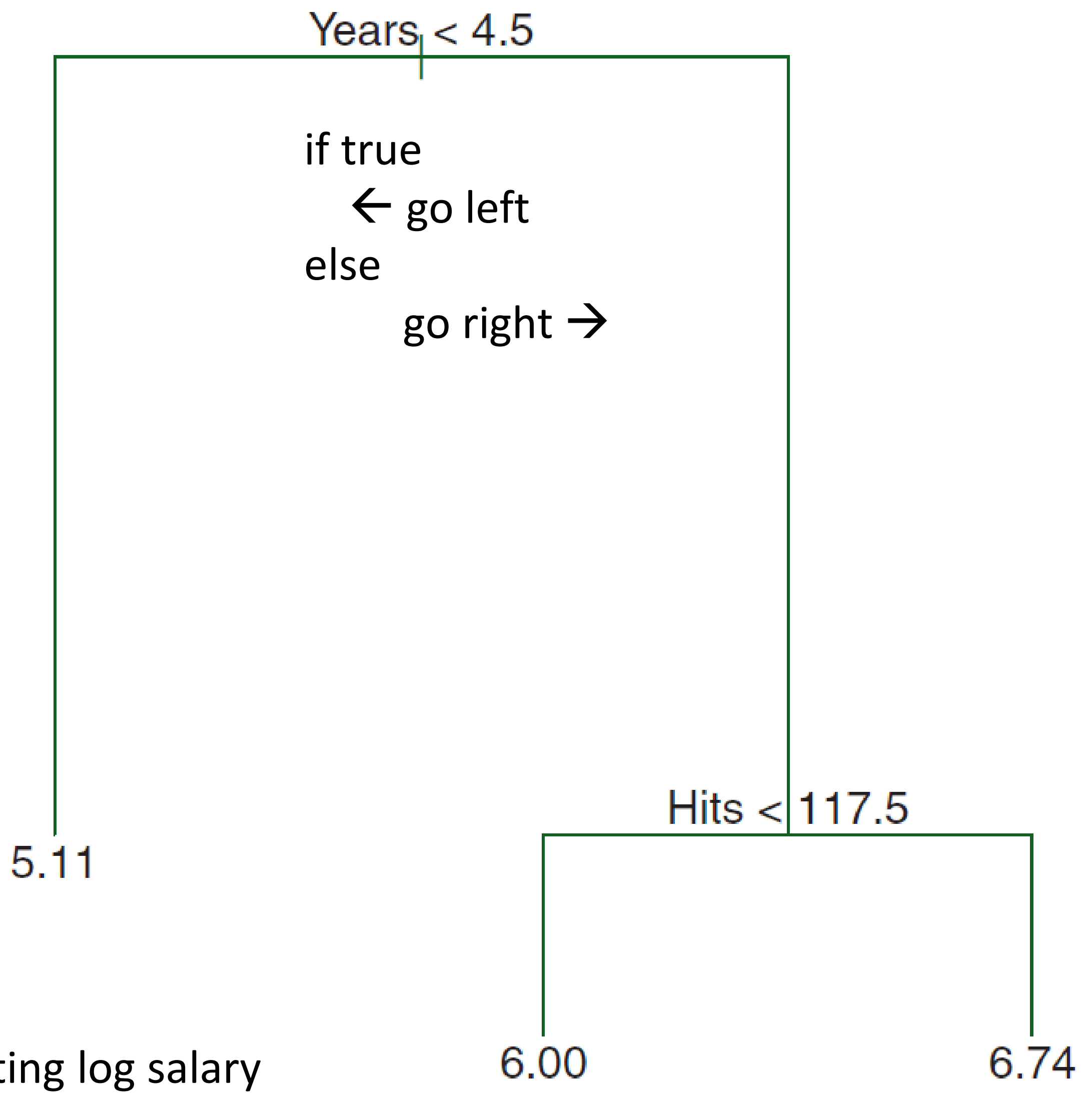
Course Outline

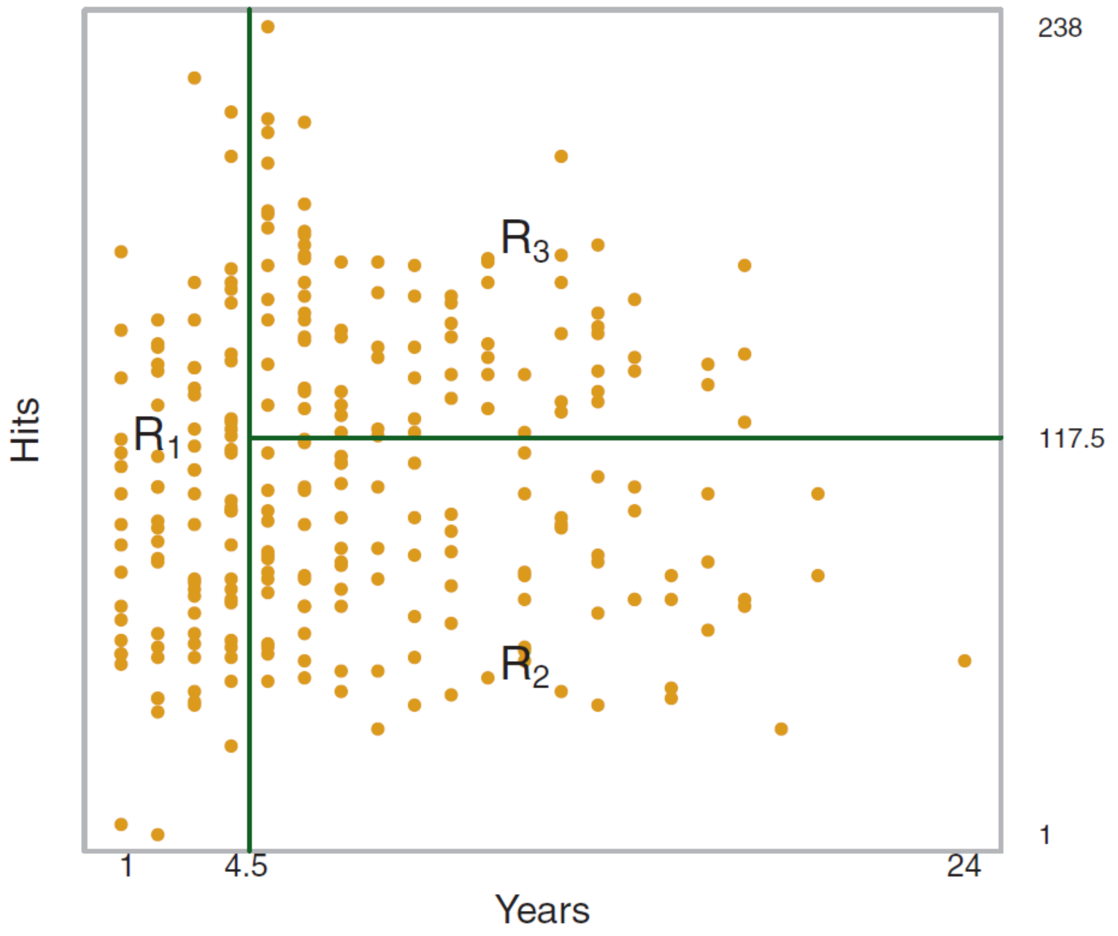
1. Introduction to Statistical Learning
2. Linear Regression
3. Classification
4. Resampling Methods
5. Linear Model Selection and Regularization
6. Moving Beyond Linearity
7. $\boxed{\text{Tree-Based Methods}}$
8. Support Vector Machines
9. Unsupervised Learning
10. Neural Networks and Genetic Algorithms

Agenda



Regression Tree to Predict Baseball Salary





predicting log salary

Steps for Constructing a Regression Tree

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.

