Classification: Tree-based methods

Rui Zhu

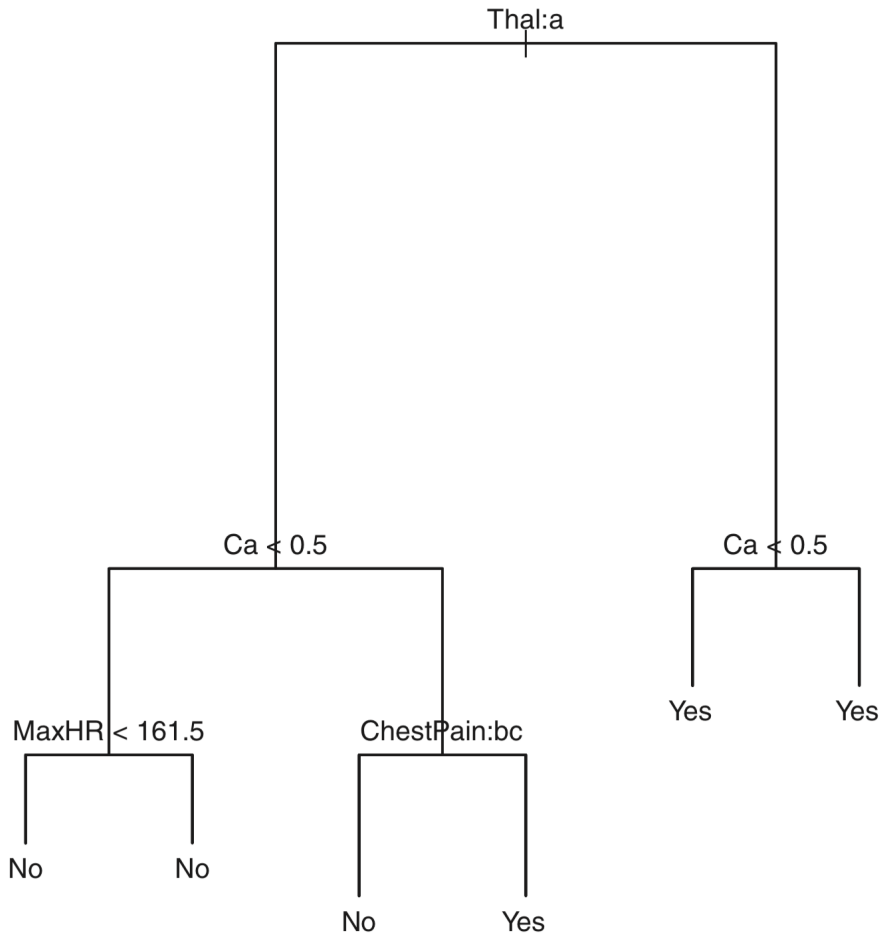
Overview

1. Decision trees
2. Bagging, Random forests, Boosting

Decision trees

- Stratify or segment the predictor space (feature space) into a number of simple regions.
- Simple and useful for interpretation
- Nice graphical representation

Example: Heart data



Basics

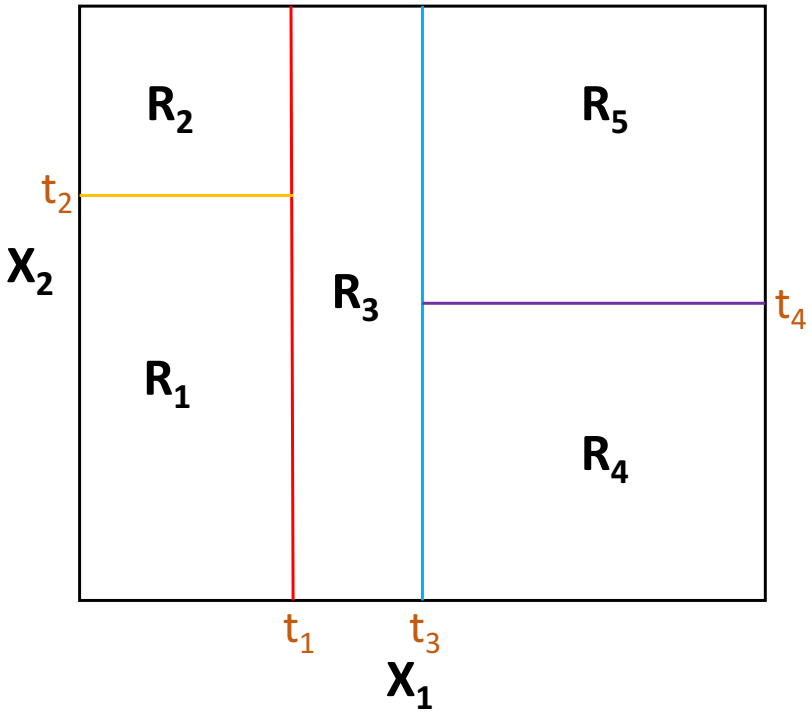
Elements in a tree:

