Topic 6. Tree-Based Methods

Outline

- Regression tree
- Classification tree
- Tree Pruning
- Improving Trees

Regression Tree

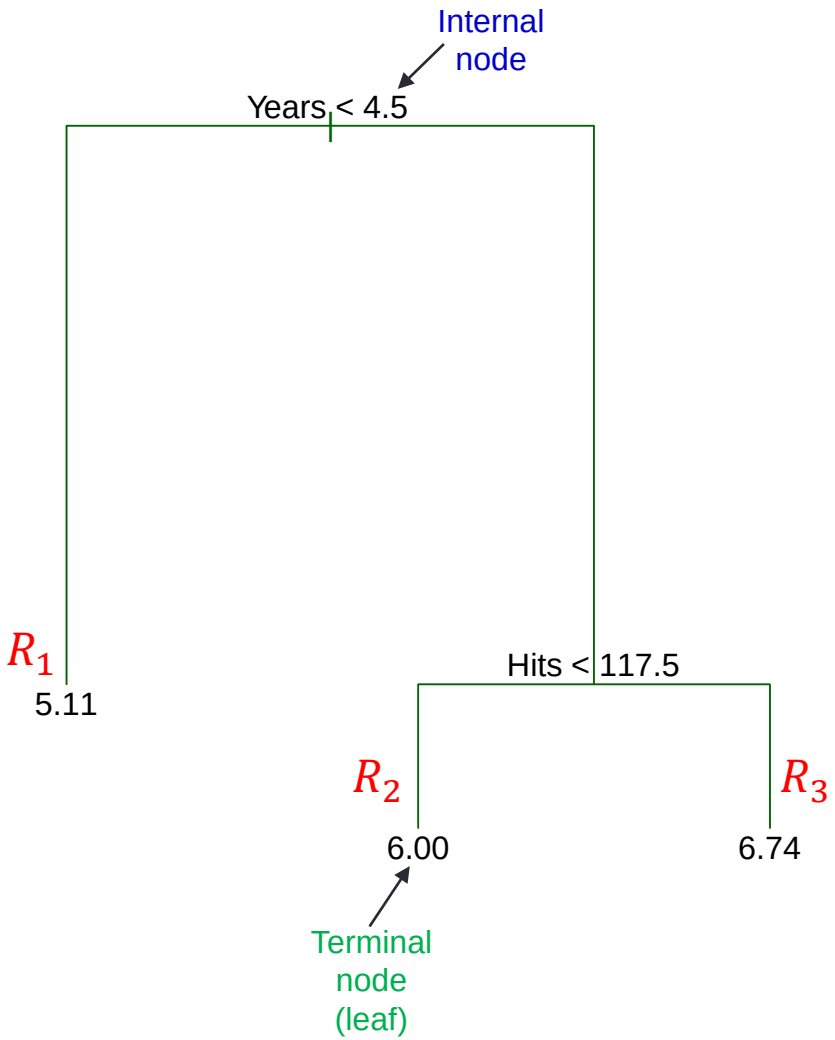
Partitioning the Predictor Space

- One way to make predictions/classification is to divide the predictor space (i.e. all the possible values for $X_1, X_2, \ldots, X_p$) into distinct regions, say $R_1, R_2, \ldots, R_k$.

- Then for every $\boldsymbol{X}$ that falls in a particular region (say R_j) we make the prediction using data in that region as training data. For example, when the response is numerical, we can use the mean of data in that region as prediction.

Regression Tree Example

- In the **Hitters** dataset, the predicted baseball player Salary is the number in each terminal node. It is the mean of the response for the observations that fall there.
- Note that Salary is measured in 1000s, and log-transformed
- The predicted salary for a player who played in the league for more than 4.5 years and had less than 117.5 hits last year is $\$1000 \times e^{6.00} = \$402,834$



Regions of Predictor Space

- The tree segments the players into three regions of predictor space:

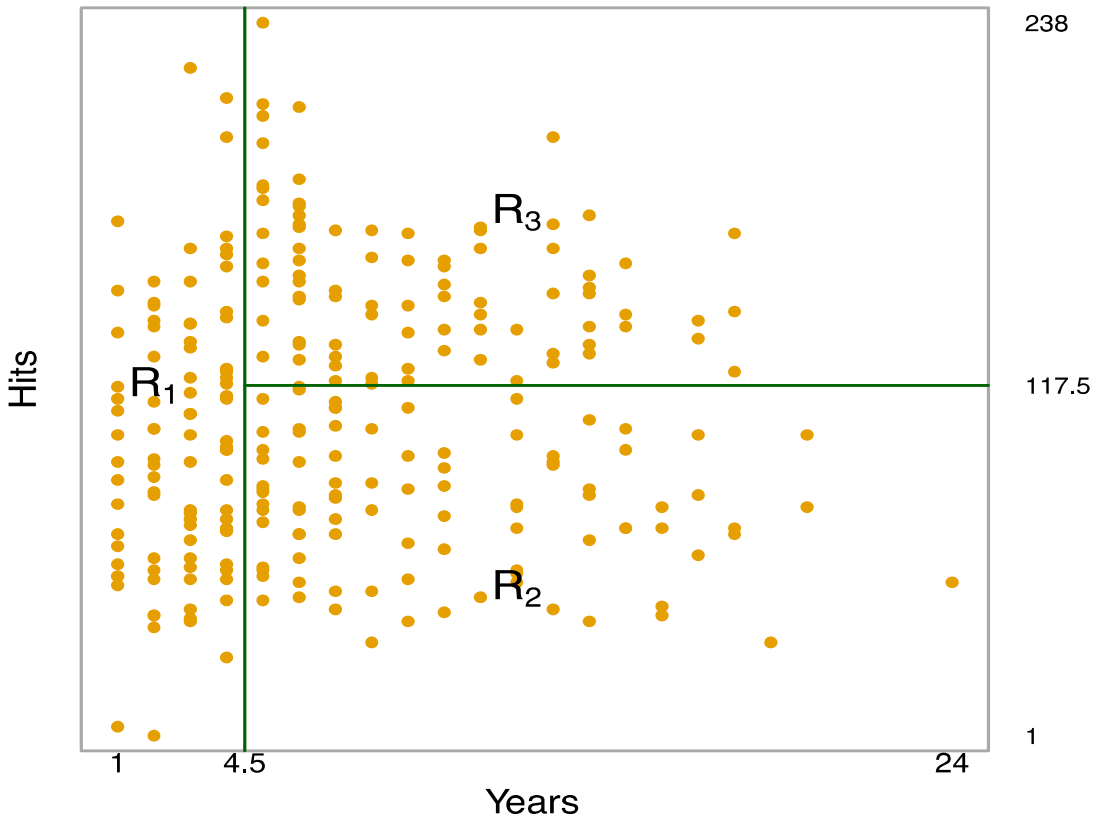
  R_1: players who have played for 4.5 or fewer years

  R_2: players who have played for 4.5 or more years and made fewer than 117.5 hits last year

  R_3: players who have played for five or more years and made at least 117.5 hits last year

