Lecture 10: Decision Trees and Random Forests

STATS 202: Data Mining and Analysis

Linh Tran

tranlm@stanford.edu

Department of Statistics
Stanford University

July 28, 2021

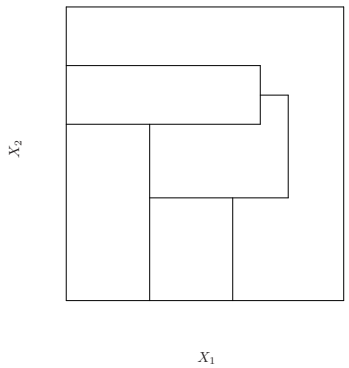
Announcements

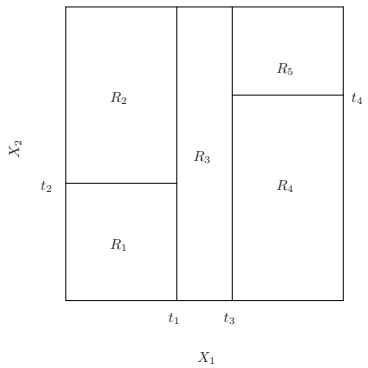
- Homework 3 due in 9 days.
- *Course survey* is (still) up
  - Up to 10 bonus points (as a percentage of class participation)
  - Closes on Friday.
- Should I move next Friday's section to this Friday?

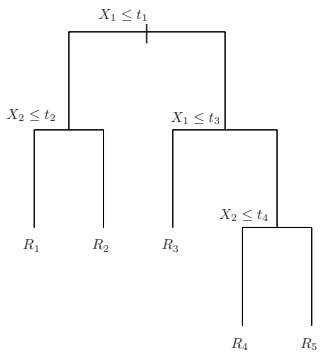
Outline

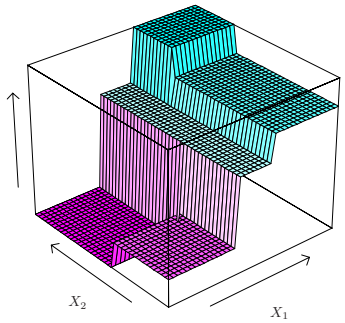
- Decision trees
  - Regression trees
  - Classification trees
  - Advantages / disadvantages
  - Misc details
- Bagging
- Random Forests

Decision trees, 10,000 foot view



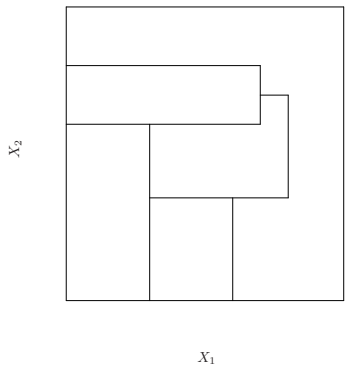


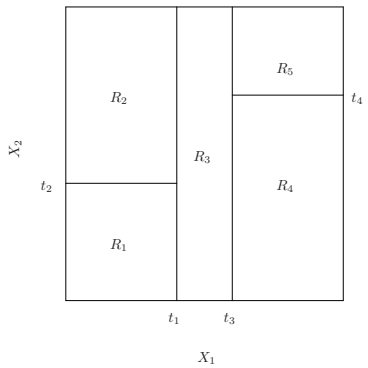


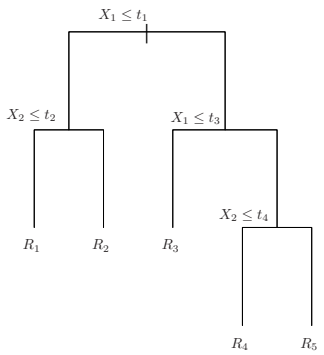


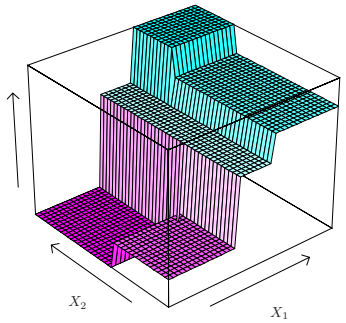
1. Find a partition of the space of predictors.
2. Predict a constant in each set of the partition.
3. The partition is defined by splitting the range of one predictor at a time.

Decision trees, 10,000 foot view





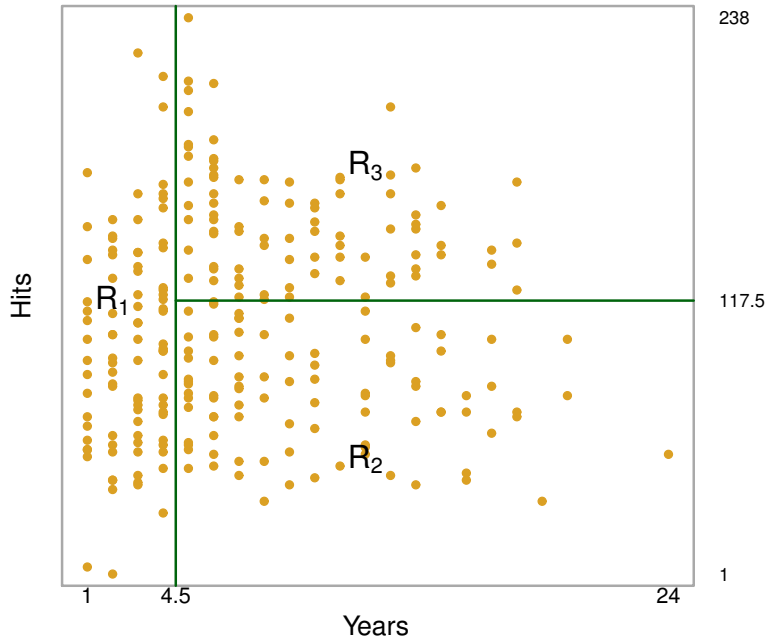




1. Find a partition of the space of predictors.
2. Predict a constant in each set of the partition.
3. The partition is defined by splitting the range of one predictor at a time.

   $\rightarrow$ Not all partitions are possible.

Example: Predicting a baseball player's salary



The prediction for a point in R_i is the average of the training points in R_i.