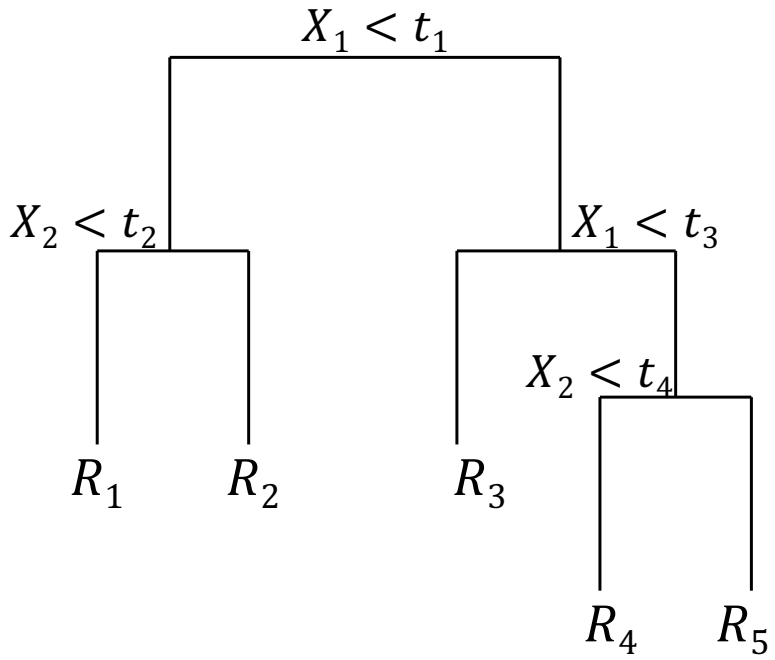
- Terminal nodes/leaves
- Internal nodes
- Branches

Decision trees:

- Upside down: the leaves are at the bottom of the tree

Stratification of the feature space





How to build decision trees?

Two steps:

- We divide the feature space into J distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.
- For every observation that falls into the region R_j, we predict that it belongs to the most commonly occurring class of training observations in R_j.

How to get the regions?

Aim: divide the predictor space into high-dimensional rectangles, or boxes, for simplicity and interpretation ability.

Find boxes $R_1, R_2, \ldots, R_J$ that minimize a criterion, for example, the error rate. [*More about the criterion later*]

However, it is computationally infeasible to consider every possible partition of the feature space into J boxes.

Recursive binary splitting: a top-down, greedy approach.

How to get the regions?

Recursive binary splitting: a top-down, greedy approach.

- Top-down: We begin at the top of the tree, all observations belong to a single region. We then successively splits the predictor space; each split is indicated via two new branches further down on the tree.
- 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.

Criterion for binary split

- Classification error rate:

$$E = 1 - \max_k(\hat{p}_{mk})$$

However, it is not sufficiently sensitive for tree-growing. Alternatively,we usually use

- Gini index:

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

A measure of total variance across the K classes; node purity.

- Entropy:

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

Criterion for binary split

- The Gini index and the entropy are quite similar numerically.
- Either the Gini index or the entropy are typically used to evaluate the quality of a particular split, since these two approaches are more sensitive to node purity than is the classification error rate.
- In interpreting the results of a classification tree, we are often interested not only in the class prediction corresponding to a particular terminal node region, but also in the class proportions among the training observations that fall into that region (node purity).

How to get the regions?

The first step: We select the feature X_j and the cutpoint s such that the regions

$$\{X|X_j < s\}, \quad \{X|X_j \geqslant s\}$$

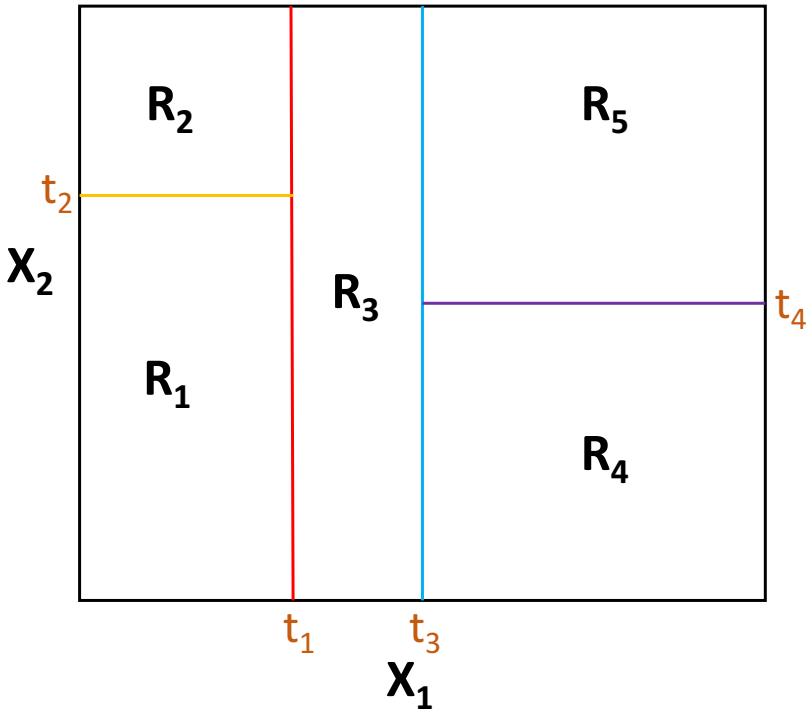
lead to the greatest possible reduction in G or D.

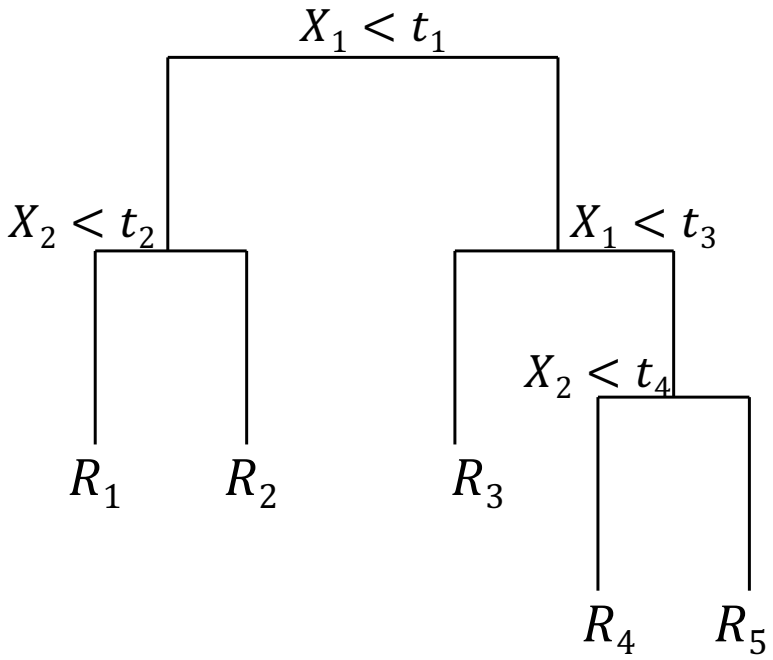
We consider all features $X_1, \ldots, X_p$, and all possible values of the cutpoint s for each of the features, and then choose the feature and cutpoint such that the resulting tree has the lowest G or D.