- Prediction is made in each region separately

Another Way of Visualizing the Decision Tree



Linear regression

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

Regression tree

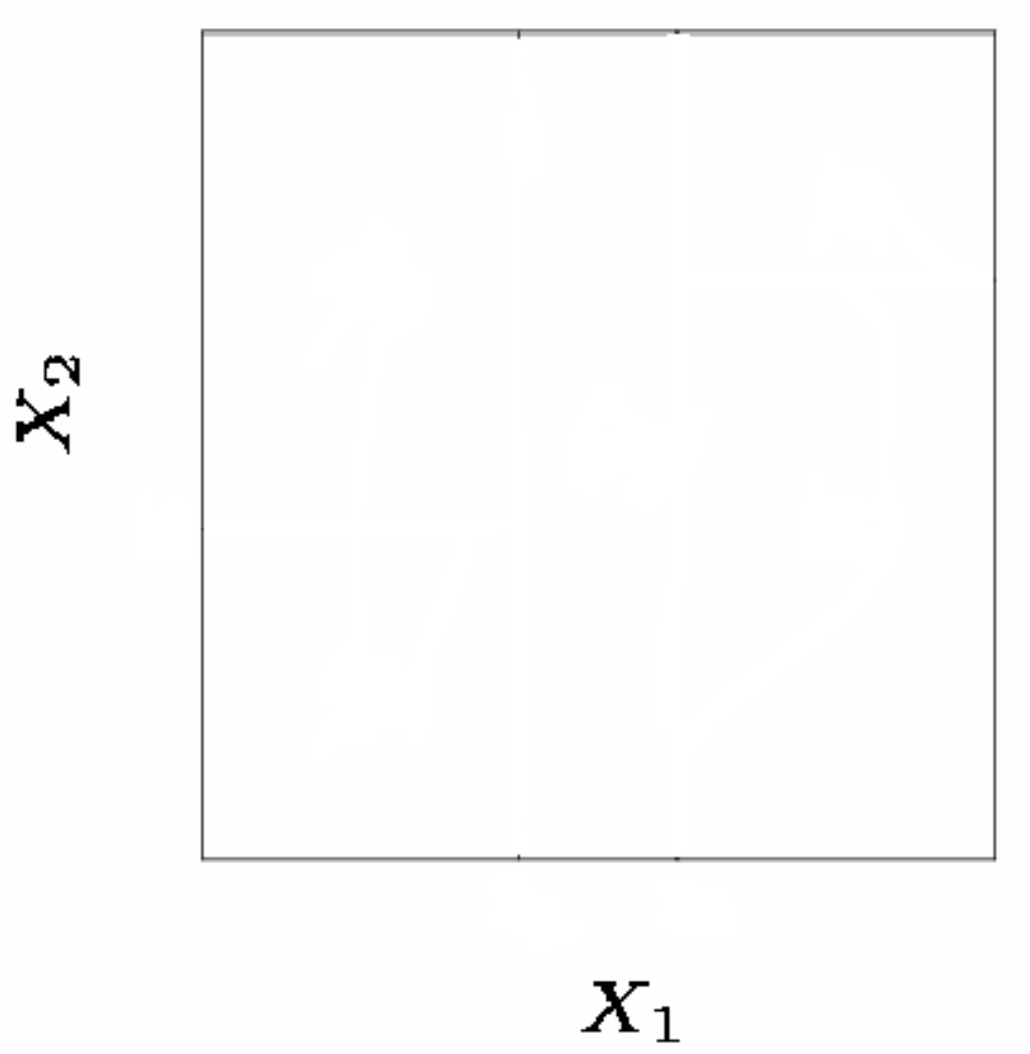
$$f(X) = \sum_{m=1}^{M} c_m \cdot 1_{(X \in R_m)}$$

Some Natural Questions

1. Where to split? i.e., how do we decide on the regions $R_1, R_2, \ldots, R_k$, or equivalently, what tree structure should we use?

2. What values should we use for the prediction in each region $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_k$?

1. Where to Split?

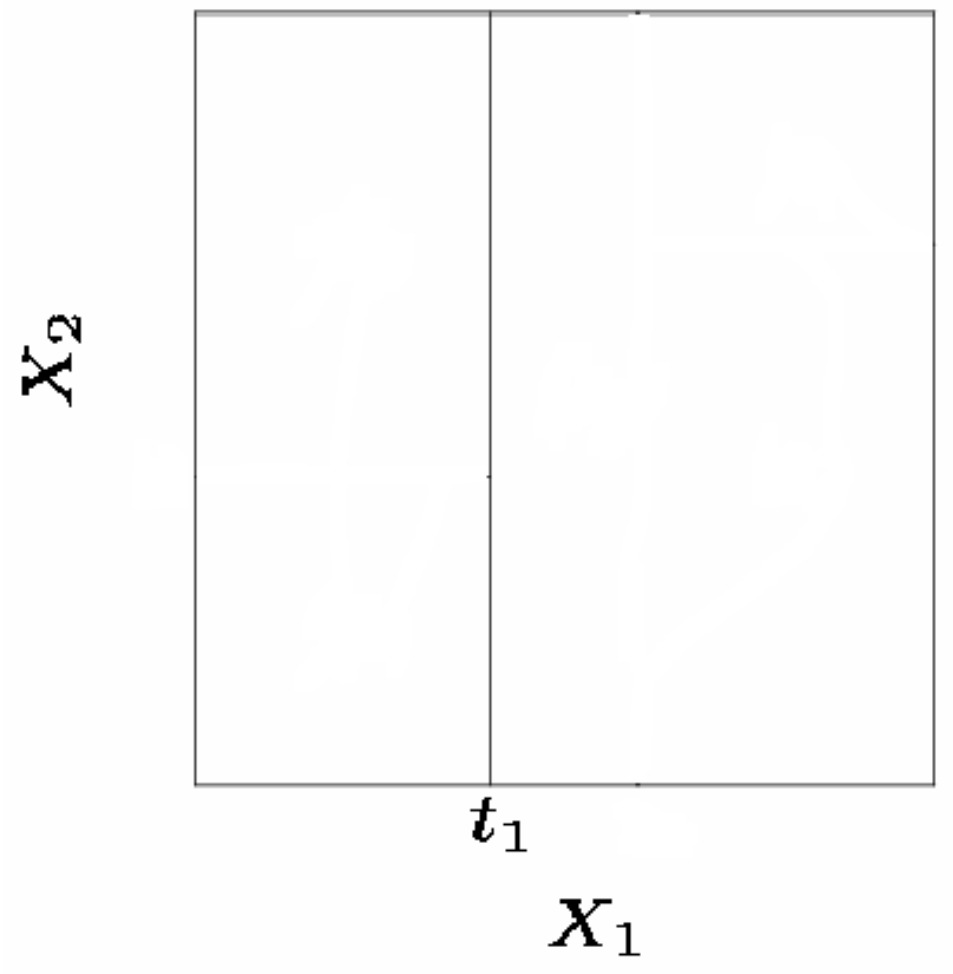


Recursive Binary Splitting—Step 1

- We consider splitting the predictor space into two regions, $X_j > s$ and $X_j < s$ for all possible values of j and s (i.e., all predictors and possible cutpoints).
- We then choose the j and s that result in the lowest RSS on the training data.

