Lecture 15: Bagging, Boosting & Random Forest

BIOS635

02/27/2020

Some slides taken/modified from other people's lectures online.

Advantages and disadvantages of trees

- ▲ Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!
- ▲ Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous chapters.
- ▲ Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- ▲ Trees can easily handle qualitative predictors without the need to create dummy variables.
- ▼ Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches seen in this book.

However, by aggregating many decision trees, the predictive performance of trees can be substantially improved. We introduce these concepts next.

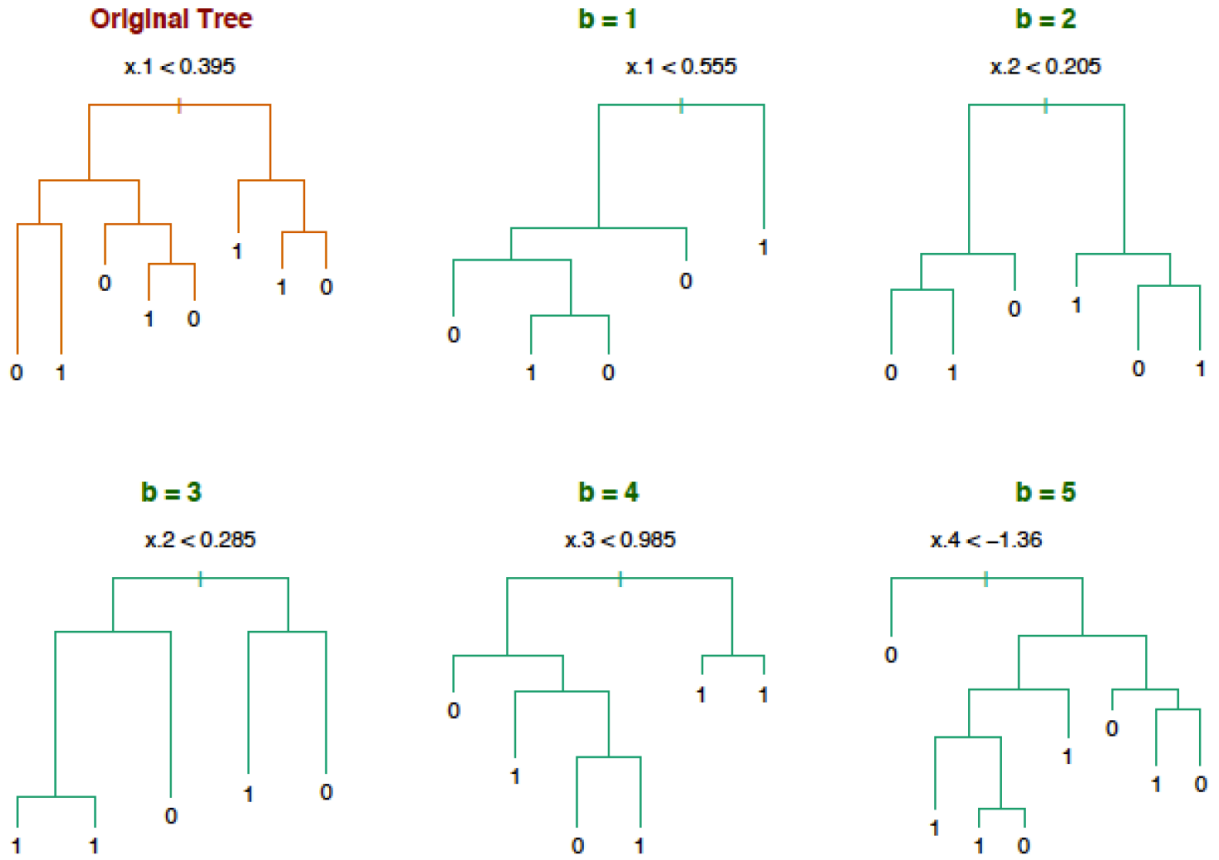
A summary of our methods so far

Method	Interpretable	Flexible	Makes assumptions?
Logistic regression	Yes	Extensible	Yes
k-NN	No	Highly	No
LDA/QDA	Sometimes	No	Yes
Trees	Extremely	Somewhat	No

- Decision trees are perhaps the most Interpretable method we've seen so far
- Trees don't assume any particular relationship between the response Y and the inputs X_j, and large trees are quite flexible
- So what's the catch?
- Turns out, Trees tend to be rather poor predictors/classifiers!
- We can fix this, if we're willing to give up Interpretability

Why are decision trees poor predictors?