Based on using the Residual Sum of Squares for our loss function: $\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$

For numeric features …

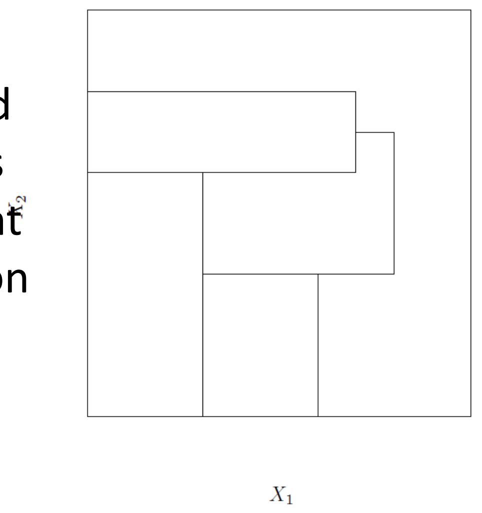
We seek the value of “j” and “s” that minimize the RSS for a split …

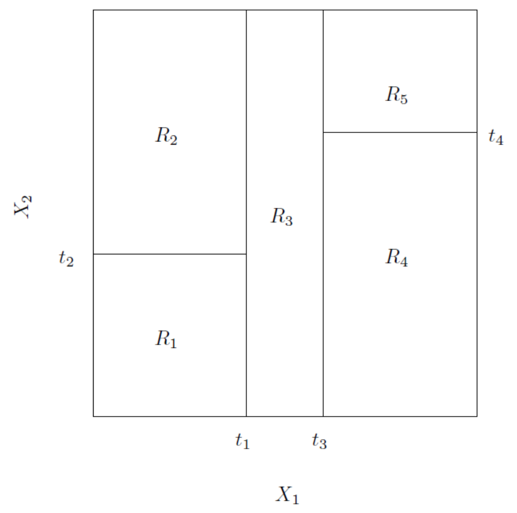
$$R_1(j,s) = \{X|X_j < s\} \text{ and } R_2(j,s) = \{X|X_j \geq s\}$$

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

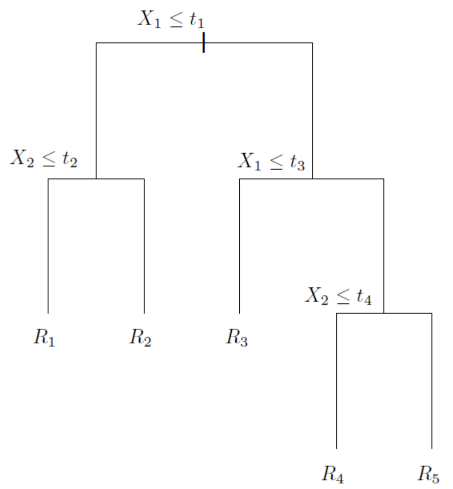
Examples of Partitioning

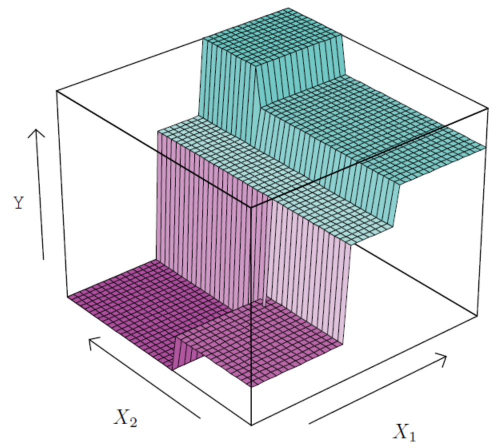
Example *not* produced by recursive binary splits [a binary split is a straight line that splits one region into two]



