MONASH University

ETC3250

Business Analytics

Week 9. Tree-Based Methods

18 September 2017

Outline

Week	Topic	Chapter	Lecturer
1	Introduction to business analytics & R	1	Souhaib
2	Statistical learning	2	Souhaib
3	Regression for prediction	3,7	Tas & David
4	Classification	4	Souhaib
5	Classification	4, 9	Souhaib
6	Model selection and resampling methods	5	Souhaib
7	Dimension reduction	6,10	Souhaib
8	Advanced regression	6	Souhaib
9	**Tree-based methods**	8	Souhaib
	Decision trees	8	Souhaib
	Random forests, bagging and boosting	8	Souhaib
	Semester break		
10	Clustering	10	Souhaib
11	Visualization and data wrangling		Souhaib
12	Project presentation		Souhaib

Building a regression tree

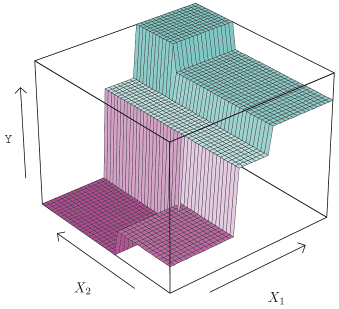
- We divide the predictor space—that is, the set of possible values for $X_1, X_2, \ldots, X_p$—into J **distinct** and **non-overlapping** regions, $R_1, R_2, \ldots, R_J$.
- The regions could have any shape. However, for simplicity and for ease of interpretation, we divide the predictor space into high-dimensional **rectangles**.
- We model the response as a constant c_j in each region

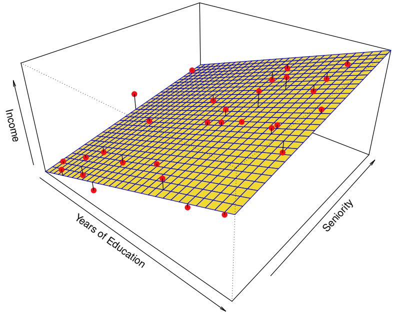
$$f(x) = \sum_{j=1}^{J} c_j \, I(x \in R_j)$$

Trees Versus Linear Models

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

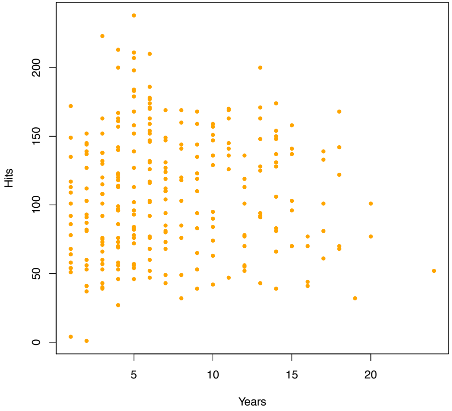
$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$



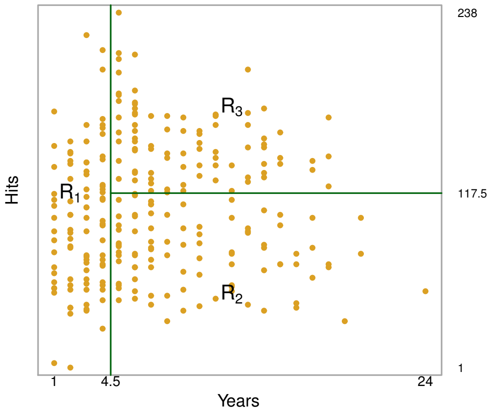


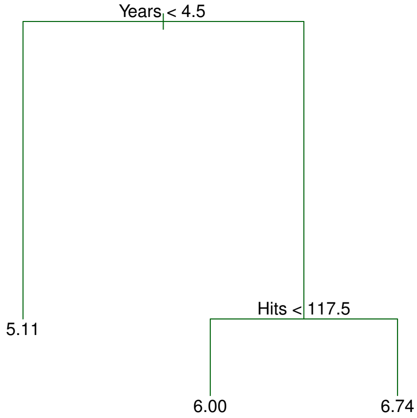
Predicting a baseball player's salary

- Y: log salary of a baseball player (in thousands of dollars)
- X_1: number of years played in the major leagues
- X_2: number of hits made in the previous year