Next: we look for the best predictor and best cutpoint in order to split the data further so as to minimize G or D within each of the resulting regions.

Stop: a stopping criterion is reached, e.g. no region contains more than five observations.

Decision trees





Tree pruning

The resulting tree might be too complex: overfitting. We usually need a smaller tree.

Grow a very large tree T_0 and then prune it back to get a subtree. However, it is too cumbersome to consider every possible subtree. We need a way to select a small set of subtrees for consideration.

Cost complexity pruning (weakest link pruning): we consider a sequence of trees indexed by a nonnegative tuning parameter α.

Tree pruning: Cost complexity pruning

For each α, we can find a subtree $T \subset T_0$ by

$$\min \sum_{m=1}^{|T|} \text{criterion} + \alpha|T|$$

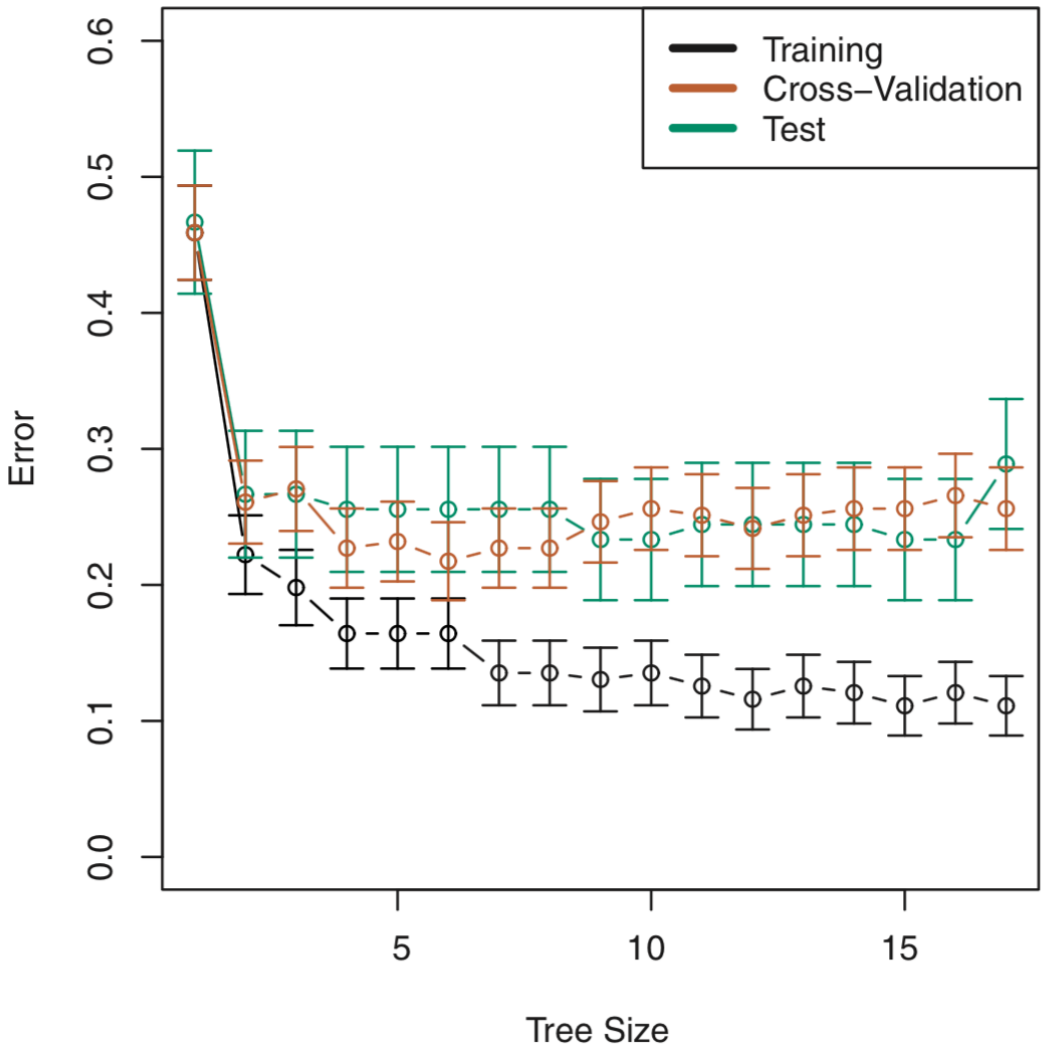
- Criterion: E, G or D
- $|T|$: number of leaves
- $\alpha = 0$: $T = T_0$
- As α increases, the quantity will tend to be minimised for a smaller subtree.
- For a sequence values of α, we can get a sequence of subtrees. Cross-validation to choose α.

Decision trees: Algorithm

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, e.g. 5.
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 criterion on the data in the left-out kth fold, as a function of α.

