Statistics 101C - Week 4 Tree Models

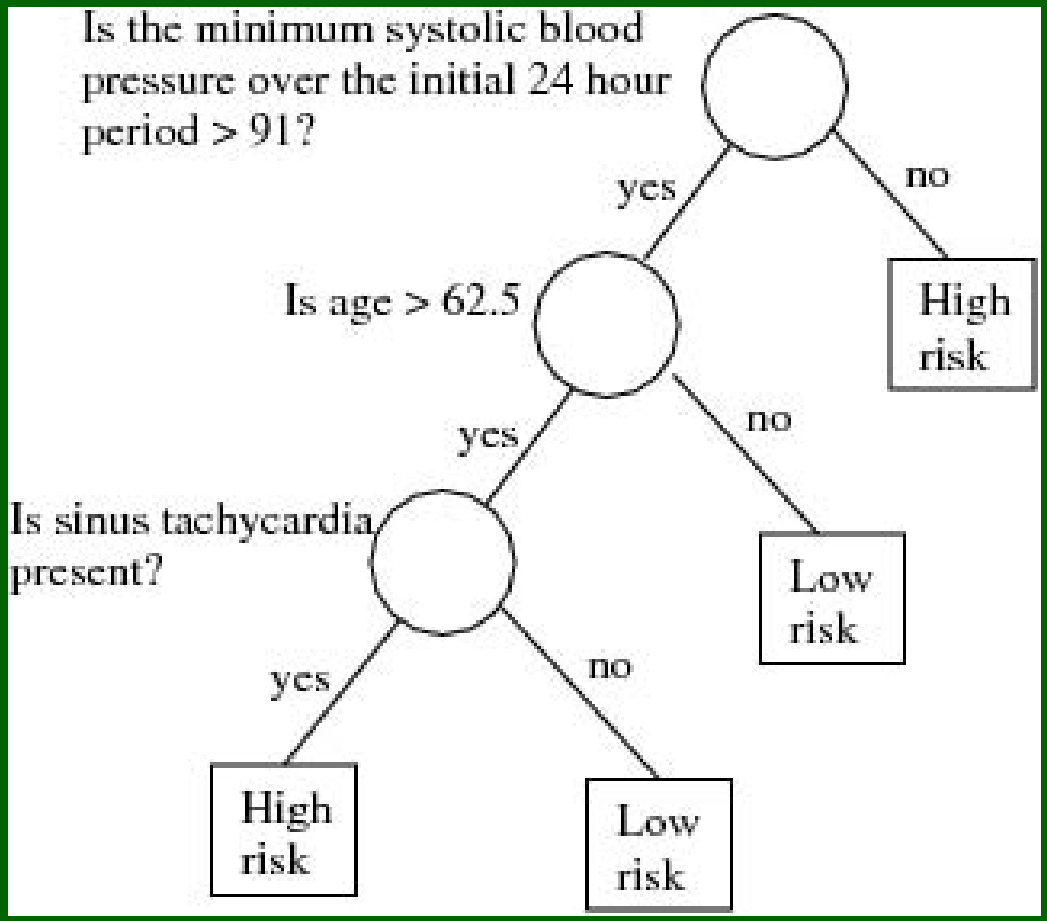
Shirong Xu

University of California, Los Angeles

shirong@stat.ucla.edu

October 22, 2024

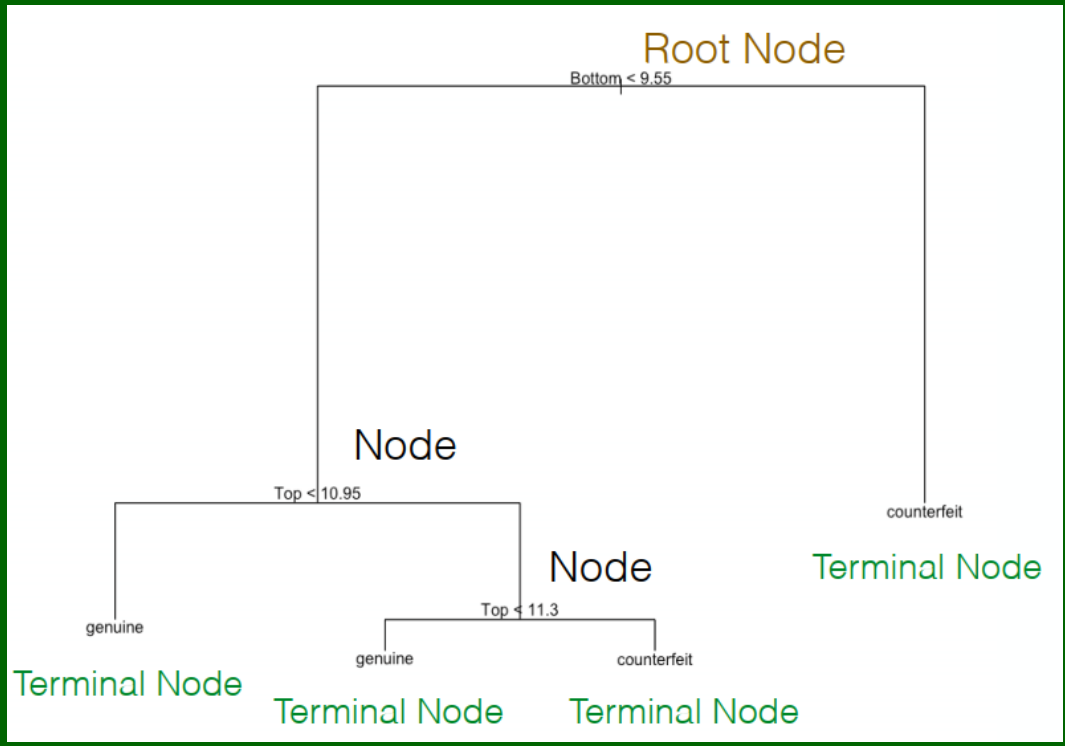
An example of classification tree



- Predict whether a patient has high risk

Some notations

- $X = (X_1, X_2, \ldots, X_p)^T \in \mathcal{X} \subset \mathcal{R}^p$, some of which may be categorical
- Terminology: node, root node, terminal node (leaf node), parent node, child node



- Trees are constructed by repeated splits of subsets of $\mathcal{X}$ into two descendant subsets, beginning with $\mathcal{X}$ itself
- A node will have only one parent node except for the root node

Some notations

- A node is denoted by t
- Its left child node is denoted by t_L and right by t_R
- The collection of all the nodes is denoted by T
- The collection of all the leaf nodes by $\widetilde{T}$
- A split is denoted by s, and the set of splits is denoted by $\mathcal{S}$

Key elements of a tree

- Questions we need to deal with in the construction of a tree
  1. The decisions when to declare a node terminal or to continue splitting it
  2. The selection of the splits
  3. The assignment of each terminal node to a class

Key elements of a tree

- Questions we need to deal with in the construction of a tree
  1. The decisions when to declare a node terminal or to continue splitting it
  2. The selection of the splits
  3. The assignment of each terminal node to a class

- In particular,
  1. A set of binary questions used as splits
  2. A goodness of split criterion $\phi(s, t)$ that can be evaluated for any split s of any node t
  3. A stop-splitting rule
  4. A rule for assigning every terminal node to a class

How to split in decision tree?

