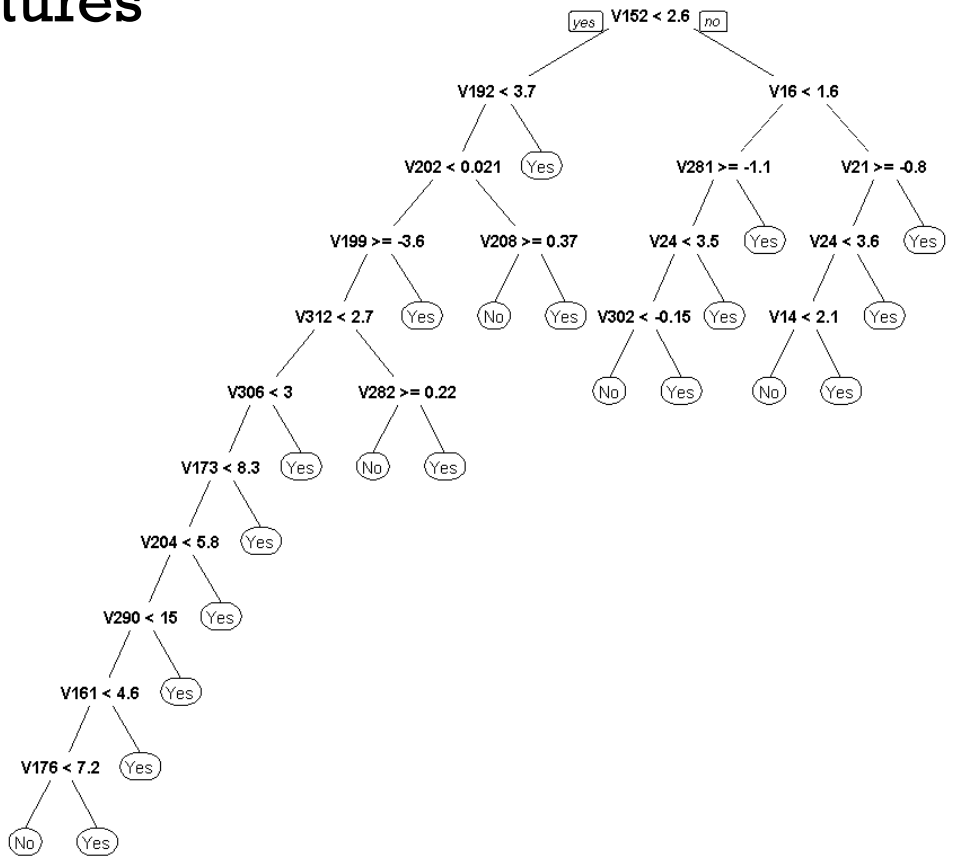
DECISION TREES

OUTLINE

- The Basics of Decision Trees
  - Regression Trees
  - Pruning Trees
  - Classification Trees
- Advanced Prediction Models (Ensemble Learning)
  - Bagging
  - Random Forests
  - Boosting

OUTLINE

- An example of making decisions based on tree representation of incident conditions (yes)
- Analyzing many features



INTRODUCTION

- Tree-based methods for regression and classification
- The idea is to stratify or segment the predictor space into a number of simple regions
- In order to make a prediction for a given observation, we typically use the mean or the mode of the training observations in the region to which it belongs
- Since the set of splitting rules used to segment the predictor space can be summarized in a tree, these types of approaches are known as decision-tree methods

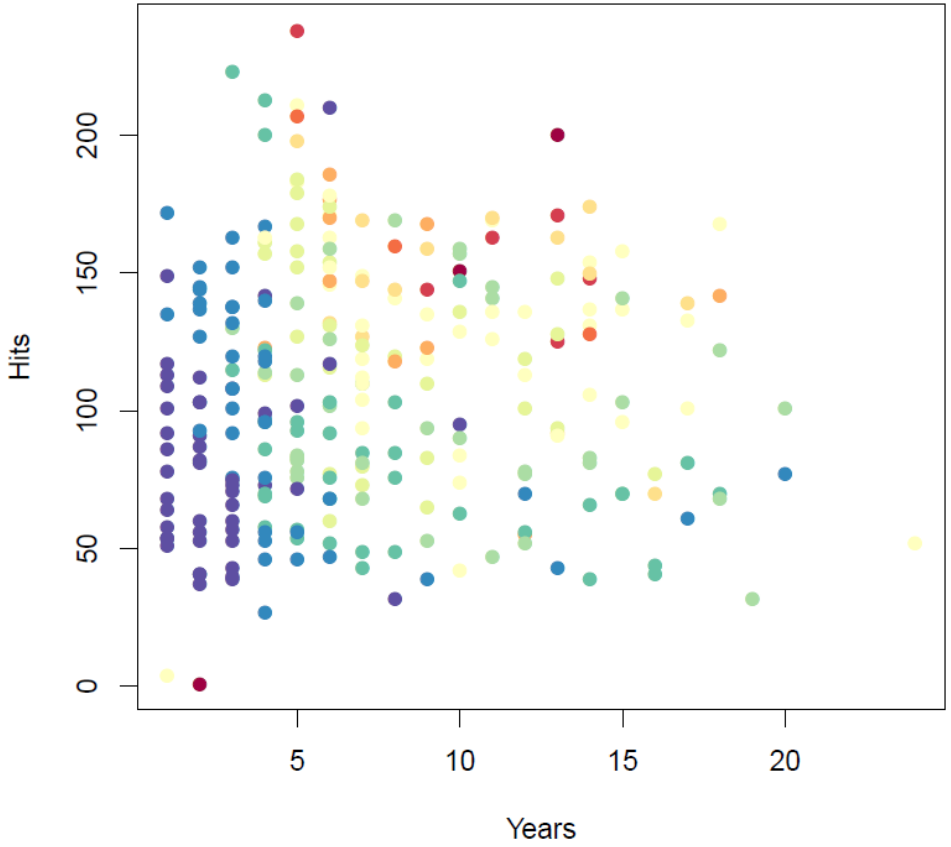
PROS AND CONS

- Tree-based methods are simple and useful for interpretation
- Typically, they are not competitive with the best supervised learning approaches in terms of prediction accuracy
- Hence we also discuss bagging, random forests, and boosting
- These methods grow multiple trees which are then combined to yield a single consensus prediction (ensemble learning)
- Combining a large number of trees can often result in dramatic improvements in prediction accuracy (transforming weak learner into a stronger one), at the expense of some loss in interpretation

REGRESSION TREES

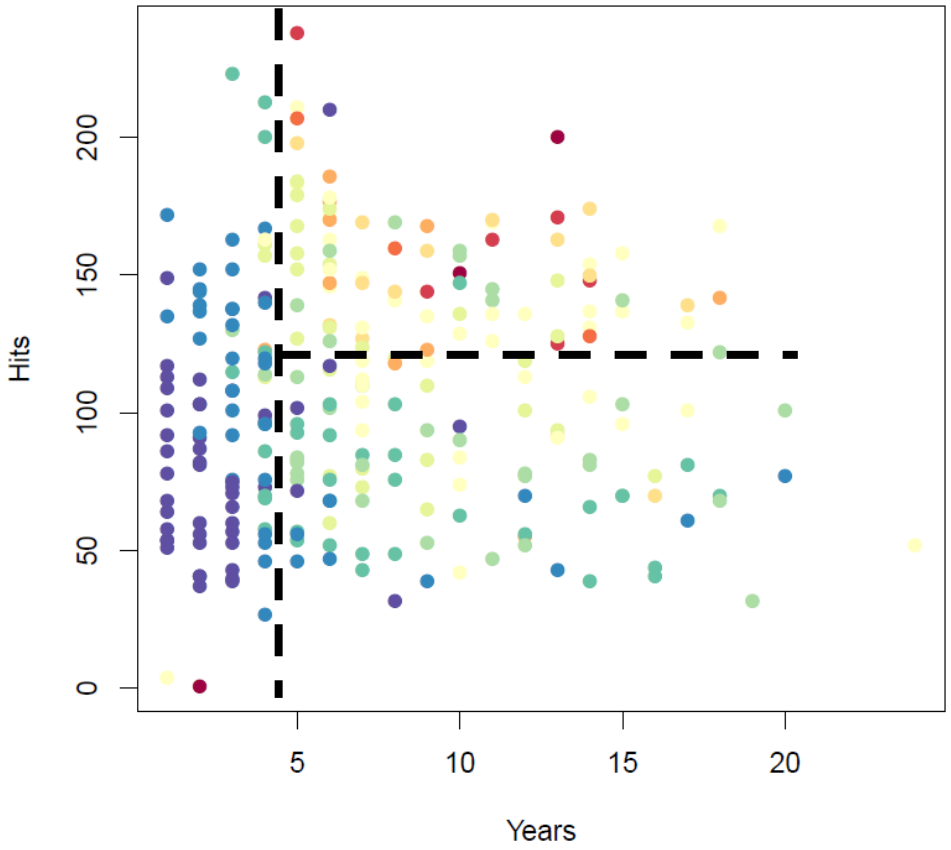
HITTERS DATA: VISUALIZATION

- Salary ~ Years + Hits
- Salary is color-coded from low (blue, green) to high (yellow, red)

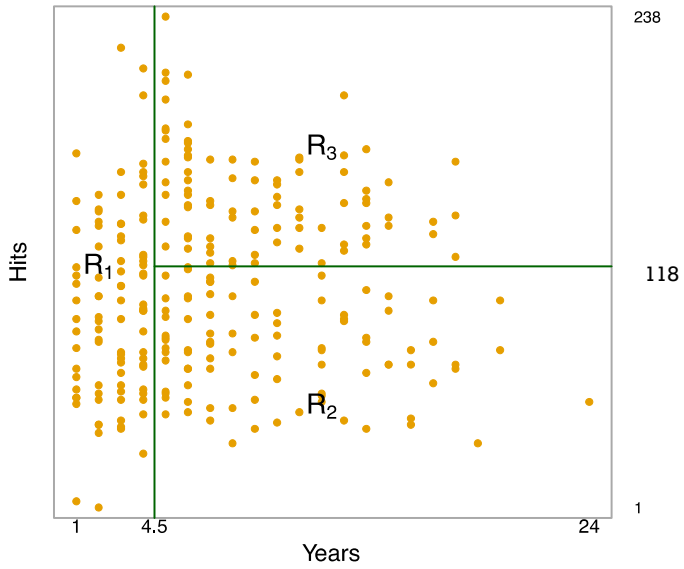


HITTERS DATA: VISUALIZATION

- Salary ~ Years + Hits
- Salary is color-coded from low (blue, green) to high (yellow, red)