Result of Step 1

- Here the optimal split was on X_1 at point t_1.

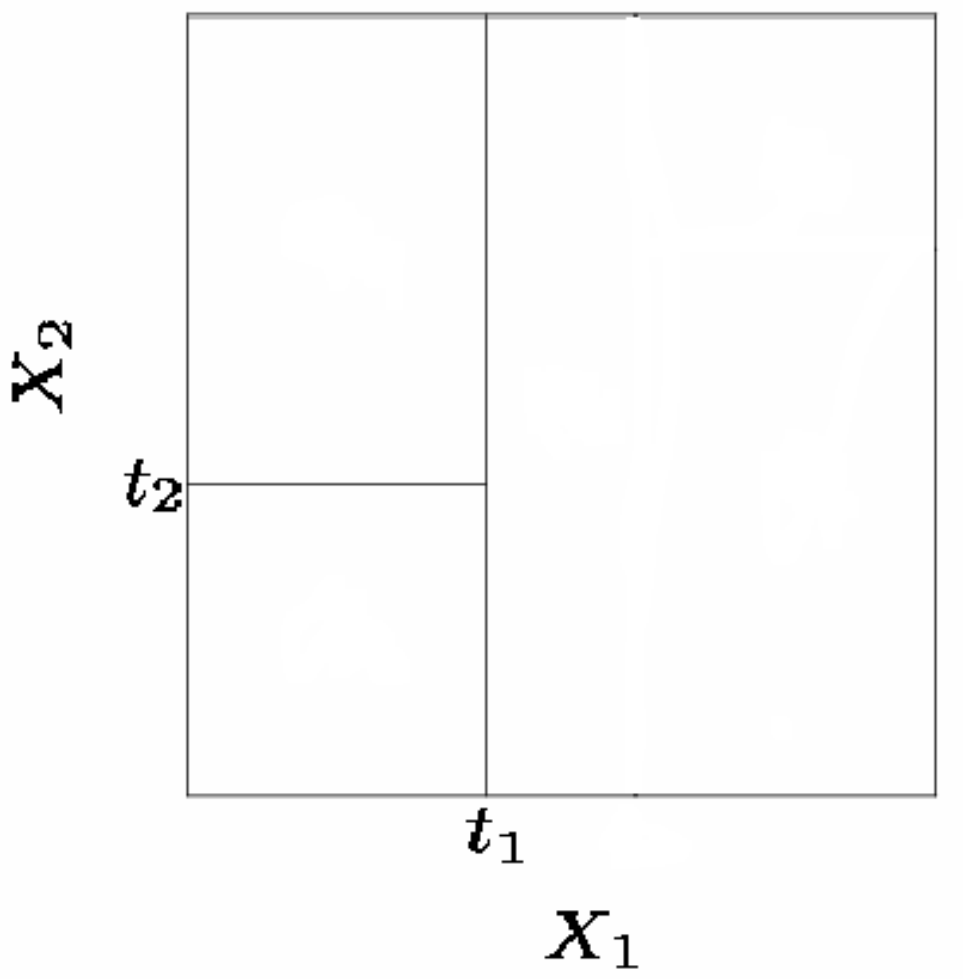


Recursive Binary Splitting—Step 2

- Now we repeat the process looking for the next best split except that we must also consider whether to split the first region or the second region.
- Again the criteria is smallest RSS.

Result of Step 2

- Here the optimal split was the left region on X_2 at point t_2.

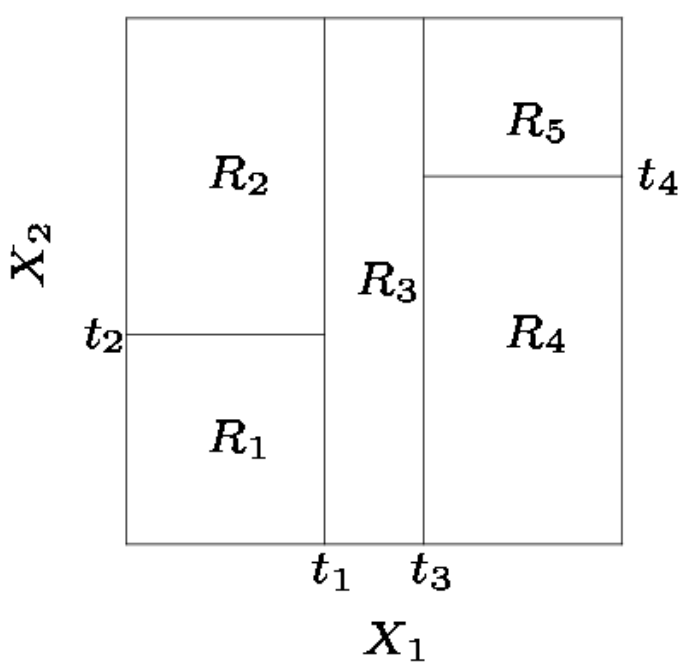


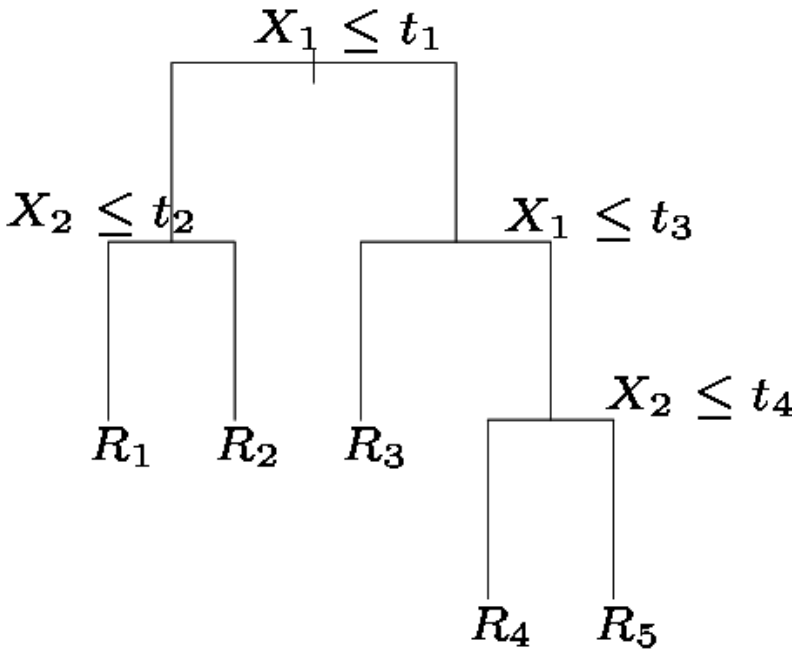
Recursive Binary Splitting—Step 3

- This process continues until our regions have too few observations to continue, e.g., all regions have 5 or fewer points.

Final Output

- When we create partitions this way we can always represent them using a tree structure.
- This provides a very simple way to explain the model to a non-expert.