- Decision trees tend to have high variance. A small change in the training data can produce big changes in the estimated Tree.



Bagging

- *Bootstrap aggregation*, or *bagging*, is a general-purpose procedure for reducing the variance of a statistical learning method; we introduce it here because it is particularly useful and frequently used in the context of decision trees.
- Recall that 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 σ^2/n.
- In other words, *averaging a set of observations reduces variance*. Of course, this is not practical because we generally do not have access to multiple training sets.

Bagging, cont'd

- Instead, we can bootstrap, by taking repeated samples from the (single) training data set.
- In this approach we generate B different bootstrapped training data sets. We then train our method on the bth bootstrapped training set in order to get $\hat{f}^{*b}(x)$, the prediction at a point x. We then average all the predictions to obtain

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

This is called *bagging*.

Bagging classification trees

- The above prescription applied to regression trees
- For classification trees: for each test observation, we record the class predicted by each of the B trees, and take a *majority vote*: the overall prediction is the most commonly occurring class among the B predictions.

Out-of-bag error estimation

- Recall, each bootstrap sample contains roughly 2/3 ($\approx 63.2\%$) of the of the training observations

- The remaining observations not used to fit a given bagged tree are called the out-of-bag (OOB) observations

- Another way of thinking about it: Each observation is OOB for roughly $B/3$ of the trees. We can treat observation i as a test point each time it is OOB.

- To form the OOB estimate of test error:
  - Predict the response for the ith observation using each of the trees for which i was OOB. This gives us roughly $B/3$ predictions for each observation.
  - Aggregate predictions into a single prediction for observation i and calculate the error of this aggregated prediction
  - Average all of the errors

Random forests

- *Random forests* provide an improvement over bagged trees by way of a small tweak that *decorrelates* the trees. This reduces the variance when we average the trees.
- As in bagging, we build a number of decision trees on bootstrapped training samples.

