COM 525000 – Statistical Learning

Lecture 8 – Tree-Based Methods

Y.-W. Peter Hong

Background Concept

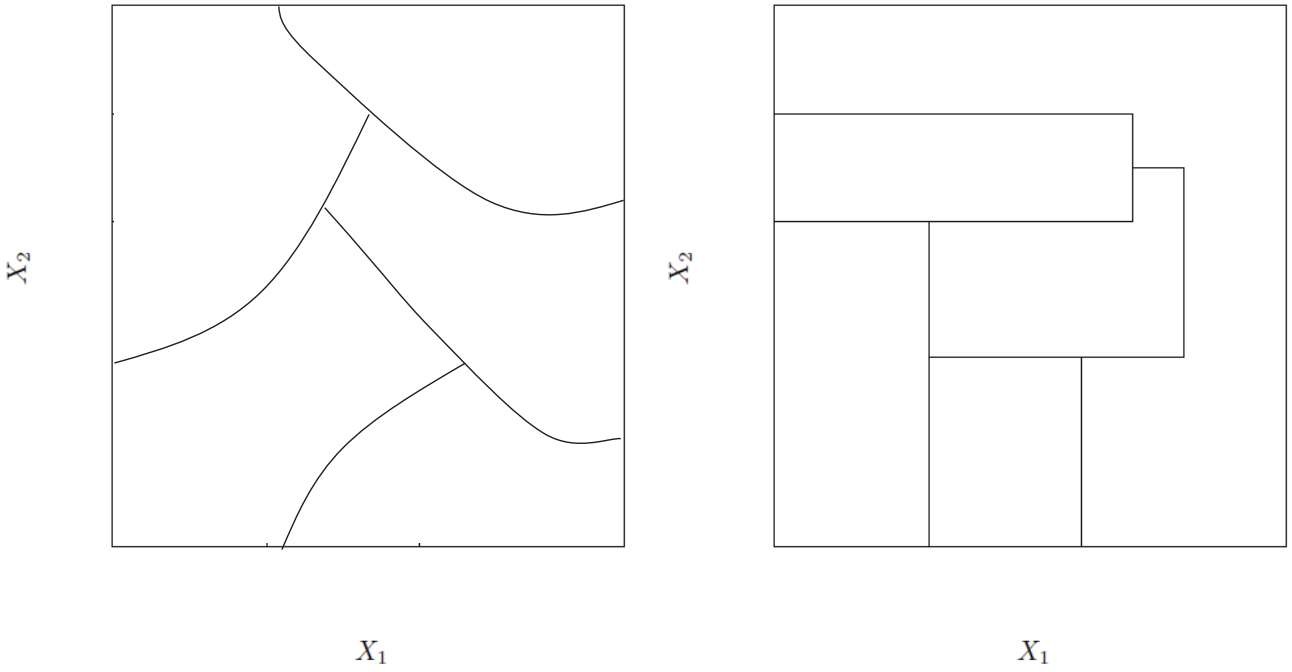
- **General Idea:**
  1. Divide the predictor space (i.e., the set of possible values for $X_1, \ldots, X_p$) into J distinct and non-overlapping regions $\mathcal{R}_1, \ldots, \mathcal{R}_J$.
  2. For observation $x \in \mathcal{R}_j$, we make the prediction

$$\hat{y} = \text{avg}(y_i | x_i \in \mathcal{R}_j) \Big(\triangleq \hat{y}_{\mathcal{R}_j} \Big).$$

- For regression problems, find regions $\mathcal{R}_1, \ldots, \mathcal{R}_J$ that minimize the RSS

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

Examples



Recursive Binary Splitting

- Recursive binary splitting determines the regions in a top-down greedy fashion.
  - Starting with all the data, select predictor X_j and cut-point s such that the splitting $\mathcal{R}_1(j,s) \triangleq \{X|X_j < s\}$ and $\mathcal{R}_2(j,s) \triangleq \{X|X_j \geq s\}$ yields the greatest RSS reduction, i.e., find j and s that minimizes

$$\sum_{i:x_i \in \mathcal{R}_1(j,s)} (y_i - \hat{y}_{\mathcal{R}_1})^2 + \sum_{i:x_i \in \mathcal{R}_2(j,s)} (y_i - \hat{y}_{\mathcal{R}_2})^2.$$

  - In the next iteration, repeat the splitting process on all resulting regions and find the best split.
  - Repeat until J regions are obtained.