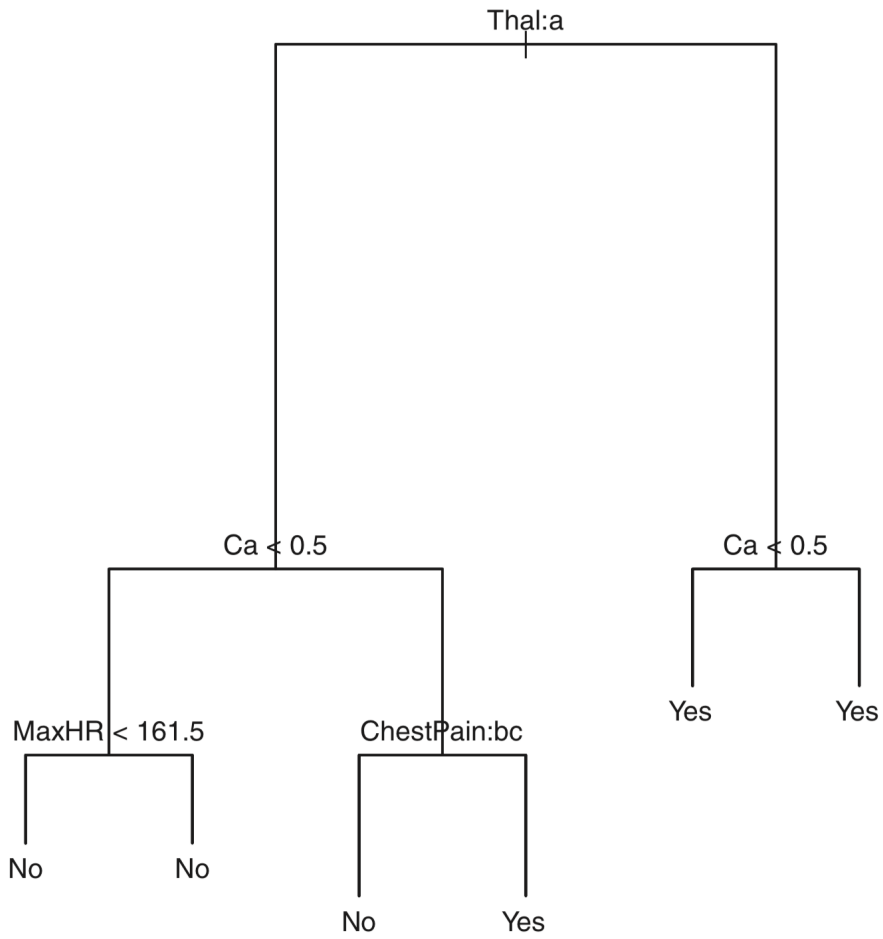
   Average the results for each value of α, and pick α to minimize the average error.
4. Output the subtree from Step 2 that corresponds to the chosen value of α.