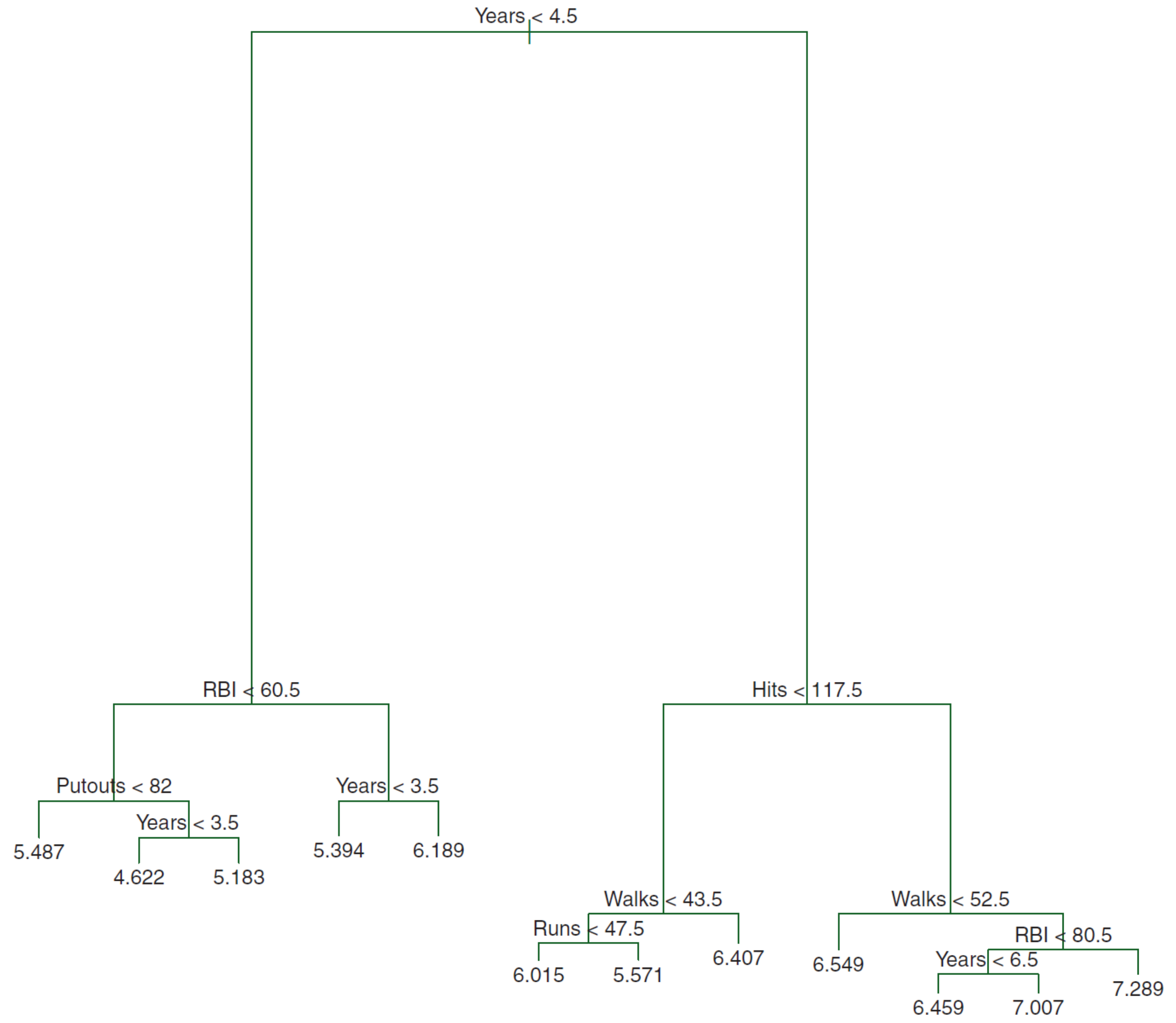


Example produced by recursive binary splits





Unpruned Tree for the Hitters Data



Complexity Pruning

- One possible strategy to avoid overfitting is to build 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 since a seemingly worthless split early on in the tree might be followed by a very good split.
- Complexity pruning is a viable alternative: pruning nodes when the following expression is reduced …

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

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 $k = 1, \ldots, K$:

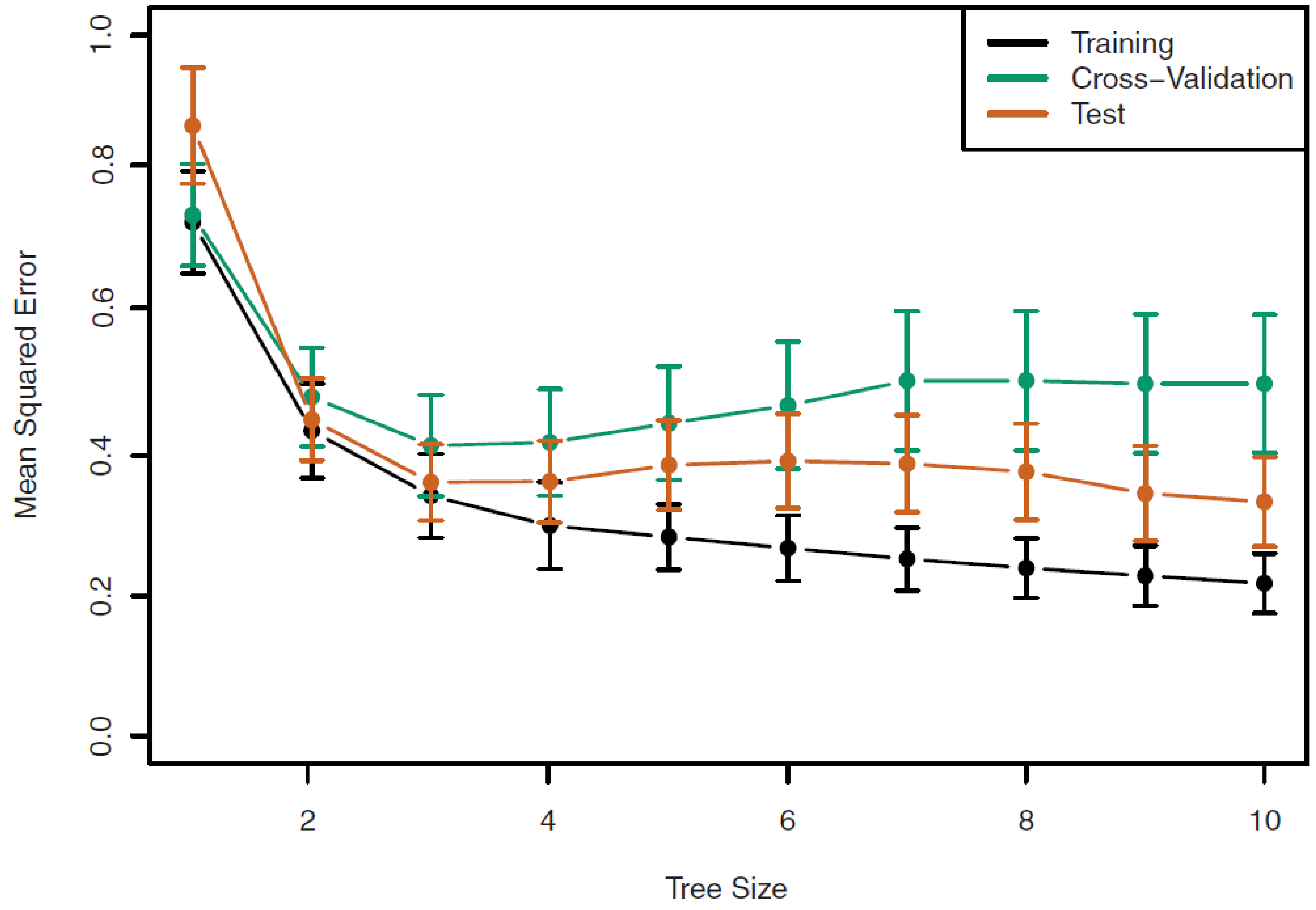
   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

   (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.

Regression Tree Analysis for the Hitters Data



Classification Trees

Splitting criteria for classification trees (our loss function) …

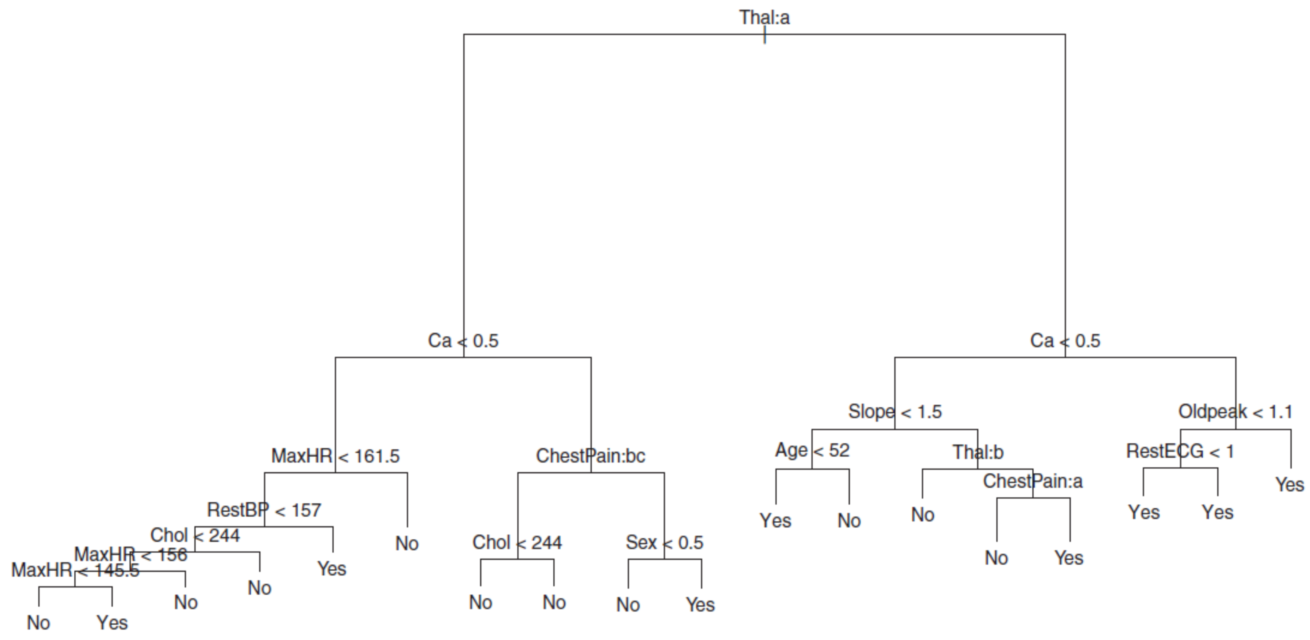
$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