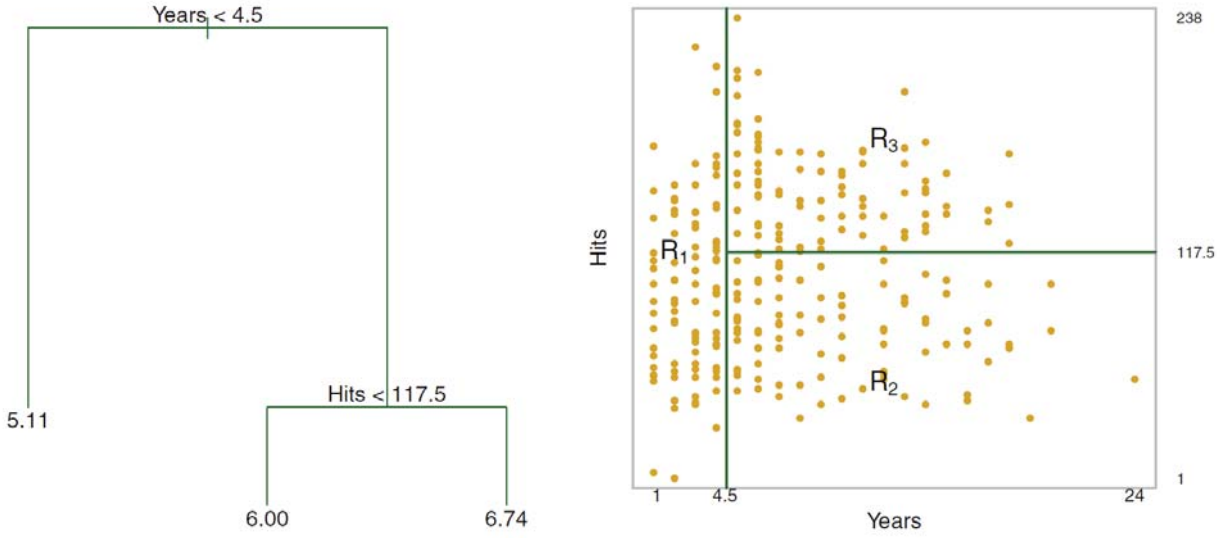
➔ This results in a tree structure!

Basic Decision Trees

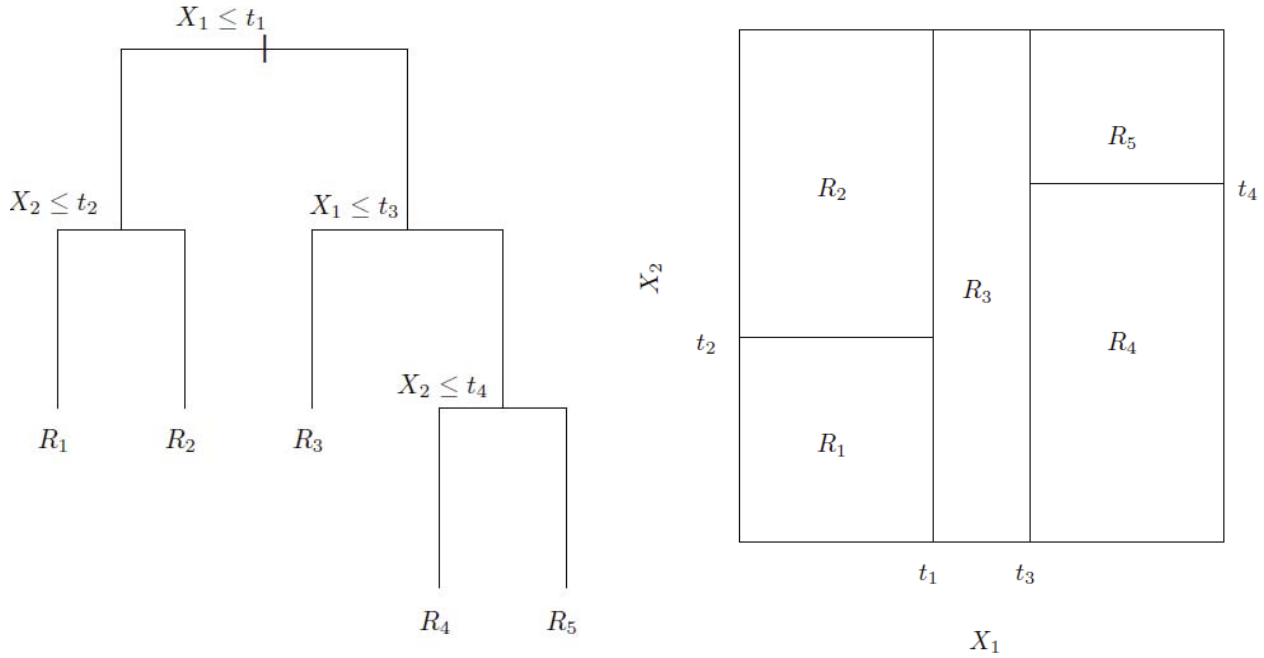
- Decision trees involve *a recursive binary splitting* of the predictor space into a number of simple regions.