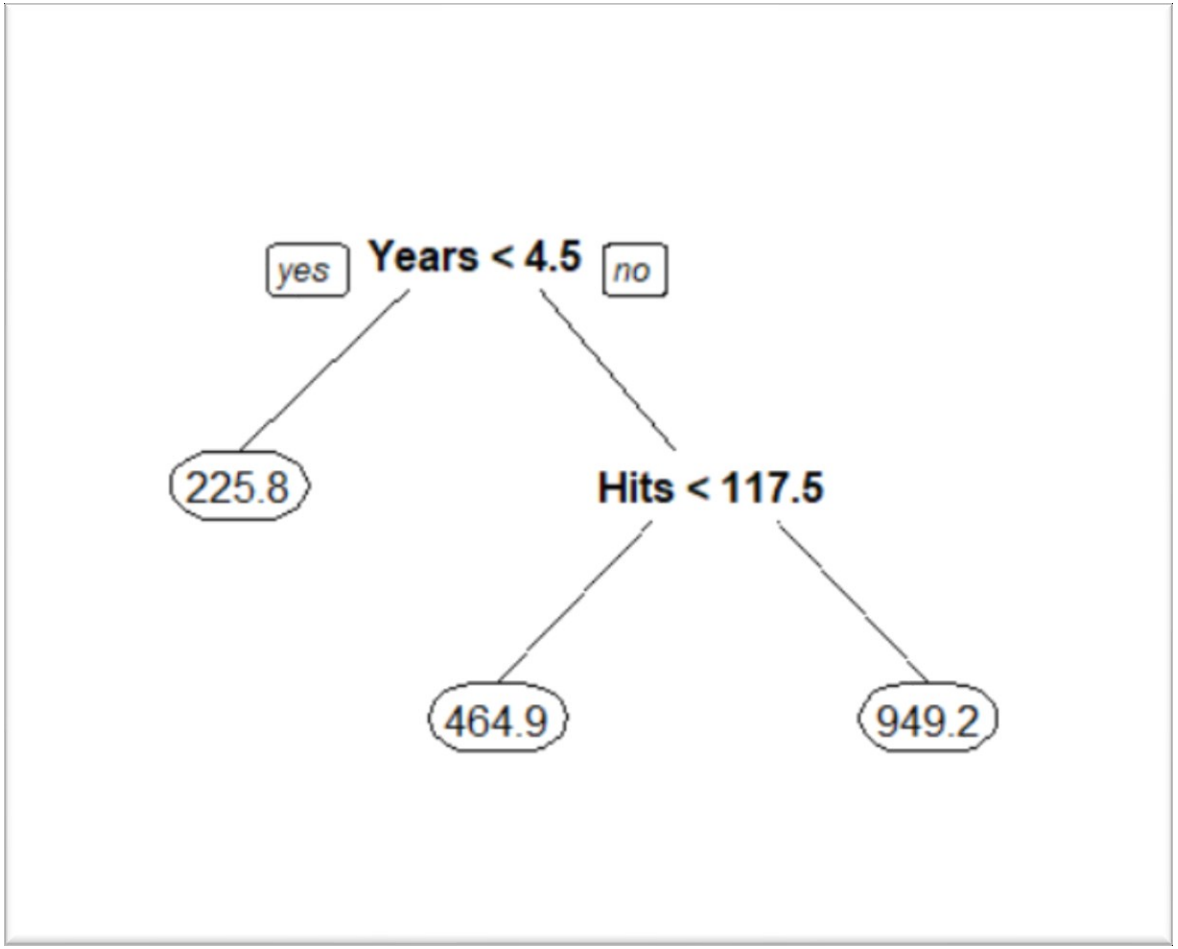


RESULTS



- Overall, the tree segments the players into three regions of predictor space:
- $R_1 = \{X \mid Years < 4.5\}$
- $R_2 = \{X \mid Years \geq 4.5, Hits < 118\}$
- $R_3 = \{X \mid Years \geq 4.5, Hits \geq 118\}$

HITTERS DATA: REGRESSION TREE



TERMINOLOGY

- The regions R_1, R_2, and R_3 are known as terminal nodes or leaves of the tree
- Decision trees are typically drawn upside down, in the sense that the leaves are at the bottom of the tree
- The points along the tree where the predictor space is split are referred to as internal nodes
- In the hitters tree, the two internal nodes are indicated by the text $Years < 4.5$ and $Hits < 118$
- We refer to the segments of the trees that connect the nodes as branches

HITTERS DATA: INTERPRETATION

- Years is the most important factor in determining Salary, and players with less experience earn lower salaries than more experienced players
- Given that a player is less experienced, the number of Hits that he made in the previous year seems to play little role in his Salary
- Among players who have been in the major leagues for five or more years, the number of Hits made in the previous year does affect Salary, and players who made more Hits last year tend to have higher salaries
- The predicted Salary for those players is given by the mean response value for the players in the data set belonging to the segments R_1, R_2, and R_3
- Surely an over-simplification, but compared to a regression model, it is easy to display, interpret and explain

TREE-BUILDING PROCESS

1. 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, $R_1, R_2, \ldots, R_J$

2. For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j

EXAMPLE

- Suppose that in Step 1, we obtain two regions, R_1 and R_2
- Suppose, the response mean of the training observations in the first region is 10
- Suppose, the response mean of the training observations in the second region is 20
- If for a given observation $X = x, x \in R_1$, we will predict a value of 10
- If for a given observation $X = x, x \in R_2$, we will predict a value of 20

TREE-BUILDING PROCESS – STEP 1

- In theory, the regions could have any shape
- However, we choose to divide the predictor space into high-dimensional rectangles, or boxes
- This is for simplicity and for ease of interpretation of the resulting predictive model

TREE-BUILDING PROCESS – STEP 1

- The goal is to find boxes $R_1, R_2, \ldots, R_J$ that minimize the RSS

$$RSS = \sum_{j=1}^{J} \sum_{i:x_i \in R_j} \left(y_i - \hat{y}_{R_j}\right)^2$$

where $\hat{y}_{R_j}$ is the mean response for the training observations within the j^{th} box

TREE-BUILDING PROCESS – STEP 1

- Unfortunately, it is computationally infeasible to consider every possible partition of the feature space into J boxes
- For this reason, we take a top-down, greedy approach that is known as recursive binary splitting
- The approach is top-down because it begins at the top of the tree and then successively splits the predictor space; each split is indicated via two new branches further down on the tree
- It is greedy because at each step of the tree-building process, the best split is made at particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step

TREE-BUILDING PROCESS – STEP 1

- We first select the predictor X_j and the cut-point s such that splitting the predictor space into the regions

$$R_1(j, s) = \{X | X_j < s\}$$

and

$$R_2(j, s) = \{X | X_j \geq s\}$$

leads to the greatest possible reduction in RSS

TREE-BUILDING PROCESS – STEP 1

- Before splitting

$$err_0 = \sum_{i:x_i \in R} (y_i - \hat{y}_R)^2$$

- After splitting

$$err_1 = \sum_{i:x_i \in R_1(j,s)} \left(y_i - \hat{y}_{R_1}\right)^2 + \sum_{i:x_i \in R_2(j,s)} \left(y_i - \hat{y}_{R_2}\right)^2$$

TREE-BUILDING PROCESS – STEP 1

- Actually, we seek the values of j and s that minimize err_1 and include X_j in a tree only if the decrease of the error is significant
- Finding the values of j and s that minimize err_1 can be done quite quickly, especially when the number of features p is not too large

TREE-BUILDING PROCESS

- Next, we repeat the process, looking for the best predictor and best cut-point in order to split the data further so as to minimize the RSS within each of the resulting regions
- However, this time, instead of splitting the entire predictor space, we split one of the two previously identified regions. We now have three regions
- Again, we look to split one of these three regions further, so as to minimize the RSS
- The process continues until a stopping criterion is reached; for instance, we may continue until no region contains more than five observations