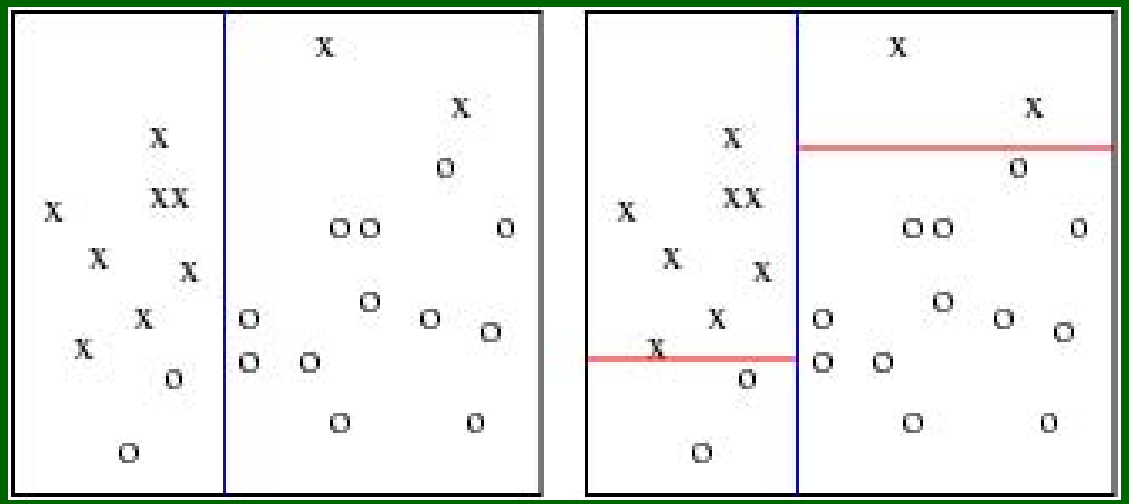
- Each split depends on the value of only one variable
- For each variable X_j, $\mathcal{S}$ includes all questions of the form

$$\{\text{Is } X_j \in A?\}$$

  - If X_j is continuous or ordinal, $A = (-\infty, c]$
  - If X_j is categorical, A is a subset of categories
- The splits for all p variables constitute the standard set of questions

How to evaluate the quality of a split?

- The goodness of split is measured by an impurity function defined for each node
- Intuitively, we want each leaf node to be "pure", that is, one class dominates



The impurity function

Let t denote a node and $I(t)$ denote the indexes of samples at the node t. Denote by $p_{tk} = P(Y = k|t) = |I(t)|^{-1} \sum_{i \in I(t)} I(y_i = k)$ the frequency of label k at the node t.

- For example, if at node t, there are 100 samples with 70 labeled 1 and 30 labeled as 0. Then

$$p_{t1} = P(Y = 1|t) = \frac{1}{100} \sum_{i \in I(t)} I(y_i = 1) = 0.7$$

$$p_{t0} = P(Y = 0|t) = \frac{1}{100} \sum_{i \in I(t)} I(y_i = 0) = 0.3$$

The impurity function

Possible impurity functions $i(t)$ are

- Misclassification error: $1 - \max_k p_{tk}$
- Gini index: $\sum_k p_{tk}(1 - p_{tk}) = 1 - \sum_k p_{tk}^2$
- Cross entropy: $-\sum_k p_{tk} \log p_{tk}$

The impurity function

Possible impurity functions $i(t)$ are

- Misclassification error: $1 - \max_k p_{tk}$
- Gini index: $\sum_k p_{tk}(1 - p_{tk}) = 1 - \sum_k p_{tk}^2$
- Cross entropy: $-\sum_k p_{tk} \log p_{tk}$

Goodness of split:

$$\phi(s, t) = i(t) - p_L \times i(t_L) - p_R \times i(t_R),$$

where p_L and p_R are proportions of the samples in node t that go to the left child t_L and the right child of t_R, respectively

- A simple criterion: stop splitting a node t when

$$\max_{s \in \mathcal{S}} \frac{|I(t)|}{n} \phi(s, t) < \beta,$$

  where β is a pre-specified threshold
- The idea is to stop splitting when the node is sufficiently pure or sufficiently small

How to determine the prediction at terminal node?