Example: (Baseball Hitters Data Set)

- Use hitters' number of years and hits to predict salary.



Five-Region Example



Tree Pruning

- **Question:** How many regions should we have?
- Approach 1: Increase J until RSS decrease is small.
- Approach 2: Tree pruning by cost-complexity pruning.
- Cost complexity pruning finds subtree $T \subset T_0$ that minimizes, for some α,

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

➔The solution lies in the sequence of subtrees obtained by *weakest link pruning*, which successively collapses the internal node that produces the smallest per-node increase in RSS.

Weakest Link Pruning

- Starting with the initial full tree T_0, replace a subtree with a leaf node to obtain a new tree T_1. Select subtree to prune by minimizing:

$$\frac{\mathrm{RSS}(T_1) - \mathrm{RSS}(T_0)}{|T_0| - |T_1|}$$

- Iterate this pruning procedure to obtain a sequence of subtrees $T_0, T_1, T_2, \ldots, T_M$, where T_M is the tree with a single leaf node.

Algorithm and CV

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.

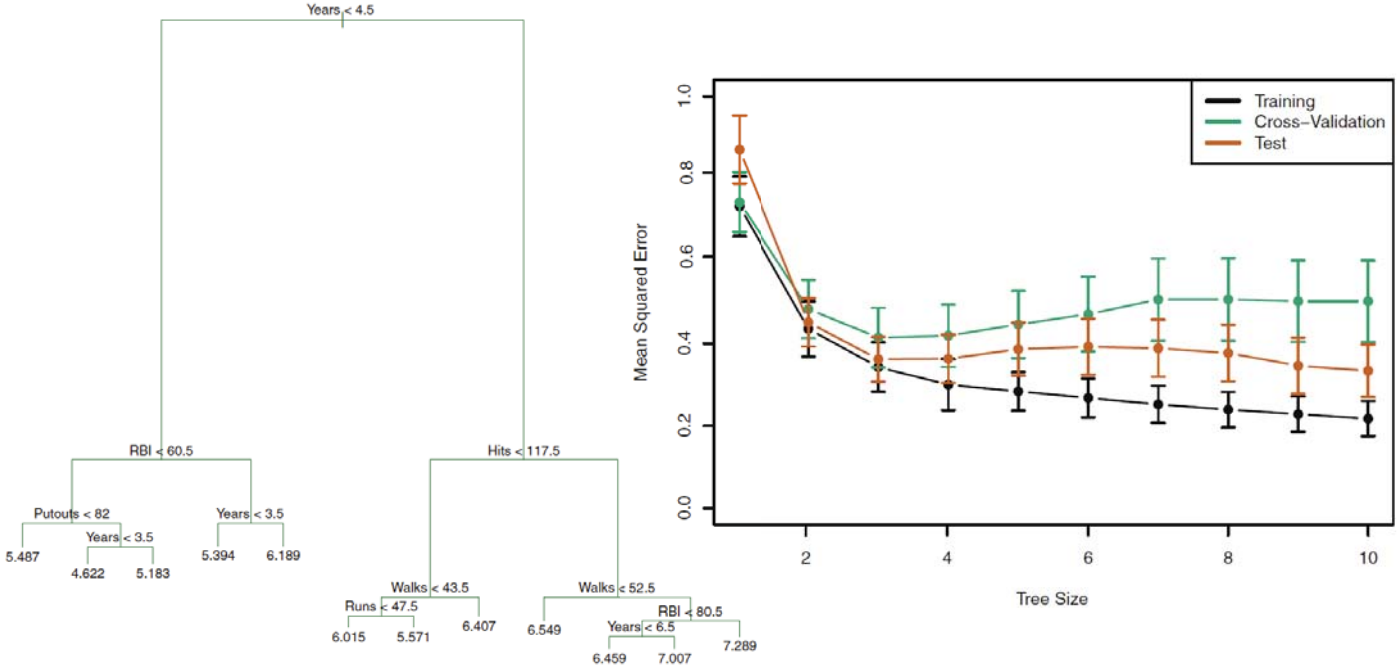
   Average the results for each value of α, and pick α to minimize the average error.