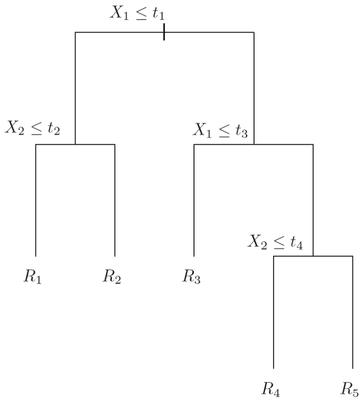
Predicting a baseball player's salary

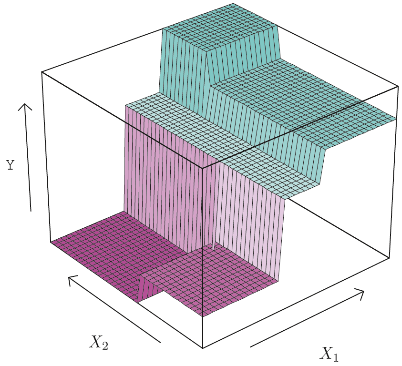




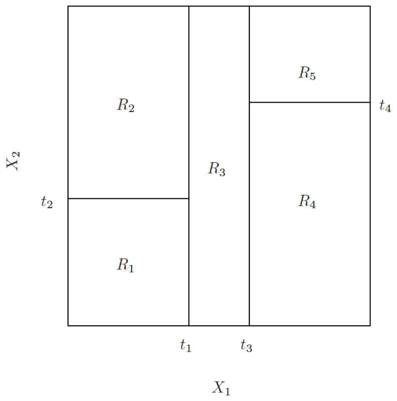
- $\boldsymbol{R_1} = \{X | Years < 4.5\}$
- $\boldsymbol{R_2} = \{X | Years \geq 4.5, Hits < 117.5\}$
- $\boldsymbol{R_3} = \{X | Years \geq 4.5, Hits \geq 117.5\}$

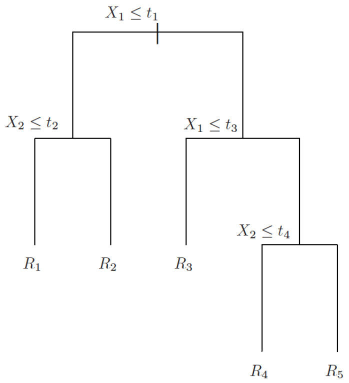
Regression tree





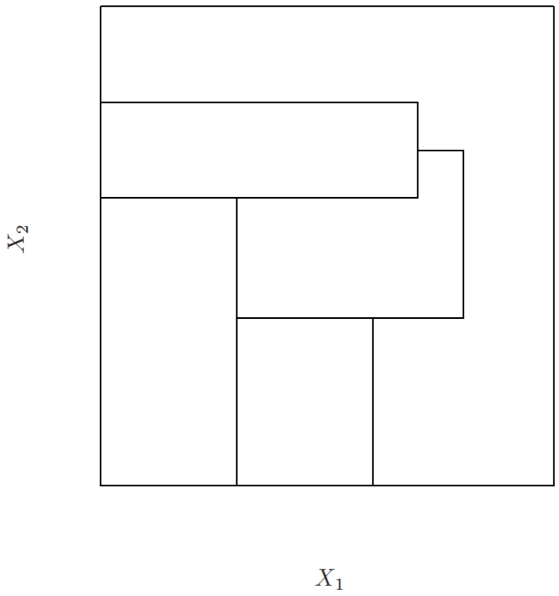
Tree-Based Methods





- $R_1, R_2, \ldots,$ and R_5 are **terminal nodes** or **leaves**.
- The points where we split are **internal nodes**.
- The segments that connect the nodes are **branches**.