How is a decision tree built?

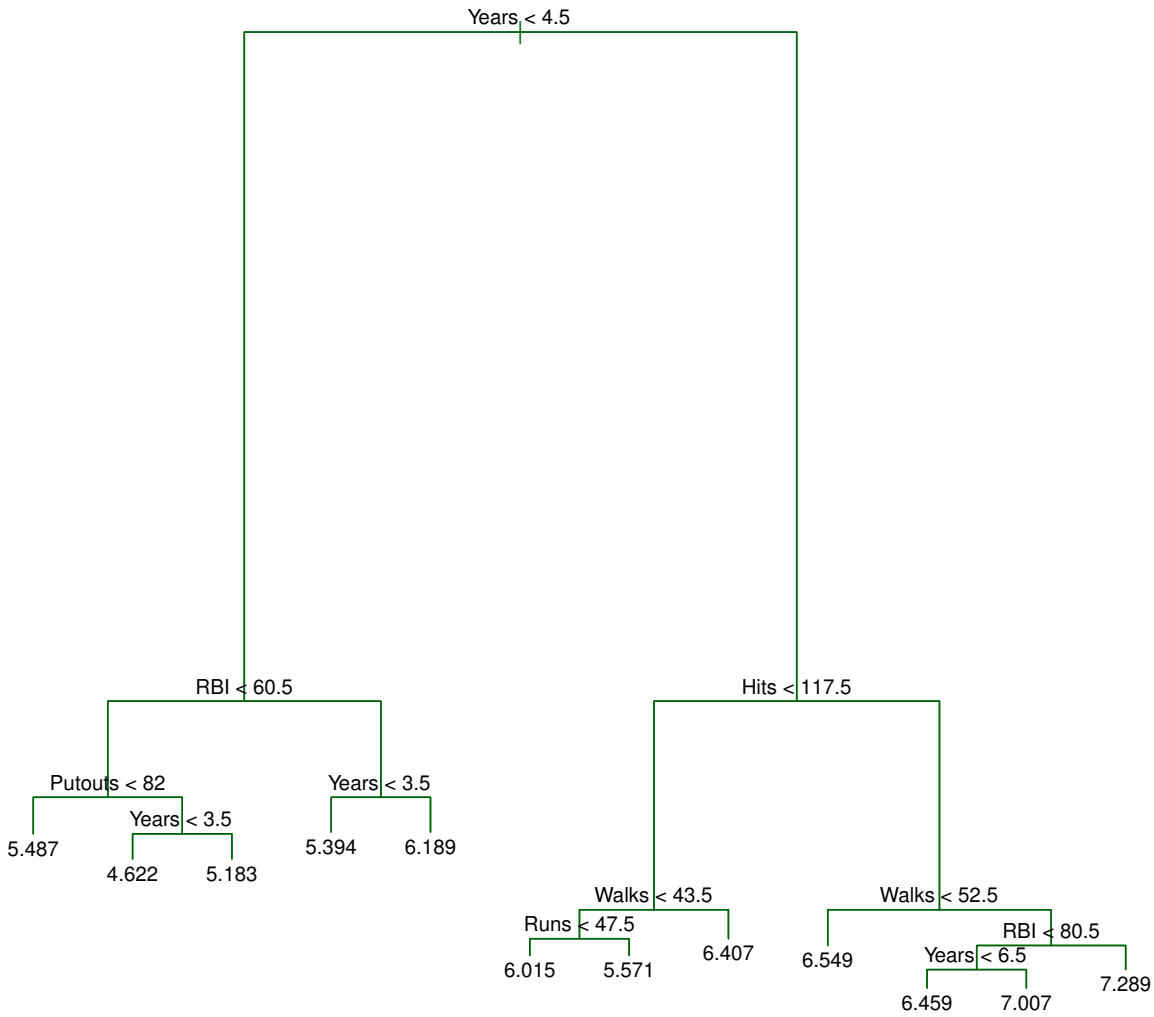
Using a *greedy* approach:

- Start with a single region R_1, and iterate:
  - Select a region R_k, a predictor X_j, and a splitting point s, such that splitting R_k with the criterion $X_j < s$ produces the largest decrease in RSS:

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \bar{y}_{R_m})^2$$

  - Redefine the regions with this additional split.
- Terminate when there are 5 observations or fewer in each region.
- This grows the tree from the root towards the leaves.

How is a decision tree built?



How do we control overfitting?

- **Idea 1:** Find the optimal subtree by cross validation.

How do we control overfitting?

- **Idea 1:** Find the optimal subtree by cross validation.

  $\rightarrow$ There are too many possibilities, so we would still over fit.

How do we control overfitting?

- **Idea 1:** Find the optimal subtree by cross validation.

  $\rightarrow$ There are too many possibilities, so we would still over fit.

- **Idea 2:** Stop growing the tree when the RSS doesn't drop by more than a threshold with any new cut.

How do we control overfitting?

- **Idea 1:** Find the optimal subtree by cross validation.

  $\rightarrow$ There are too many possibilities, so we would still over fit.

- **Idea 2:** Stop growing the tree when the RSS doesn't drop by more than a threshold with any new cut.

  $\rightarrow$ In our greedy algorithm, it is possible to find good cuts after bad ones.

How do we control overfitting?

Solution:

Prune a large tree from the leaves to the root.

- **Weakest link pruning:**
  - Starting with T_0, substitute a subtree with a leaf to obtain T_1, by minimizing:
$$\frac{RSS(T_1) - RSS(T_0)}{|T_0| - |T_1|}.$$
  - Iterate this pruning to obtain a sequence $T_0, T_1, T_2, \ldots, T_m$ where T_m is the null tree.
  - Select the optimal tree T_i by cross validation.

How do we control overfitting?

... or an equivalent procedure

- **Cost complexity pruning:**
  - Solve the problem:

$$\text{minimize} \sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \bar{y}_{R_m})^2 + \alpha |T|.$$

  - When $\alpha = \infty$, we select the null tree.
  - When $\alpha = 0$, we select the full tree.
  - The solution for each α is among $T_1, T_2, \ldots, T_m$ from weakest link pruning.
  - Choose the optimal α (the optimal T_i) by cross validation.