Decision trees



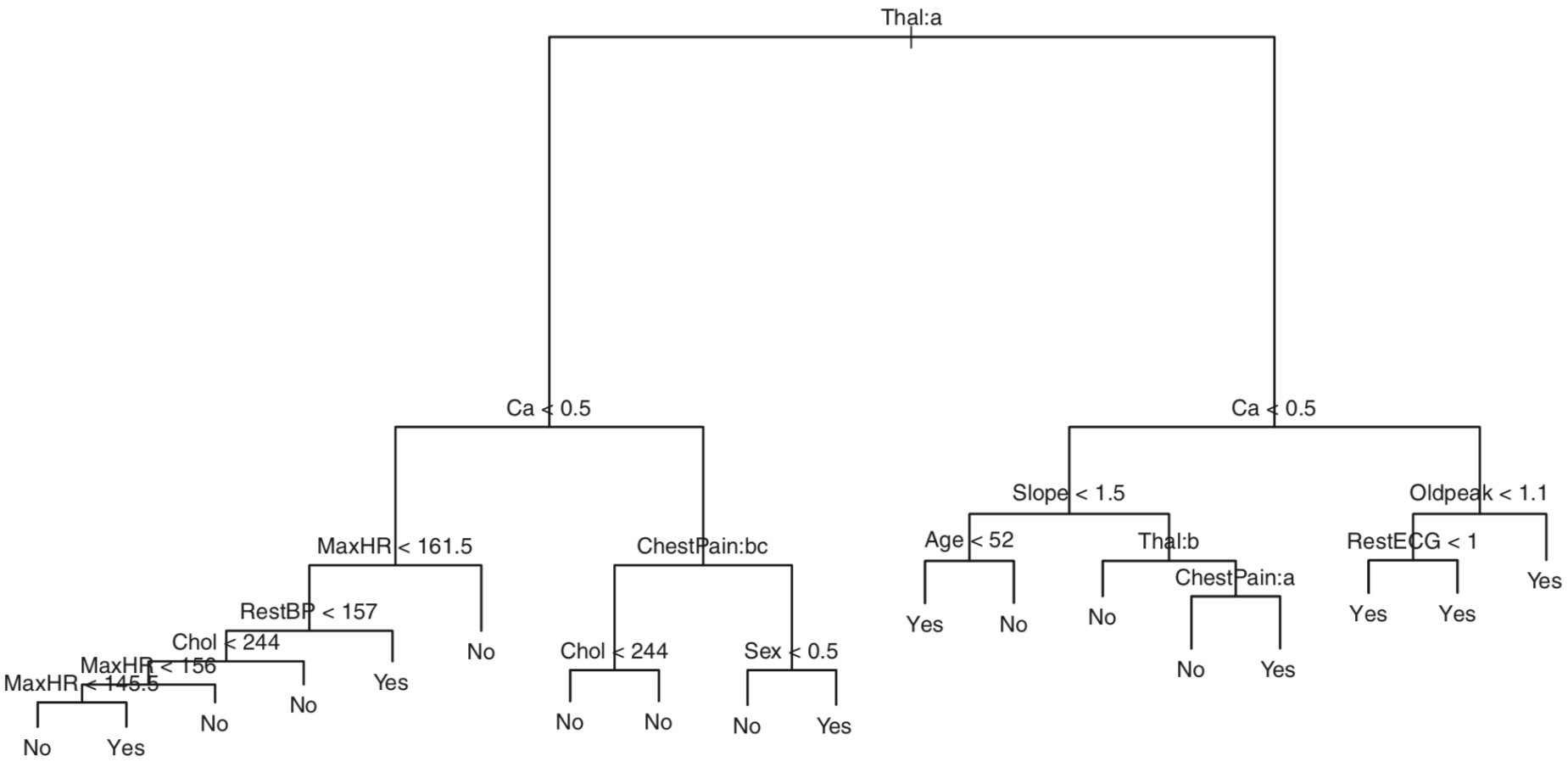
Decision trees

Can be directly applied to qualitative features.



Decision trees

Importance of node purity for the interpretation classification results.



Decision trees

Advantages:

- Easy interpretation.
- Nice graphical representation.
- Can be directly applied to qualitative features.

Disadvantages:

- Not competitive with the best supervised learning approaches in terms of prediction accuracy.
- Can be very non-robust.

By aggregating many decision trees, the predictive performance of trees can be substantially improved, at the expense of some loss in interpretation.

Three methods to talk about: bagging, random forests, boosting

Bagging

Decision trees suffer from high variance!

To reduce variance: averaging a set of observations.

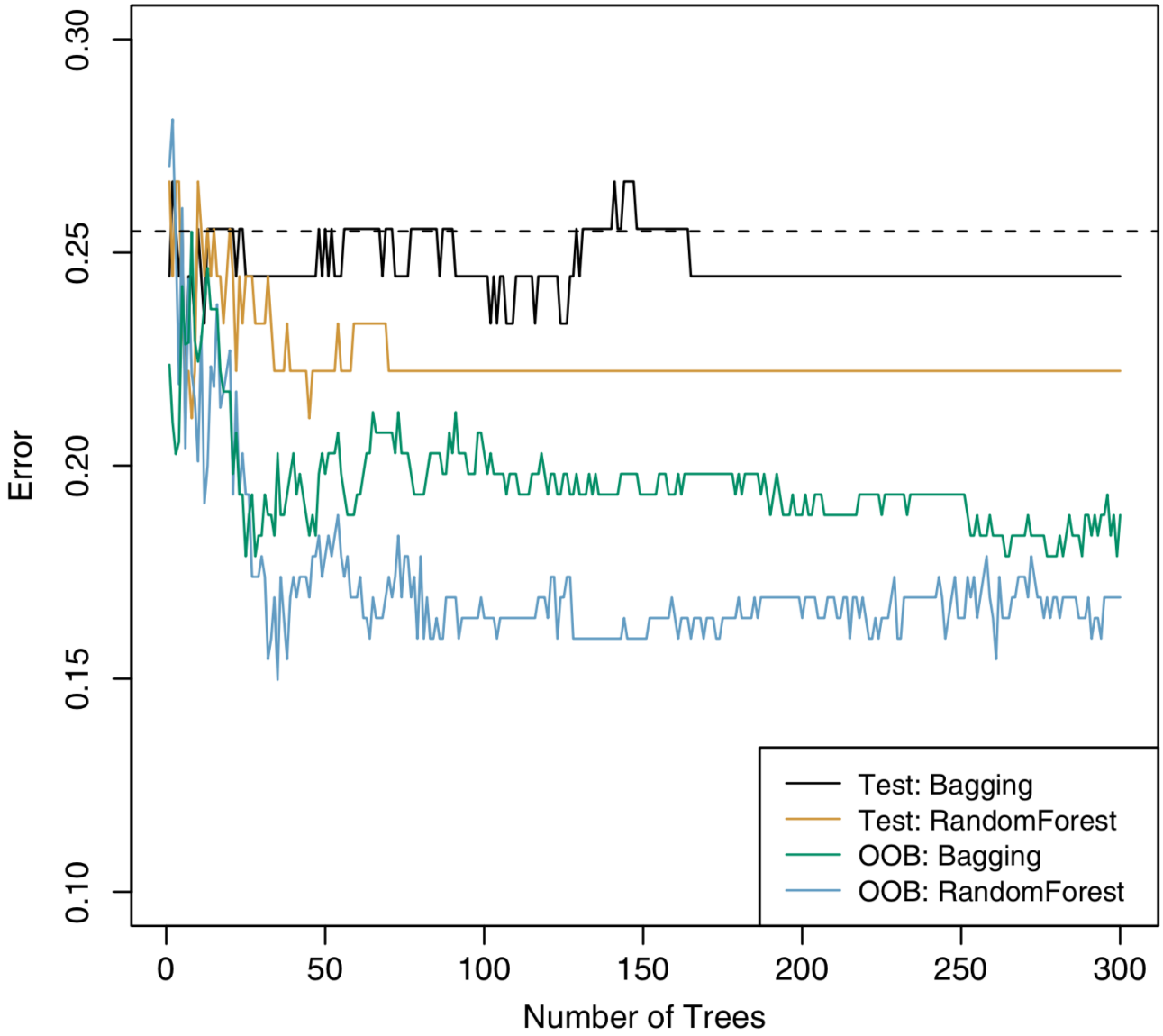
Bagging

Bootstrap aggregation (Bagging): Take repeated samples from the training data and get B different bootstrapped training datasets.

- Train B classification trees on the B training datasets.
- Predict B classification results for a test instance.
- Majority vote.

Use B that is sufficiently large.

Bagging


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

Bagging: OOB observations

Estimate test error by the out-of-bag (OOB) observations to avoid high computational cost by cross-validation.