Not all partitions are possible



Building a regression tree

1. Given a partition $R_1, R_2, \ldots, R_J$, what are the optimal values of c_j if we want to minimize $\sum_i (y_i - f(x_i))^2$?
2. How do we construct the regions $R_1, ..., R_J$?

1. The best c_j is just the average of y_i in region R_j:
$$\hat{c}_j = \text{average}(y_i | x_i \in R_j).$$

2. Finding the best binary partition in terms of minimum sum of squares is generally *computationally infeasible*. For this reason, we take a *top-down*, *greedy* approach that is known as **recursive binary splitting**.

Building a regression tree

1. Given a partition $R_1, R_2, \ldots, R_J$, what are the optimal values of c_j if we want to minimize $\sum_i (y_i - f(x_i))^2$?
2. How do we construct the regions $R_1, ..., R_J$?

1. The best c_j is just the average of y_i in region R_j:

$$\hat{c}_j = \text{average}(y_i | x_i \in R_j).$$

2. Finding the best binary partition in terms of minimum sum of squares is generally *computationally infeasible*. For this reason, we take a *top-down*, *greedy* approach that is known as **recursive binary splitting**.

Building a regression tree

1. Given a partition $R_1, R_2, \ldots, R_J$, what are the optimal values of c_j if we want to minimize $\sum_i (y_i - f(x_i))^2$?
2. How do we construct the regions $R_1, ..., R_J$?

1. The best c_j is just the average of y_i in region R_j:

$$\hat{c}_j = \text{average}(y_i | x_i \in R_j).$$

2. Finding the best binary partition in terms of minimum sum of squares is generally *computationally infeasible*. For this reason, we take a *top-down*, *greedy* approach that is known as **recursive binary splitting**.

Recursive binary splitting

1. **Top-down**: it begins at the top of the tree (all observations belong to a single region) and then successively splits the predictor space; each split is indicated via two new branches further down on the tree
2. **Greedy**: at each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

Recursive binary splitting

1. Start with a single region R_1 (entire input space), and iterate:
    1. Select a region R_m, a predictor X_j , and a splitting point s, such that splitting R_m with the criterion $X_j < s$ produces the **largest decrease in RSS**
    2. Redefine the regions with this additional split.
2. Continues until stopping criterion, e.g. $N_m < 5$.

$$\mathrm{RSS}(T) = \sum_{m=1}^{|T|} N_m Q_m(T), \quad N_m = \#\{x_i \in R_m\},$$

where

$$Q_m(T) = \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2$$

and $|T|$ is the number of terminal nodes in T.

What size of tree?

- The process described above may produce good predictions on the **training set**, but is likely to **overfit** the data (trees are very flexible).
- A smaller tree with fewer splits (that is, fewer regions) might lead to **lower variance** and better interpretation at the cost of a **little bias**.
- Tree size is a tuning parameter governing the **model's complexity**, and the optimal tree size should be adaptively chosen from the data
- One possible alternative is to produce splits only if the decrease in the RSS exceeds some **(high) threshold**. The problem is that a worthless split early on in the tree might be followed by a very good split.

Tree Pruning

Tree pruning grows a **very large tree** T_0, and then **prune** it back in order to obtain a subtree. The weakest link pruning procedure is:

1. Starting with 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

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

2. Iterate this pruning to obtain a sequence $T_0, T_1, T_2, \ldots, T_R$ where T_R is the tree with a single leaf node.
3. Select the optimal tree T_j by cross validation

Tree Pruning - equivalent procedure

The cost complexity criterion is given by

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T|$$

where $\alpha \geq 0$ is a tuning parameter that governs the tradeoff between tree size and its goodness of fit to the data.