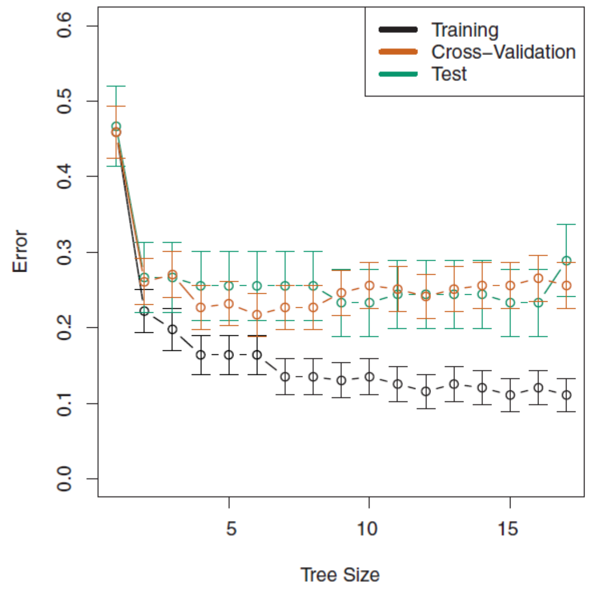
The Gini index
[a measurement of uncertainty;
compare to the variance of a
Bernoulli random variable]

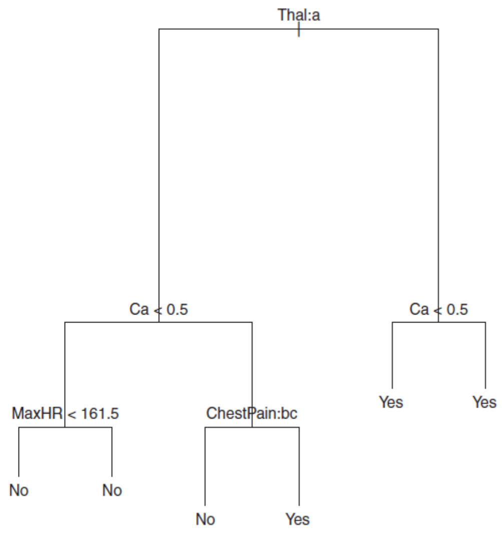
$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$$

Entropy
[an alternative to the Gini index;
refer to this as entropy
(*not* Cross Entropy)]

Classification Tree to Predict Heart Disease



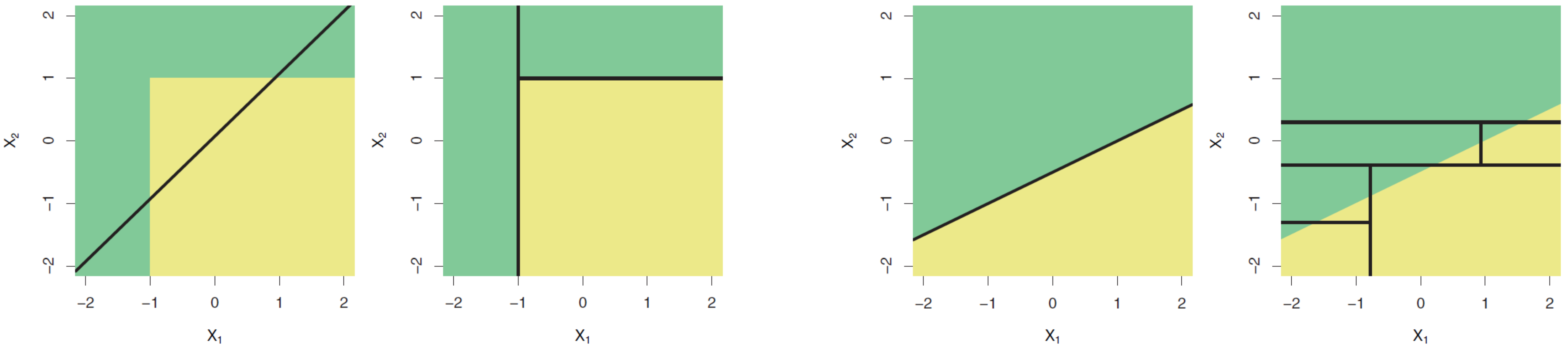




Tree versus Linear Model

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

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



No free lunch: no model always wins
“if an algorithm performs well on a certain class of problems
then it necessarily pays for that with degraded performance on the set of all remaining problems”

Advantages and Disadvantages of Trees

▲ 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 chapters.

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

▲ Trees can easily handle qualitative predictors without the need to create dummy variables.

▼ Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches seen in this book.

▼ Additionally, trees can be very non-robust. In other words, a small change in the data can cause a large change in the final estimated tree.