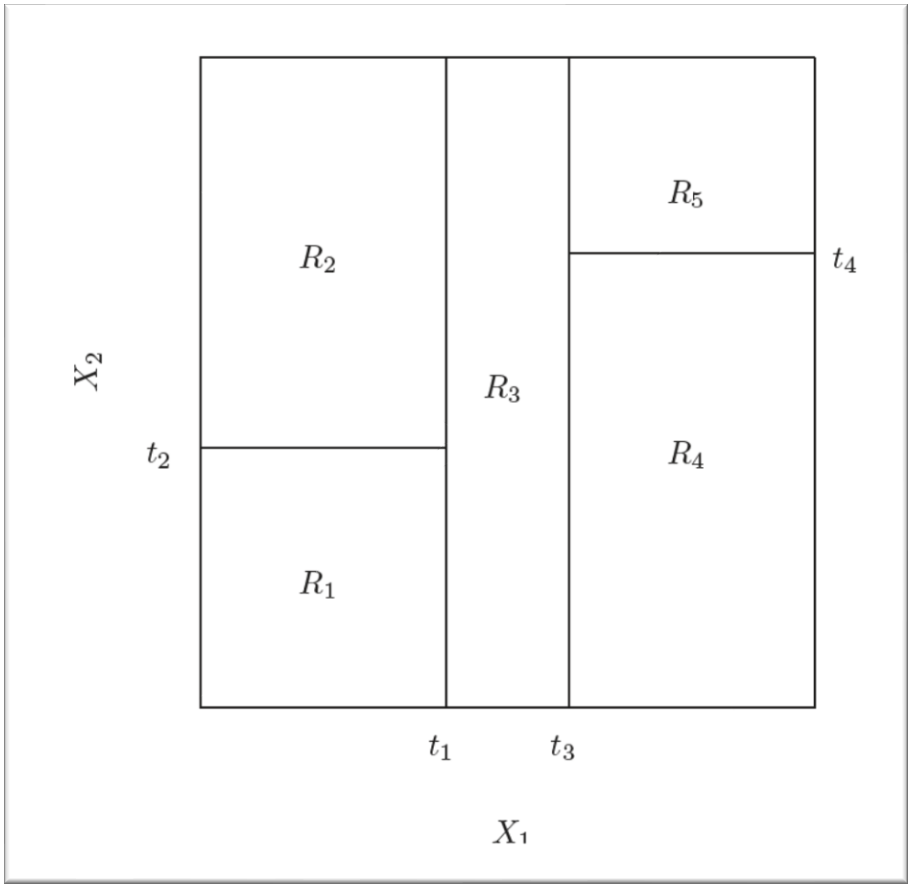
TREE-BUILDING PROCESS – STEP 2

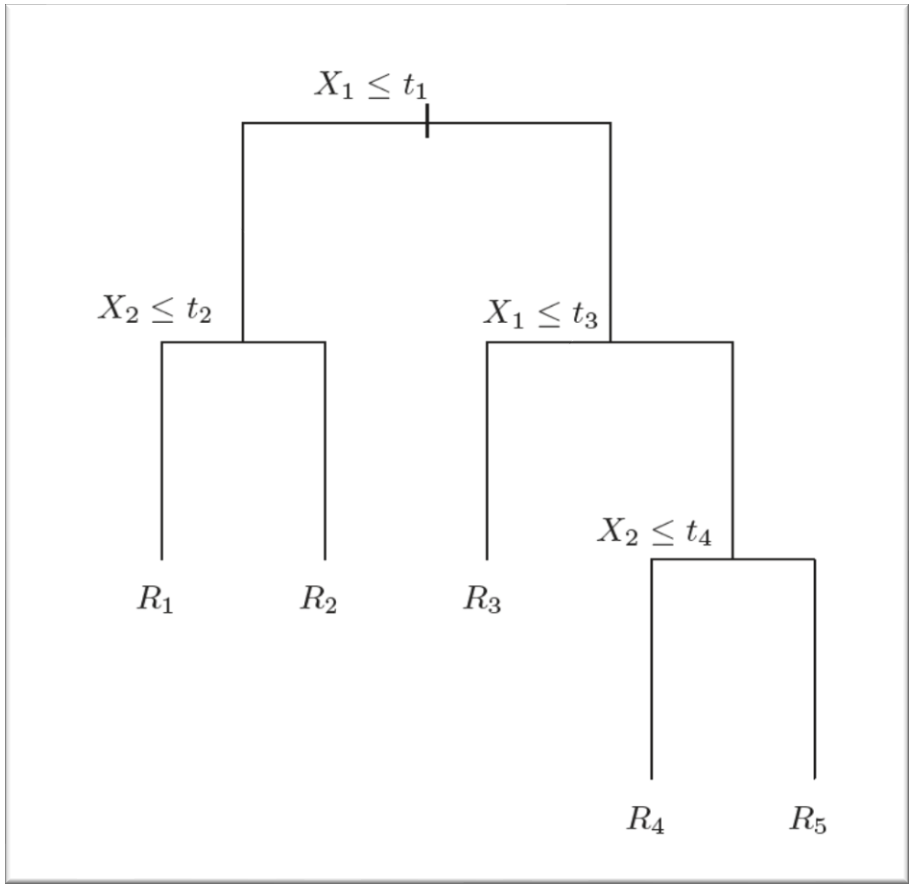
- Once the regions

$$R_1, \ldots, R_J$$

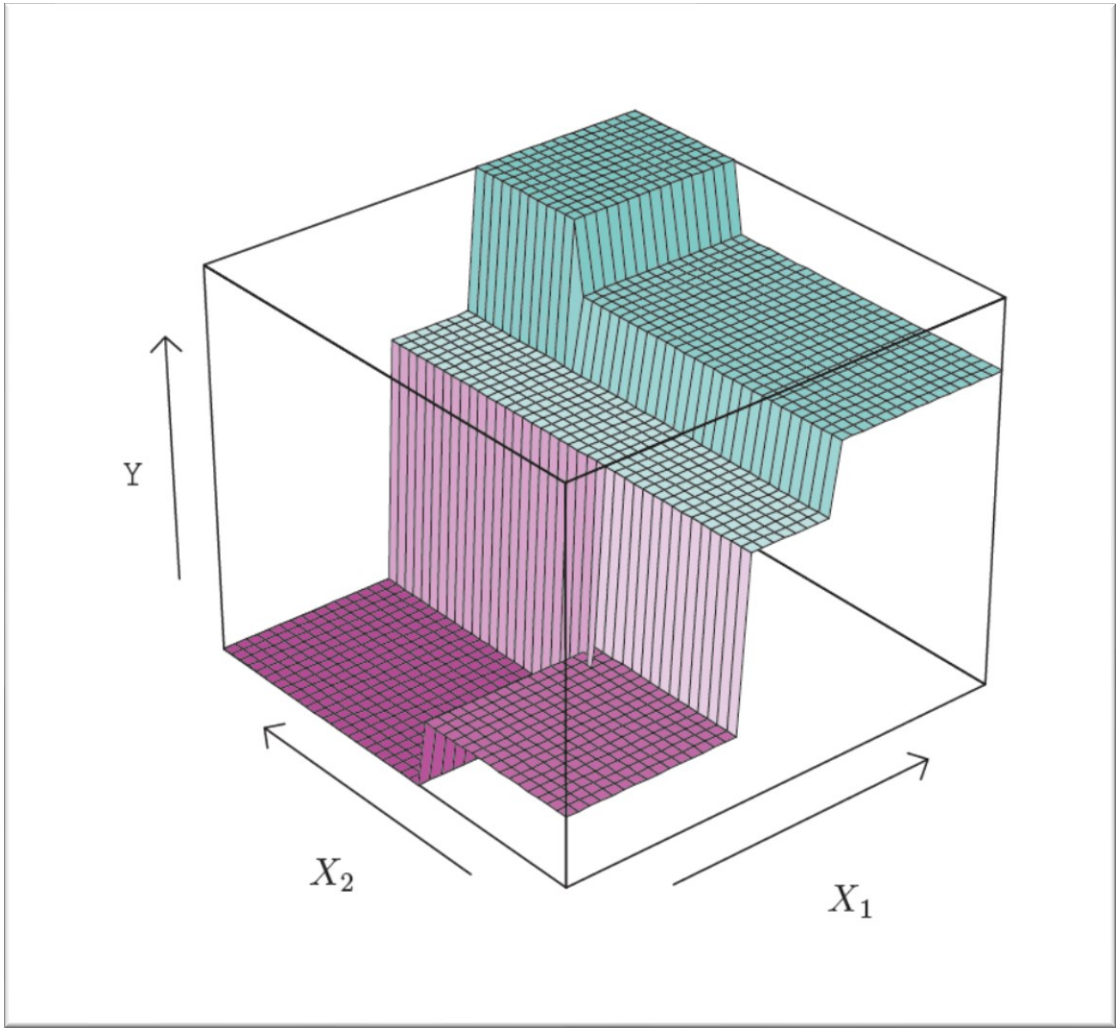
have been created, we predict the response for a given test observation using the mean of the training observations in the region to which that test observation belongs

EXAMPLE OF A RECURSIVE BINARY SPLITTING

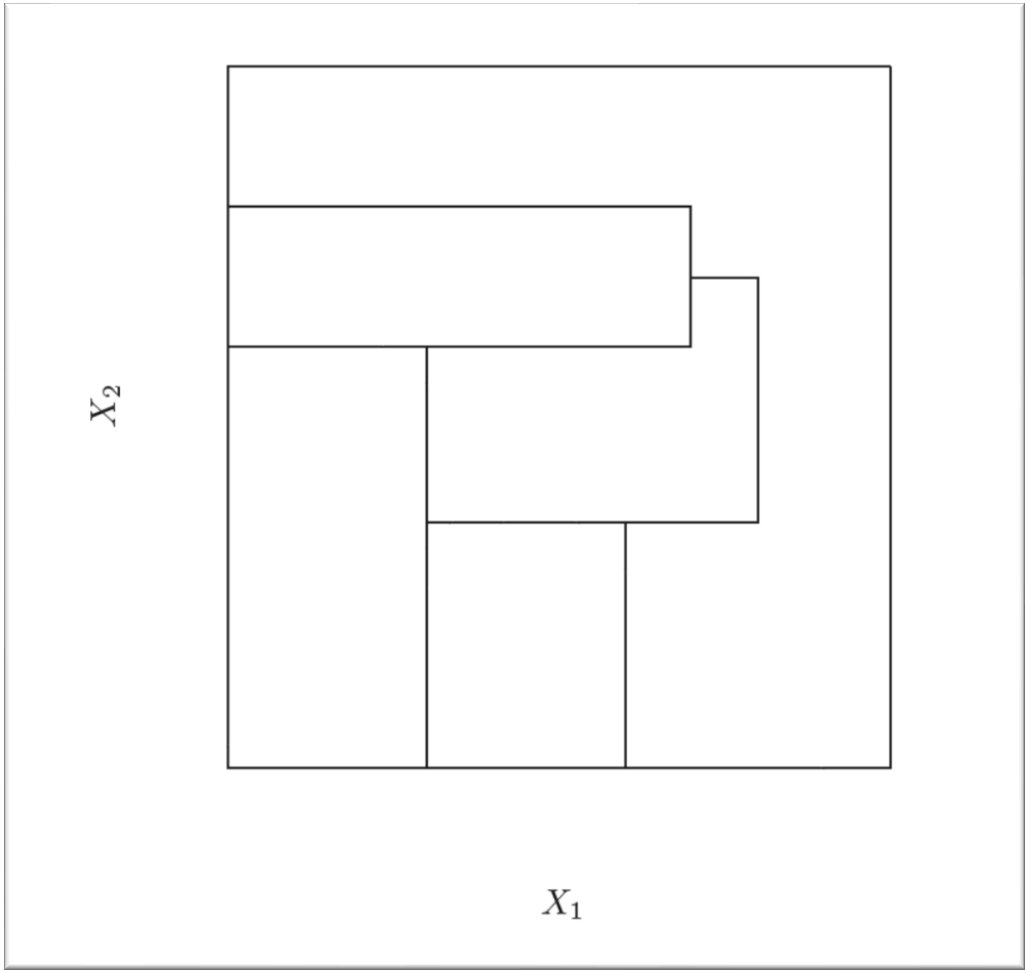




A PERSPECTIVE PLOT OF THE PREDICTION SURFACE CORRESPONDING TO THE PREVIOUS TREE



A PARTITION OF TWO-DIMENSIONAL FEATURE SPACE THAT COULD NOT RESULT FROM RECURSIVE BINARY SPLITTING



TREE PRUNING

IMPROVING TREE ACCURACY

- A large tree (with many terminal nodes) may tend to overfit the training data, leading to poor test set performance
- This is because the resulting tree might be too complex
- Generally, we can improve accuracy by pruning the tree i.e. cutting off some of the terminal nodes
- A smaller tree with fewer splits (that is, fewer regions $R_1, \ldots, R_J$) might lead to lower variance and better interpretation at the cost of a little bias

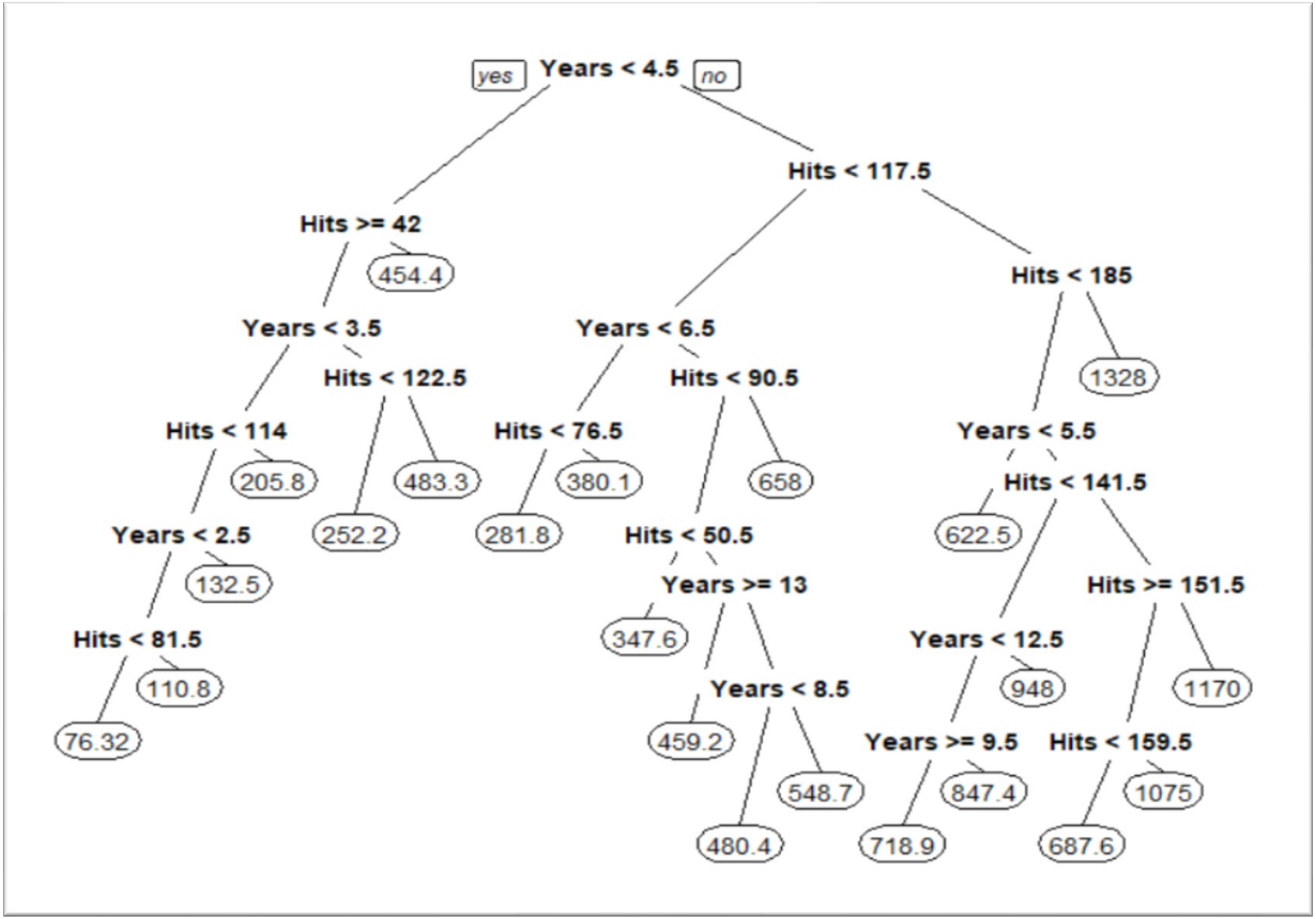
IMPROVING TREE ACCURACY

- One possible alternative to this process described above is to grow the tree only so long as the decrease in the RSS due to each split exceeds some (high) threshold
- This strategy will result in smaller trees, but is too short-sighted
- A seemingly worthless split early on in the tree might be followed by a very good split - that is, a split that leads to a large reduction in RSS later on

IMPROVING TREE ACCURACY

- A better strategy is to grow a very large tree T_0, and then prune it back in order to obtain a subtree
- How do we know how far back to prune the tree?
- Given a subtree, we can estimate its test error using cross validation
- However, estimating the cross-validation error for every possible subtree would be too cumbersome, since there is an extremely large number of possible subtrees
- Instead, we need a way to select a small set of subtrees for consideration

BIG REGRESSION TREE - T_0



COST COMPLEXITY PRUNING

- Cost complexity pruning - also known as weakest link pruning - is used to do this
- Rather than considering every possible subtree, we consider a sequence of trees indexed by a tuning parameter α
- For each value of α there exists a subtree $T \in T_0$ such that

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} \left(y_i - \hat{y}_{R_m}\right)^2 + \alpha\,|T| \rightarrow min$$

