Class 8: Tree-Based Methods

MSA 8150: Machine Learning for Analytics

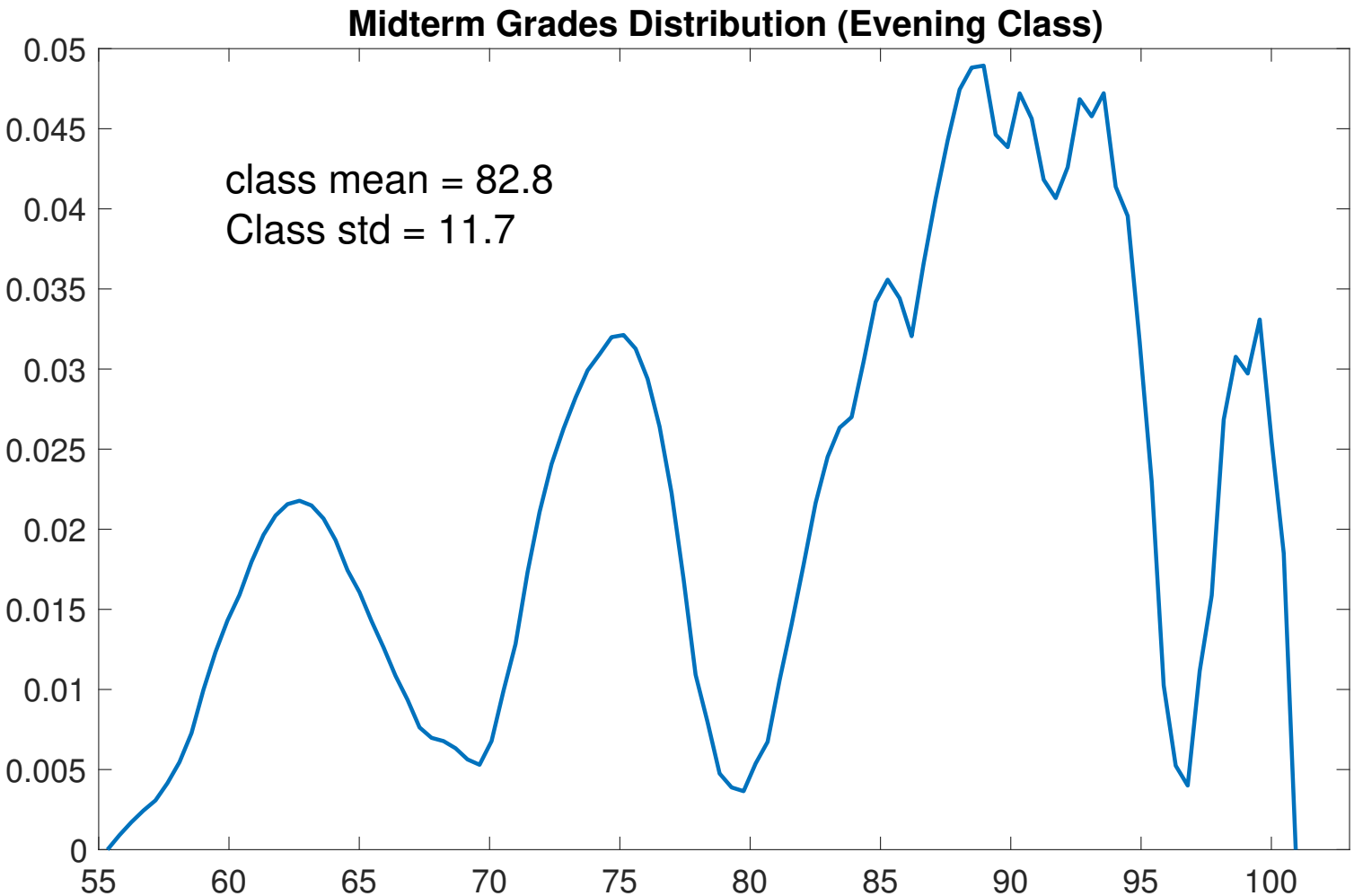
Alireza Aghasi

Institute for Insight, Georgia State University

Midterm Exam

Midterm Grade Distribution

- You did a good job!