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

Example

Example (Hitters' Data Set):

- 132 training and 131 testing samples; 6-fold CV.



Classification Trees

- For regions $\mathcal{R}_1, \ldots, \mathcal{R}_J$, we define

$$\hat{p}_{mk} = \frac{1}{|\mathcal{R}_m|} \sum_{i: x_i \in \mathcal{R}_m} I(y_i = k)$$

  as the proportion of class k observations in $\mathcal{R}_m$, and classify the observations in region $\mathcal{R}_m$ as

$$k(m) = \arg\max_k \hat{p}_{mk}.$$

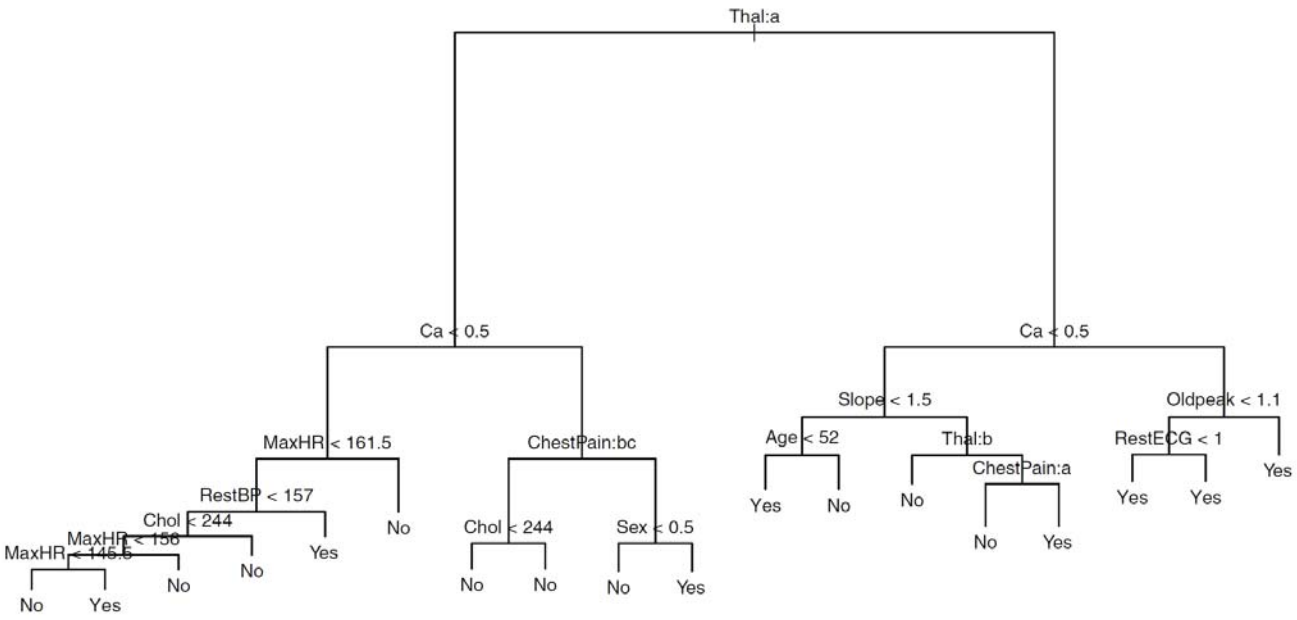
- Different measures of impurity:
  - ✓Classification Error Rate: $E = 1 - \max_k \hat{p}_{mk}$.
  - ✓Gini Index: $G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$.
  - ✓Cross-Entropy or Deviance: $D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$.