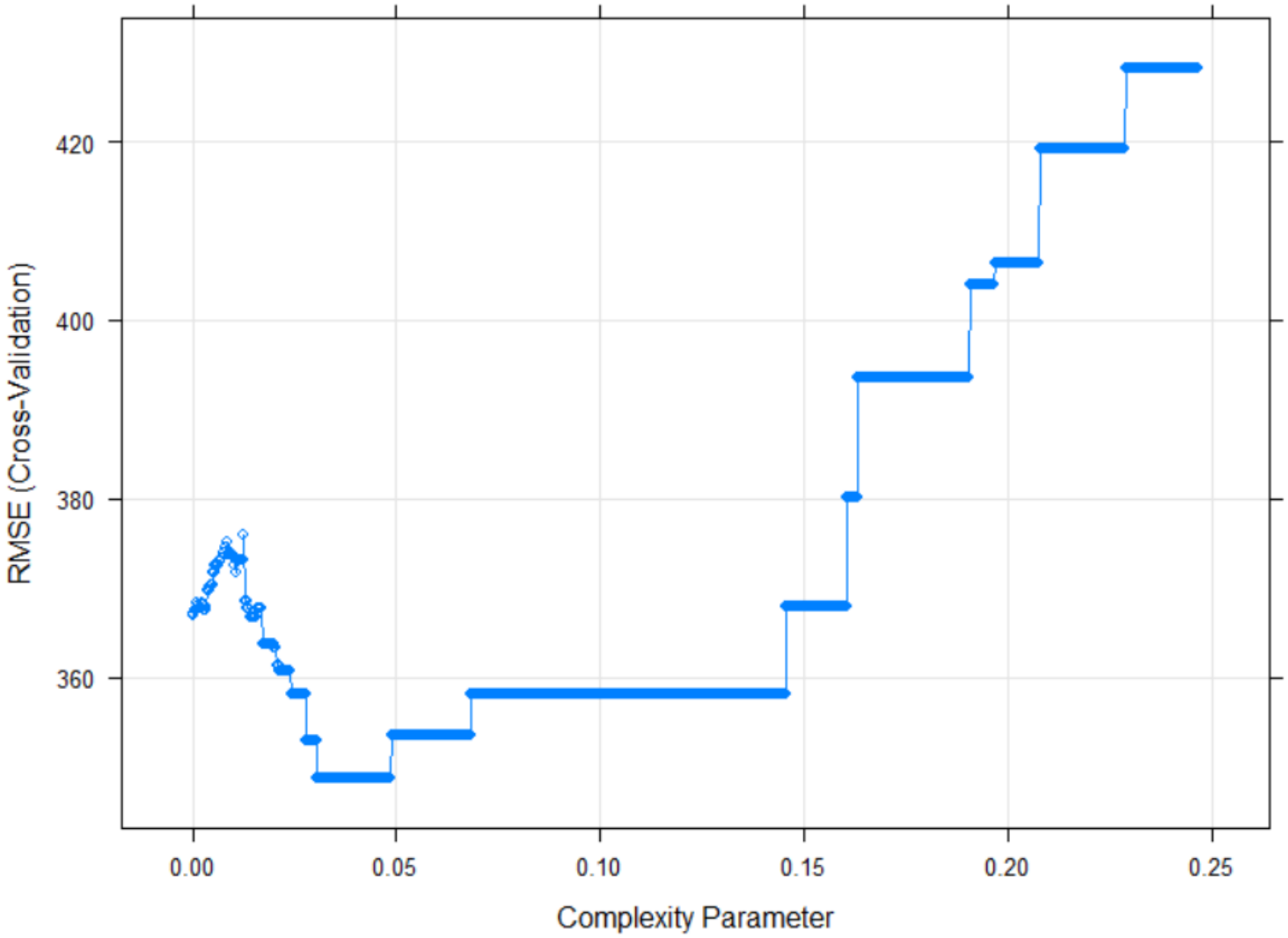
CHOOSING THE BEST SUBTREE

- The tuning parameter controls a trade-off between the subtree's complexity and its fit to the training data

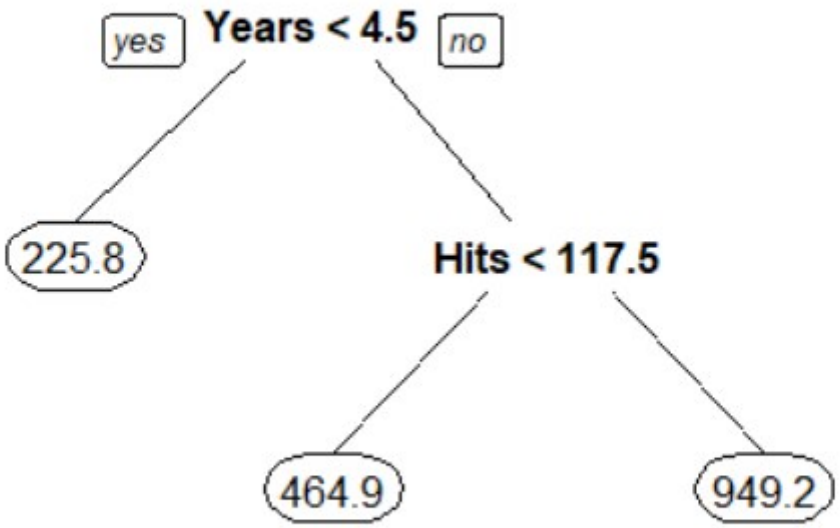
- We select an optimal value $\hat{\alpha}$ using cross-validation

- We then return to the full data set and obtain the subtree corresponding to $\hat{\alpha}$