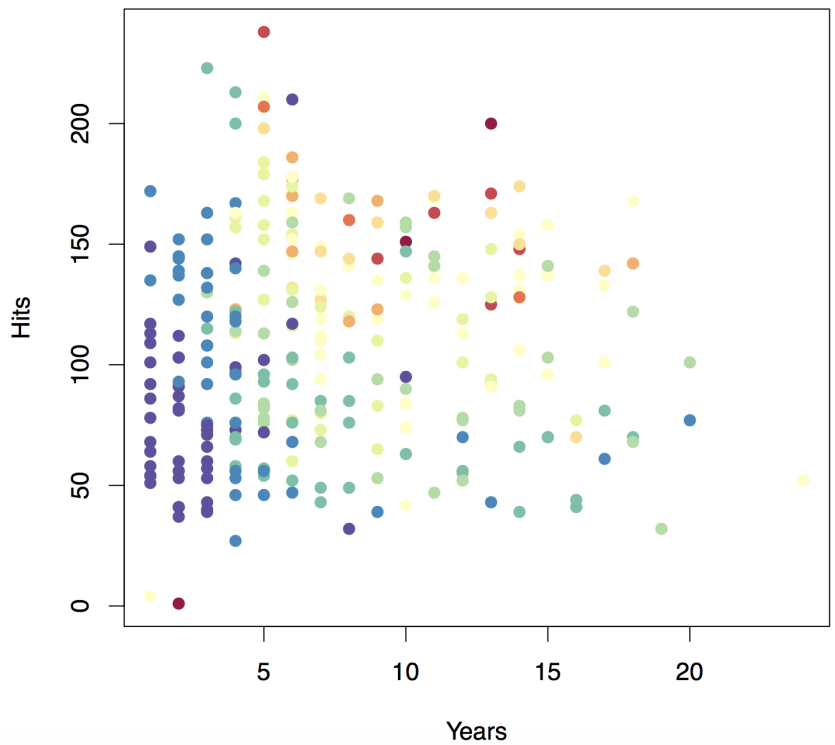
Introduction

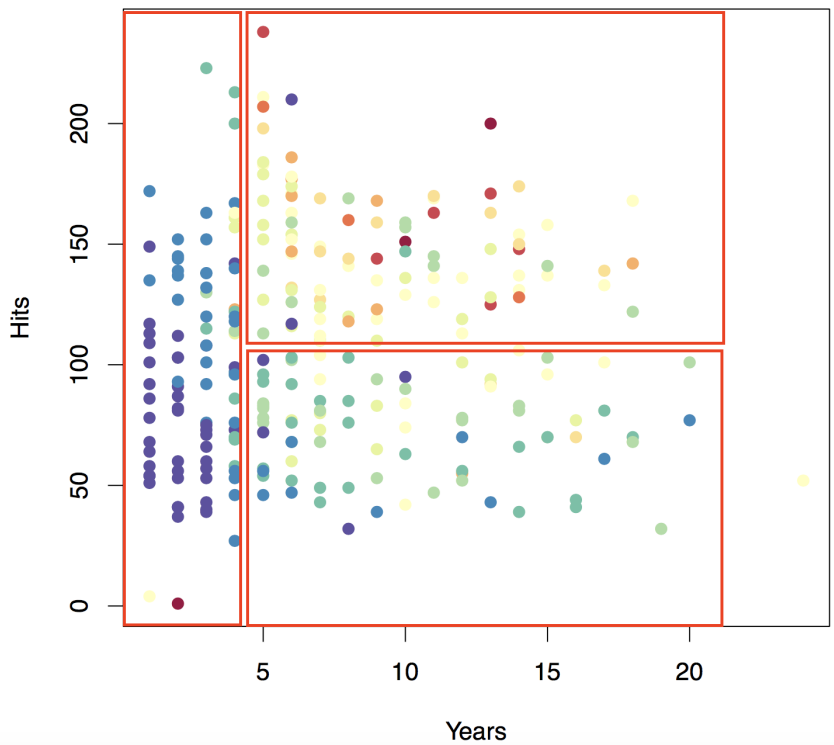
Tree-Based Methods

- Tree-based methods can be used for both regression and classification
- The main idea is to split the feature space into simpler regions
- Cons & Pros:
  - simple and easy to interpret
  - in their simple forms they do not compare well with other supervised learning techniques
  - using some randomization and boot-strapping technique their accuracy can significantly improve and even exceed other techniques

Regression & Decision Trees

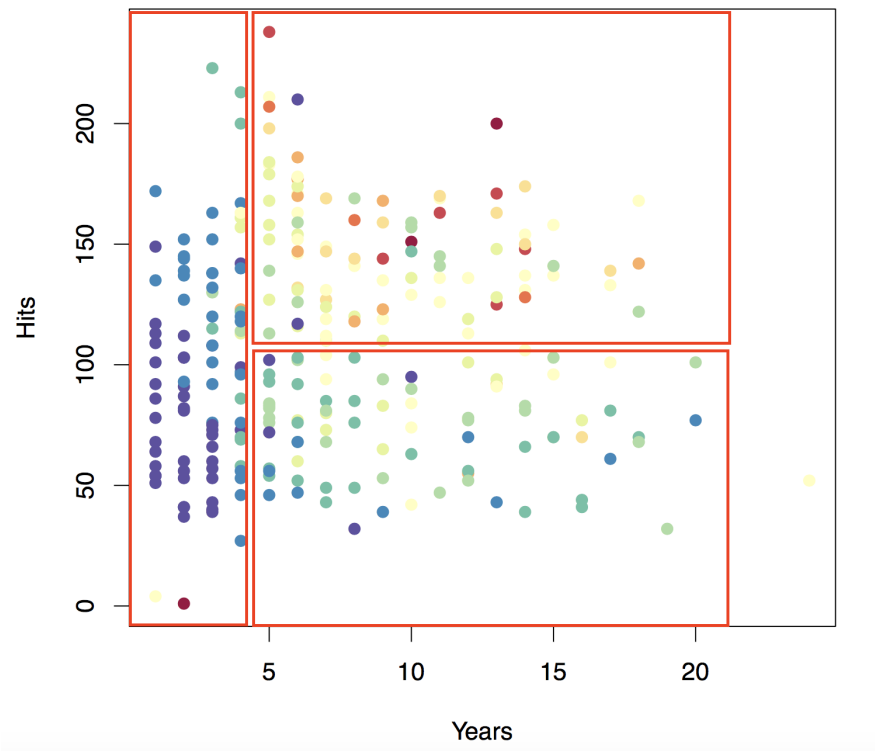
- Baseball salary data (salary in color vs two features)





Regression & Decision Trees

- Baseball salary data (salary in color vs two features)