Bagging: Bootstrap Aggregation

- Each tree in an “ensemble” (a set of trees) is built from a bootstrap sample of the data

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

- This reduces variance (overfitting) by preventing the learning algorithm from using all the data

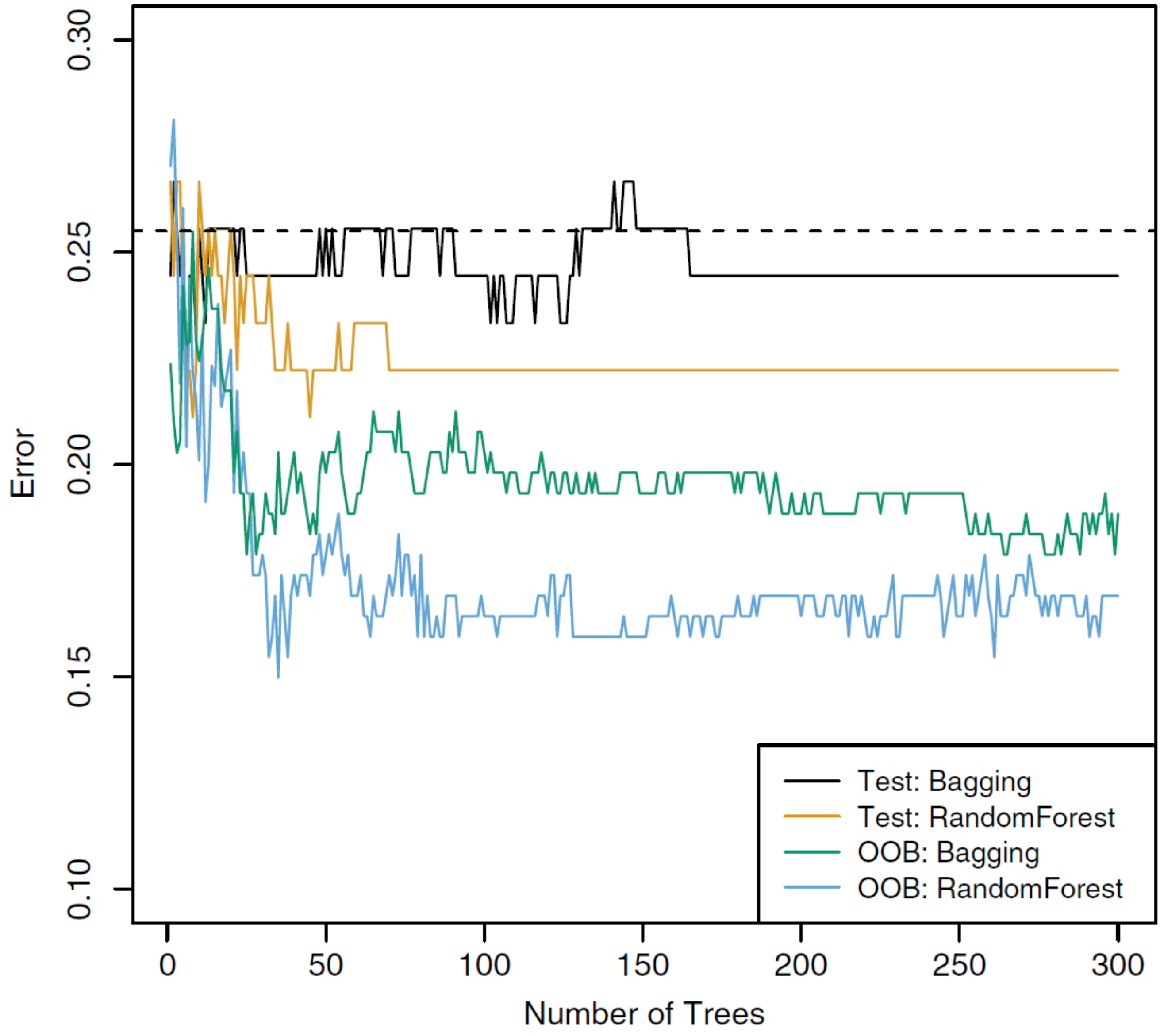
Out Of Bag (OOB) Error Estimates

- Recall that around 36.8% of the data will not be selected as part of a bootstrap sample
- We can estimate the error for each observation using the 0.368 * B trees where the observation was not used as part of the training data