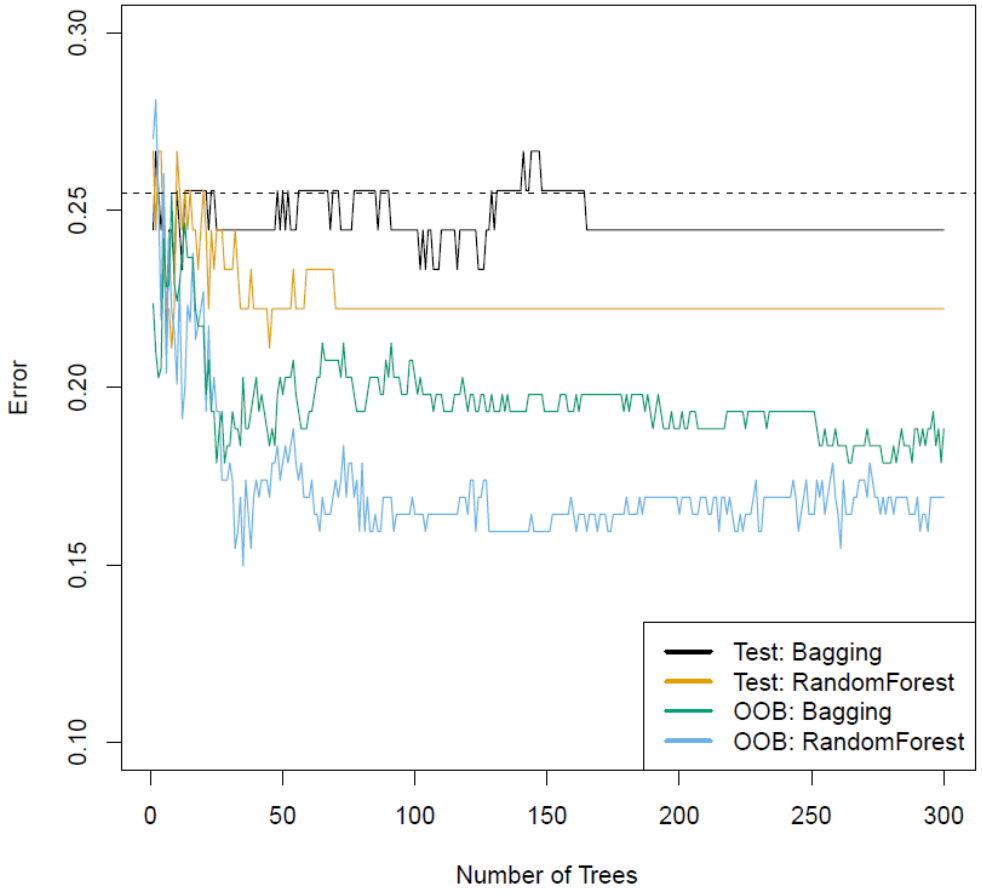
Random forests

- *Random forests* provide an improvement over bagged trees by way of a small tweak that *decorrelates* the trees. This reduces the variance when we average the trees.
- As in bagging, we build a number of decision trees on bootstrapped training samples.
- But when building these decision trees, each time a split in a tree is considered, *a random selection of m predictors* is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors.

Random forests

- *Random forests* provide an improvement over bagged trees by way of a small tweak that *decorrelates* the trees. This reduces the variance when we average the trees.
- As in bagging, we build a number of decision trees on bootstrapped training samples.
- But when building these decision trees, each time a split in a tree is considered, *a random selection of m predictors* is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors.
- A fresh selection of m predictors is taken at each split, and typically we choose $m \approx \sqrt{p}$ — that is, the number of predictors considered at each split is approximately equal to the square root of the total number of predictors (4 out of the 13 for the Heart data).

Bagging/random forest on the heart data



Bagging and random forest results for the Heart data.

- The test error (black and orange) is shown as a function of B, the number of bootstrapped training sets used.
- Random forests were applied with $m = \sqrt{p}$.
- The dashed line indicates the test error resulting from a single classification tree.
- The green and blue traces show the OOB error, which in this case is considerably lower

Example: gene expression data

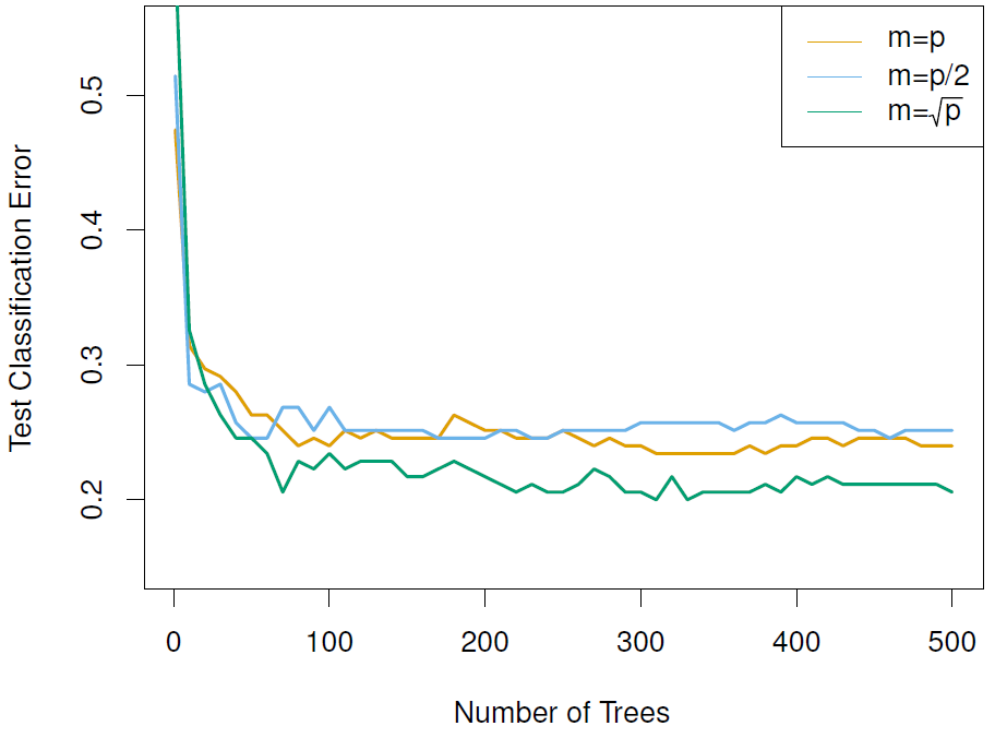
- We applied random forests to a high-dimensional biological data set consisting of expression measurements of 4,718 genes measured on tissue samples from 349 patients.
- There are around 20,000 genes in humans, and individual genes have different levels of activity, or expression, in particular cells, tissues, and biological conditions.
- Each of the patient samples has a qualitative label with 15 different levels: either normal or one of 14 different types of cancer.