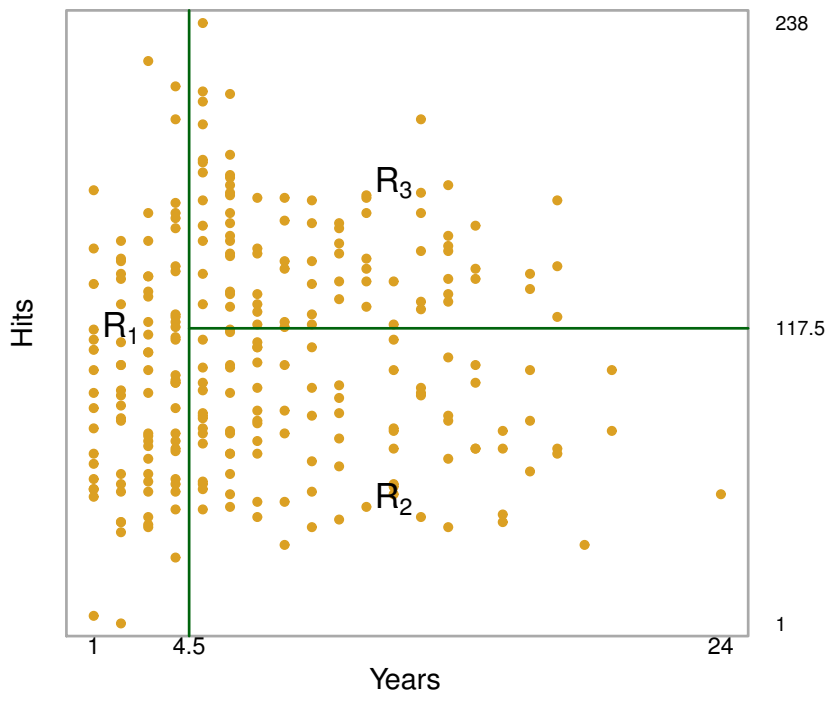
Regression & Decision Trees

- At each node ($X_j < a$) indicates the left branch as the corresponding path, and the right-hand branch corresponds to $X_j \geq a$

- The tree has three terminal nodes, or leaves. The number in each leaf is the mean of the response for the observations that fall there

Regression & Decision Trees

- $R_1 = \{\boldsymbol{X} : \mathit{years} < 4.5\}$
- $R_2 = \{\boldsymbol{X} : \mathit{years} \geq 4.5, \mathit{Hits} < 117.5\}$
- $R_3 = \{\boldsymbol{X} : \mathit{years} \geq 4.5, \mathit{Hits} \geq 117.5\}$



Few Definitions

- Regions R_1, R_2, R_3 each of which can be represented by one terminal node of the tree are called **leaves** (leaf)
- The points along the tree where the predictor space is split are referred to as **internal nodes**
- As an example, in our example the two internal nodes are indicated by the text points on the graph where $\textit{Years} < 4.5$ and $\textit{Hits} < 117.5$

Simple Interpretation

- Players with less years earn less salary
- For players who have been in the leagues for five or more years, the number of Hits made in the previous year does increase Salary
- This simple interpretation is a major advantage of decision trees, although the output values only sweep three values unlike more flexible regression models

Construction of a Decision Tree

- The main idea is to partition the feature space into non-overlapping regions $R_1, \cdots, R_J$ such that they cover the entire feature space
- Each observation that falls in R_j takes the mean of the other responses in that region as the response
- For simpler interpretation we are interested in R_j regions that are rectangular (boxed-shape in high dimensions)
- The objective to minimize is

$$\mathcal{C}(R_1, \cdots, R_J) = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \bar{y}_{R_j})^2,$$

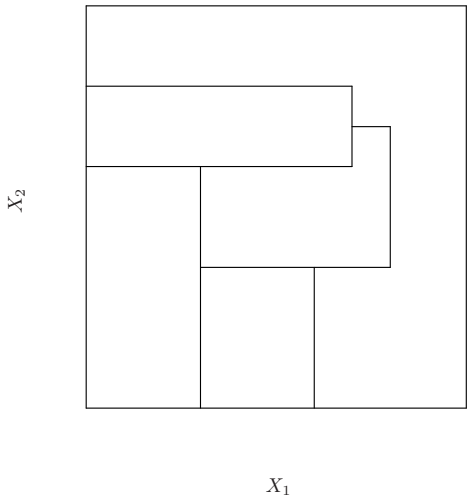
  where $\bar{y}_{R_j}$ is the mean of the responses in region R_j
- Yet, another NP-hard problem!

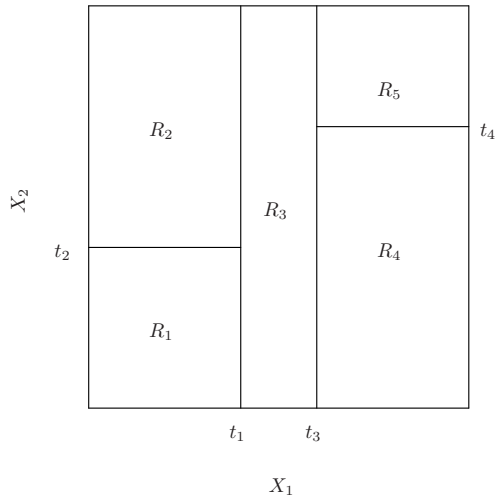
Construction of a Decision Tree: Greed is Good

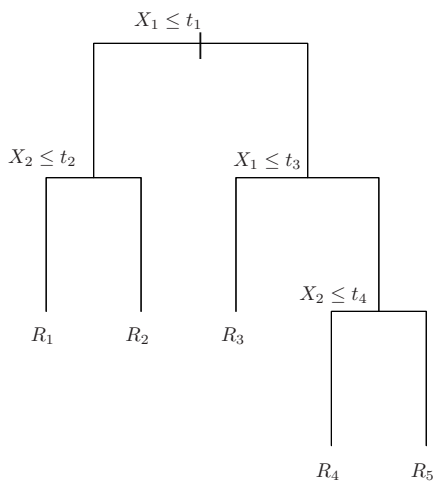
- We take a greedy approach and select features and cut-points one after the other
- We first select a feature X_j and cut point s such that splitting the predictor space into the regions $\{\boldsymbol{X} : X_j < s\}$ and $\{\boldsymbol{X} : X_j \geq s\}$ yields the largest reduction in the RSS
- We split the data further so as to minimize the RSS within each of the resulting regions, but, this time, instead of splitting the entire predictor space, we split one of the two previously identified regions (three regions are constructed)
- Among the resulted three regions we split one that causes more improvement in RSS cost. The process continues until a stopping criterion is reached (e.g., condition on the minimum number of y_j's in each region)
- For prediction we just go through the tree path