- Large/small values of α result in smaller/larger trees T_α
- The idea is to find, for each α, the subtree $T_\alpha \subseteq T_0$ to minimize $C_\alpha(T)$ (which is unique).
- Fact: the solution for each α is among $T_0, T_1, T_2, \ldots, T_R$ from weakest link pruning

Tree Pruning

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 $k = 1, \ldots, K$:

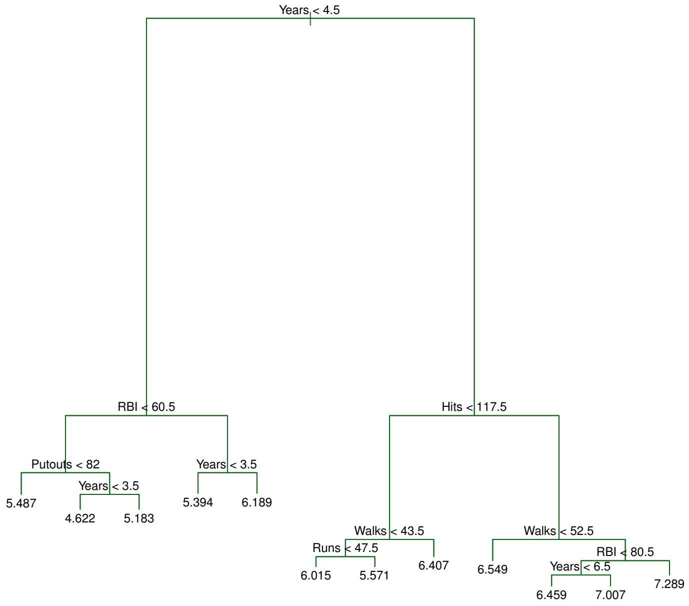
   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

   (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.

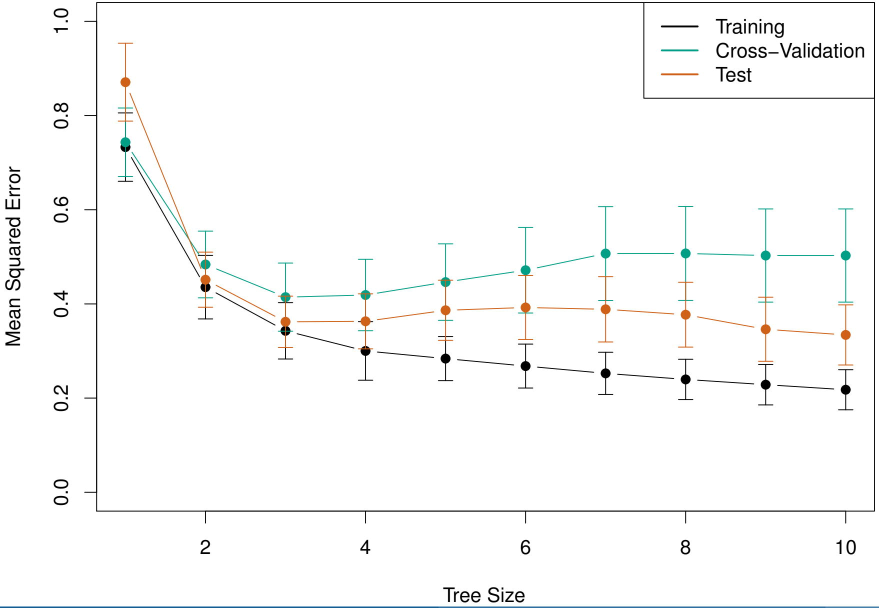
4. Return the subtree from Step 2 that corresponds to the chosen value of α.

Example



Years < 4.5
RBI < 60.5
Hits < 117.5
Putouts < 82
Years < 3.5
Years < 3.5
5.487
4.622
5.183
5.394
6.189
Walks < 43.5
Runs < 47.5
6.015
5.571
6.407
Walks < 52.5
6.549
RBI < 80.5
Years < 6.5
6.459
7.007
7.289

Example


Training
Cross−Validation
Test
Mean Squared Error
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
Tree Size