We can predict the class for the ith observation using each of the trees in which that observation was OOB.

An OOB prediction can be obtained in this way for each of the n observations, from which the overall OOB classification error can be computed.

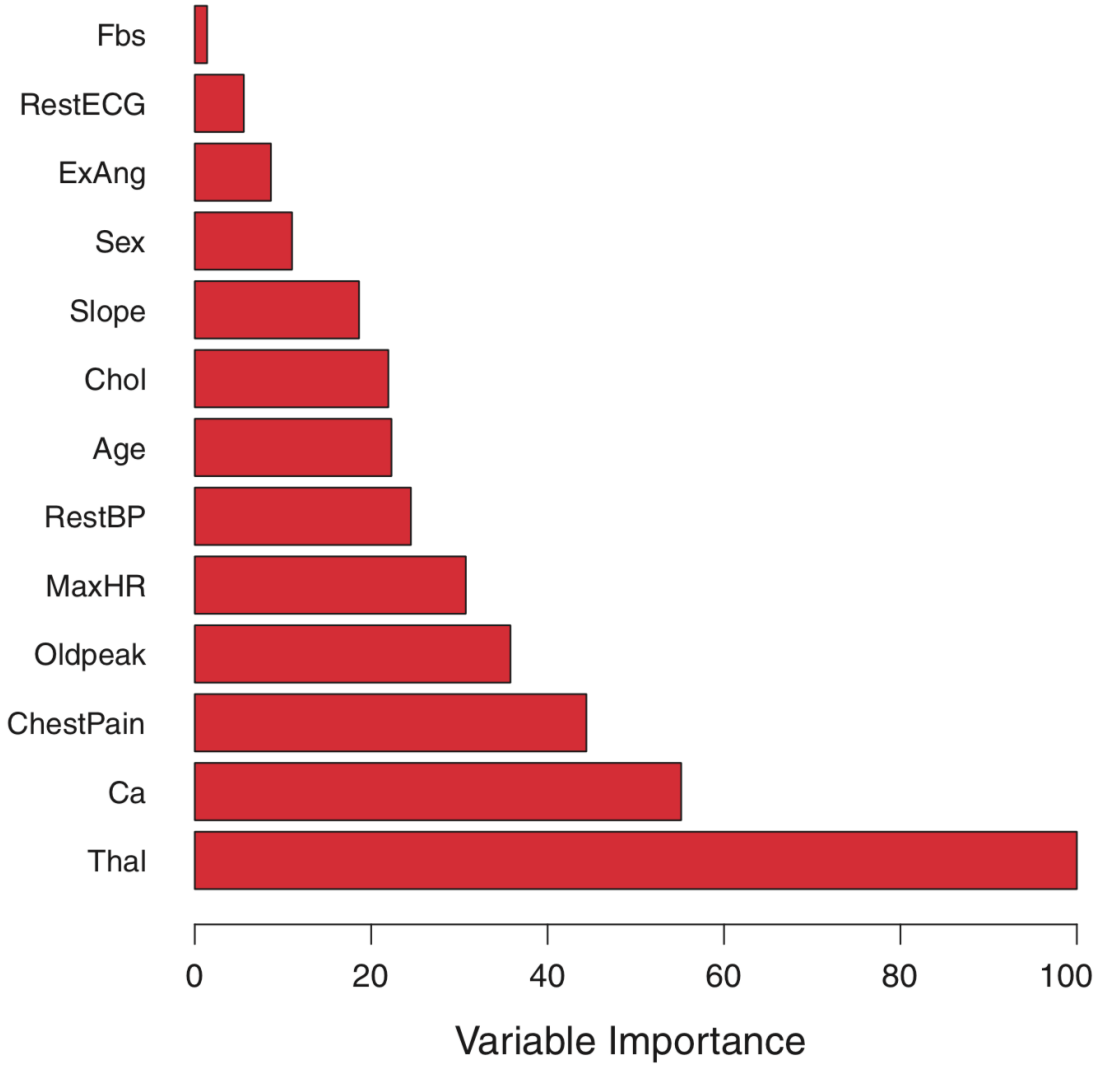
The resulting OOB error is a valid estimate of the test error for the bagged model, since the response for each observation is predicted using only the trees that were not fit using that observation.

Bagging: Variable importance measures

Variable importance measures:

We can add up the total amount that the Gini index is decreased by splits over a given predictor, averaged over all B trees.

Bagging: Variable importance measures



Random forests

Averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities.

Aim: decorrelate the trees.