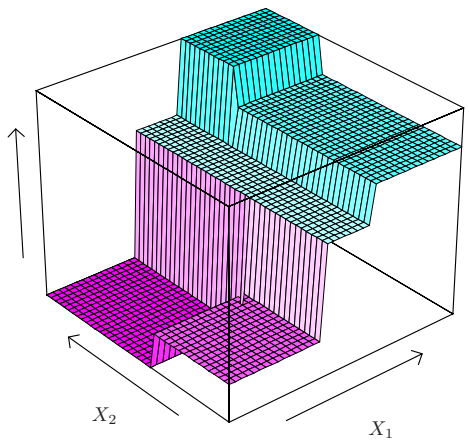
Construction of a Decision Tree

- Possible DT outcomes (top left not possible) and resulting tree regressor









Prevent Overfitting via Tree Pruning

- The proposed scheme to grow decision trees can easily end up with overfitting
- Such phenomenon can be prevented by pruning the tree and excluding some branches
- One idea is to continue the splitting only to the extent that the RSS reduction is smaller than some relatively larger quantity
- This is something like an incomplete construction of the tree
- Another approach would be to construct a large tree and then prune it to the extent that we get an optimal design

Pruning Procedure

- We consider a sequence of α-parameterized trees with $T \subset T_0$ that minimize

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

  where $|T|$ denotes the cardinality or number of elements in T (number of terminal nodes).
- Similar to Lasso α balances the model complexity and bias (simplicity vs fit to the training data)
- Optimal α is found via cross validation

Algorithm

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 $k = 1, \ldots, K$:

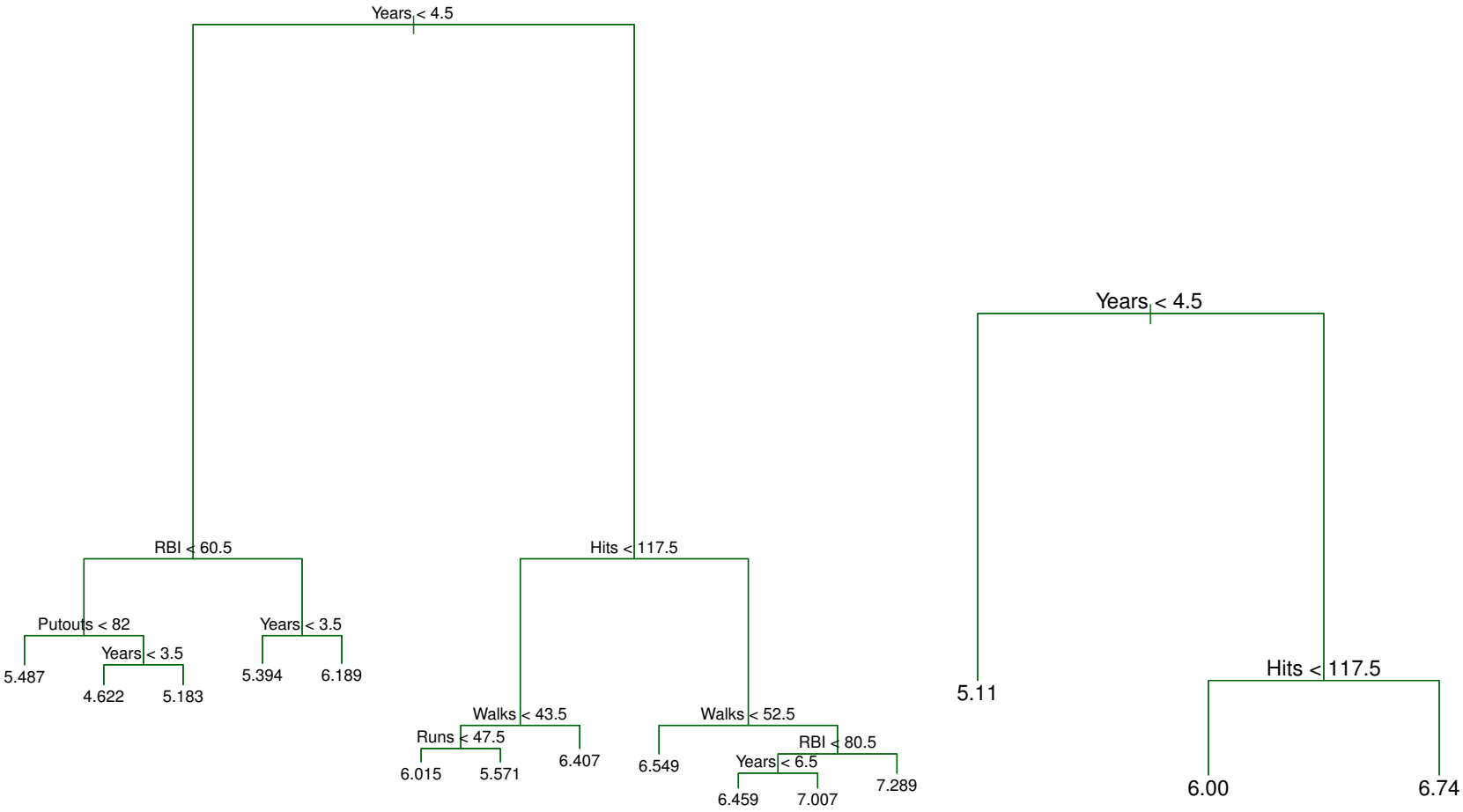
   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