Example: gene expression data

- We applied random forests to a high-dimensional biological data set consisting of expression measurements of 4,718 genes measured on tissue samples from 349 patients.
- There are around 20,000 genes in humans, and individual genes have different levels of activity, or expression, in particular cells, tissues, and biological conditions.
- Each of the patient samples has a qualitative label with 15 different levels: either normal or one of 14 different types of cancer.
- We use random forests to predict cancer type based on the 500 genes that have the largest variance in the training set.
- We randomly divided the observations into a training and a test set, and applied random forests to the training set for three different values of the number of splitting variables m.

"Unsupervised" screening of features.

Results: gene expression data



- Curves show Test misclassification rates for a 15-class problem with $p = 500$ predictors and under 200 observations used for training
- x-axis gives number of trees (number of bootstrap samples used)
- $m = p$ corresponds to bagging.
- A single classification tree has an error rate of 45.7%.

Summary: random forests

- Random forests have two tuning parameters:
  - $m =$ the number of predictors considered at each split
  - $B =$ the number of trees (number of bootstrapped samples)

- Increasing B helps decrease the overall variance of our estimator

- $m \approx \sqrt{p}$ is a popular choice

- Cross-validation can be used across a grid of (m, B) values to find the choice that gives the lowest CV estimate of test error

- Out-of-bag (OOB) error rates are commonly reported
  - Each observation is OOB for around $B/3$ of the trees
  - Can get a test error estimate for each observation each time it is OOB
  - Average over all the OOB errors to estimate overall test error

- RF's are parallelizable: You can distribute the computations across multiple processors and build all the trees *in parallel*

Random forests v.s. trees

- We liked trees because the model fit was very easy to understand
  - Large trees wind up hard to interpret, but small trees are highly interpretable
- With random forests, we're averaging over a bunch of bagged trees, and each tree is built by considering a small random subset of predictor variables at each split.
- This leads to a model that's *essentially* uninterpretable
- The Good: Random forests are very flexible and have a *somewhat justifiable* reputation for not overfitting
  - Clearly RF's can overfit: If we have 1 tree and consider all the variables at each split, $m = p$, we just have a single tree
  - If $m \ll p$ and the number of trees is large, RF's tend not to overfit
  - You should still use CV to get error estimates and for model tuning! Don't simply rely on the reputation RF's have for not overfitting.

Boosting

- Recall that bagging involves creating multiple copies of the original training data set using the bootstrap, fitting a separate decision tree to each copy, and then combining all of the trees in order to create a single predictive model.
- Notably, each tree is built on a bootstrap data set, independent of the other trees.
- Boosting works in a similar way, except that the trees are grown *sequentially*: each tree is grown using information from previously grown trees.

Boosting algorithm 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:
   - 2.1 Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).
   - 2.2 Update $\hat{f}$ by adding in a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x).$$