Classification trees

- A classification tree is used to predict a **qualitative response** rather than a quantitative one
- We predict that each observation belongs to the **most commonly occurring class** of training observations in the region to which it belongs
- Just as in the regression setting, we use recursive binary splitting to grow a classification tree
- However RSS cannot be used as a criterion for making the binary splits. A natural alternative to RSS is the classification error rate:

$$1 - \max_k \hat{p}_{mk},$$

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

Classification trees

- Classification error is not sufficiently sensitive for tree-growing. In practice two other measures are used
- The *Gini index* measures total variance across the K classes:

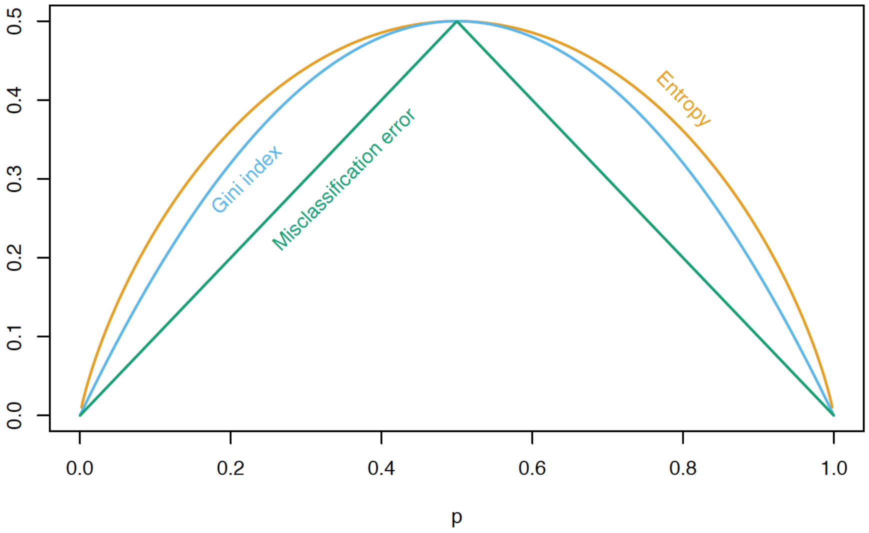
$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

- An alternative to the Gini index is *entropy*, given by

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

- If all of the $\hat{p}_{mk}$'s are close to zero or one, both G and D are small. It is a measure of **node purity**, i.e. a small value indicates that a node contains predominantly observations from a single class.

Classification trees



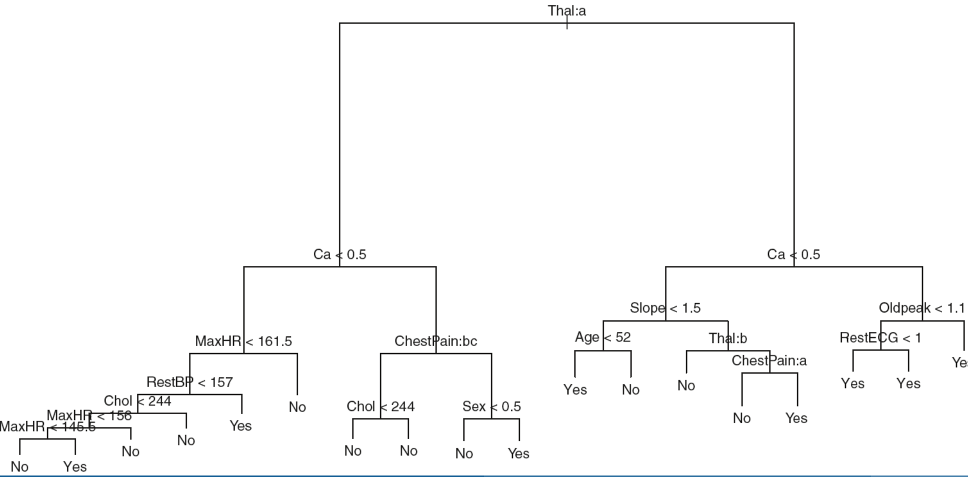
Classification trees

- When <u>building</u> a classification tree, either the Gini index or the entropy are typically used to evaluate the quality of a particular split
- Any of these three approaches might be used when <u>pruning</u> the tree, but the classification error rate is preferable if prediction accuracy of the final pruned tree is the goal