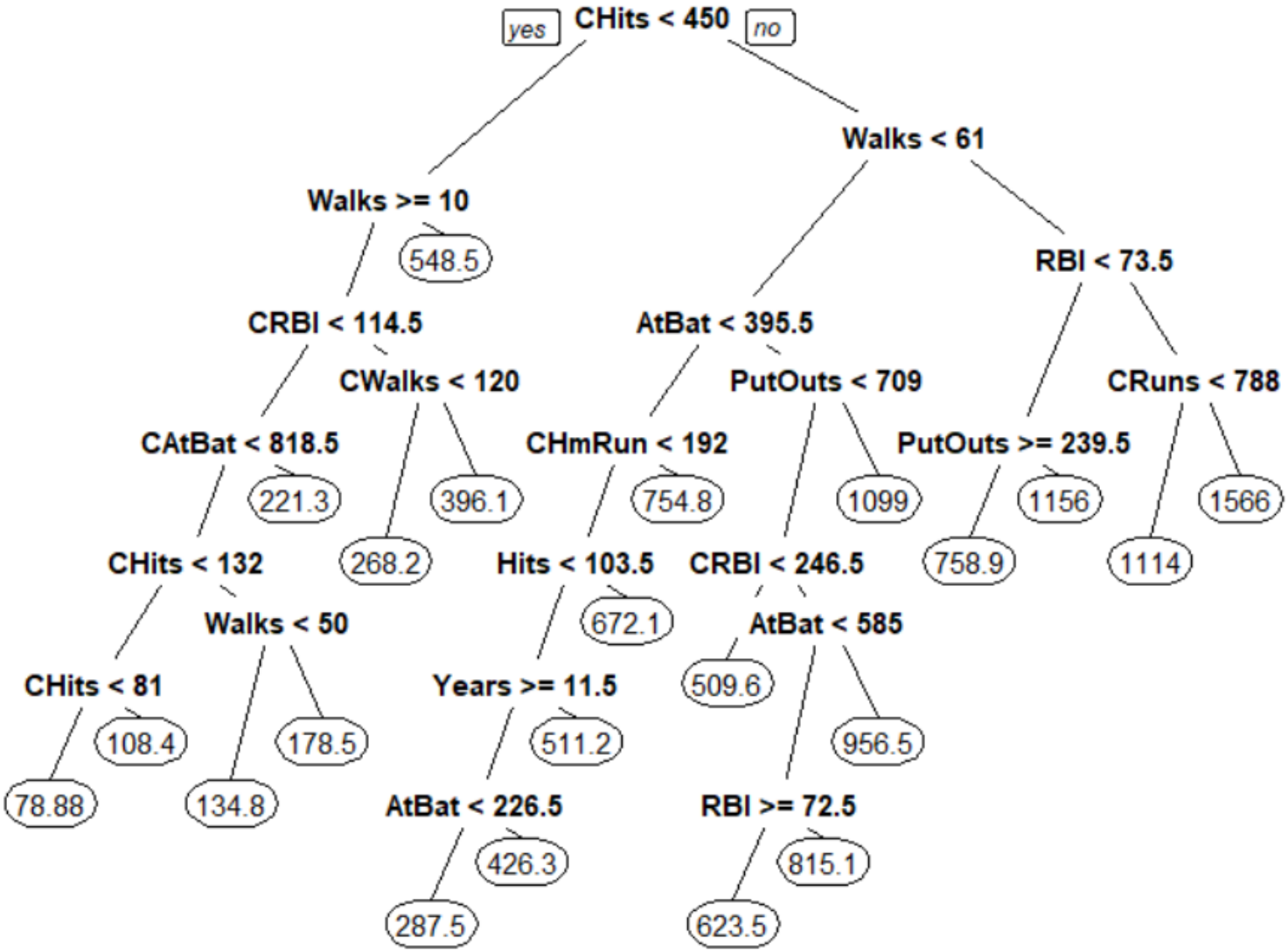
HITTERS DATA: CV RESULTS



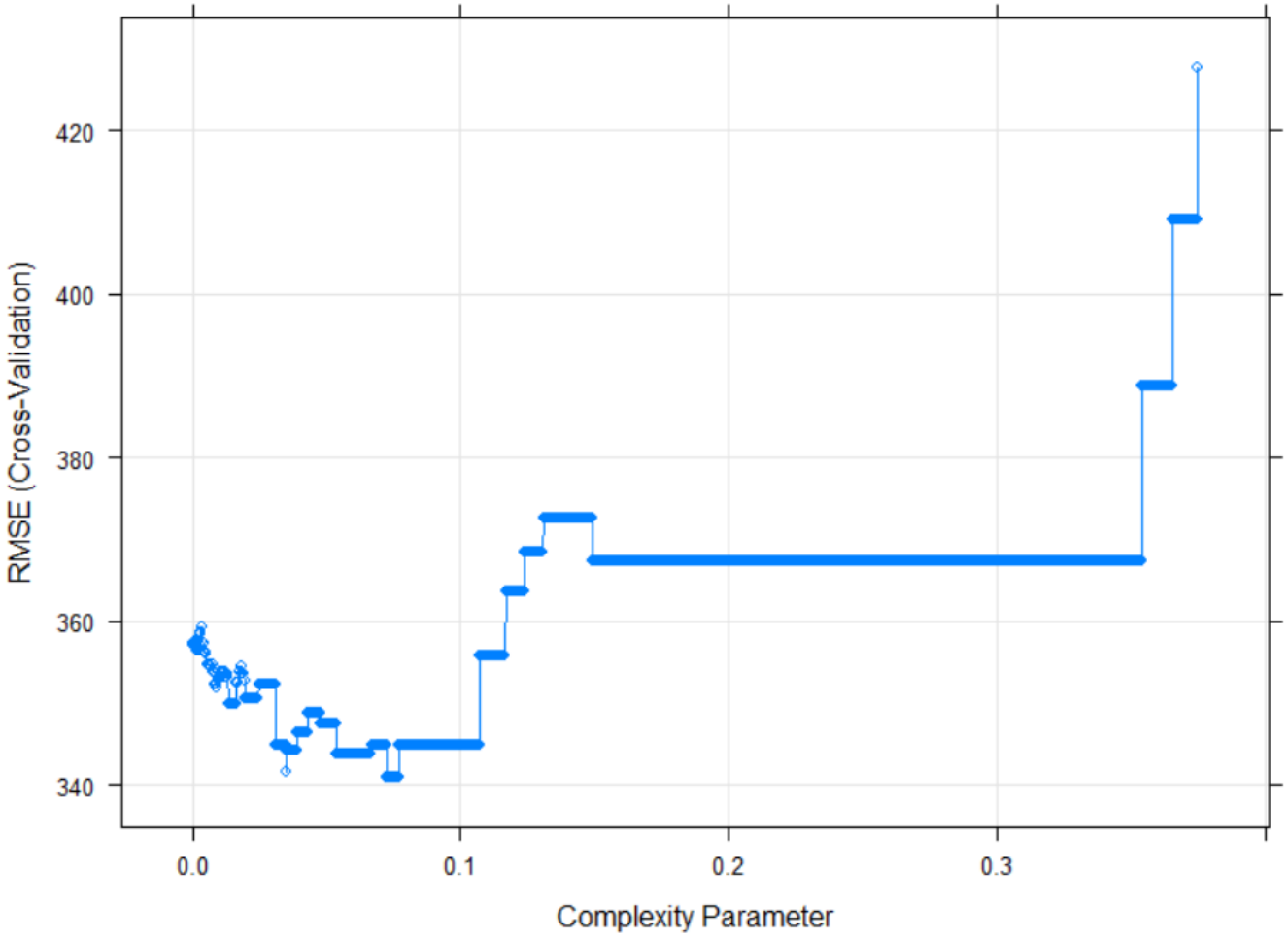
HITTERS DATA: REGRESSION TREE



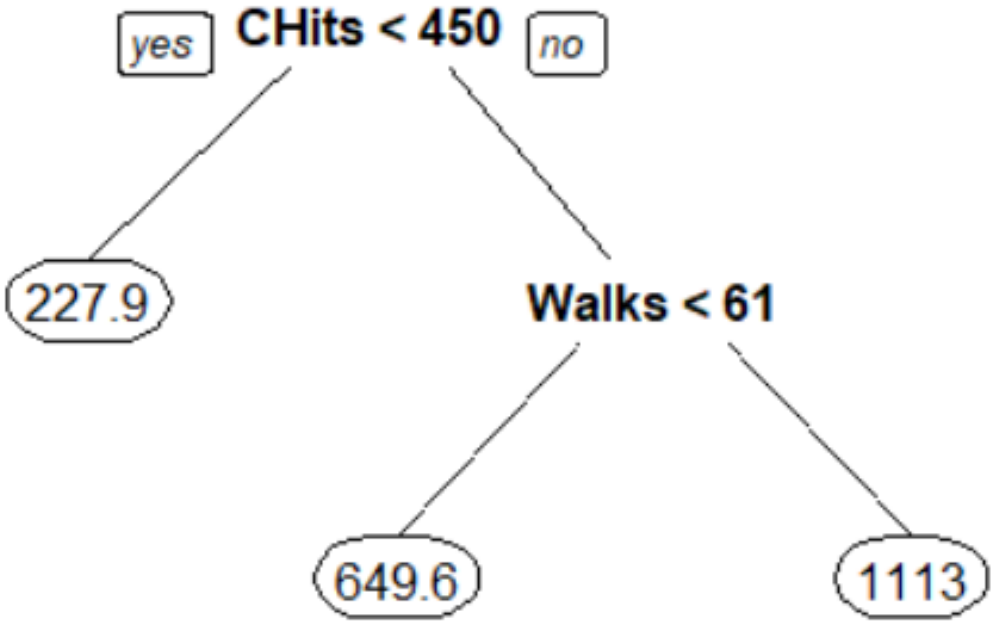
ENTIRE HITTERS DATA



OPTIMAL TREE PRUNING



OPTIMAL TREE



CLASSIFICATION TREES

GROWING A CLASSIFICATION TREE

- A classification tree is very similar to a regression tree except that we try to make a prediction for a **categorical** rather than **continuous** Y
- Recall that for a regression tree, the **predicted response** for an observation is given by the **mean response** of the training observations that belong to the same terminal node
- In contrast, for a classification tree, we predict that each observation belongs to the ***most commonly*** *occurring class* of training observations in the region to which it belongs

ERROR RATE

- Just as in the regression setting, we use recursive binary splitting to grow a classification tree

- In the classification setting, RSS cannot be used as a criterion for making the binary splits

- A natural alternative to RSS is the **classification error rate**

ERROR RATE

- Since, we plan to assign an observation in a given region to the *most commonly occurring class* of training observations in that region, the classification **error rate** is the fraction of the training observations in that region that do not belong to the most common class:

$$E_m = 1 - \max_k p_{mk}$$

where p_{mk} represents the proportion of training observations in the m^{th} region that are from the k^{th} class.

GINI INDEX

- However classification error is not sufficiently sensitive for tree-growing, and in practice two other measures are preferable
- The **Gini index** is defined by

$$G_m = \sum_{k=1}^{K} p_{mk}(1 - p_{mk})$$

- The Gini index takes on a small value if all of the p_{mk} are close to zero or one
- For this reason, the Gini index is referred to as a measure of **node purity** - a small value indicates that nodes contain predominantly observations from a single class

ENTROPY

- An alternative to the Gini index is **entropy**

$$D_m = -\sum_{k=1}^{K} p_{mk}\, log(p_{mk})$$

- Like the Gini index, the entropy will take on a small value if the m-th node is pure
- It turns out that the Gini index and the cross-entropy are very similar numerically

PARAMETERS PREFERENCES

- When building a classification tree, 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

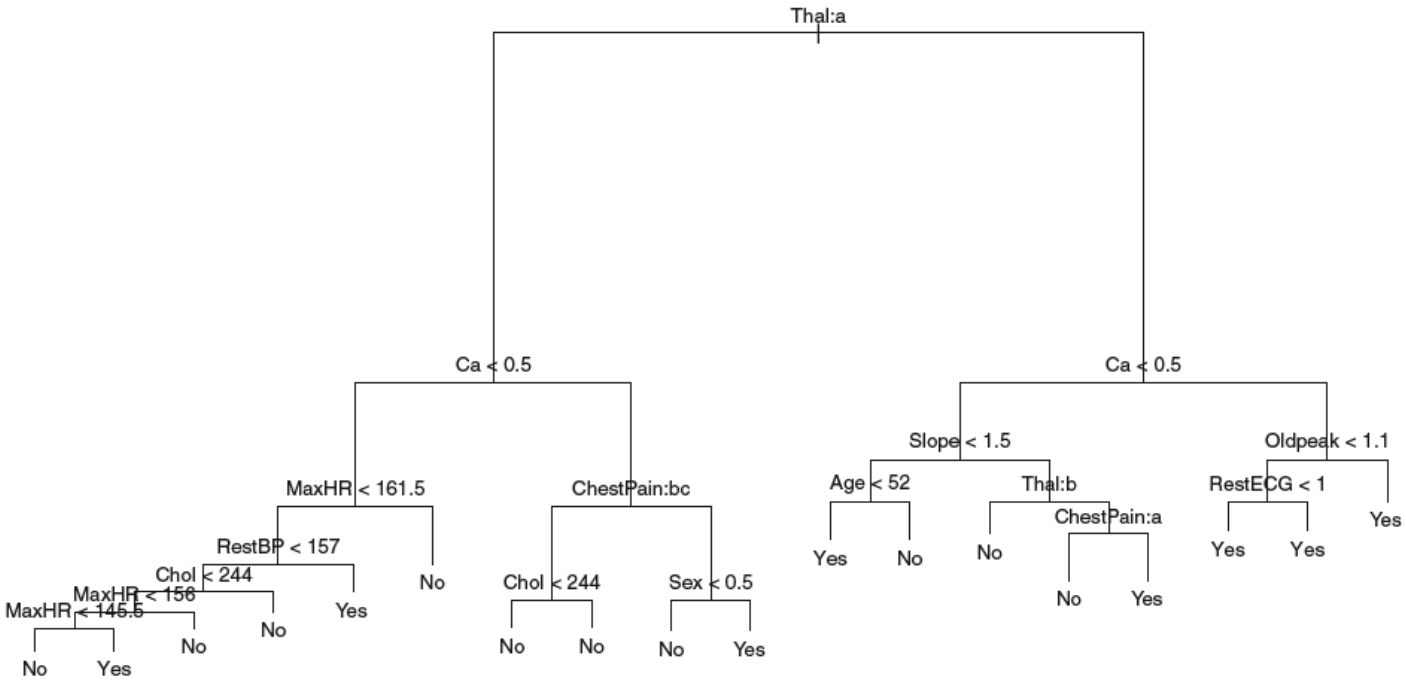
- Any of these three approaches might be used when **pruning** the tree, but the classification **error rate** is preferable if prediction accuracy of the final pruned tree is the goal

HEART DATA

- These data contain a binary outcome HD (heart disease) for 303 patients who presented with chest pain
- An outcome value of Yes indicates the presence of heart disease based on an angiographic test, while No means no heart disease
- There are 13 predictors including Age, Sex, Chol (a cholesterol measurement), and other heart and lung function measurements

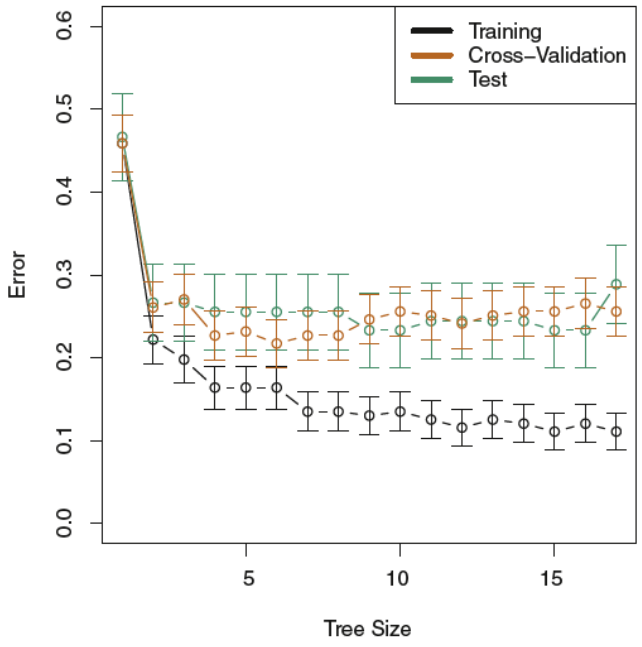
HEART DATA

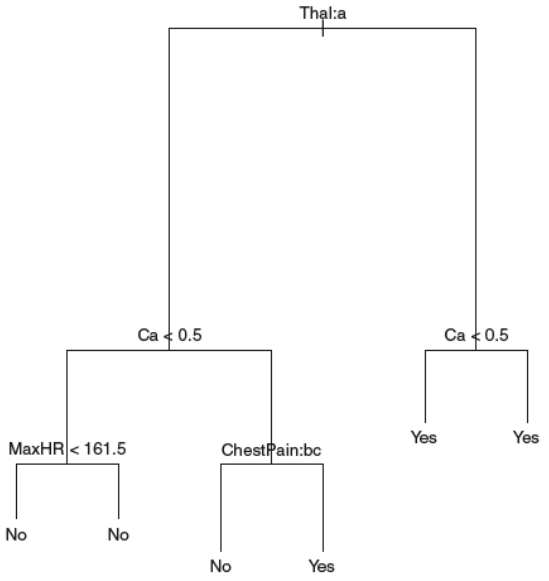
- Unpruned tree



HEART DATA

- *Cross-validation error, training, and test error, for different sizes of the pruned tree*
- *The pruned tree corresponding to the minimal cross-validation error*





HEART DATA

- There is a surprising characteristic in the trees: some of the splits yield two terminal nodes that have the *same predicted value*

- For instance, consider the split RestECG<1 near the bottom right of the unpruned tree

- Regardless of the value of RestECG, a response value of Yes is predicted for those observations