- A class assignment rule assigns a class $k = \{1, \ldots, K\}$ to every terminal node $t \in \widetilde{T}$
- The class assigned to node $t \in \widetilde{T}$ is denoted by $\kappa(t)$

- For 0-1 loss, the class assignment rule is

$$\kappa(t) = \underset{k}{\operatorname{argmax}} \ \hat{p}_{tk}$$

Right-sized trees

- Denote the misclassification error of a tree T by $R(T)$, then

$$\text{The Training Error : } R(T) = \sum_{t \in \tilde{T}} p(t) i(t),$$

where $p(t)$ is the proportion of observations in node t, and $i(t)$ is defined by the misclassification error

- $R(T)$ is biased downward, as

$$p(t)i(t) \geq p(t_L)i(t_L) + p(t_R)i(t_R),$$

thus the larger a tree is, the smaller its misclassification error (Overfitting!)

Pruning

- Grow a very large tree T_{max}
  1. Until all terminal nodes are pure, containing only one class
  2. When the number of data in each terminal node is no greater than a certain threshold
  3. As long as the tree is sufficiently large

- A "selective" pruning procedure is needed
  1. The pruning is optimal in a certain sense
  2. The search for different ways of pruning should be of manageable computational load

Minimal cost pruning

- For any subtree T of T_{max}, define its cost function as

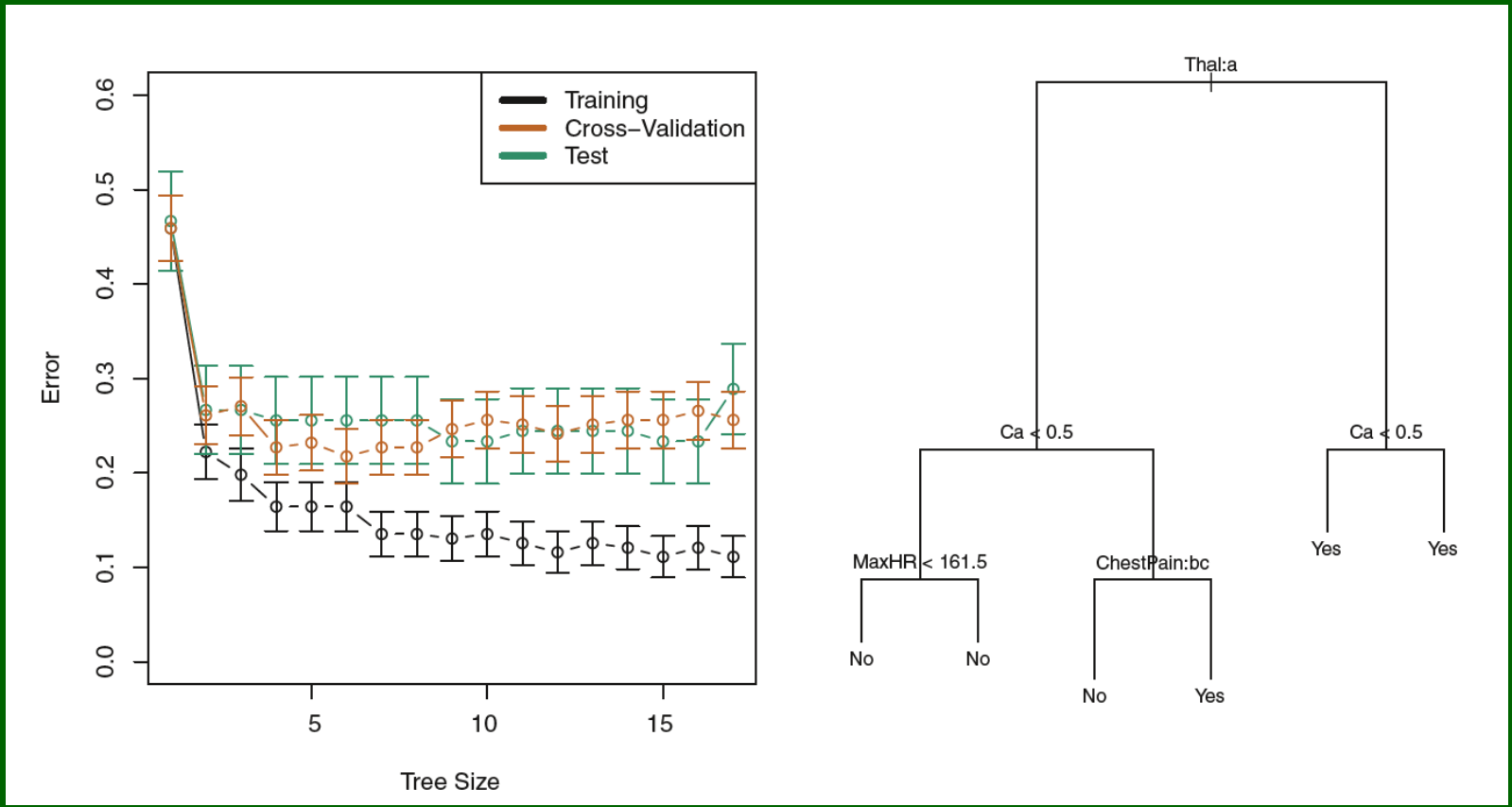
$$R_\lambda(T) = R(T) + \lambda|\widetilde{T}|,$$