Random Forest

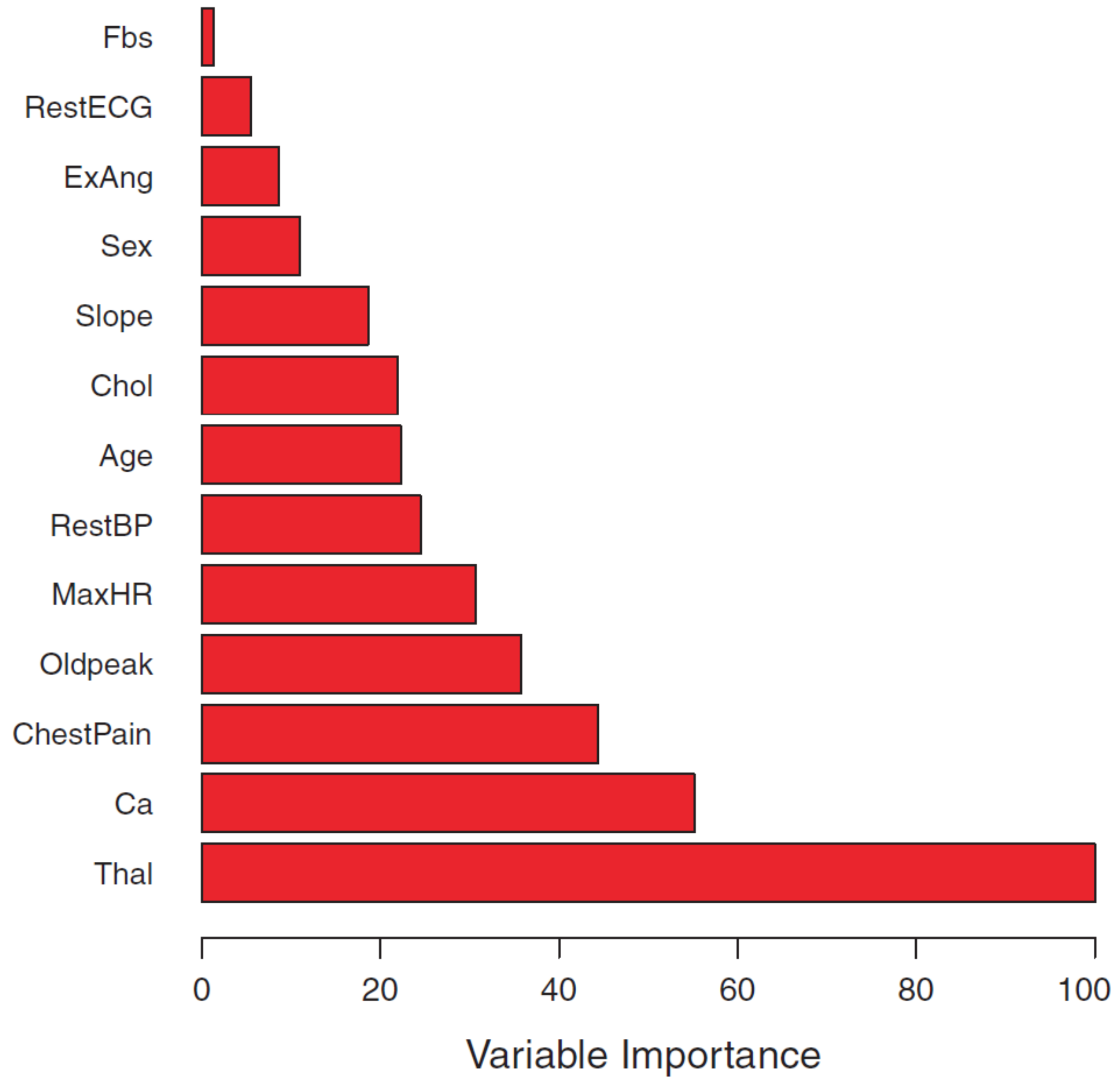
- Each tree in an "ensemble" (a set of trees) is built from a bootstrap sample of the data [so a Random Forest uses bagging]
- Only a randomly selected subset of predictors is considered when splitting a region [often, m = sqrt(p) is used as the predictor count]
- This helps to further reduce variance by promoting diversity among (decorrelating) the trees [for example, in bagging we might expect the best feature to appear at the top of every tree in the ensemble; but this would be unlikely to happen for a random forest]