   - 2.3 Update the residuals,

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

3. Output the boosted model,

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

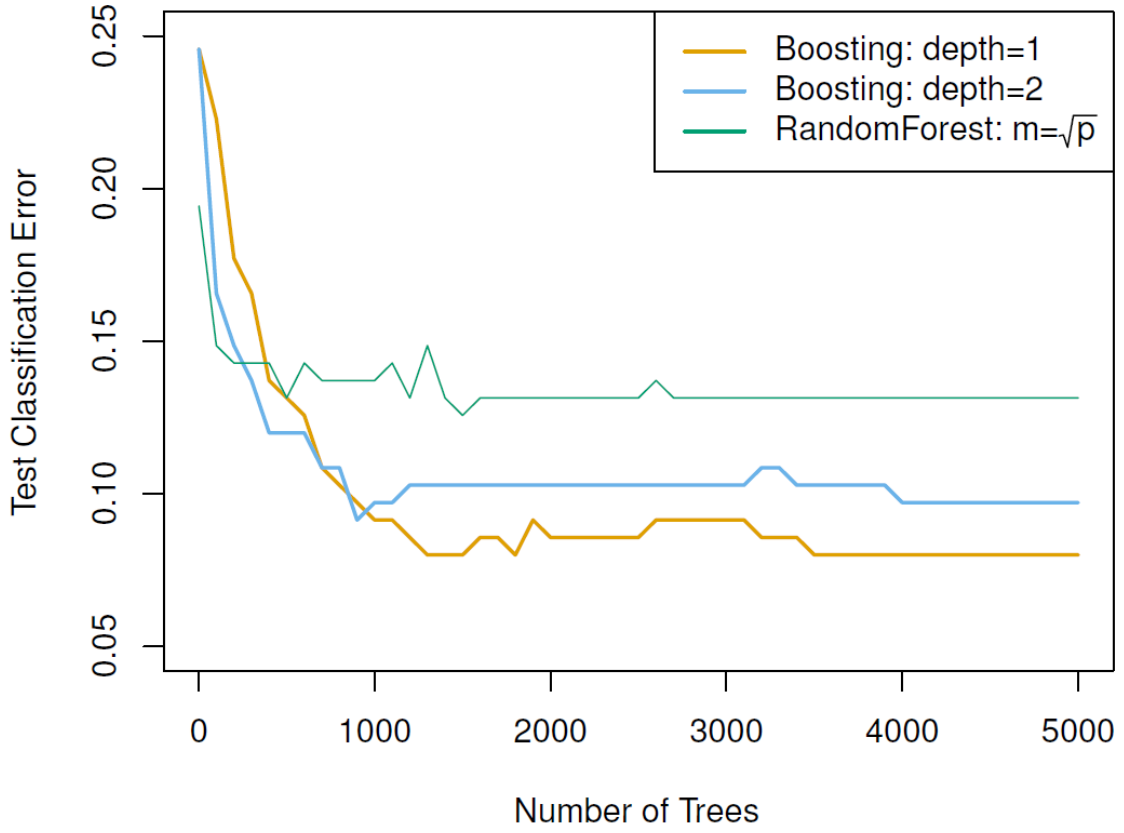
What's boosting trying to do?

- Boosting works best if d (the size of each tree) is small
- Given the current model, we fit a decision tree to the *residuals* from the model
- This new tree helps us perform just a little bit better in places where the current model wasn't doing well
- Unlike bagging, where each tree is large and tries to model the entire (bootstrapped) data well, each tree in boosting tries to incrementally improve the existing model
- Think of boosting as learning slowly: We use small trees, try to make incremental improvements, and further slow down the learning process by incorporating the *shrinkage parameter* λ

Boosting for classification

- Boosting for classification is similar in spirit to boosting for regression, but is a bit more complex.
- The R package `gbm` (gradient boosted models) handles a variety of regression and classification problems.

Gene expression data, cont'd



- Results from performing boosting and random forests on the fifteen-class gene expression data set in order to predict *cancer* versus *normal*.