  where λ is a tuning parameter
- For each λ, solve

$$T(\lambda) = \underset{T}{\operatorname{argmin}} \; R_\lambda(T)$$

  - To search for the pruning branch, many technique has been proposed, such as the weakest-link cutting
- Optimal λ can be determined by cross validation

Example of a pruned tree



Extension to regression tree

- $\mathcal{S}$ remains the same
- Impurity function: $i(t) = \sum_{i \in t}(y_i - \text{avg}(y|t))^2$
- Goodness of fit: $\phi(s, t) = i(t) - i(t_L) - i(t_R)$
- Stopping rule remains the same
- Response assignment rule: $\kappa(t) = \text{avg}(y|t)$

Tree-based methods

- Can handle continuous and categorical predictors and responses. So, they can be applied to both classification and regression problems.
- Tree-based methods involve segmenting the predictor space into a number of simple regions
- In each region, we typically use the mean or the mode of the training observations in the region to make prediction
- Region-wise constant, and completely nonparametric

Advantages of tree-based methods

- Easy to interpret
- Handles both categorical and ordered variables in a simple and natural way
- Automatic stepwise variable selection and complexity reduction
- It provides an estimate of conditional class probability
- It is invariant under all monotone transformations of individual ordered variables
- Very nice and intuitive graphical display

Issues of tree-based methods

- Categorical variable with q classes produces $2^q - 1$ possible splits
  - Simplification is possible for binary classification with Gini index or cross entropy, and regression with squared error loss
  - But it is unclear for multiclass classification
- Cost-sensitive (non-standard) classification with unequal misclassification costs
- Splits based on combined variables
- Lack of smoothness (blockwise constant)
- High variance, constructed tree is very sensitive to sample

Bagging (bootstrap aggregation)

Bagging is a technique to reduce the variance of an estimated prediction function

1. Draw bootstrap samples $(x_1^{*b}, y_1^{*b}), \ldots, (x_n^{*b}, y_n^{*b})$; $b = 1, \ldots, B$
2. For each bootstrap sample, fit the tree model $\hat{f}^{*b}(x)$
3. The bagging estimate is

$$\hat{f}_{bag}(x) = \begin{cases} \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x), & \text{for regression/classification;} \\ \text{mode}(\hat{f}^{*1}(x), \ldots, \hat{f}^{*B}(x)), & \text{for classification} \end{cases}$$

Why it works: some intuitions

- Pretend we draw bootstrap sample from the true distribution rather than $\mathcal{T}$ (so independent),

$$E(Y - \hat{f}(X))^2 \geq E(Y - \hat{f}_{bag}(X))^2,$$

as $\hat{f}_{bag}(x) = E_{\mathcal{T}}(\hat{f}(x))$

- Key statistical idea is averaging reduces variance
- "Wisdom of crowds": the collective knowledge of a diverse and independent body of people exceeds the knowledge of any single individual

Random forest

Random forest is similar to bagging, but different as it averages *de-correlated* trees