Cross validation

1. Construct a sequence of trees $T_0, \ldots, T_m$ for a range of values of α.
2. Split the training points into 10 folds.
3. For $k = 1, \ldots, 10$,
   - For each tree T_i, use every fold except the kth to estimate the averages in each region.
   - For each tree T_i, calculate the RSS in the test fold.
4. For each tree T_i, average the 10 test errors, and select the value of α that minimizes the error.

Cross validation

1. Construct a sequence of trees $T_0, \ldots, T_m$ for a range of values of α.
2. Split the training points into 10 folds.
3. For $k = 1, \ldots, 10$,
   - For each tree T_i, use every fold except the kth to estimate the averages in each region.
   - For each tree T_i, calculate the RSS in the test fold.
4. For each tree T_i, average the 10 test errors, and select the value of α that minimizes the error.

THIS IS THE WRONG WAY TO DO CROSS VALIDATION!

Cross validation, the right way

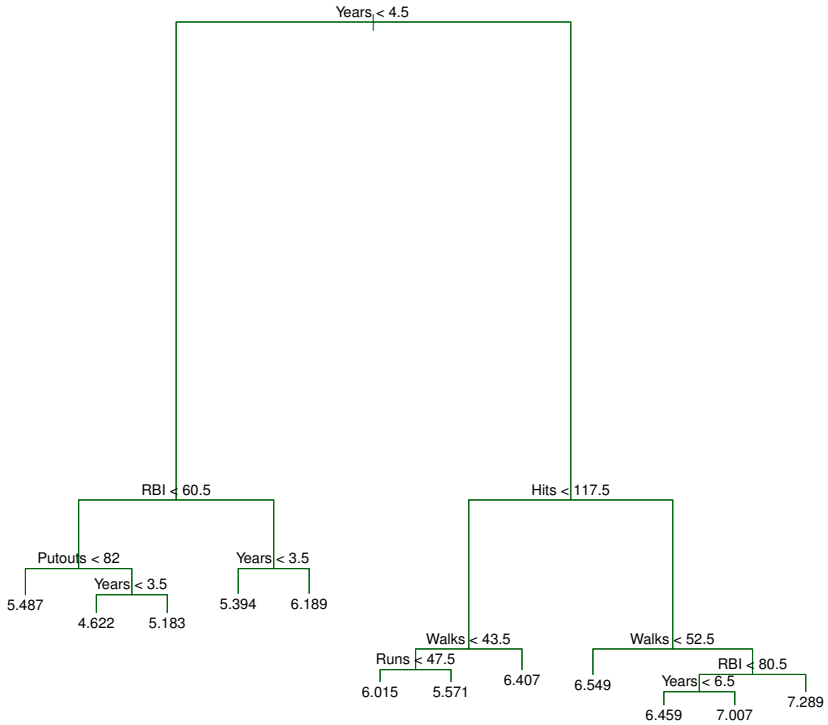
1. Split the training points into 10 folds.
2. For $k = 1, \ldots, 10$, using every fold except the kth:
   - Construct a sequence of trees $T_1, \ldots, T_m$ for a range of values of α, and find the prediction for each region in each one.
   - For each tree T_i, calculate the RSS on the test set.
3. For each tree T_i, average the 10 test errors, and select the value of α that minimizes the error.

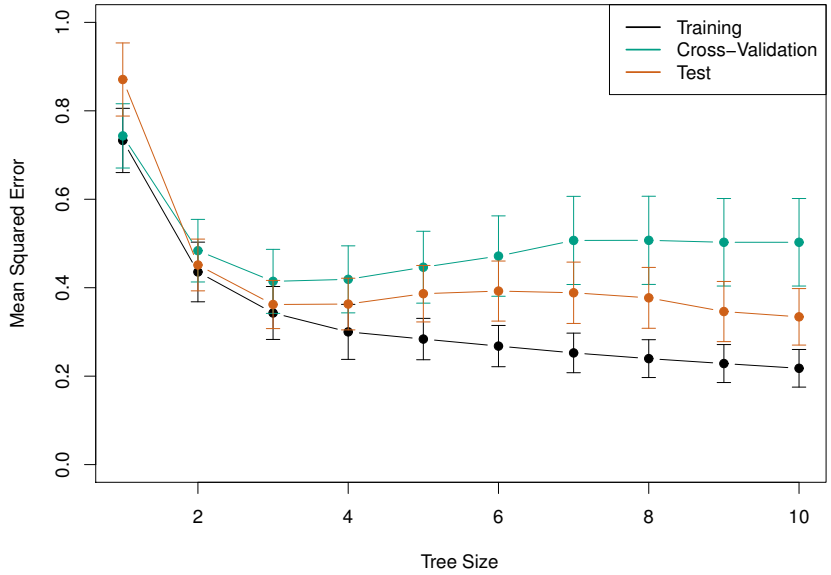
Cross validation, the right way

1. Split the training points into 10 folds.
2. For $k = 1, \ldots, 10$, using every fold except the kth:
   - Construct a sequence of trees $T_1, \ldots, T_m$ for a range of values of α, and find the prediction for each region in each one.
   - For each tree T_i, calculate the RSS on the test set.
3. For each tree T_i, average the 10 test errors, and select the value of α that minimizes the error.

Note: We are doing all fitting, including the construction of the trees, using only the training data.

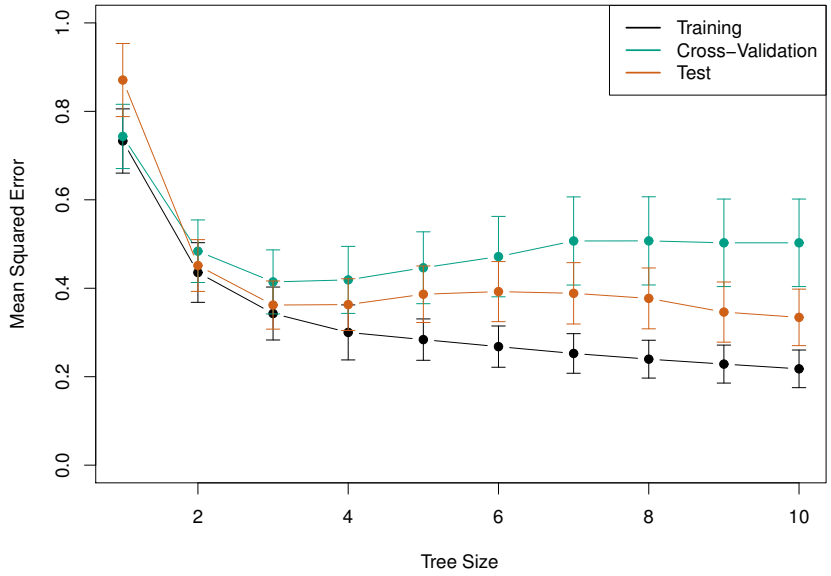
Example. Predicting baseball salaries





Unpruned tree (size=12)

Example. Predicting baseball salaries



Short tree (size=3)

Classification trees

Work much like regression trees.