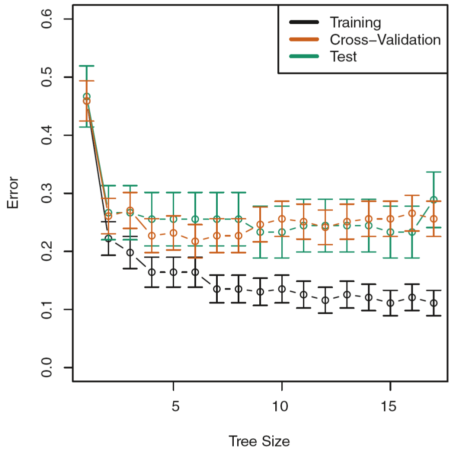
Example

Y: presence of heart disease (Yes/No)

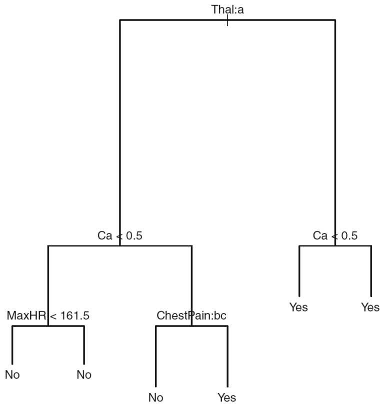
$\boldsymbol{X}$: heart and lung function measurements



Example

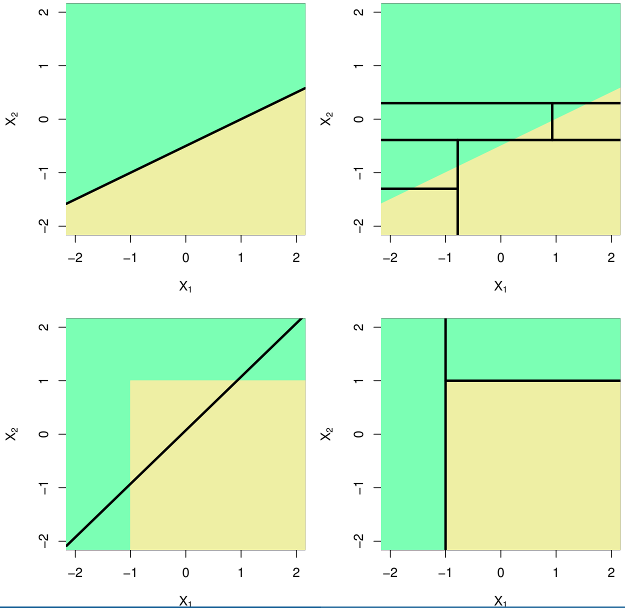


Training
Cross–Validation
Test
Error
0.0
0.1
0.2
0.3
0.4
0.5
0.6
5
10
15
Tree Size



Thal:a
Ca < 0.5
Ca < 0.5
MaxHR < 161.5
ChestPain:bc
Yes
Yes
No
No
No
Yes

Trees Versus Linear Models



Advantages and Disadvantages of Trees

(+) Trees are flexible models

(+) Trees can be easily interpreted

(+) Trees can easily handle qualitative predictors without the need to create dummy variables + missing values

(-) Trees are unstable and can be very non-robust: a small change in the data can cause a large change in the final estimated tree. The decision trees suffer from high variance ☹

$\rightarrow$ The predictive performance of trees can be substantially improved by **aggregating many decision trees**, using methods like <u>bagging</u>, <u>random forests</u>, and <u>boosting</u>.