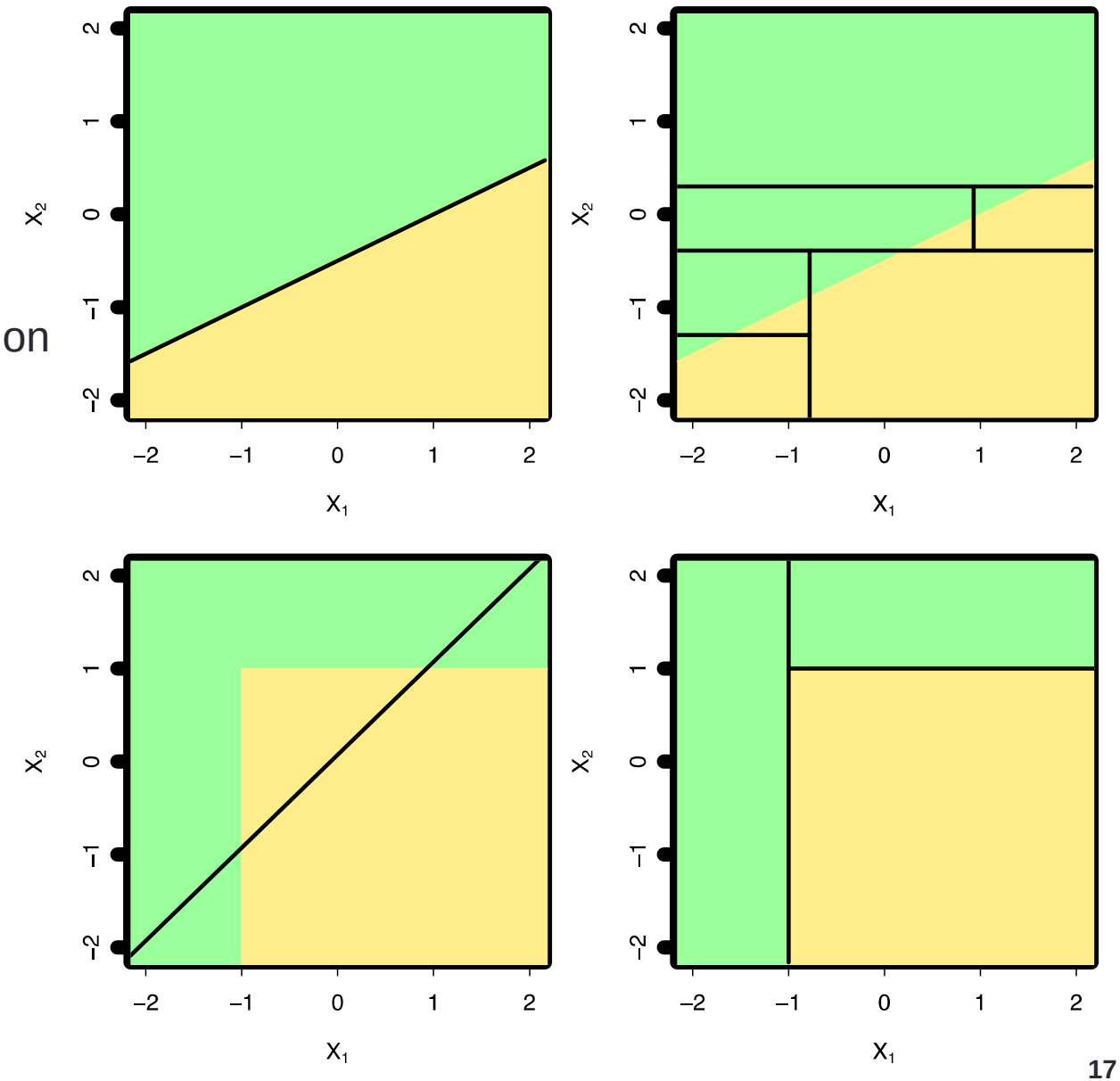


2. What Values Used for Predictions?

- Find Predictions $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_k$
- Simple!
- For region R_j, the best prediction is simply the *average* of all the responses in the training data that fell in this region.

Trees vs. Linear Model: Classification Example

- Top row: the true decision boundary is linear
  - Left: linear model (good)
  - Right: decision tree
- Bottom row: the true decision boundary is non-linear
  - Left: linear model
  - Right: decision tree (good)



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 highly non-linear and complex, then decision trees would outperform classical approaches.

Pros of Decision Trees

- Trees are very easy to explain to people, probably even easier than linear regression!
- Some people believe that decision trees more closely mirror human decision-making than do classical regression and classification approaches.
- Trees can be displayed graphically, and are easily interpreted even by non-expert (especially if they are small).
- They can easily handle qualitative predictors without the need to create dummy variables.

Cons of Decision Trees

- Trees don’t have the same prediction accuracy as some of the more complicated approaches that we examine in this course.

- Trees can be very non-robust. A small change in the data can cause a large change in the final estimated tree.

Classification Tree

Classification Trees

- Suppose for example we have two regions R_1 and R_2 with $\hat{y}_1 = 0$, $\hat{y}_2 = 1$.

- Then for any value of $\boldsymbol{X}$ such that $\boldsymbol{X} \in R_1$, we would predict 0, otherwise if $\boldsymbol{X} \in R_2$, we would predict 1.

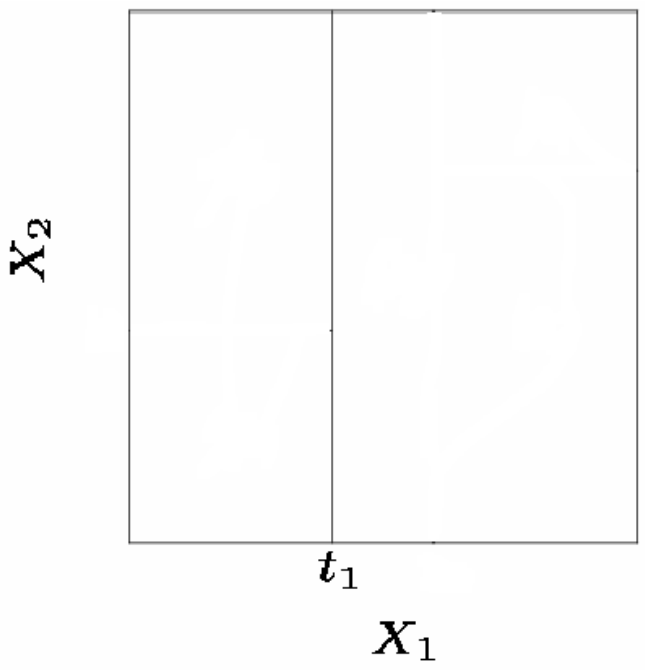
- The two questions:

  (1) how to split the predictor space?

  (2) what is used for prediction in each region?

1. Recursive Binary Splitting

- Follow the same procedure as in regression tree
- The only difference is that *RSS will not be used for making the binary splits. Criteria defined for classification will be used*.



Performance Measures in Classification Trees

- **Classification error rate**

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

- **Gini index**

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

- **Cross-entropy**

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

- **Deviance**

$$-2 \sum_m \sum_k n_{mk} \log \hat{p}_{mk}$$

$\hat{p}_{mk}$: proportion of training observations in the mth region that are from the kth class.