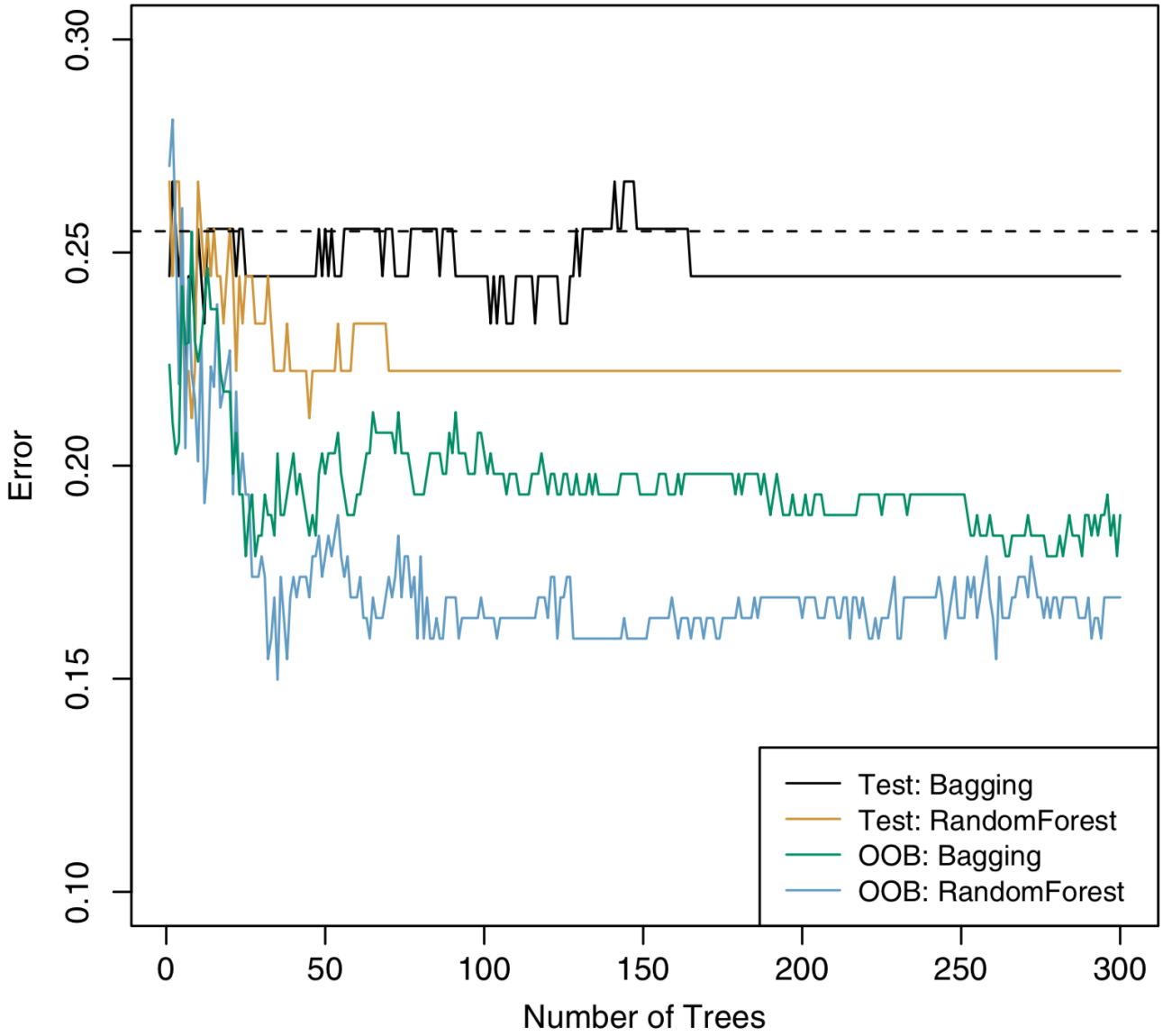
We build B decision trees on bootstrapped training samples.

When building these decision trees, each time we split it, a random sample of m features is chosen as split candidates from the full set of p features.

A fresh sample of m features is taken at each split, and typically we choose $m = \sqrt{p}$.

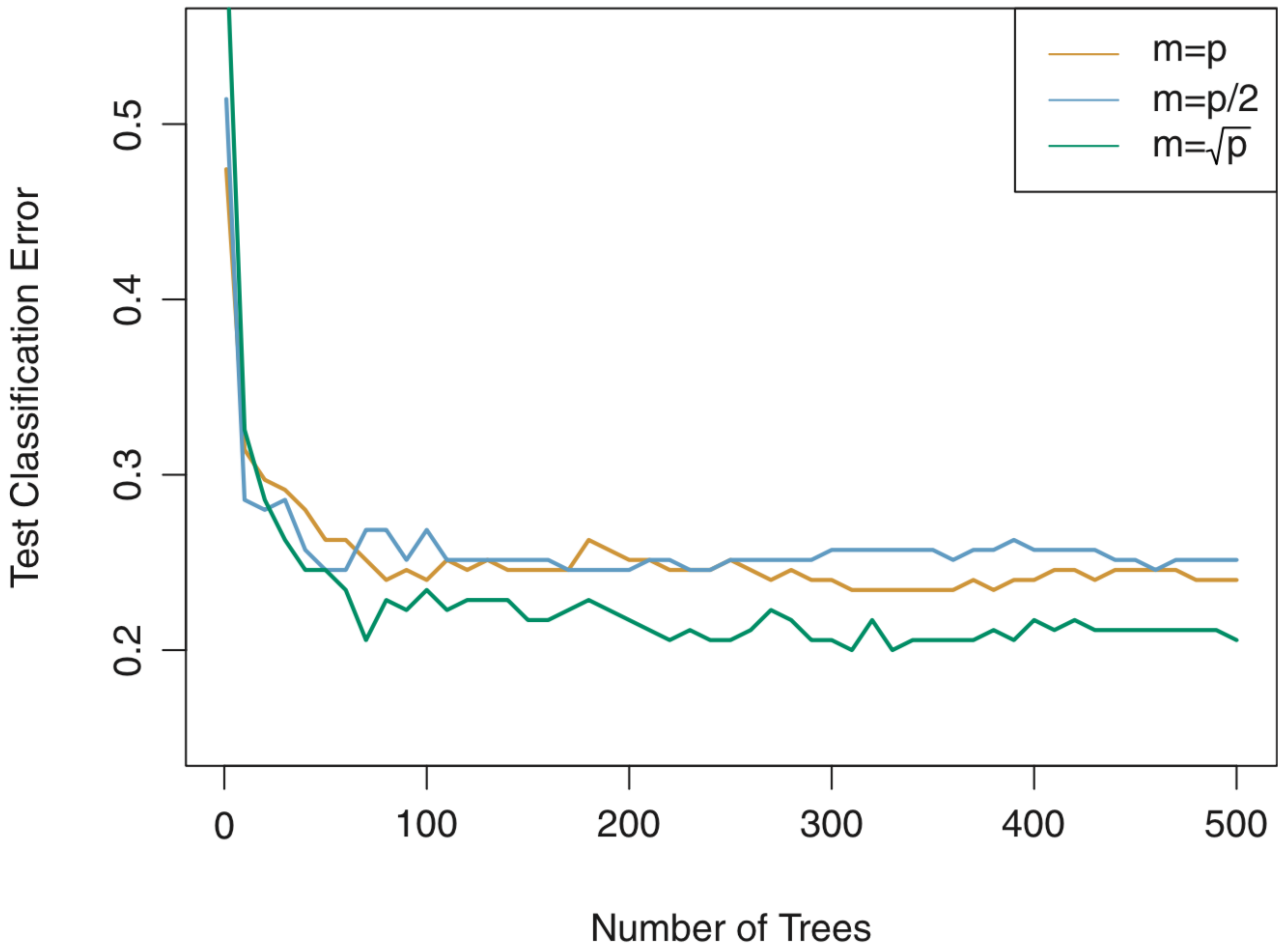
If $m = p$, bagging=random forests.