Aggregation of prediction models

- Given a set of n independent observations $Z_1, \ldots, Z_n$, each with variance σ^2, the variance of the mean $\bar{Z}$ of the observations is given by $\frac{\sigma^2}{n}$.
- Aggregation procedure:
  1. Take B training sets from the <u>population</u>:
  $$D_1, D_2, \ldots, D_B$$
  2. Build a separate prediction model using each $D_{(\cdot)}$:
  $$\hat{f}_1(x), \hat{f}_2(x), \ldots, \hat{f}_B(x)$$
  3. Average the resulting predictions:
  $$\hat{f}_{\text{avg}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_b(x)$$

Bootstrap aggregation = bagging

1. Take B different bootstrapped training sets:

$$D_1, D_2, \ldots, D_B$$

2. Build a separate prediction model using each $D_{(\cdot)}$:

$$\hat{f}_1(x), \hat{f}_2(x), \ldots, \hat{f}_B(x)$$

3. Average the resulting predictions:

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

Bootstrap aggregation = bagging

- Bootstrap aggregation, or bagging, is a general-purpose procedure for reducing the variance of a statistical learning method
- We have introduced the bootstrap for uncertainty quantification. The bootstrap can be used in a completely different context, in order to improve statistical learning methods
- While bagging can improve predictions for *many* regression methods, it is particularly useful for decision trees
- Bagging is more likely to improve the results for high variance and low bias prediction models.

Bagging for decision trees

- Construct B regression trees using B bootstrapped training sets, and average the resulting predictions.
- These trees are grown deep, and are **not pruned**.
- Each individual tree has **high variance, but low bias**. Why?
- Averaging these B trees **reduces the variance**. Why?
- For classification trees, there are few possible aggregation methods, but the simplest is the **majority vote**.

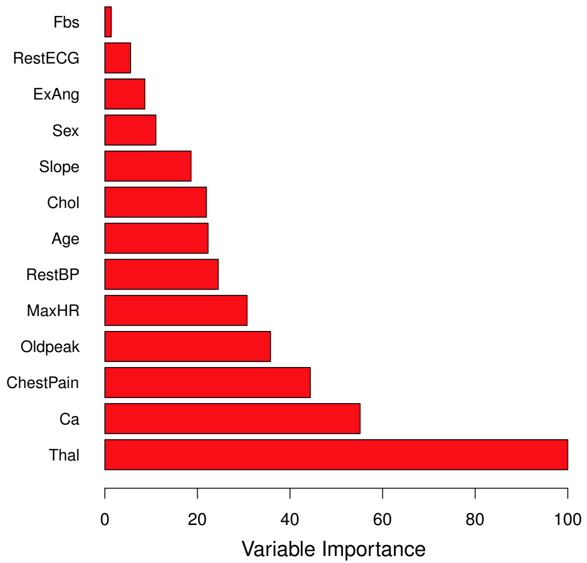
Out-of-Bag Error Estimation

- No need to use (cross-)validation to **estimate the test error** of a bagged model = convenient for large datasets.
- On average, each bagged tree makes use of around **two-thirds of the observations**. Prove it.
- The remaining one-third of the observations not used to fit a given bagged tree are referred to as the **out-of-bag** (OOB) observations.
- We can predict the response for the *i*th observation using each of the trees in which that observation was OOB. This will yield around $\frac{B}{3}$ predictions for the *i*th observation.
- To obtain a single prediction for the *i*th observation, we can average these predicted responses (regression) or can take a majority vote (classification).

Variable Importance Measures

- Bagging improves <u>prediction accuracy</u> at the expense of <u>interpretability</u>.
- It is no longer clear which variables are most important to the procedure.
- Record the total amount that the RSS/Gini is decreased due to splits over a given predictor, averaged over all B trees. A large value indicates an important predictor.

Variable Importance Measures



Bias and variance tradeoff

$$\hat{f}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$$

$$\mathrm{MSE} = E[(y - \hat{f}^B(x))^2]$$

MSE = NOISE + BIAS^2 + VARIANCE

- BIAS: If trees are sufficiently deep, they have very small bias. Why?
- VARIANCE = Var$(\frac{1}{B}\sum_{b=1}^{B} T_b(x)) = ?$

Aggregation with correlation

- Given a set of B independent observations $Z_1, \ldots, Z_B$, each with variance σ^2, the variance of the mean $\bar{Z}$ of the observations is given by $\frac{\sigma^2}{B}$.
- If the variables are simply i.d. (identically distributed, but not necessarily independent) with positive pairwise correlation ρ ($\rho > 0$), then

$$\mathrm{Var}(\bar{Z}) = \rho\sigma^2 + \sigma^2 \frac{1-\rho}{B}$$

How do we reduce it?