Tuning parameters for boosting

1. The *number of trees* B. Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select B.

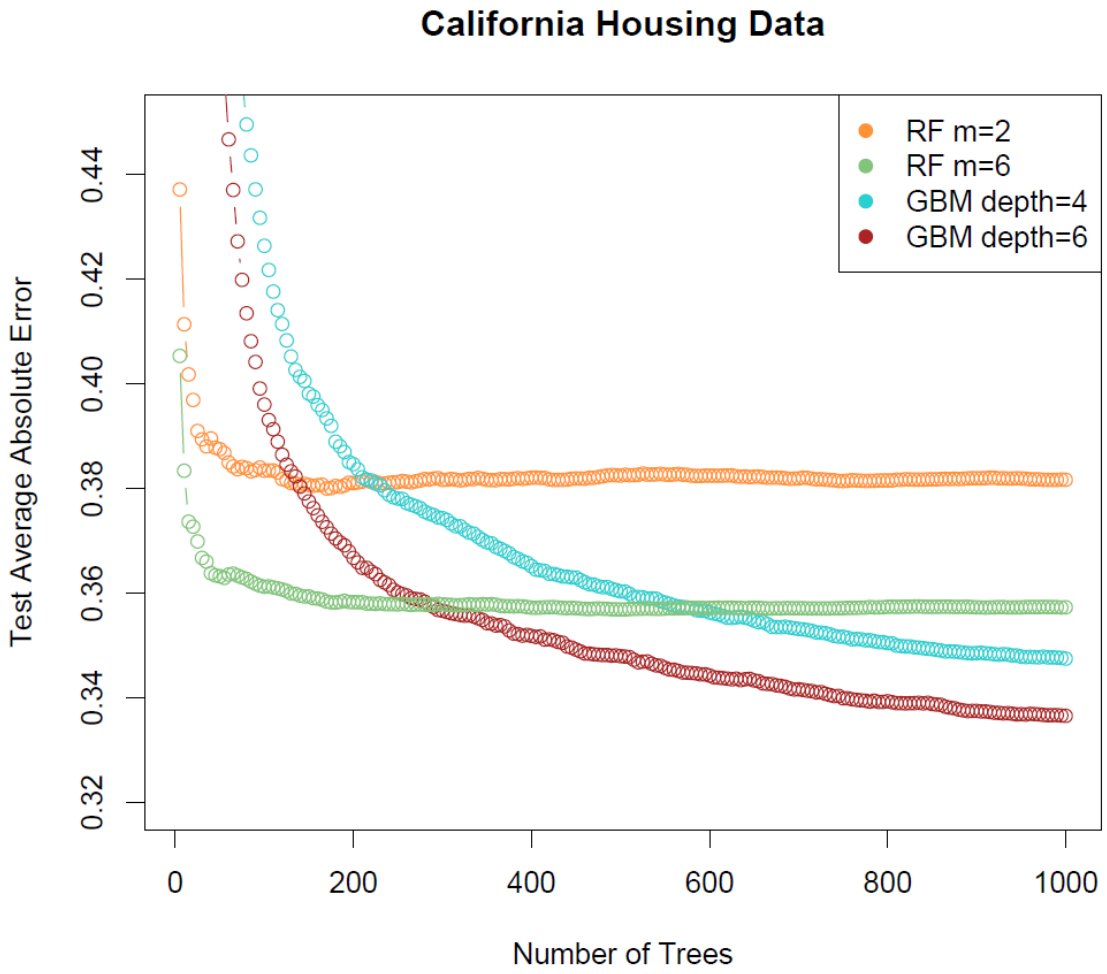
Tuning parameters for boosting

1. The *number of trees* B. Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select B.
2. The *shrinkage parameter* λ, a small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem. Very small λ can require using a very large value of B in order to achieve good performance.