1. Draw bootstrap samples $(x_1^{*b}, y_1^{*b}), \ldots, (x_n^{*b}, y_n^{*b})$
2. For each bootstrap sample, grow a tree by repeating:
   a. Randomly select a subset of the p variables
   b. Pick the best split among the chosen subset
   c. Split the node
3. The random forest estimate is constructed similarly as in bagging (average for regression, and majority vote for classification)

Why it works: some intuitions

- In bagging, the bootstrap trees are correlated, and correlation limits the benefit of averaging
- Averaging B variables with variance σ^2 and correlation ρ yields variance

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

- In random forest, the correlation is reduced by selection of variables in growing trees
- The price it has to pay is the increment of bias and variance, which, fortunately, is usually small

Boosting

- Boosting also involves growing multiple trees, but sequentially
- Each tree is grown using information from previously grown trees

- Iteratively learning weak classifiers
- Final result is the weighted sum of the results of each weak classifiers
- Many different boosting algorithms, and adaboost (adaptive boosting) is the first
- Gradient boosting machine (GBM) is now one of the most competitive approach; XGBoost

Adaboost for binary classification trees

1. Initialize $w_{1,i} = 1/n$; $i = 1, \ldots, n$
2. For $m = 1$ to M:
   a. Fit a weak classifier $h_m(\mathbf{x}) : \mathcal{R}^p \to \{-1, 1\}$ to the training data with weights $w_{m,i}$
   b. Compute weighted misclassification error:

$$e_m = \sum_{i=1}^{n} w_{m,i} I(y_i \neq h_m(\mathbf{x}_i))$$

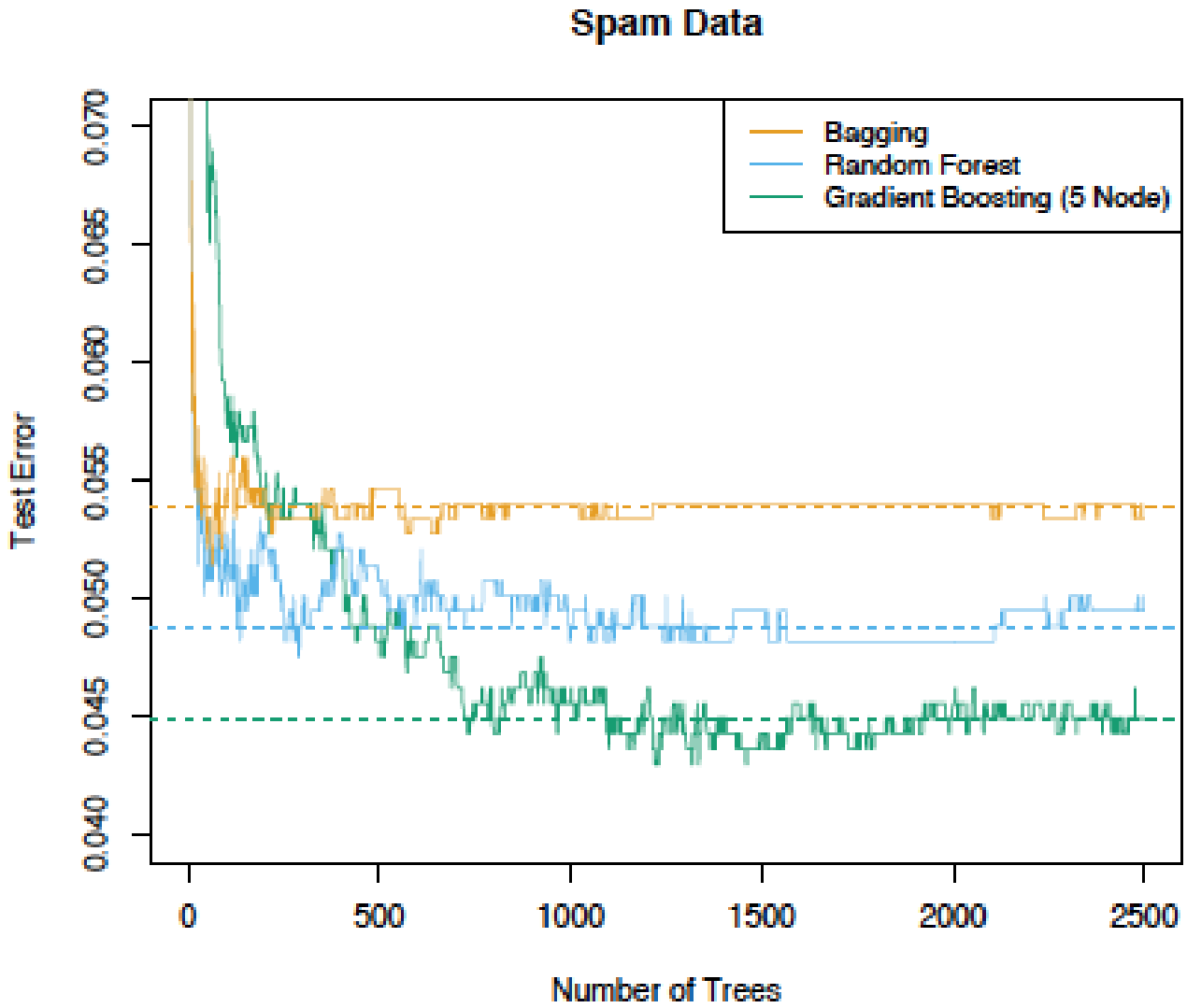
   c. Compute $\alpha_m = \frac{1}{2} \log((1 - e_m)/e_m)$
   d. Update $w_{m+1,i} = w_{m,i} \exp(-y_i \alpha_m h_m(\mathbf{x}_i))/Z_m$, where $Z_m = \sum_i w_{m,i} \exp(-y_i \alpha_m h_m(\mathbf{x}_i))$
3. Output $f(x) = \frac{\sum_m \alpha_m h_m(\mathbf{x})}{\sum_m \alpha_m}$ and $G(\mathbf{x}) = \text{sign}(f(\mathbf{x}))$