2. Predictions

- Find Predictions $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_k$
- For region R_j, the best prediction is the *most commonly occurring class* of training observations in that region.

True Pruning

Improving Tree Accuracy

- A large tree (i.e., one with many terminal nodes) may tend to overfit the training data.
- A small tree tends to have lower variance and better interpretation.
- Generally, we can improve accuracy by "pruning" the tree, i.e., cutting off some of the terminal nodes.

(allthingsclipart.com)

Tree Pruning

- Grow a very large tree and then prune it back to obtain a subtree.
- Find a number of subtrees in this way.
- Use cross validation to estimate test error of each subtree.
- Choose the subtree that has lowest test error.

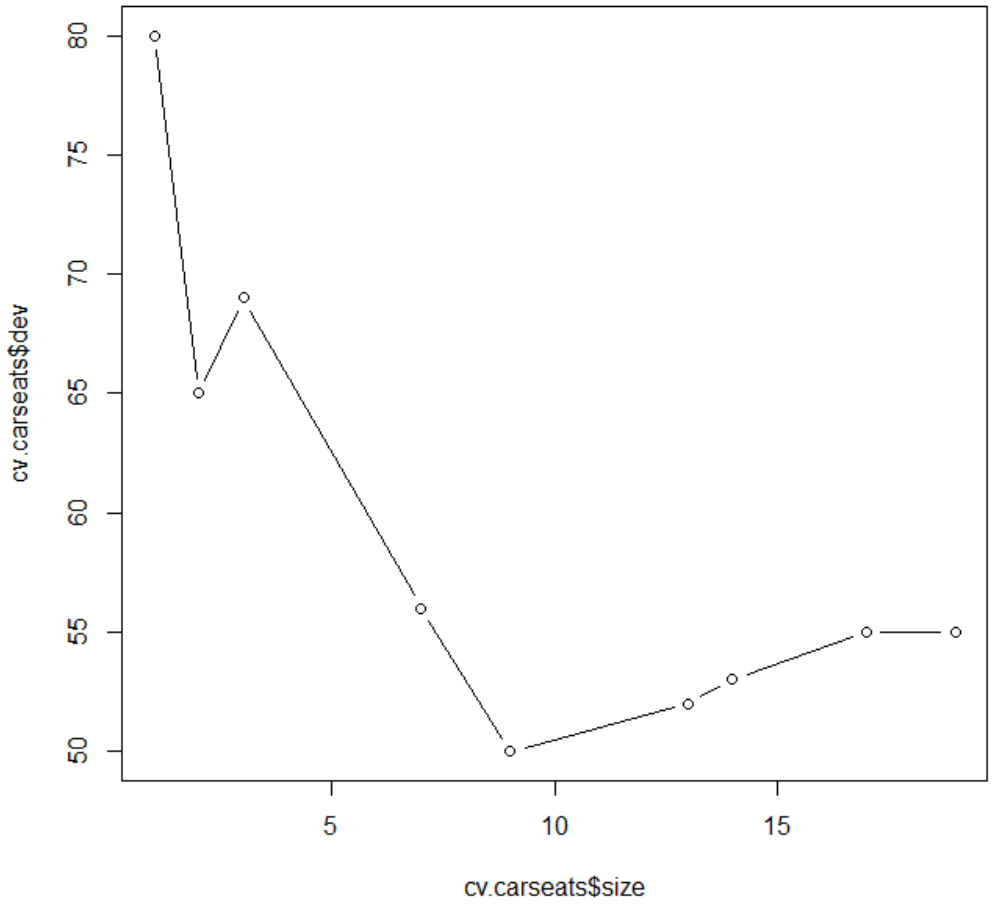
Carseats Data Set

- Carseats data set
- Predict Sales (child car seat sales) in 400 locations based on predictors such as Price, Urban (No/Yes, indicate whether the store is in an urban or rural location), US (No/Yes, indicate whether the store is in the US or not) and ShelveLoc (*Bad/Medium/Good*, indicate the quality of the shelving location, i.e., the space within a store in which the car seat is displayed).
- Use ifelse() function to create a binary response variable, which takes on Yes if Sales>8, and No otherwise. Then a classification tree is fit for the data.

(**Note:** the purpose of creating a binary response from the continuous response Sales is only to illustrate how to build a classification tree!)

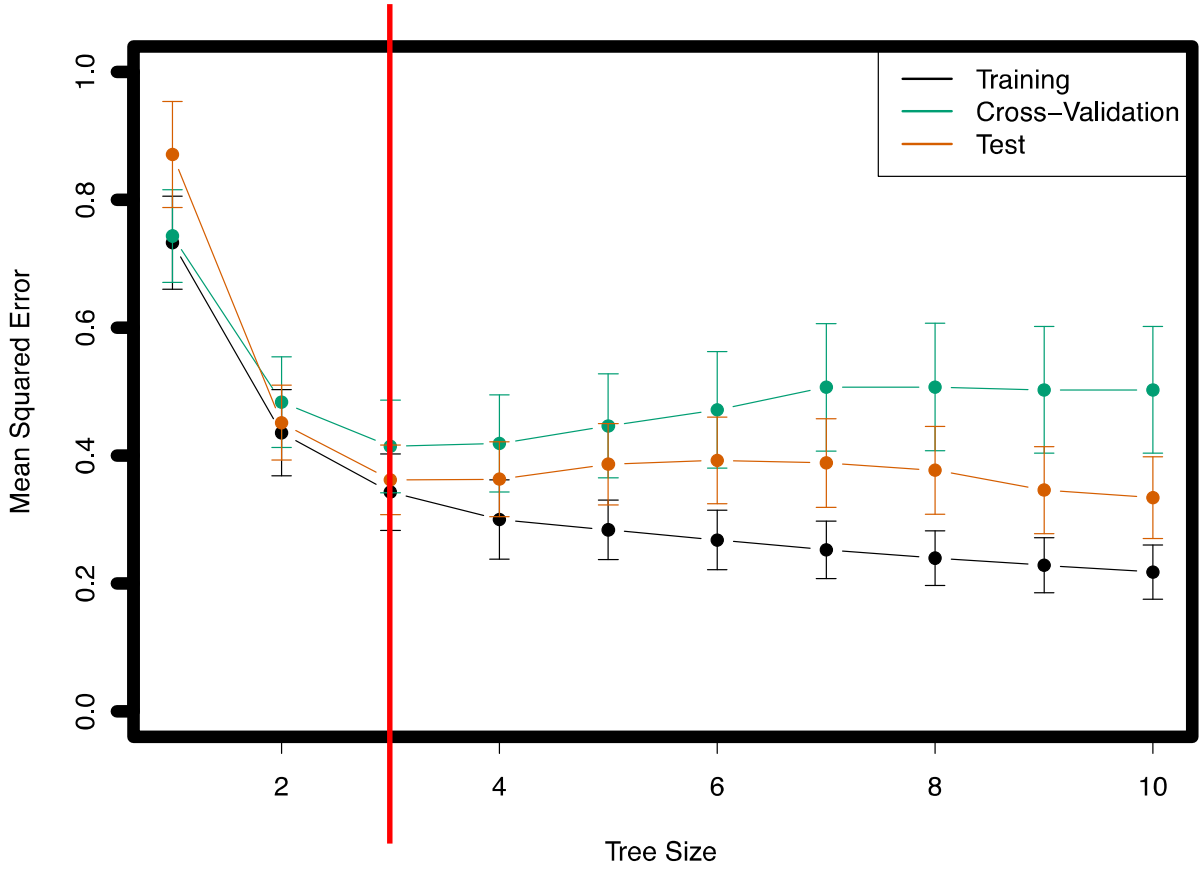
Tree Pruning (Cont'd)

```
##visualize results
plot(cv.carseats$size,cv.carseats$dev,type="b")
```