- We predict the response by **majority vote**, i.e. pick the most common class in every region.
- Instead of trying to minimize the RSS:

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \bar{y}_{R_m})^2$$

we minimize a classification loss function.

  - Multiple losses to choose from

Classification losses

- The 0-1 loss or misclassification rate:

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} \mathbf{1}(y_i \neq \hat{y}_{R_m})$$

Classification losses

- The 0-1 loss or misclassification rate:

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} \mathbf{1}(y_i \neq \hat{y}_{R_m})$$

- The Gini index:

$$\sum_{m=1}^{|T|} q_m \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

where $\hat{p}_{m,k}$ is the proportion of class k within R_m, and q_m is the proportion of samples in R_m.

Classification losses

- The 0-1 loss or misclassification rate:

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} \mathbf{1}(y_i \neq \hat{y}_{R_m})$$

- The Gini index:

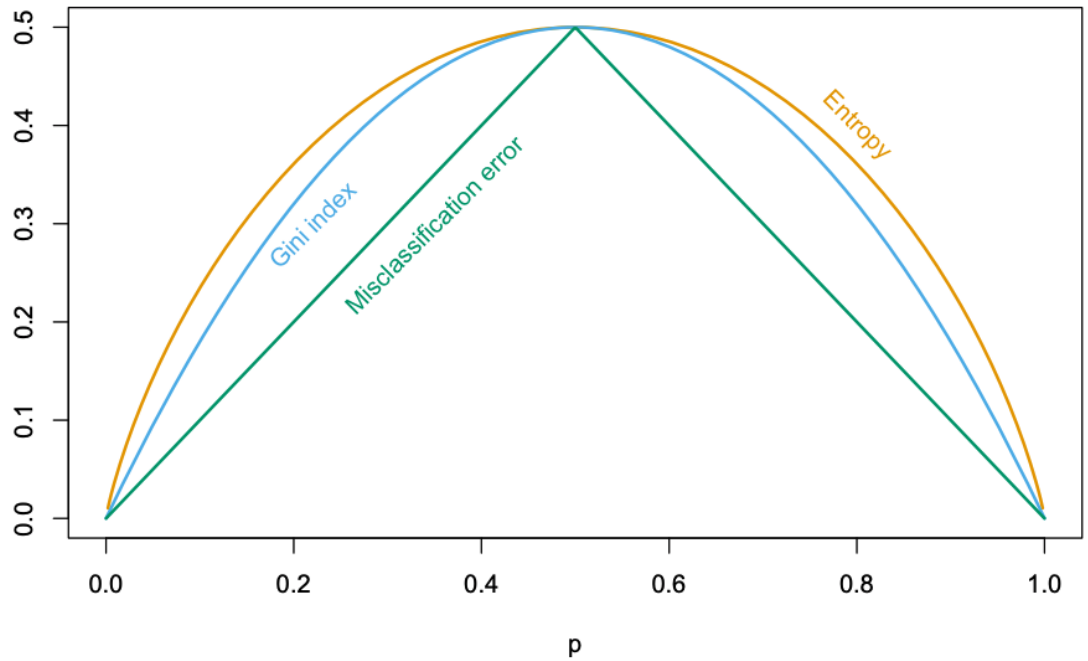
$$\sum_{m=1}^{|T|} q_m \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

  where $\hat{p}_{m,k}$ is the proportion of class k within R_m, and q_m is the proportion of samples in R_m.

- The entropy:

$$-\sum_{m=1}^{|T|} q_m \sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk}).$$

Classification losses



Losses for 2-class classication, as a function of the proportion p. Entropy has been scaled to pass through (0.5, 0.5).

Classification losses

Remarks:

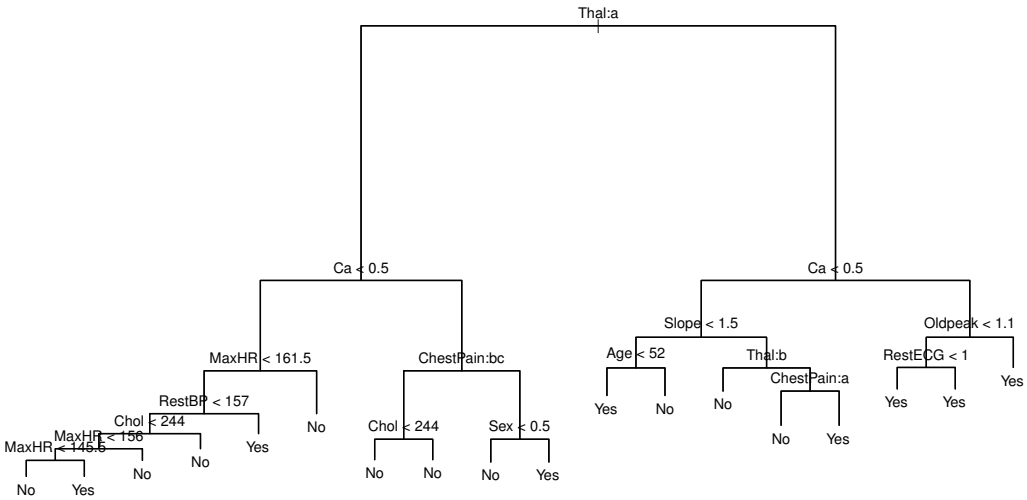
- **Motivation for the Gini index:**

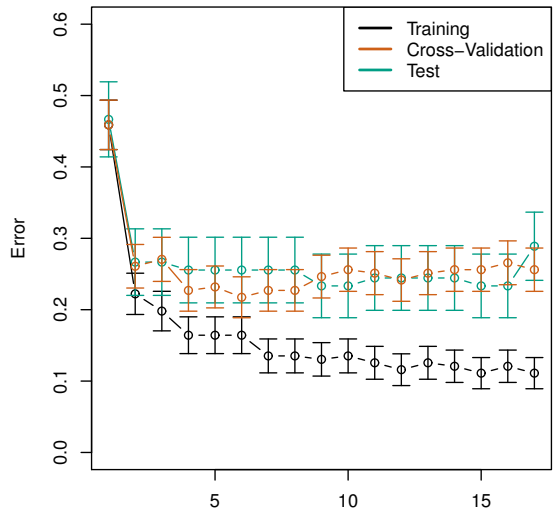
  If instead of predicting the most likely class, we predict a random sample from the distribution $(\hat{p}_{1,m}, \hat{p}_{2,m}, \ldots, \hat{p}_{K,m})$, the Gini index is the expected misclassification rate.

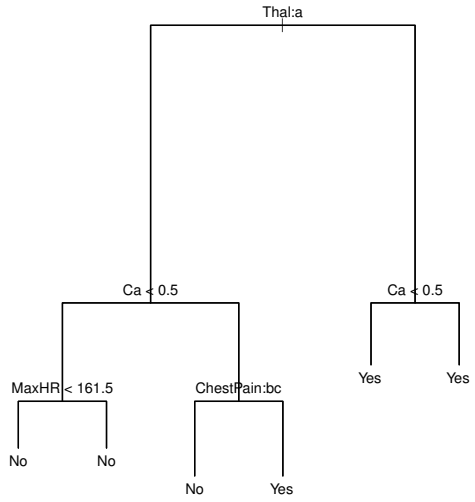
- The Gini index and entropy are better measures of the purity of a region, i.e. they are low when the region is mostly one category.

- It is typical to use the Gini index or entropy for growing the tree, while using the misclassification rate when pruning the tree.

Example. Heart dataset.





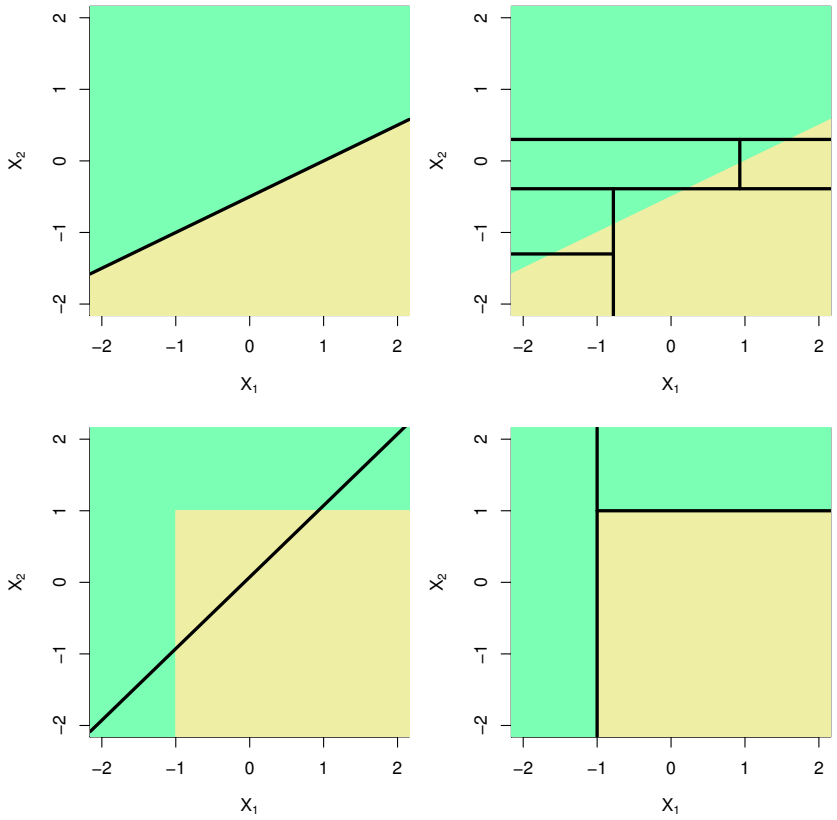


Decision tree advantages

- Very easy to interpret!
- Closer to human decision-making.
- Can capture complex interactions between variables.
- Easy to visualize graphically.
- Easily handle qualitative predictors and missing data.