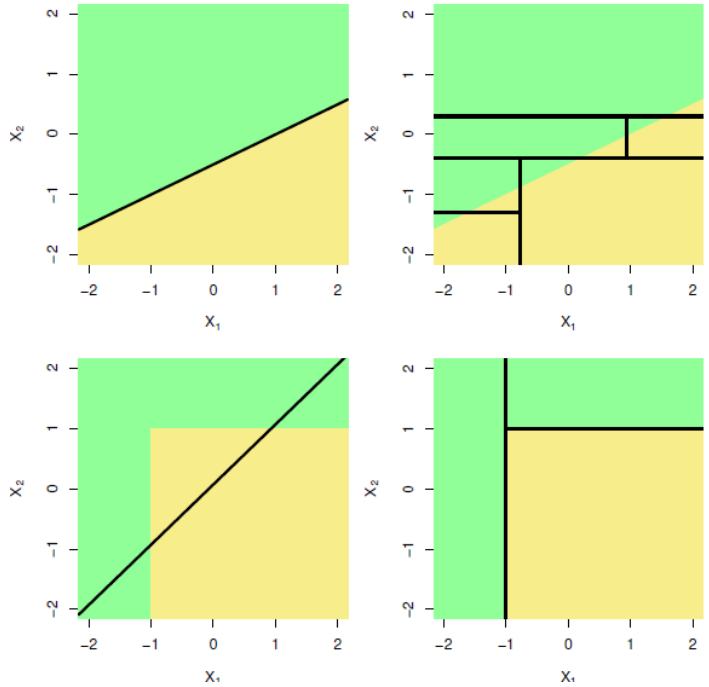
HEART DATA

- Why, then, is the split performed at all? The split is performed because it leads to increased *node purity*
- That is, all 9 of the observations corresponding to the right-hand leaf have a response value of Yes, whereas 7/11 of those corresponding to the left-hand leaf have a response value of Yes
- Why is node purity important? Suppose that we have a test observation that belongs to the region given by that right-hand leaf. Then we can be pretty certain that its response value is Yes. In contrast, if a test observation belongs to the region given by the left-hand leaf, then its response value is probably Yes, but we are much less certain
- Even though the split RestECG<1 does not reduce the classification error, it improves the Gini index and the entropy, which are more sensitive to node purity

TREES VS. LINEAR MODELS

- Which model is better?
  - If the relationship between the predictors and response is linear, then classical linear models such as linear regression would outperform regression trees
  - On the other hand, if the relationship between the predictors is non-linear, then decision trees would outperform classical approaches.

TREES VS. LINEAR MODEL



- Top Row: True linear boundary; Bottom row: true non-linear boundary
- Left column: linear model; Right column: tree-based model

PROS AND CONS OF DECISION TREES

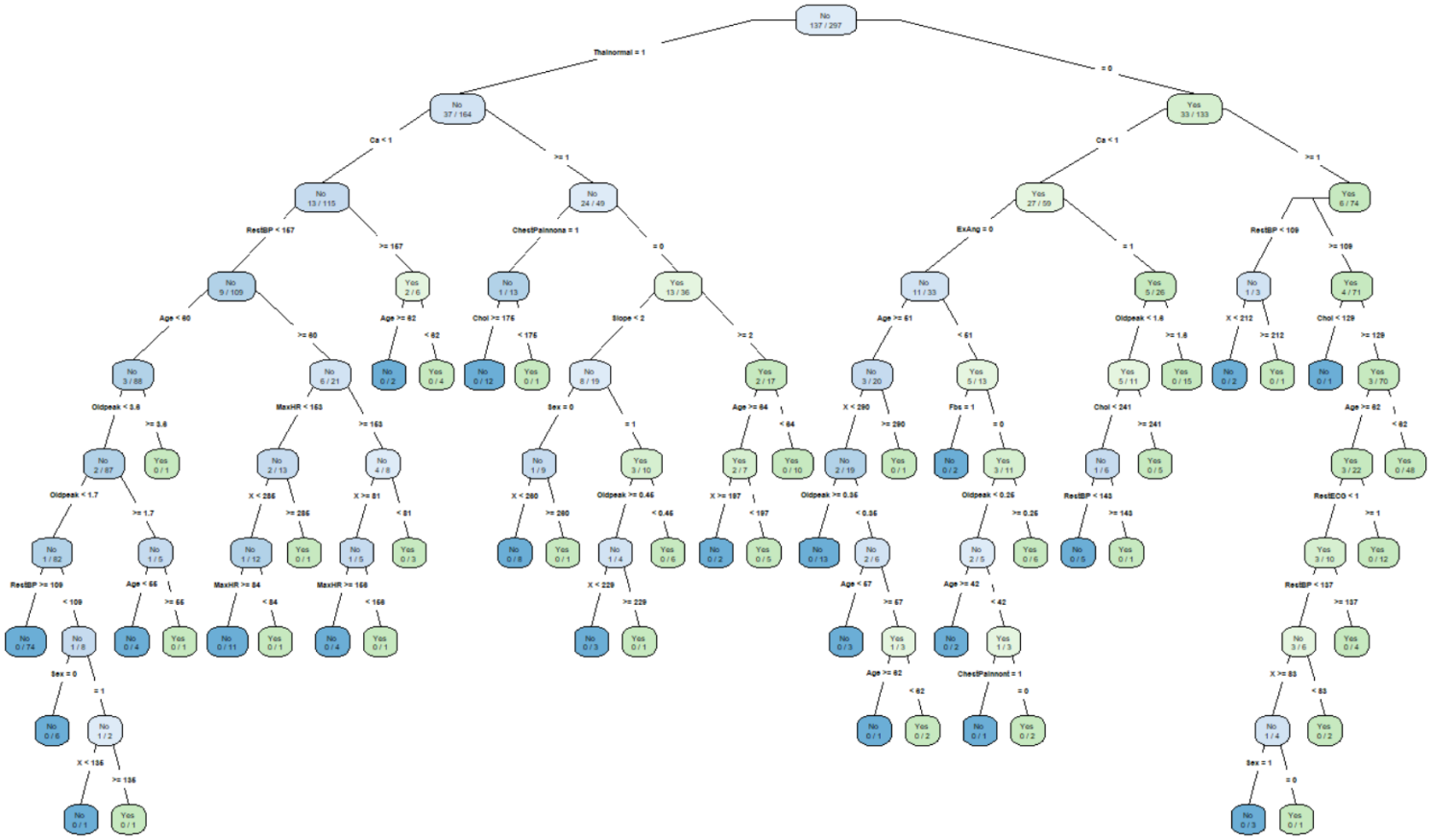
- **Pros:**
  - Trees are very easy to explain to people (probably even easier than linear regression)
  - Trees can be plotted graphically, and are easily interpreted even by non-expert
  - They work fine on both classification and regression problems
- **Cons:**
  - Trees don't have the same prediction accuracy as some of the more complicated approaches

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

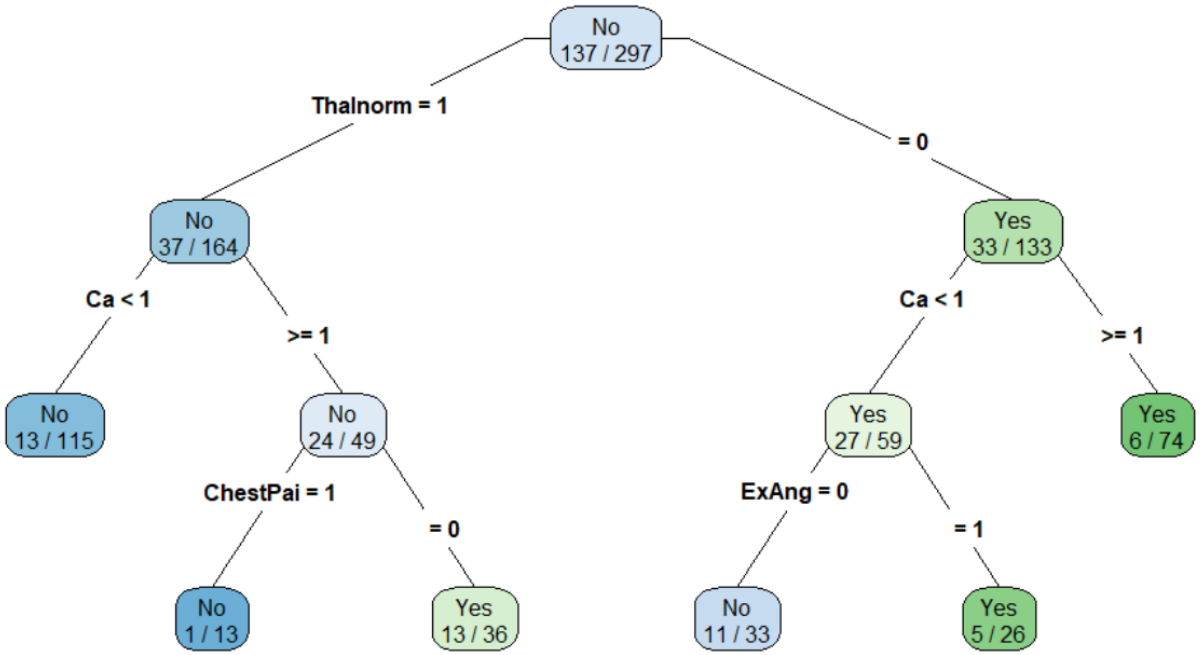
HEART DATA

- **'data.frame':** **297 obs. of 15 variables:**
- **X** : int 1 2 3 4 5 6 7 8 9 10 ...
- **Age** : int 63 67 67 37 41 56 62 57 63 53 ...
- **Sex** : int 1 1 1 1 0 1 0 0 1 1 ...
- **ChestPain** : Factor w/ 4 levels "asymptomatic",.. : 4 1 1 2 3 3 1 1 1 1 ...
- **RestBP** : int 145 160 120 130 130 120 140 120 130 140 ...
- **Chol** : int 233 286 229 250 204 236 268 354 254 203 ...
- **Fbs** : int 1 0 0 0 0 0 0 0 0 1 ...
- **RestECG** : int 2 2 2 0 2 0 2 0 2 2 ...
- **MaxHR** : int 150 108 129 187 172 178 160 163 147 155 ...
- **ExAng** : int 0 1 1 0 0 0 0 1 0 1 ...
- **Oldpeak** : num 2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
- **Slope** : int 3 2 2 3 1 1 3 1 2 3 ...
- **Ca** : int 0 3 2 0 0 0 2 0 1 0 ...
- **Thal** : Factor w/ 3 levels "fixed", "normal",.. : 1 2 3 2 2 2 2 2 3 3 ...
- **AHD** **: Factor w/ 2 levels "No", "Yes“** **: 1 2 2 1 1 1 2 1 2 2 ...**

HEART DATA


No 137 / 297
Thalnormal = 1
= 0
No 37 / 164
Yes 33 / 133
Ca < 1
>= 1
Ca < 1
>= 1
No 13 / 115
No 24 / 49
Yes 27 / 59
Yes 6 / 74
RestBP < 157
>= 157
ChestPainnona = 1
= 0
ExAng = 0
= 1
RestBP < 109
>= 109
No 9 / 105
Yes 2 / 6
No 1 / 13
Yes 13 / 36
No 11 / 33
Yes 5 / 26
No 1 / 3
Yes 4 / 71
Age < 60
>= 60
Age >= 62
< 62
Chol >= 176
< 176
Slope < 2
>= 2
Age >= 61
< 61
Oldpeak < 1.6
>= 1.6
X < 212
>= 212
Chol < 129
>= 129
No 3 / 88
No 6 / 21
No 0 / 2
Yes 0 / 4
No 0 / 12
Yes 0 / 1
No 8 / 19
Yes 2 / 17
No 3 / 20
Yes 5 / 13
Yes 5 / 11
Yes 0 / 15
No 0 / 2
Yes 0 / 1
No 0 / 1
Yes 3 / 70
Oldpeak < 0.6
>= 0.6
MaxHR < 163
>= 163
Sex = 0
= 1
Age >= 64
< 64
X < 290
>= 290
Fbs = 1
= 0
Chol < 241
>= 241
Age >= 62
< 62
No 2 / 87
Yes 0 / 1
No 2 / 13
No 4 / 8
No 1 / 9
Yes 3 / 10
Yes 2 / 7
Yes 0 / 10
No 2 / 19
Yes 0 / 1
No 0 / 2
Yes 3 / 11
No 1 / 6
Yes 0 / 5
Yes 3 / 22
Yes 0 / 48
Oldpeak < 1.7
>= 1.7
X < 286
>= 286
X >= 81
< 81
X >= 197
< 197
Oldpeak >= 0.36
< 0.36
Oldpeak < 0.25
>= 0.25
RestBP >= 143
< 143
RestECG < 1
>= 1
No 1 / 82
No 1 / 5
No 1 / 12
Yes 0 / 1
No 1 / 5
Yes 0 / 3
No 0 / 8
Yes 0 / 1
No 1 / 4
Yes 0 / 6
No 0 / 2
Yes 0 / 5
No 0 / 13
No 2 / 6
No 2 / 5
Yes 0 / 6
No 0 / 5
Yes 0 / 1
Yes 3 / 10
Yes 0 / 12
RestBP >= 109
< 109
Age < 55
>= 55
MaxHR >= 84
< 84
MaxHR >= 168
< 168
Oldpeak >= 0.45
< 0.45
Age < 67
>= 67
Age >= 42
< 42
RestBP < 137
>= 137
No 0 / 74
No 1 / 8
No 0 / 4
Yes 0 / 1
No 0 / 11
Yes 0 / 1
No 0 / 4
Yes 0 / 1
No 0 / 3
Yes 0 / 1
No 0 / 3
Yes 1 / 3
No 0 / 2
Yes 1 / 3
No 3 / 6
Yes 0 / 4
Sex = 0
= 1
Age >= 62
< 62
ChestPainnont = 1
= 0
X >= 83
< 83
No 0 / 6
No 1 / 2
No 0 / 1
Yes 0 / 2
No 0 / 1
Yes 0 / 2
No 1 / 4
Yes 0 / 2
X < 135
>= 135
Sex = 1
= 0
No 0 / 1
Yes 0 / 1
No 0 / 3
Yes 0 / 1