Random forests



Random forests

Small m is helpful when we have a large number of correlated predictors, e.g. gene expression data.



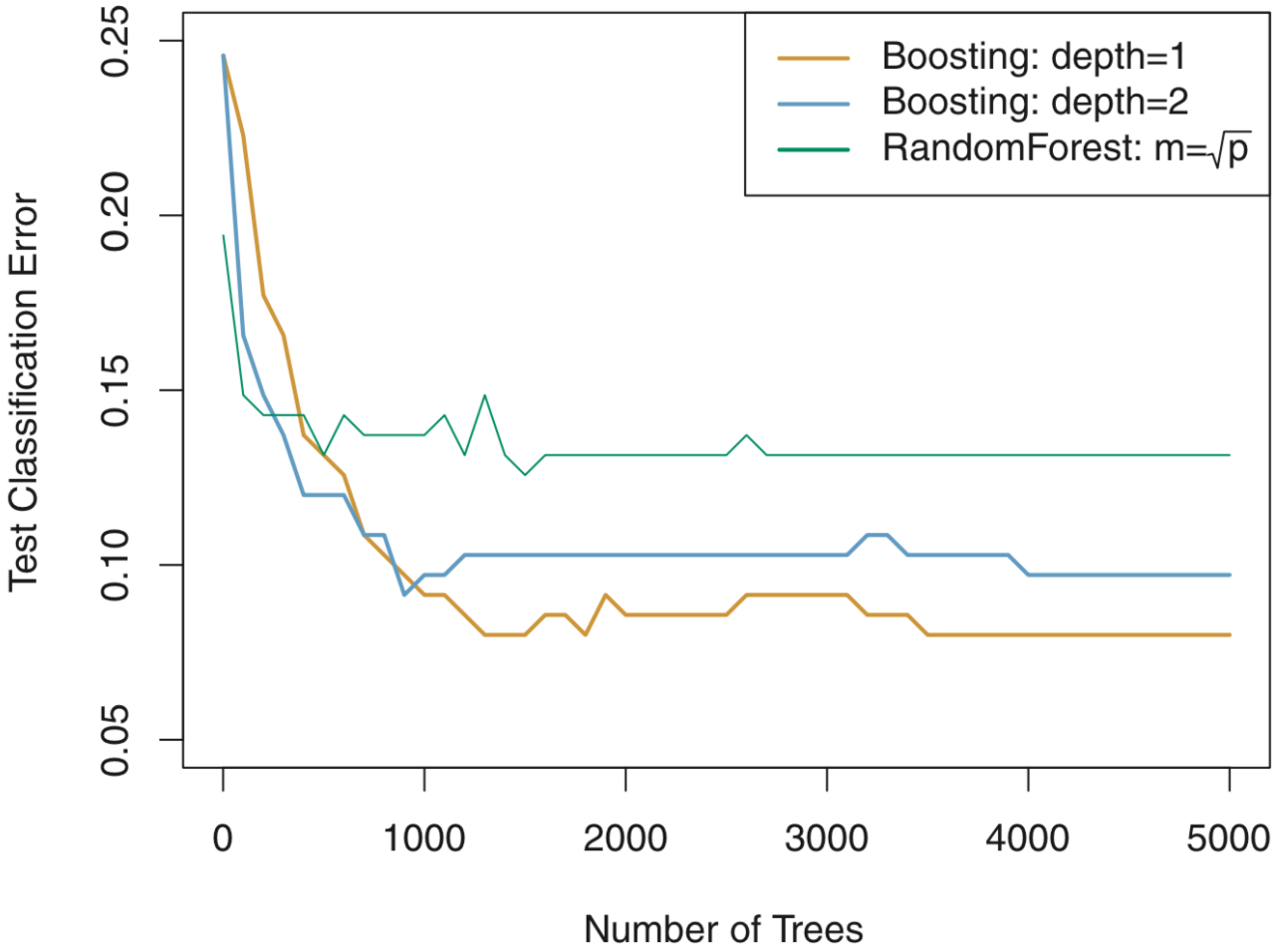
Boosting

- The trees are grown sequentially: each tree is grown using information from previously grown trees.
- Each tree is fit on a modified version of the original dataset.
- Learns slowly.

Tuning parameters:

- B: number of trees
- λ: shrinkage parameter, usually 0.01 or 0.001
- d: number of splits in each tree, interaction depth, usually 1 works well

Boosting


Boosting: depth=1
Boosting: depth=2
RandomForest: m=√p
Test Classification Error
0.05
0.10
0.15
0.20
0.25
0
1000
2000
3000
4000
5000
Number of Trees