Boosting for regression trees

- Initialize $f_0(\mathbf{x}) = \bar{y}$
- Compute $g_m(x_i) = y_i - f_{m-1}(\mathbf{x}_i) = r_i$
- Fit a regression tree h_m to the training data $(\mathbf{x}_i, r_i)$
- Update $f_m(x) = f_{m-1}(\mathbf{x}) + \alpha h_m(\mathbf{x})$, and iterate

Remarks: This boils down to the standard approach of iteratively fitting the residuals

Spam detection



Variable Importance

Random Forests and bagged trees are a black box. We can't see what's going on. But "Importance" is a statistic that attempts to give us some insight.

- Permutation Importance or Mean Decrease in Accuracy (MDA)
- Gini Importance or Mean Decrease in Impurity (MDI)

Variable Importance: Permutation Importance

- Suppose that we have trained a random forest and we have validation data.
- We record the prediction error of $\widehat{f}(\boldsymbol{x})$ on the validation data. This serves as a baseline.
- Next, a variable is "taken out" by having all of its values permuted in the validation dataset. We then compute the prediction error of $\widehat{f}(\boldsymbol{x})$ on the perturbed validation data.
- The importance statistic is the difference between the error due to the perturbation and the baseline. A larger difference means an important predictor for $\widehat{f}(\boldsymbol{x})$.

Variable Importance: Gini Importance

Consider a parent node t and two child nodes t_L and t_R, resulting from splitting at value .

- Gini impurity index of node t: , where is the proportion of training observations in a region that are from class .
- $j(s) = q_{l,s} i(t_{l,s}) + q_{r,s} i(t_{r,s})$ used to evaluate the split s, where and $q_{r,s}$ denotes the proportions of observations.
- The decrease in impurity, I is defined as: $I = i(t) - j(s)$.

To calculate the actual mean decrease in impurity importance metric for each variable, values would be averaged over all splits in the forest involving the variable.

Variable importance