HEART DATA


No
137 / 297
Thalnorm = 1
= 0
No
37 / 164
Yes
33 / 133
Ca < 1
>= 1
Ca < 1
>= 1
No
13 / 115
No
24 / 49
Yes
27 / 59
Yes
6 / 74
ChestPai = 1
= 0
ExAng = 0
= 1
No
1 / 13
Yes
13 / 36
No
11 / 33
Yes
5 / 26

BAGGING

PROBLEM

- Decision trees discussed earlier suffer from high variance!
  - If we randomly split the training data into 2 parts, and fit decision trees on both parts, the results could be quite different
- We would like to have models with low variance
- Bootstrap aggregation or bagging is a general purpose procedure for reducing the variance of a statistical learning method

WHY BAGGING?

- Bagging is an extremely powerful idea based on two things:
  - Averaging: reduces variance!
  - Bootstrapping: plenty of training datasets!
- Why does averaging reduces variance?
  - 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

WHY BAGGING?

- To apply bagging to regression trees, we simply construct *B* regression trees using *B* bootstrapped training sets, and average the resulting predictions
- These trees are grown deep, and are not pruned
- Hence each individual tree has high variance, but low bias
- Averaging these *B* trees reduces the variance
- Bagging has been demonstrated to give impressive improvements in accuracy by combining together hundreds or even thousands of trees into a single procedure

REGRESSION BAGGING

- We generate B different bootstrapped training data sets
- We then train our method on the b^{th} bootstrapped training set in order to get $f^b(x)$, the prediction at a point x
- We then average all the predictions to obtain

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

CLASSIFICATION BAGGING

- We generate B different bootstrapped training data sets
- We then train our method on the b^{th} bootstrapped training set in order to get $f^b(x)$, the class prediction for a point x
- 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

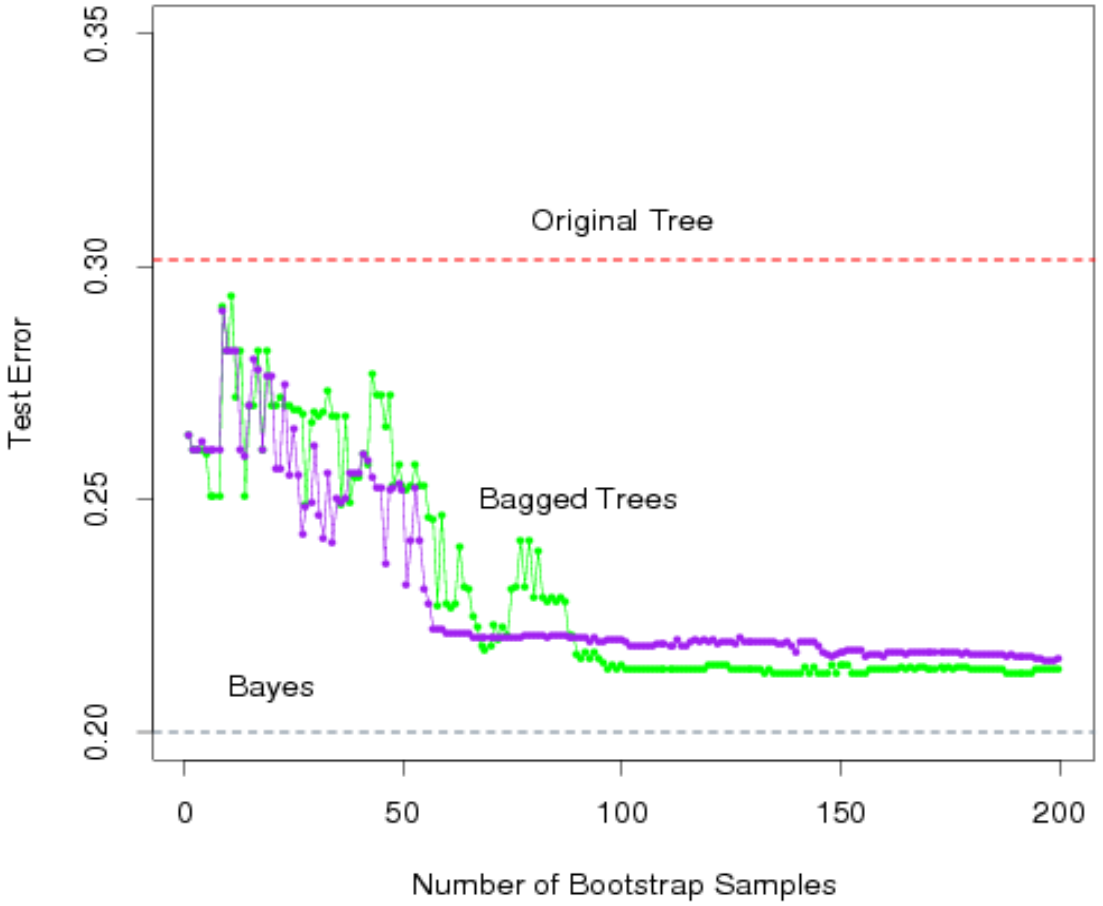
- It turns out that there is a very straightforward way to estimate the test error of a bagged model without performing the cross-validation
- Recall that the key to bagging is that trees are repeatedly fit to bootstrapped subsets of the observations
- It can be proved that on average, each bagged-tree makes use of two-thirds of the observations
- 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

OUT-OF-BAG ERROR ESTIMATION

- 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 $B/3$ predictions for the i^{th} observation, which we average (for regression) or take a majority vote (for classification)
- Overall OOB MSE or OOB classification error can be obtained
- This estimate is essentially the LOOCV error for bagging, if B is large

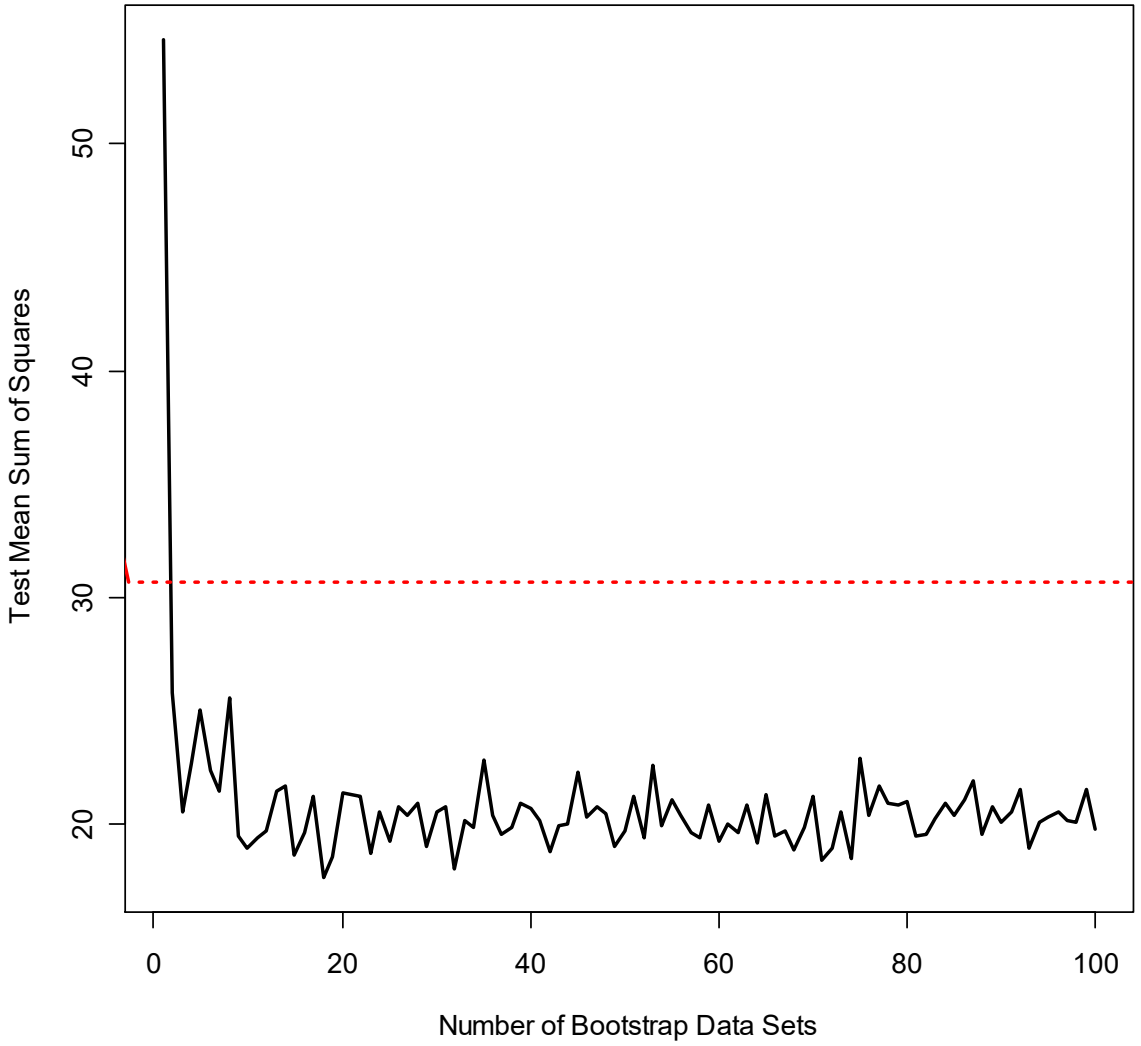
CLASSIFICATION: COMPARISON OF ERRORS

- Here the green line represents a simple majority vote approach
- The purple line corresponds to averaging the probability estimates
- Both do far better than a single tree (dashed red) and get close to the Bayes error rate (dashed grey)



REGRESSION: COMPARISON OF ERRORS

- The red line represents the test mean sum of squares using a single tree.
- The black line corresponds to the bagging error rate