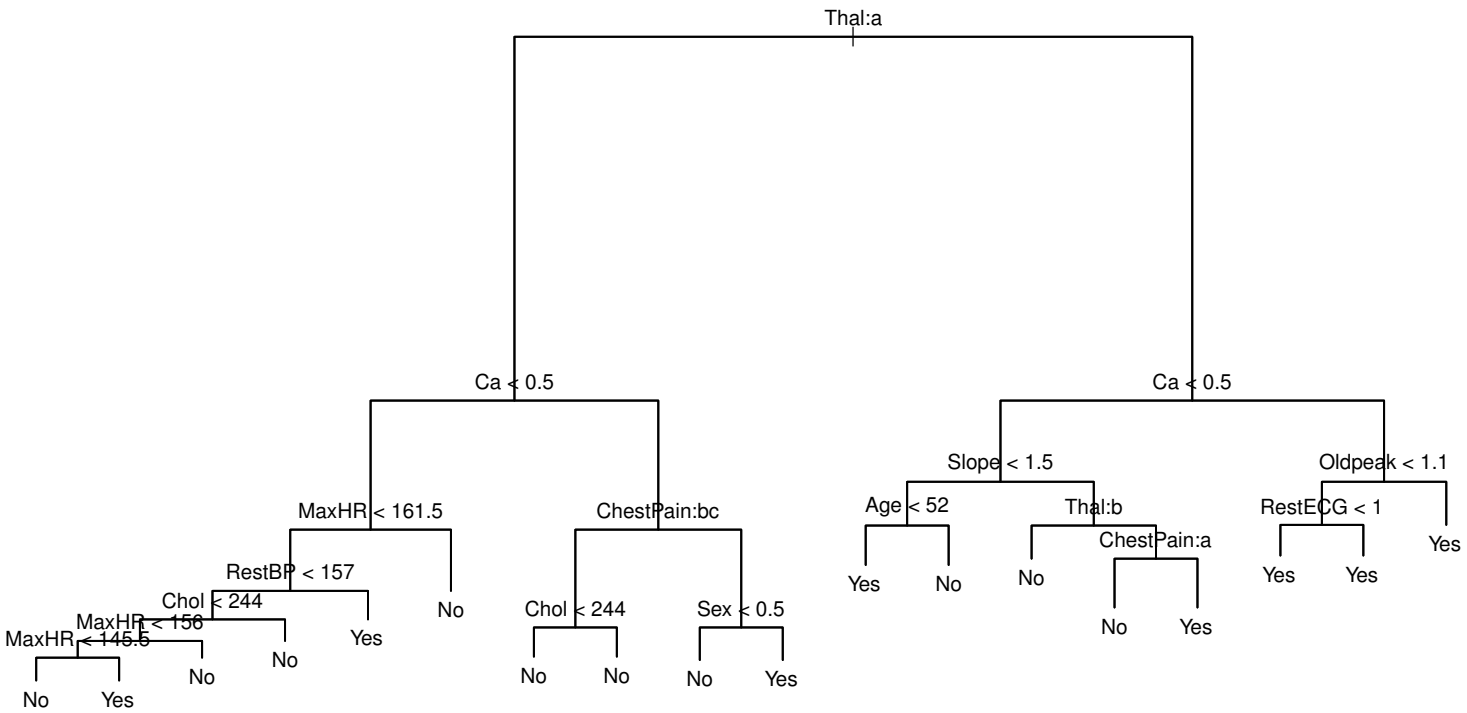
Decision tree disadvantages

- Doesn't capture simple (e.g. linear) relationships well.
- Less accurate than other ML methods.
- Can have high variance.



Example. Heart dataset.

Question: How do we deal with categorical predictors?



Categorical predictors

- If there are only 2 categories, then the split is obvious. We don't have to choose the splitting point s, as for a numerical variable.
- If there are more than 2 categories:
  - Order the categories according to the average of the response:

$$\texttt{ChestPain} : \texttt{a} > \texttt{ChestPain} : \texttt{c} > \texttt{ChestPain} : \texttt{b}$$

  - Treat as a numerical variable with this ordering, and choose a splitting point s.
- This is the optimal way of partitioning.

Missing data

Problem: If a sample is missing variable X_j, and a tree contains a split according to $X_j > s$, then we may not be able to assign the sample to a region.

Missing data

Problem: If a sample is missing variable X_j, and a tree contains a split according to $X_j > s$, then we may not be able to assign the sample to a region.

Solution:

- When choosing a new split with variable X_j (growing the tree):
  - Only consider the samples which have the variable X_j.
  - In addition to choosing the best split, choose a second best split using a different variable, and a third best, ...
- To propagate a sample down the tree, if it is missing a variable to make a decision, try the second best decision, or the third best, etc...

Bagging

Recall:

- Bagging = Bootstrap Aggregating
- We replicate our dataset by sampling with replacement:
  - Original dataset: $\texttt{x} = \texttt{c}(\texttt{x1}, \texttt{x2}, \ldots, \texttt{x100})$
  - Bootstrap samples:
    $\texttt{boot1} = \texttt{sample}(\texttt{x}, 100, \texttt{replace} = \texttt{True})$, ...,
    $\texttt{bootB} = \texttt{sample}(\texttt{x}, 100, \texttt{replace} = \texttt{True})$.
- We average the predictions of a model fit to many Bootstrap samples:

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

When does Bagging make sense?

When a regression method or a classifier has a tendency to overfit, Bagging reduces the variance of the prediction.

- When n is large, the empirical distribution is similar to the true distribution of the samples.
- Bootstrap samples are like independent realizations of the data.
- Bagging amounts to averaging the fits from many independent datasets, which would reduce the variance by a factor $1/B$, i.e. $\frac{1}{B}\sigma^2$.

Bagging decision trees

- **Disadvantage:** Every time we fit a decision tree to a bootstrap sample, we get a different tree T^b.

  $\rightarrow$ Loss of interpretability

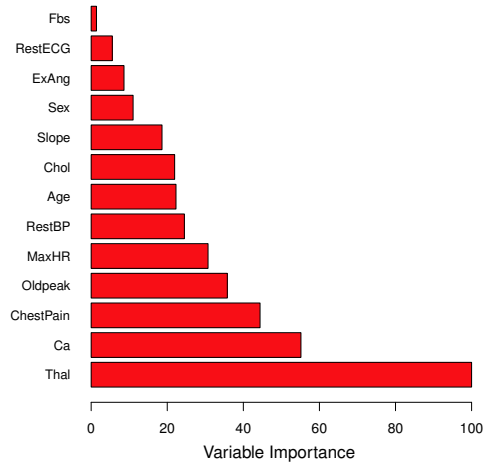
Bagging decision trees

- **Disadvantage:** Every time we fit a decision tree to a bootstrap sample, we get a different tree T^b.

  $\rightarrow$ Loss of interpretability

Variable importance:

- For each predictor, add up the total by which the RSS (or Gini index) decreases every time we use the predictor in T^b.
- Average this total over each Boostrap estimate $T^1, \ldots, T^B$.