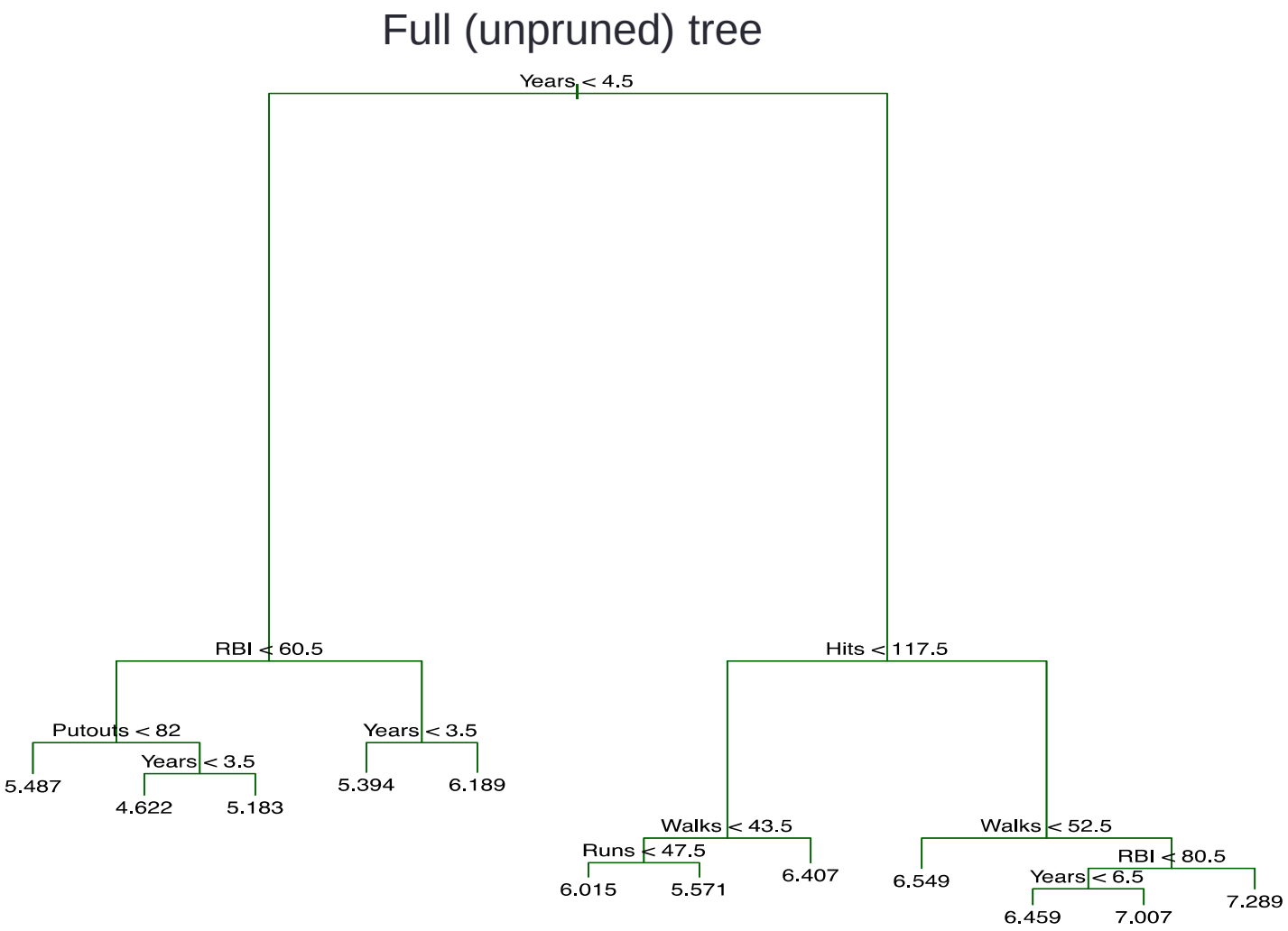


Example: Baseball Players' Salaries

The minimum cross validation error occurs at a tree size of 3



Example: Baseball Players' Salaries



Example: Baseball Players' Salaries

- Cross validation indicated that the minimum RSS is achieved when the tree size is three (i.e. there are 3 terminal nodes).

Improving Trees

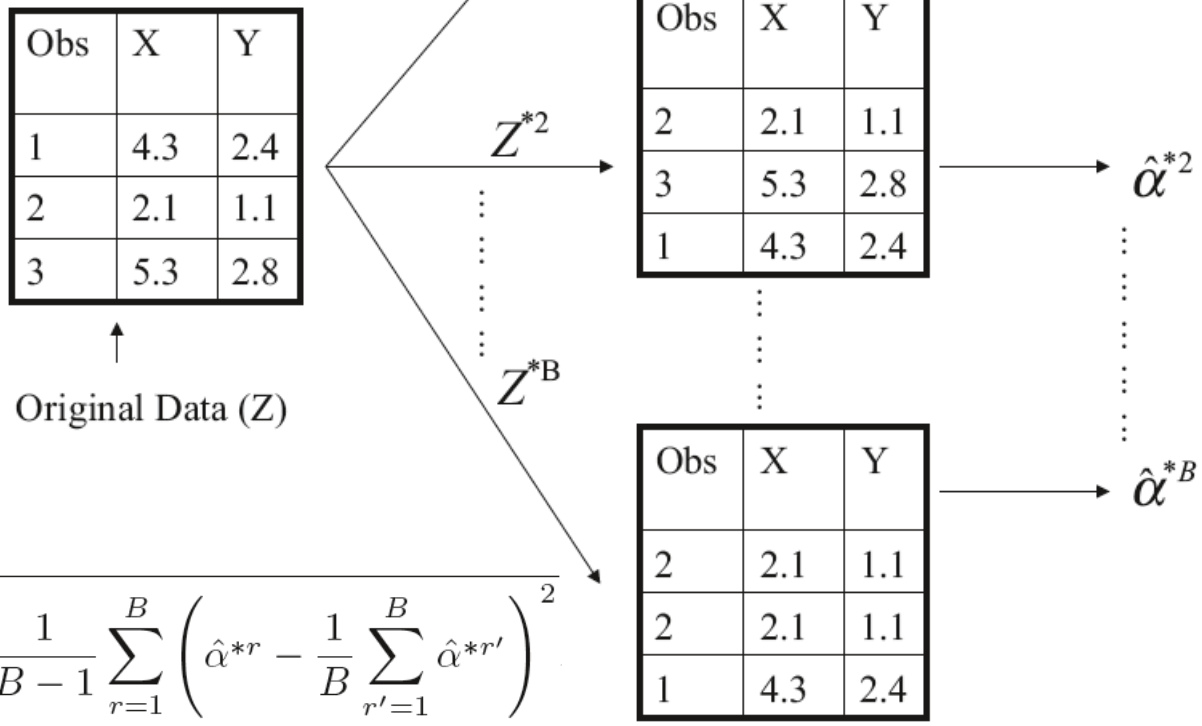
Motivation

- 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 (and thus high prediction accuracy).
- Three methods to improve trees: *Bagging*, *Random forests*, *Boosting*