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

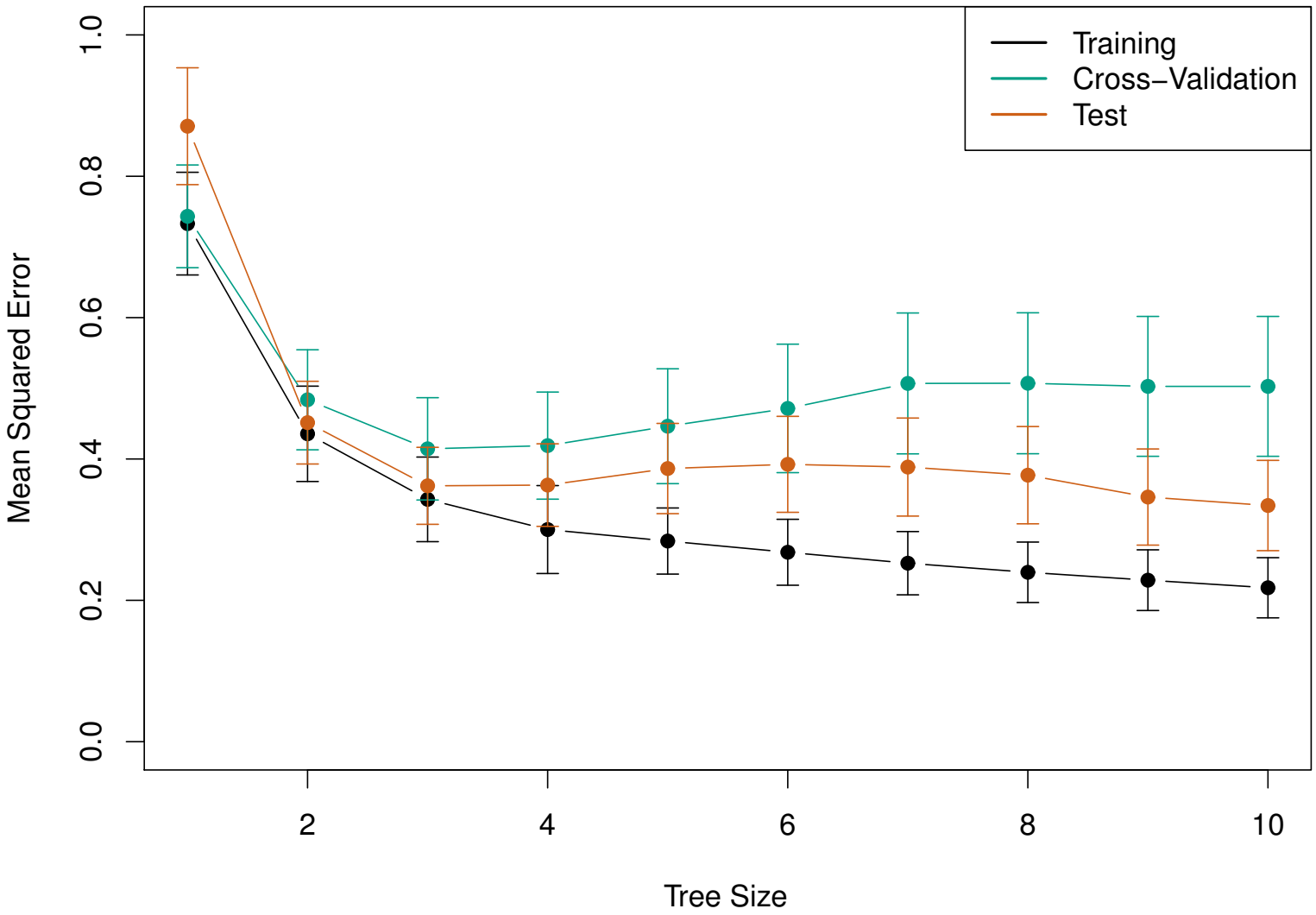
   Average the results for each value of α, and pick α to minimize the average error.

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

Example of Outcome



Parameter Selection



Decision Trees for Classification

- In classification each R_j would contain one class as the dominant population and the prediction label is the dominant class in the region it belongs to
- For classification we cannot form an RSS but instead we can use the **classification error rate**
- Definition of classification error rate: fraction of the training observations in a designated region that do not belong to the most common class:

$$E = 1 - \max_k(p_{j,k})$$

  where $p_{j,k}$ represents the proportion of training observations in the j-th region that are from the k-th class
- Ideally, we would want to have one class dominant in each region and the remainder classes to be minor

Gini Index and Cross Entrpy

- It turns out that classification error rate is not a very sensitive objective for classification
- Instead we may consider the Gini index:

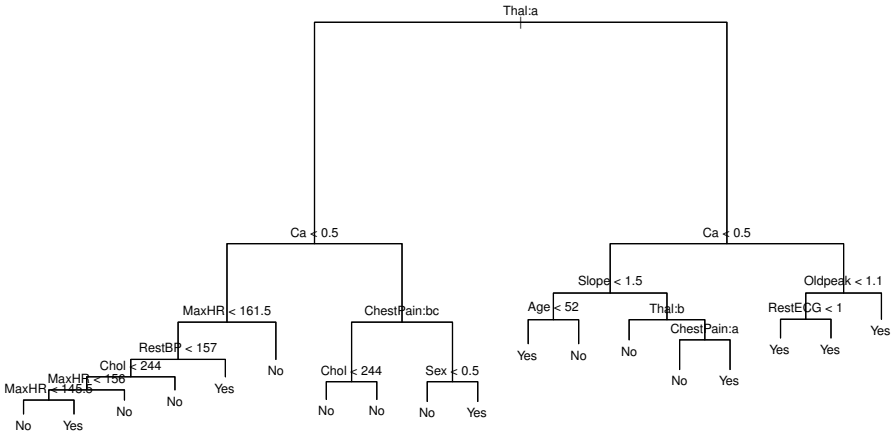
$$G = \sum_{k=1}^{K} p_{j,k}(1 - p_{j,k})$$