➔ Built using the last two, but pruned using the first.

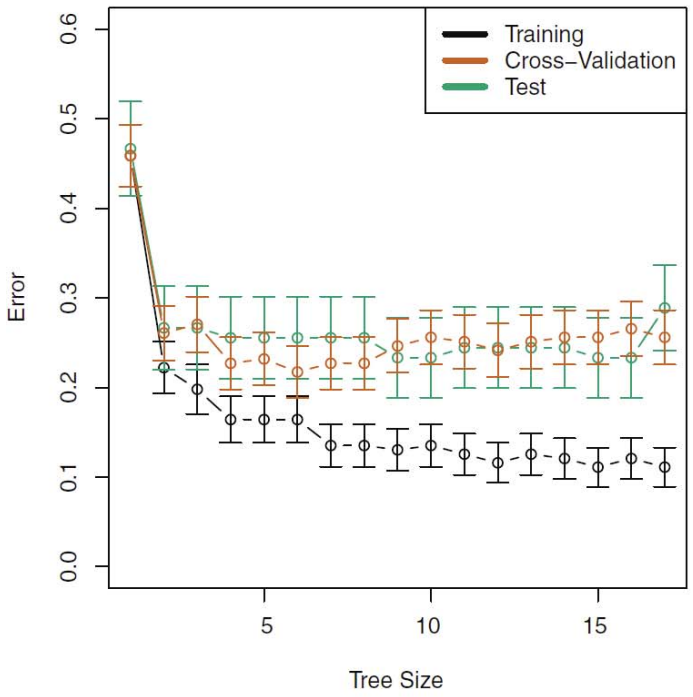
Example (1/2)

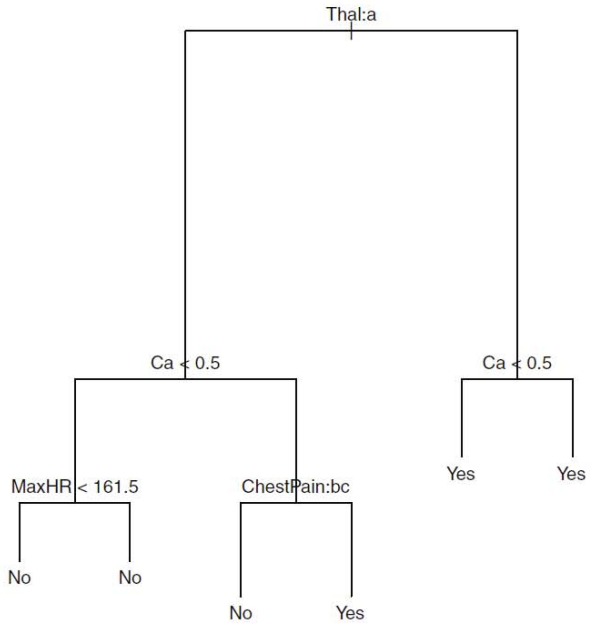
Example (Heart Data Set):

- 303 patients; 13 predictors (e.g., Age, Sex, Cholesteral, … etc); binary response (i.e., heart disease or not)