Recall: Idea of The Bootstrap

- The bootstrap approach uses a computer to emulate the process of obtaining new sample sets, so that we can estimate the variability of $\hat{\alpha}$ without generating additional samples.
- Rather than repeatedly obtaining independent data sets from the population, we instead obtain distinct data sets by repeatedly sampling observations from the original data set.

Illustration on A Dataset with $n = 3$



$$\mathrm{SE}_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1}\sum_{r=1}^{B}\left(\hat{\alpha}^{*r} - \frac{1}{B}\sum_{r'=1}^{B}\hat{\alpha}^{*r'}\right)^2}$$

1. Bagging

- Bagging: **b**ootstrap **agg**regat**ing**.
- Bagging is an extremely powerful idea based on two things:
  - Bootstrapping: plenty of training datasets!
  - Averaging: reduces variance!
- Why does averaging reduces variance?
  - Averaging a set of observations reduces variance. 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.

Bagging for Regression Trees

- Generate B different bootstrapped training datasets
- Construct B regression trees using the training datasets
- Average the resulting predictions

Note: These trees are not pruned, so each individual tree has low bias but high variance. Averaging these trees reduces variance, and thus we end up lowering both variance and bias.

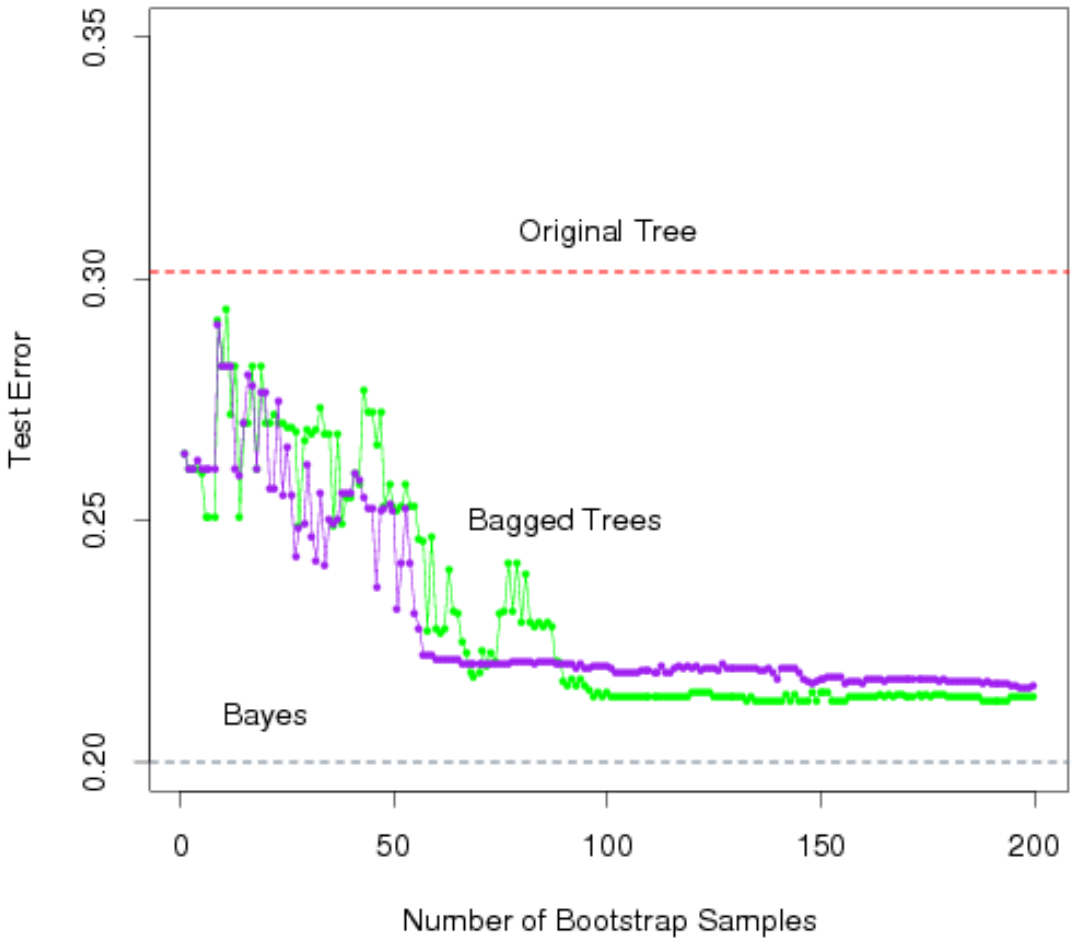
Bagging for Classification Trees

- Generate B different bootstrapped training datasets
- Construct B classification trees using the training datasets
- For prediction, there are two approaches:
  1. Record the class that each bootstrapped data set predicts and provide an overall prediction to the most commonly occurring one (majority vote).
  2. If our classifier produces probability estimates we can just average the probabilities and then predict to the class with the highest probability

Both methods work well.

A Comparison of Error Rates

- 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).



Advantage/Disadvantage of Bagging

- Bagging typically improves the accuracy over prediction using a single tree, but it is now *hard to interpret* the 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, and it is no longer clear which variables are most important to the procedure.
- Thus bagging improves prediction accuracy at the expense of interpretability.