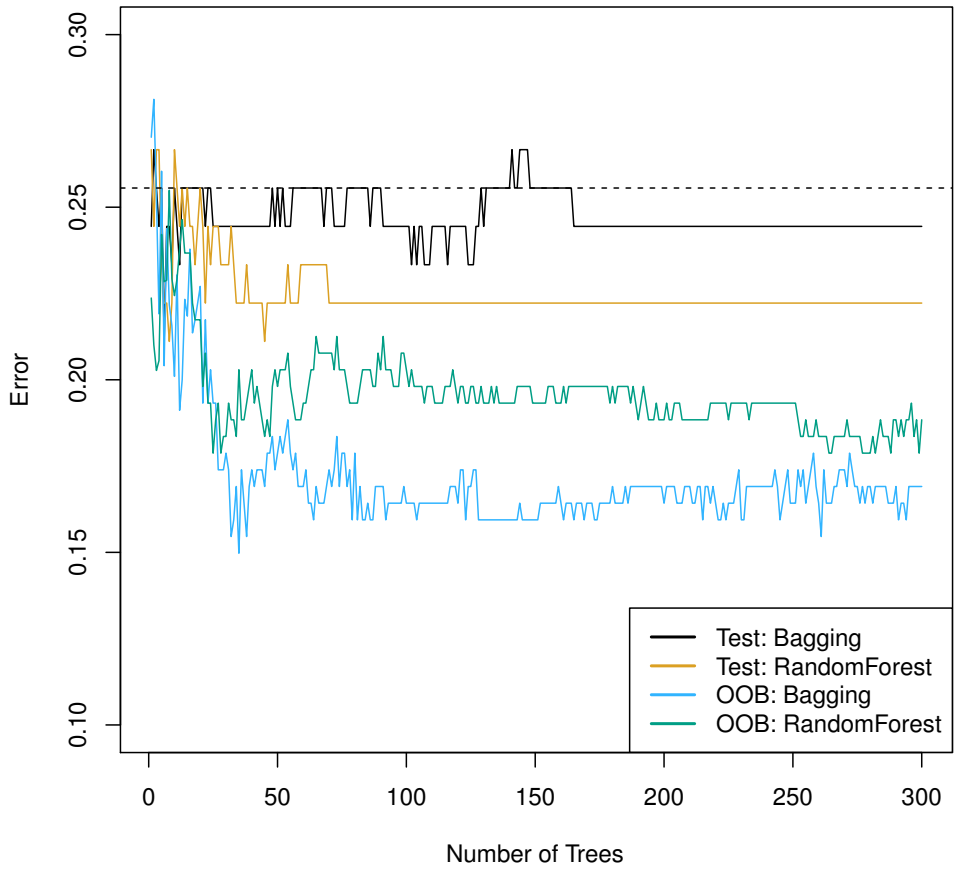
Out-of-bag (OOB) error

To estimate the test error of a bagging estimate, we *could* use cross-validation.

Or instead, could just use observations that weren't sampled

- Each time we draw a Bootstrap sample, we only use 63% of the observations.
- **Idea:** use the rest of the observations as a test set.
- **OOB error:**
  - For each sample x_i, find the prediction $\hat{y}_i^b$ for all bootstrap samples b which do not contain x_i. There should be around $0.37B$ of them. Average these predictions to obtain $\hat{y}_i^{\text{oob}}$.
  - Compute the error $(y_i - \hat{y}_i^{\text{oob}})^2$.
  - Average the errors over all observations $i = 1, \ldots, n$.

Out-of-bag (OOB) error



The test error decreases as we increase B
(dashed line is the error for a plain decision tree).

Random Forests

In general, bagging has a problem:

$\rightarrow$ The trees produced by different Bootstrap samples can be very similar.

Specifically: The variance from bagging is

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

Lowering ρ can lower our variance

Random Forests:

- We fit a decision tree to different Bootstrap samples.
- When growing the tree, we select a random sample of $m < p$ predictors to consider in each step.
- This will lead to very different (or "uncorrelated") trees from each sample.
- Finally, average the prediction of each tree.

Random Forests algorithm

Algorithm 15.1 *Random Forest for Regression or Classification.*

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.

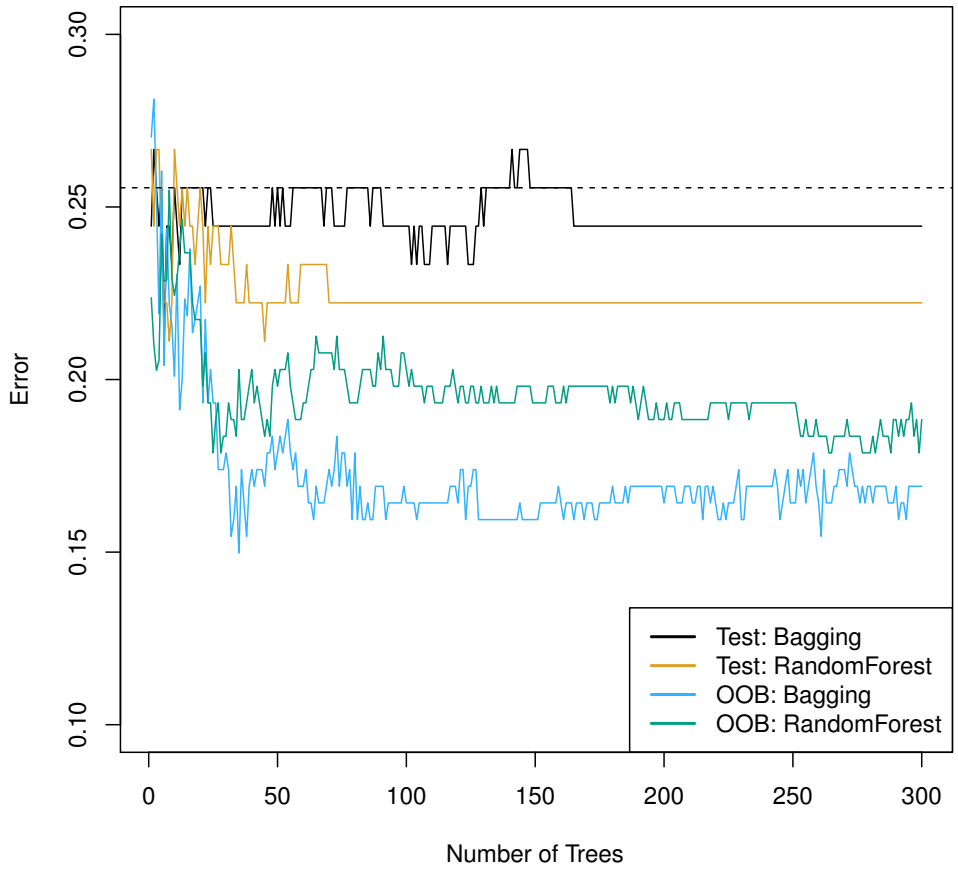
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