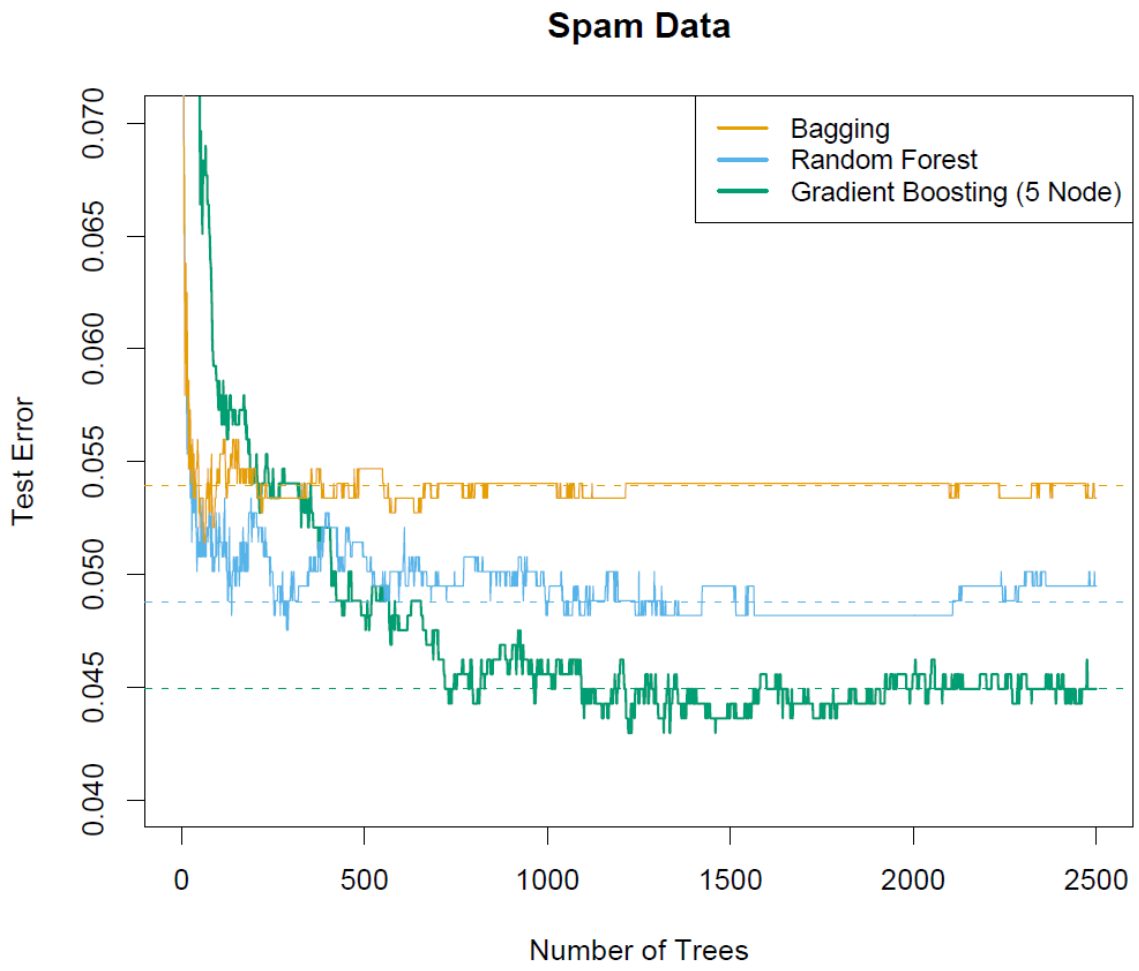
Tuning parameters for boosting

1. The *number of trees* B. Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select B.
2. The *shrinkage parameter* λ, a small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem. Very small λ can require using a very large value of B in order to achieve good performance.
3. The *number of splits* d in each tree, which controls the complexity of the boosted ensemble. Often $d = 1$ works well, in which case each tree is a *stump*, consisting of a single split and resulting in an additive model.

Another regression example



Another classification example



Bagging, boosting, interactions

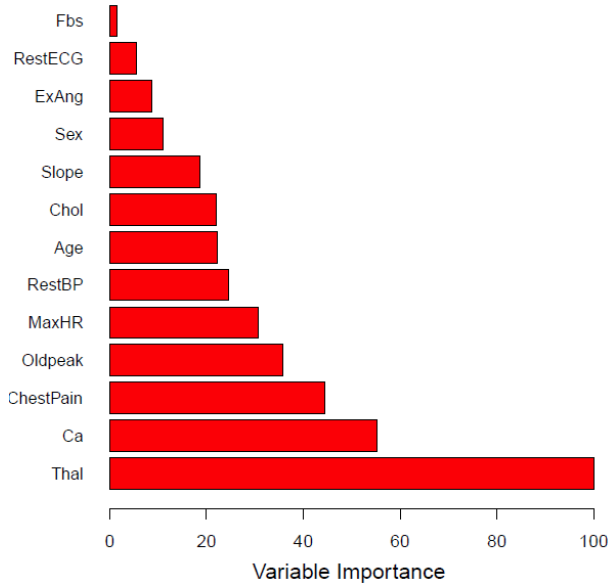
- Ensemble methods feel like *black boxes*: they make predictions by combining the results of hundreds of separate models
- Such models are able to capture complex interactions between predictors, which additive models are unable to do
- E.g., Suppose that your most profitable customers are young women and older men. A linear model would say:

$$\texttt{profit} \approx \beta_0 + \beta_1 I(\texttt{female}) + \beta_2 \texttt{age}$$

- This doesn't capture the interaction between `age` and `gender`
- Trees (and ensembles of trees) do a great job of capturing interactions
- Indeed, a tree with d splits can capture up to d-way interactions

Variable importance measure

- For bagged/RF regression trees, we record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees. A large value indicates an important predictor.
- Similarly, for bagged/RF classification trees, we add up the total amount that the Gini index is decreased by splits over a given predictor, averaged over all B trees.



Variable importance plot for the `Heart` data

Summary

- Decision trees are simple and interpretable models for regression and classification
- However they are often not competitive with other methods in terms of prediction accuracy
- Bagging, random forests and boosting are good methods for improving the prediction accuracy of trees. They work by growing many trees on the training data and then combining the predictions of the resulting ensemble of trees.
- The latter two methods— random forests and boosting— are among the state-of-the-art methods for supervised learning. However their results can be difficult to interpret.