FEATURE IMPORTANCE

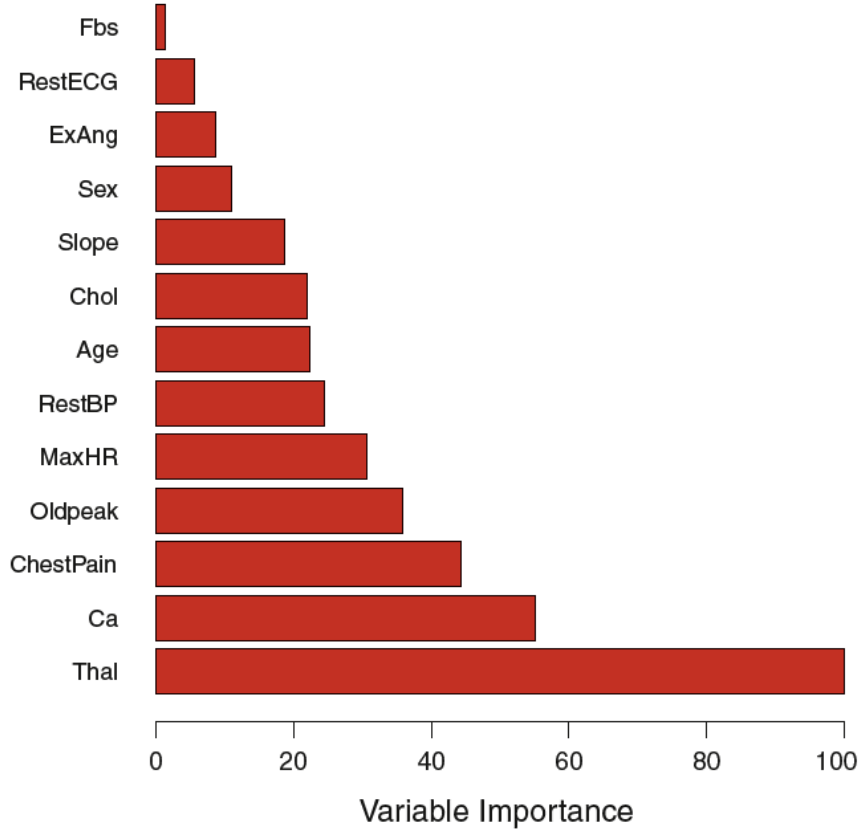
- Although the collection of bagged trees is much more difficult to interpret than a single tree, one can obtain an overall summary of the importance of each predictor
  - using the RSS for bagging regression trees or
  - the Gini index for bagging classification trees

- In the case of bagging regression trees, we can 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, in the context of bagging classification trees, we can add up the total amount that the Gini index is decreased by splits over a given predictor, averaged over all *B* trees

HEART DATA: FEATURE IMPORTANCE

- Variable importance is computed using the mean decrease in Gini index, and expressed relative to the maximum (Thal: Thallium stress test)



DISADVANTAGE OF BAGGING

- As we have discussed, bagging typically results in improved accuracy over prediction using a single tree
- Unfortunately, however, it can be difficult to interpret the resulting model
- When we bag a large number of trees, it is no longer possible to represent the resulting statistical learning procedure using a single tree
- Thus, bagging improves prediction accuracy at the expense of interpretability

RANDOM FORESTS

DISADVANTAGE OF BAGGING

- Suppose that there is one very strong predictor in the data set, along with a number of other moderately strong predictors
- Then in the collection of bagged trees, most or all of the trees will use this strong predictor in the top split
- Consequently, all of the bagged trees will look quite similar to each other
- Hence the predictions from the bagged trees will be highly correlated
- In particular, this means that bagging will not lead to a substantial reduction in variance over a single tree in this setting

DISADVANTAGE OF BAGGING

- Unfortunately, averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities

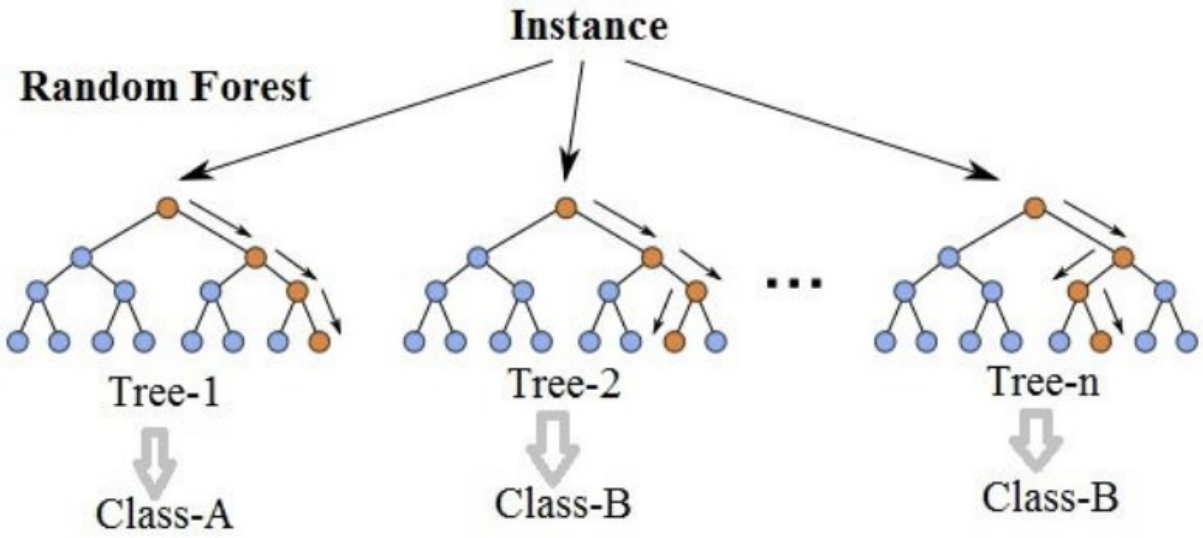
- This means that bagging will not lead to a substantial reduction in variance over a single tree in this setting

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
- 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 $m = \sqrt{p}$

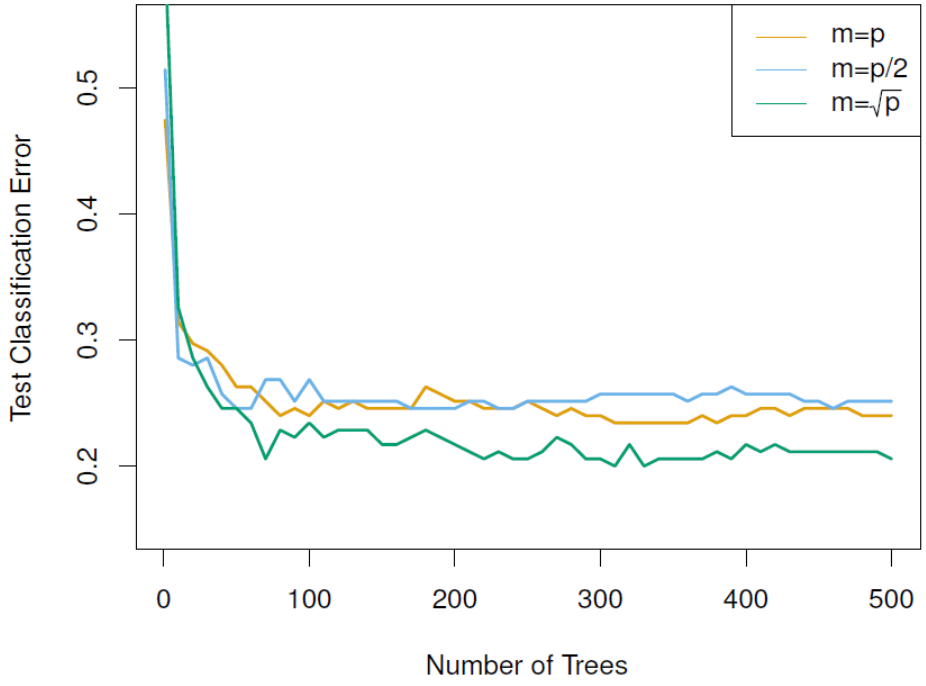
RANDOM FORESTS

- Weak learners classify observations differently



EXAMPLE

- Results from RF for the 15-class gene expression data set with $p = 500$ predictors
- Random forests ($m < p$) lead to a slight improvement over bagging ($m = p$)
- A single classification tree has an error rate of 45.7%



VARIABLE IMPORTANCE AND RELATIVE INFLUENCE PLOTS

- 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 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

BOOSTING

BOOSTING

- Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification

- 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

- In Boosting, the trees are grown sequentially:
  - each tree is grown using information from previously grown trees

- Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead learns slowly

- Boosting does not involve bootstrap sampling; instead each tree is fit on a modified version of the original data set

THE IDEA BEHIND BOOSTING

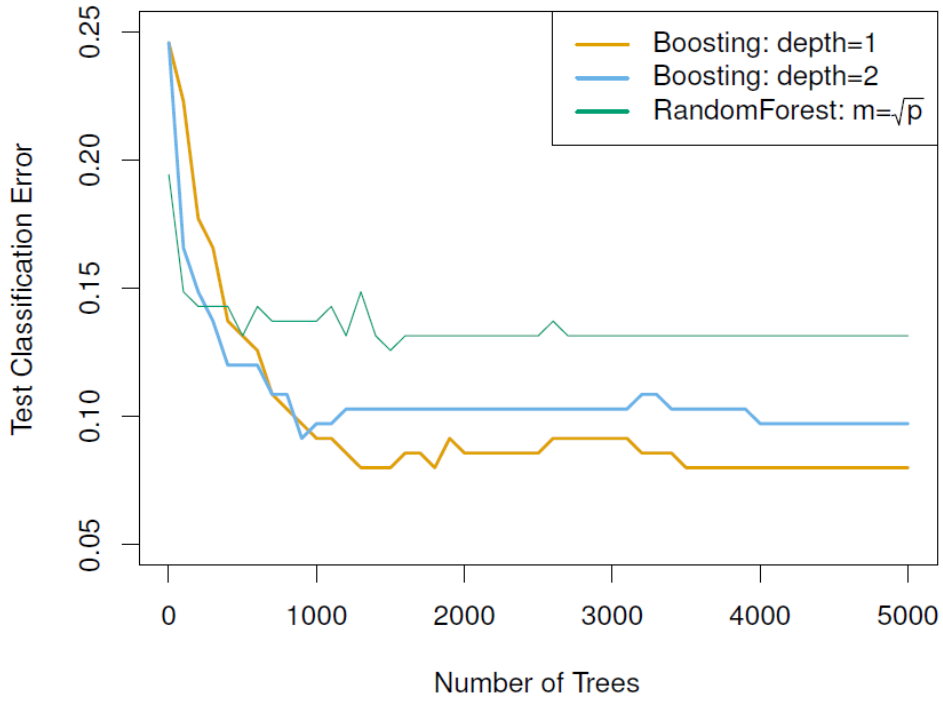
- Given the current model, we fit a decision tree to the residuals from the model
- We then add this new decision tree into the fitted function in order to update the residuals
- Each of these trees can be rather small, with just a few terminal nodes, determined by the parameter d in the algorithm
- By fitting small trees to the residuals, we slowly improve prediction in areas where it does not perform well
- The shrinkage parameter λ slows the process down even further, allowing more and different shaped trees to attack the residuals

THE ALGORITHM FOR REGRESSION

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 $\hat{f}^b(x)$ 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
   $$\hat{f}(x) = \hat{f}(x) + \lambda \hat{f}^b(x)$$
   c) Update the residuals
   $$r_i = r_i - \lambda \hat{f}^b(x_i)$$
3. Output the boosted model
$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

GENE EXPRESSION DATA (CLASSIFICATION)

- Boosting applied to the 15-class cancer gene expression data set, in order to develop a classifier that can distinguish the normal class from the 14 cancer classes



DETAILS OF PREVIOUS FIGURE

- Boosting for classification will be discussed at the end of the slide deck
- But the results of the previous figure are from performing boosting and random forests on the fifteen-class gene expression data set in order to predict cancer versus normal
- The test error is displayed as a function of the number of trees For the two boosted models, $\lambda = 0.01$
- Depth-1 trees slightly outperform depth-2 trees, and both outperform the random forests, although the standard errors are around 0.02, making none of these differences significant
- The test error rate for a single tree is 24%

TUNING PARAMETERS FOR BOOSTING

- 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

- 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

- 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 consists of a single split
- More generally d is the interaction depth, and controls the interaction order of the boosted model, since d splits can involve at most d variables

BOOSTING FOR CLASSIFICATION

- We omit details here
- The main idea is to increase the weights of misclassified training data points to be selected for further learning
- Algorithms:
  - AdaBoost (Adaptive Boosting)
  - Gradient Boosting
  - XGBoost (Extreme Gradient Boosting)
- For a high-level overview
  - https://hackernoon.com/boosting-algorithms-adaboost-gradient-boosting-and-xgboost-f74991cad38c
  - https://hackernoon.com/gradient-boosting-and-xgboost-90862daa6c77