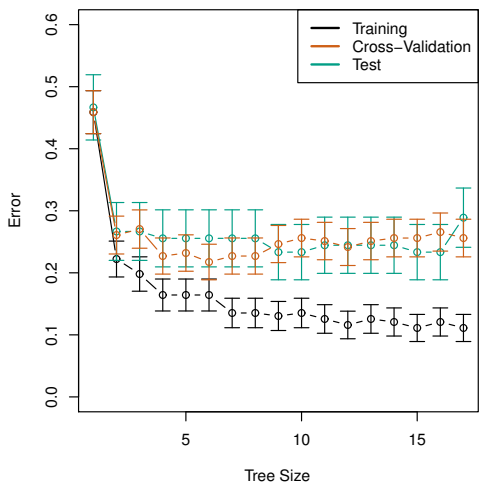
  which becomes small when all $p_{j,k}$ are close to zero or one (one class dominant)
- Another measure which numerically presents similar sensitivity is the cross entropy

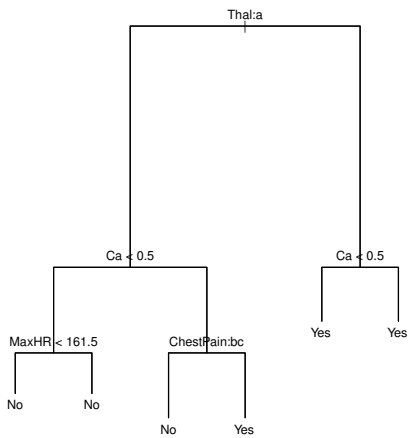
$$D = -\sum_{k=1}^{K} p_{j,k} \log p_{j,k}$$

Example

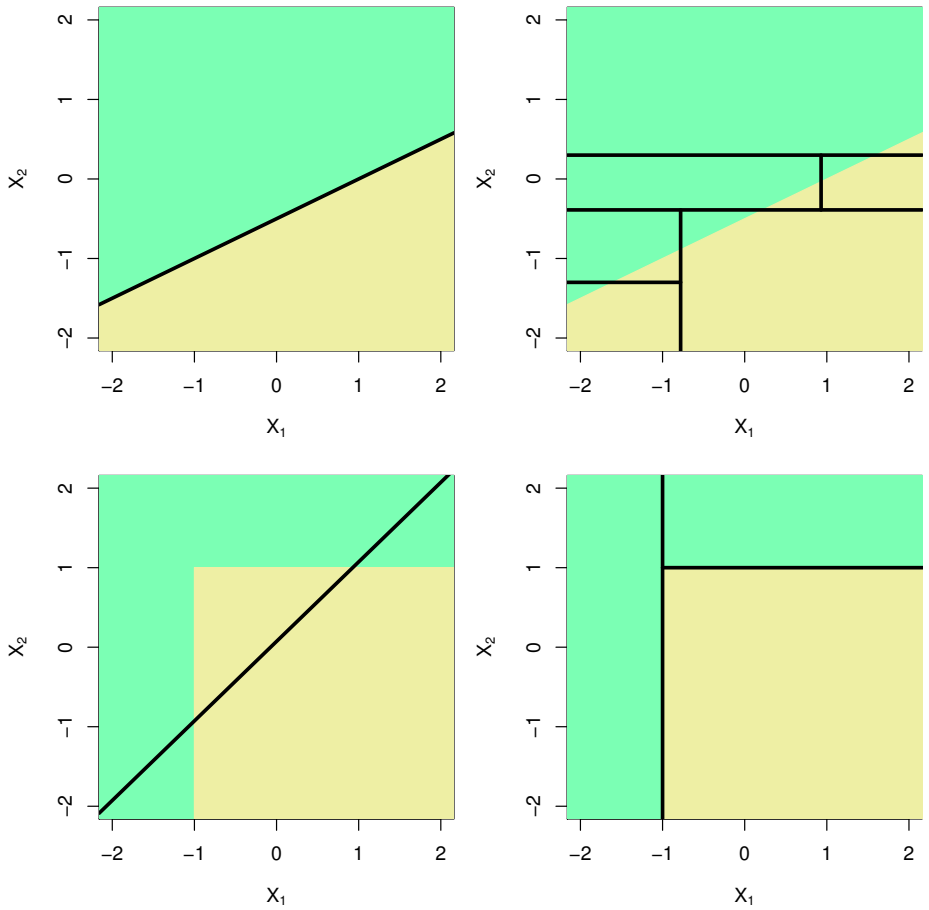
- Cross -validation error, training, and test error, for different sizes of the pruned tree







Trees or Linear Models?



Cons & Pros of Trees

- Trees are very easy to explain and interprete
- Decision trees are more closely similar to human decision-making than do the regression and classification we visited previously
- Trees can be displayed graphically, perfect for demonstrations
- Can easily handle qualitative predictors without the need to create dummy variables
- The main concern with the trees is their accuracy, which in fact can be fixed with bagging/boosting techniques and randomization