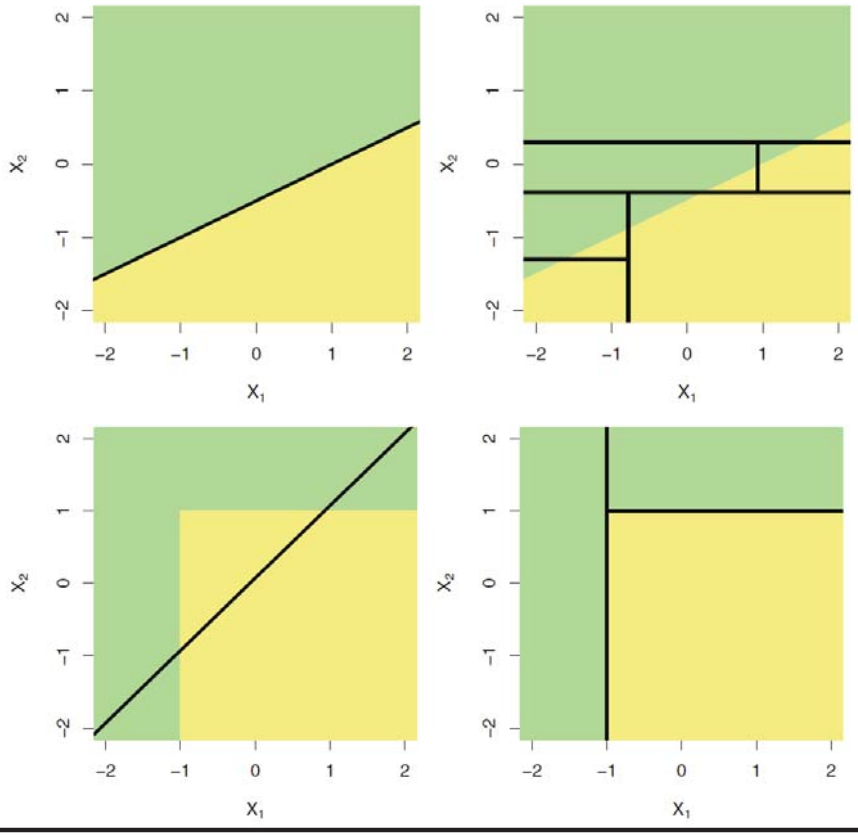


Example (2/2)





Tree vs Linear Models



Pros and Cons of Decision Trees

- Pros:
  - Easy to interpret and explain.
  - Mirrors human decision-making process. (?)
  - Easily handles qualitative predictors.
- Cons:
  - Bad predictive accuracy.
  - Non-robust to variations in data set.

Bootstrap Aggregation – *Bagging*

- Bootstrap aggregation, or **bagging**, is a general purpose procedure for reducing the variance of statistical learning methods through averaging.
- Ideally, with B different data sets, we can get the average prediction

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

- In bagging, we generate B bootstrapped training data sets to get $\hat{f}^{*1}, \ldots, \hat{f}^{*B}$, and compute

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

➔ For classification, we can instead apply majority vote.

Out-of-Bag Error Estimation

- The i-th observation is an out-of-bag (OOB) observation for data set $\mathcal{D}^{*b}$ if $(x_i, y_i) \notin \mathcal{D}^{*b}$.
- The probability that $(x_i, y_i) \notin \mathcal{D}^{*b}$ is

- Let $\mathcal{B}^{(-i)} \triangleq \{b | (x_i, y_i) \notin \mathcal{D}^{*b}\}$. For each i, we can obtain OOB prediction

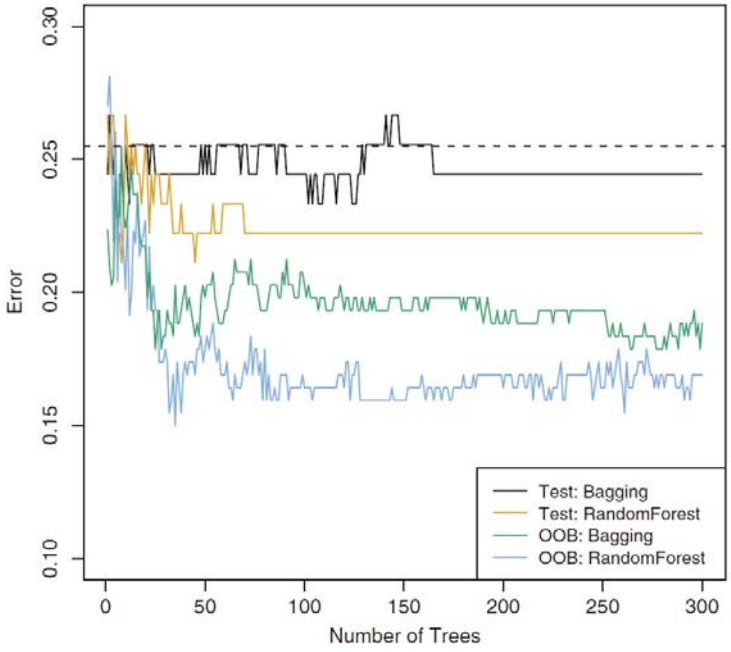
$$\hat{f}_{\text{avg}}(x_i) = \frac{1}{|\mathcal{B}^{(-i)}|} \sum_{b \in \mathcal{B}^{(-i)}} \hat{f}^{*b}(x_i)$$

and OOB test error estimate

$$\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}_{\text{avg}}^{(-i)}(x_i))^2.$$