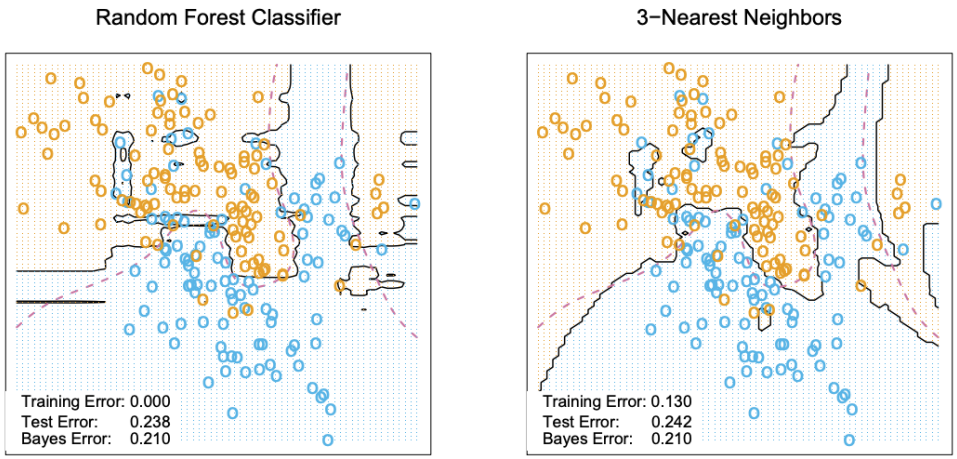
Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

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

Random Forests vs. Bagging

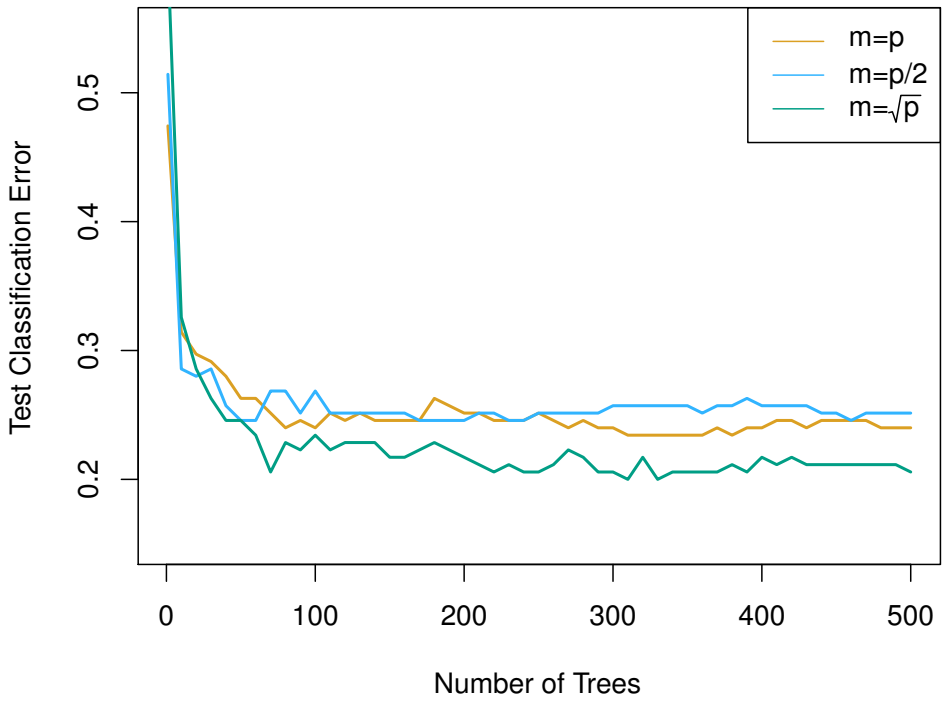


Random Forests vs. KNN



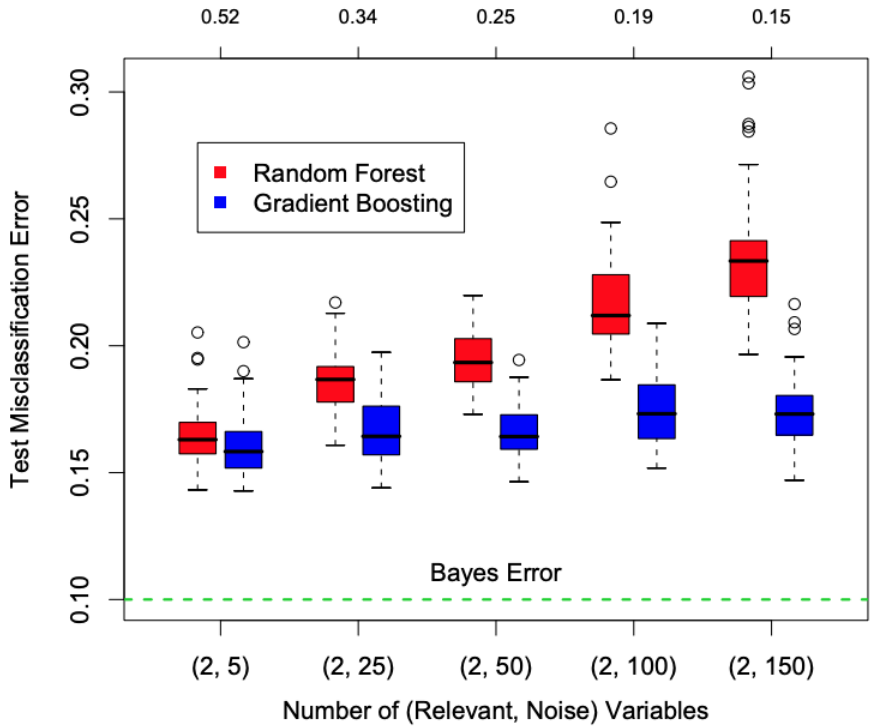
Random forest can be thought of as weighted voting of the closest points.

Random Forests, choosing m



The optimal m is usually around $\sqrt{p}$,
but this can be used as a tuning parameter.

Overfitting with Random Forests



Yes, we can overfit using Random Forests!

References

[1] ISL. Chapter 8

[2] ESL. Chapter 9.2,15