Bagging

Bagging

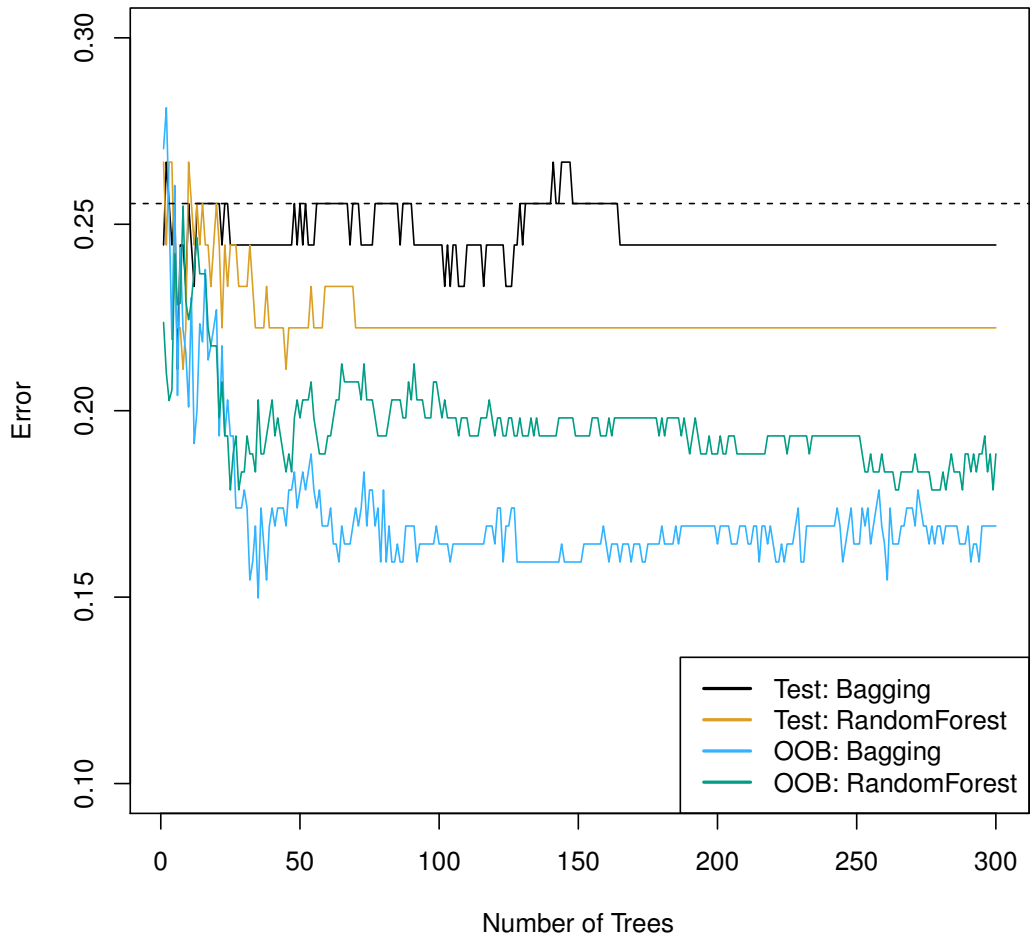
- Bagging (also referred to as Bootstrap aggregating) is a technique based on bootstrapping to reduce the variance of learning models
- The idea is simply averaging various models which may potentially reduce the variance
- While we do not have access to multiple training sets to train multiple models and then average across them, we can still take bootstrap samples and generate B different training data sets
- Associated with each bootstrap data set we can train a model $f_b(x)$ and take the average model to be

$$f_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} f_b(x)$$

- For classification still a similar procedure holds except that the final decision is based on the majority vote across the bootstrapped models