Bagging versus Random Forest for the Heart Data

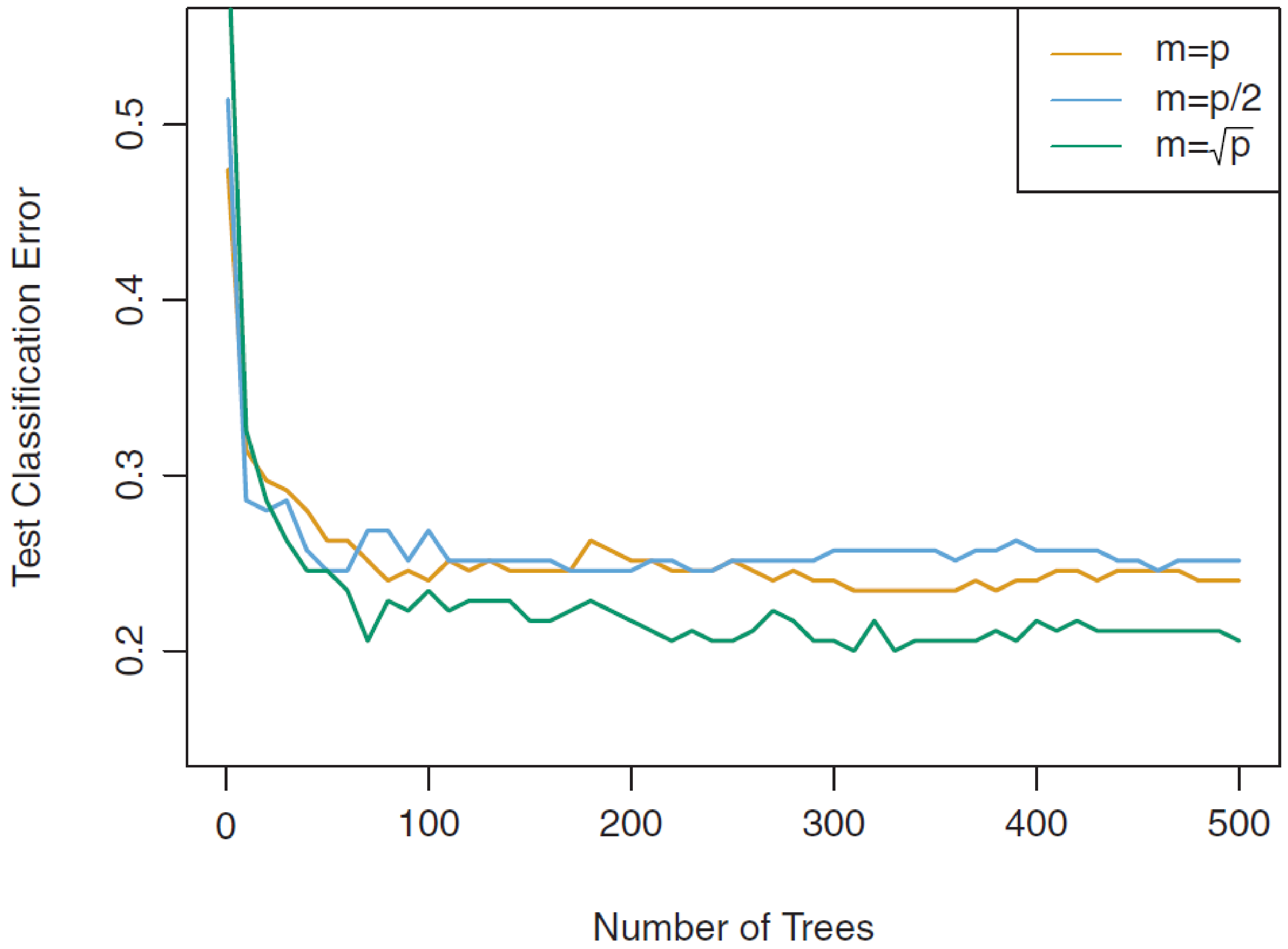


The dashed line denotes the error for a single tree

Variable Importance: Normalized Mean Gini Decrease



Random Forest for Predicting Cancer Type



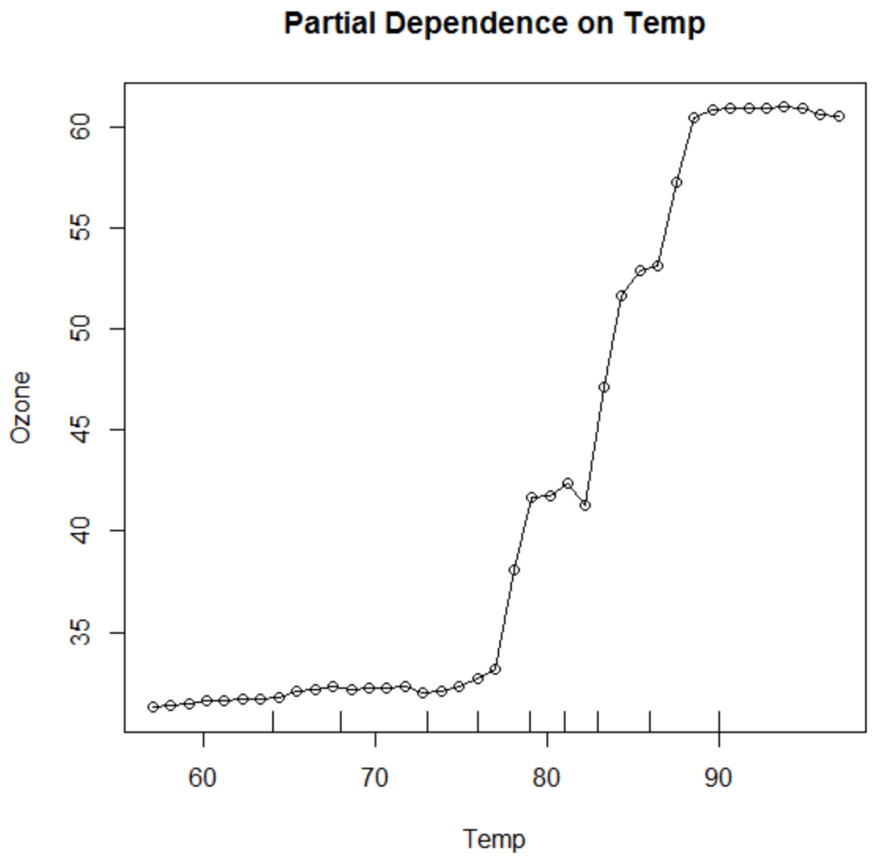
A single classification tree has an error rate of 0.457

Partial Dependence Plot

How does the expected prediction change as the value of a variable changes?

```
> library(randomForest)
randomForest 4.6-12
Type rfNews() to see new features/changes/bug fixes.
> data(airquality)
> airquality = na.omit(airquality)
> set.seed(2^17 - 1)
> model = randomForest(Ozone ~ ., data = airquality, importance = T)
> imp = importance(model)
> impvar = rownames(imp)[order(imp[,1], decreasing = T)]
> coordinates = partialPlot(model, pred.data = airquality, impvar[1],
+                           xlab = impvar[1], ylab = "Ozone",
+                           main = paste("Partial Dependence on", impvar[1]))


> averagePredictions = array(length(coordinates$x))
> for (i in 1:length(coordinates$x)) {
+     modified = airquality
+     modified$Temp = coordinates$x[i]
+     averagePredictions[i] = mean(predict(model, newdata = modified))
+ }
> points(coordinates$x, averagePredictions)
```



Boosting for Regression Trees

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   (a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).
   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{8.10}$$
   (c) Update the residuals,
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$
3. Output the boosted model,
$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

How does (8.11) relate to a gradient?

Boosting has [at least] 3 Tuning Parameters

1. The number of trees B. Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select B.
2. The shrinkage parameter λ, a small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem. Very small λ can require using a very large value of B in order to achieve good performance.
3. The number d of splits in each tree, which controls the complexity of the boosted ensemble. Often d = 1 works well, in which case each tree is a stump, consisting of a single split. In this case, the boosted stump ensemble is fitting an additive model, since each term involves only a single variable. More generally d is the interaction depth, and controls the interaction order of the boosted model, since d splits can involve at most d variables.

Agenda