Example

Example (Heart Data Set):



Remark: In bagging, predictions of different trees are often correlated (e.g., split strong predictors first).

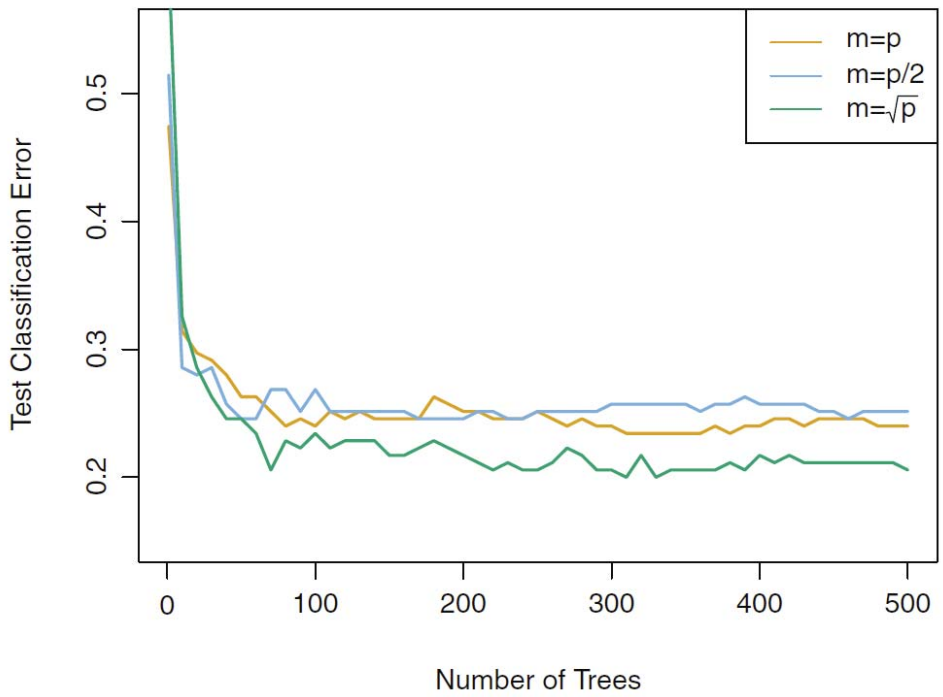
Random Forests

- **Random forest** is like bagging but decorrelates the trees by allowing each split to consider only a random m out of p predictors. (Typically, $m = \sqrt{p}$.)
- Also useful when applied to a large number of correlated predictors.

Example (Cancer Data Set):

- Expressions from 500 genes of 349 patients.
- Classification between 15 classes, namely, normal (class 1) and 14 different cancer types (classes 2 to 15)
- Error rate of a single tree is 45.7%; and the null rate is 75.4%.

Example



Remark: Bagging and random forest both involve fitting of trees *independently* to different bootstrap data sets.

Boosting

- **Boosting** also involves combining a large number of decision trees $\hat{f}^1, \ldots, \hat{f}^B$ that are grown *sequentially* (rather than independently thru bootstrap sampling).

  ➔ See algorithm on next page.

- Key Idea: Fit each tree to the current residuals of the responses (rather than the original responses).
- Three tuning parameters:
  1. Increasing B may result in overfitting (but slowly).
  2. The shrinkage parameter λ controls learning rate.
  3. Small trees are sufficient (e.g., with $d = 1$, boosting essentially is fitting to an additive model).

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

Adaboost for Classification

- Boosting for classification problems can be done with Adaboost. (Let output variable be $Y \in \{-1, +1\}$).

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Example

Example (Cancer Data Set):