Bagging

- We can use bagging for random forests to improve the accuracy



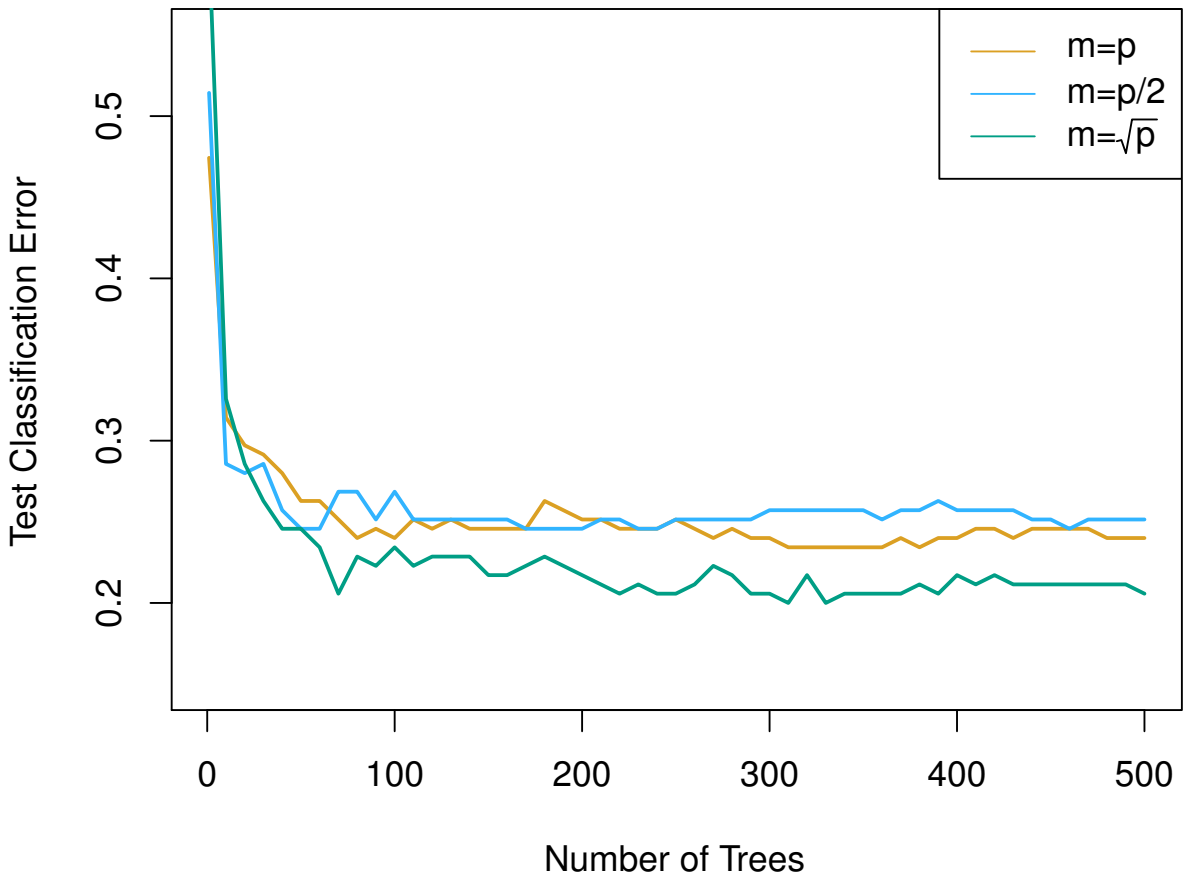
Out of Bag Estimate of the Test Error

- This is an efficient way of estimating the test error for bagged models and is similar to LOOCV
- Every time we fit a model based on a bagging set, part of the data (almost $1/3$) is not present and are called out of bag (OOB) observations
- To get an estimate of the test and to predict the response for the i-th observation we can take average of the $B/3$ OOB models that do not contain the i-th sample in the training
- You can immediately see some connection between this and LOOCV when B is very large

Random Forests

- They work in a similar way as bagged trees and only are equipped with a way of decorrelating the bagged models and further reduce the model variance
- For each bagged set we fit a decision tree, but for each split we only use a fraction of all available features (m instead of p)
- Optimally we take $m \approx \sqrt{p}$
- While this initially seems like a suboptimal way of fitting, it causes more decorrelation among the produced models and hence the variance of the average is further reduced

Example: Gene Expression



Boosting

Boosting

- Boosting is also a general technique like bagging
- In the context of decision trees recall that bagging constructed different versions of the training data and built a different model for each, later averaged
- In this process each tree is built independent of the other trees constructed
- The main difference of boosting is that trees are grown sequentially and in a sequential and dependent way

Boosting Algorithm

Algorithm 8.2 *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,

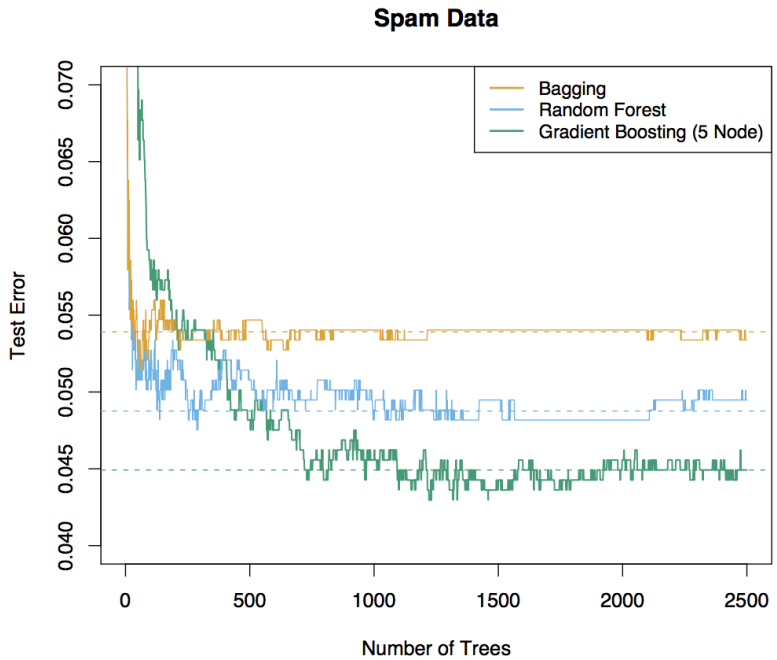
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$

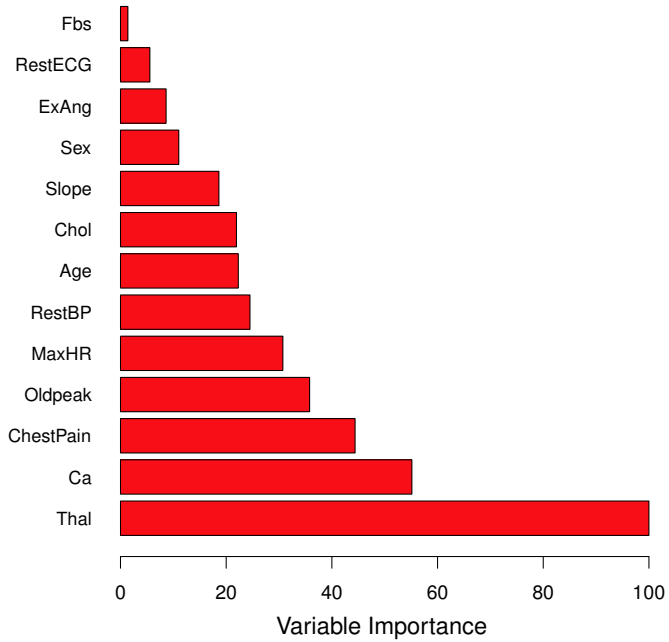
3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

- λ is normally a small number like 0.01 or 0.001
- Unlike bagging large B in the boosting can cause overfitting

Some Final Notes





- The number of splits d controls the tree depth and in boosting often d as low as 1 work fine
- By recording the total amount the RSS or classification errors are reduced by splitting over a variable, we can take an average over all B bagged models an generate variable importance
- RF and boosting technique are among the most state-of-the-art techniques with nice general prediction accuracies

Questions?

References


https://www.alsharif.info/iom530, 2013.


J. Friedman, T. Hastie, and R. Tibshirani.
The elements of statistical learning.
Springer series in statistics, 2nd edition, 2009.


G. James, D. Witten, T. Hastie, and R. Tibshirani.
https://lagunita.stanford.edu/c4x/HumanitiesScience/StatLearning/asset/trees.pdf, 2013.


G. James, D. Witten, T. Hastie, and R. Tibshirani.
An introduction to statistical learning: with applications in R, volume 112.
Springer, 2013.