Aggregation with correlation

$$\text{Var}(\bar{Z}) = \rho\sigma^2 + \sigma^2\frac{1-\rho}{B}$$

- $\rho\sigma^2$: decreases if ρ decreases (i.e. if m decreases)
- $\sigma^2\frac{1-\rho}{B}$: decreases if B increases (irrespective of ρ)

Random Forests

- Averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities.
- Random forests provide an improvement over bagged trees by **decorrelating the trees**.
- Each time a split in a tree is considered, a <u>random sample of m predictors</u> is chosen as split candidates from the full set of p predictors
- The split is allowed to use only one of those m predictors. We typically choose $m \approx \sqrt{p}$.
- If a random forest is built using $m = p$, then this amounts simply to bagging.

Random Forests

1. For $b = 1$ to B:

   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.

   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

      i. Select m variables at random from the p variables.

      ii. Pick the best variable/split-point among the m.

      iii. Split the node into two daughter nodes.

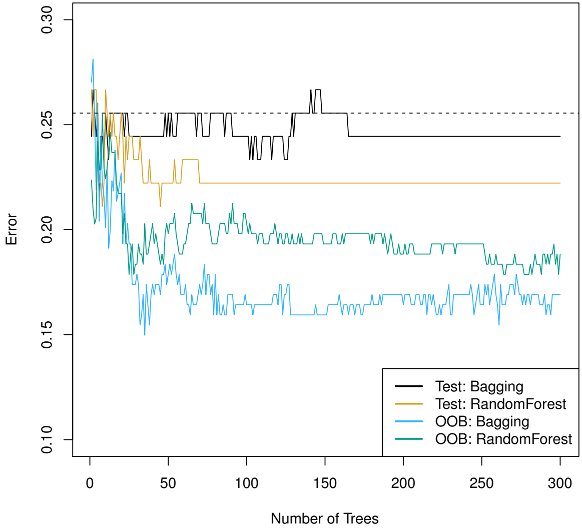
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

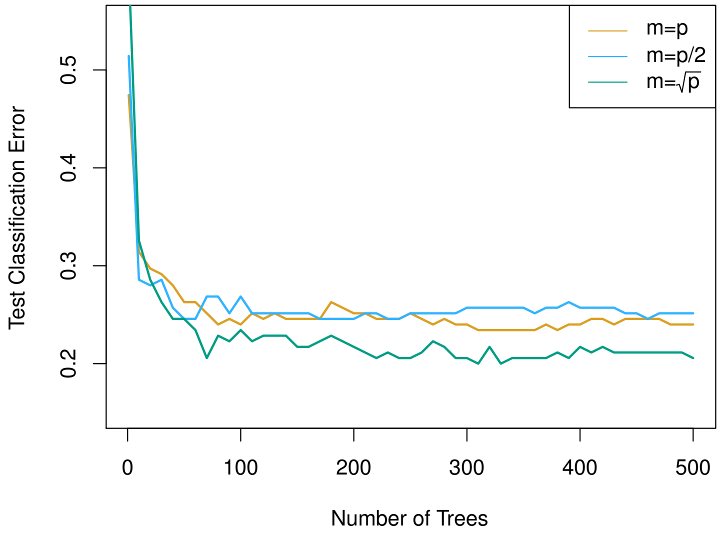
Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Example



Error
0.10
0.15
0.20
0.25
0.30
0
50
100
150
200
250
300
Number of Trees
Test: Bagging
Test: RandomForest
OOB: Bagging
OOB: RandomForest

Example



m=p
m=p/2
m=$\sqrt{p}$
Test Classification Error
0.2
0.3
0.4
0.5
0
100
200
300
400
500
Number of Trees

Boosting

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