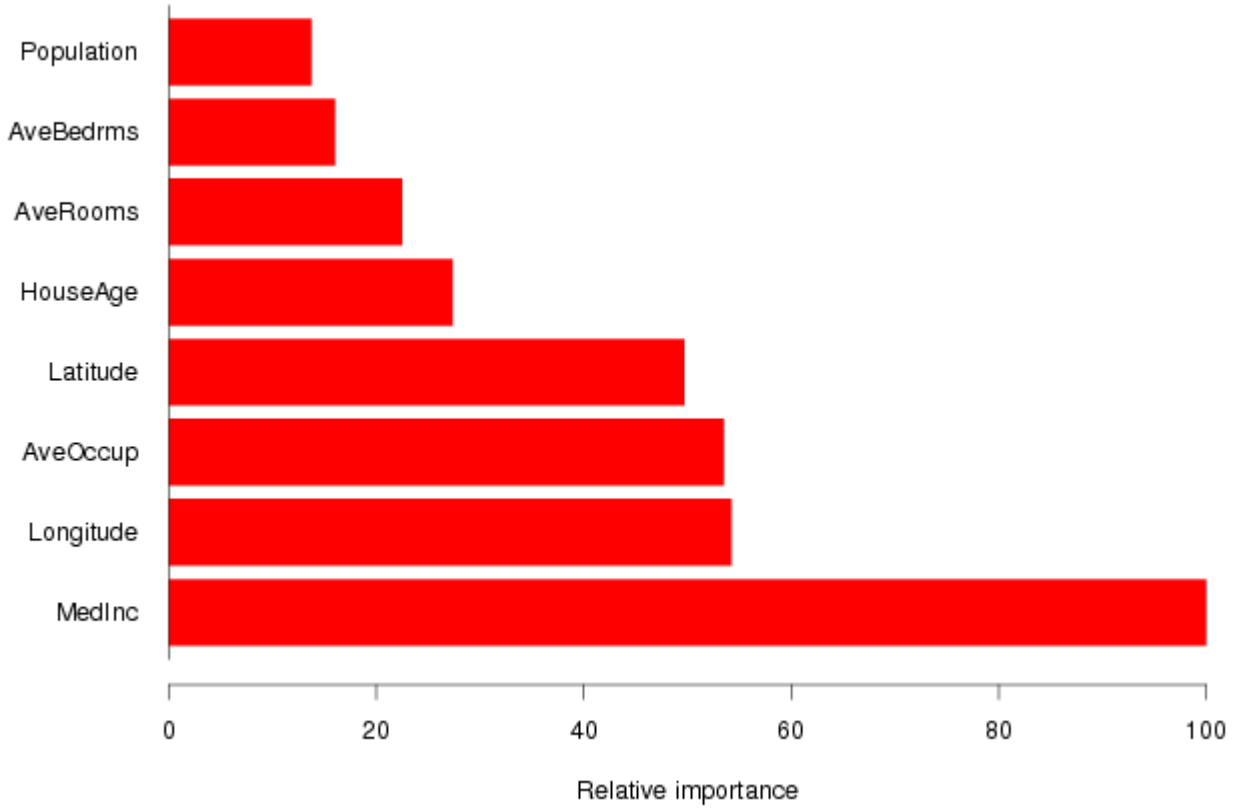
Variable Importance Measures

- Although the collection of bagged trees is much more difficult to interpret than a single tree, we can obtain an overall summary of the importance of each predictor.
- **How**: use the amount of RSS (for bagging regression trees) or the Gini index (for bagging classification trees) decreased due to splits over the predictor.
- A large value indicates an important predictor.

```
bag.boston$importance
```

Example: Boston Data

- Median income is the most important variable.
- Longitude, Latitude and Average occupancy are the next most important.



2. Random Forests

- It is a very efficient statistical learning method.
- It builds on the idea of bagging, but it provides an improvement because it de-correlates the trees.
- How does it work?
  - Build a number of decision trees on bootstrapped training sample,
  - For each tree, each time a split in a tree is considered, **a random sample of m predictors** is chosen as split candidates from among the p predictors.

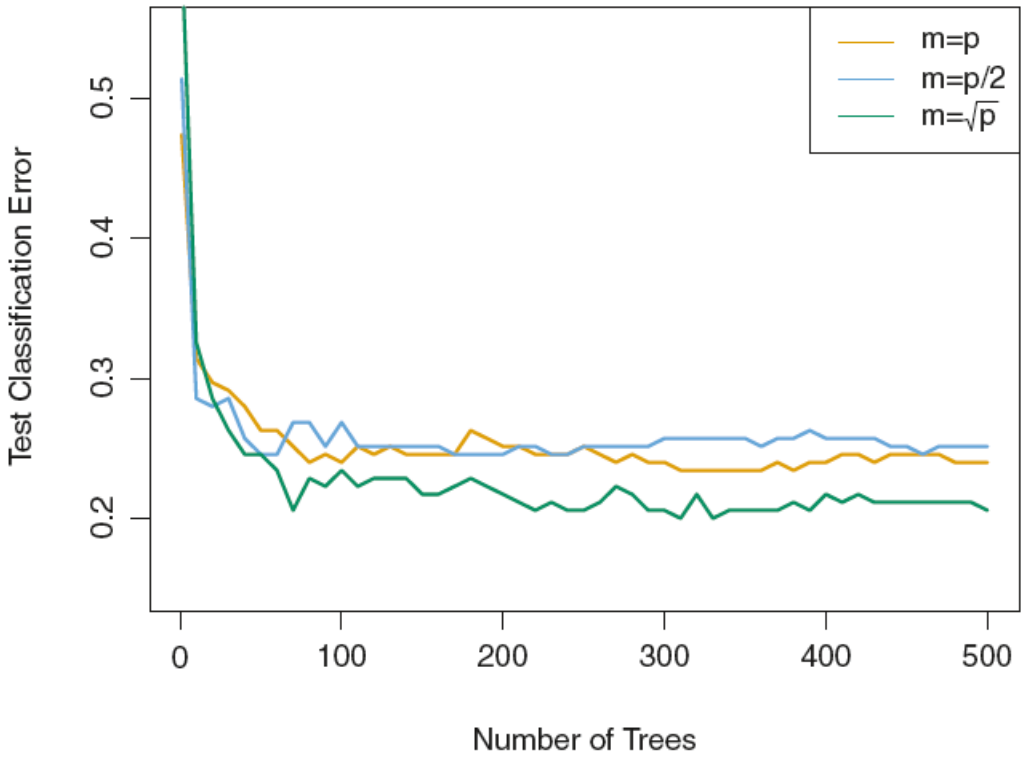
  (Usually $m = \sqrt{p}$ is used for classification)

Why *m* Predictors in Splitting?

- Suppose that we have a very strong predictor in the data set along with a number of other moderately strong predictor, then in the collection of bagged trees, most or all of them will use the very strong predictor for the first split!
- All bagged trees will look similar. Hence all the predictions from the bagged trees will be highly correlated.
- Averaging many highly correlated quantities does not lead to a large variance reduction, and thus random forests "de-correlates" the bagged trees leading to more reduction in variance.

Random Forest with different values of “m”

- Notice when random forests are built using $m = p$, then this amounts simply to bagging.



3. Boosting

- Bagging grow trees simultaneously:
  1. creating multiple copies of the original training data set using the bootstrap
  2. fitting a separate decision tree to each copy
  3. combining all the trees to generate a single predictive model

- Boosting grows trees sequentially: each tree is grown using information from previously grown trees.

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

Algorithm of Boosting for Regression Trees

1. Initially (*b=1),* Build a “small tree” with small number of splits *d<5*
2. If the residuals (training errors) are large
   (1) Scale the residuals by a shrinkage parameter (to control learning rate)
   (2) Fit another small tree to predict residuals
3. Repeat step 2 until the specified number of steps B is reached or the error reaches desired level
4. Output the boosted model

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

Idea behind Boosting

- Fitting a single large tree may cause overfitting.
- Boosting *learns slowly*: in each step, fit a very small tree to the residuals, so that improve $\hat{f}$ in areas where it does not perform well.
- In general, statistical learning approaches that learn slowly tend to perform well.