8.1 Decision Tree Basics

Tree-Based Methods

- In this section, we describe *tree-based* methods for regression and classification.
- These involve stratifying or segmenting the predictor space into a number of simple regions.
- Since the set of splitting rules used to segment the predictor space can be summarized in a tree, these approaches are called *decision trees*.

Pros and Cons

- Pros: decision trees are simple and useful for interpretation.
- Cons: they are not typically competitive with other supervised learning approaches.
- However, other methods that utilize multiple trees to generate predictions are often much more accurate (if less interpretable):
  - Bagging
  - Random forests
  - Boosting
- We will talk about these methods in Section 8.2

The Basics of Decision Trees

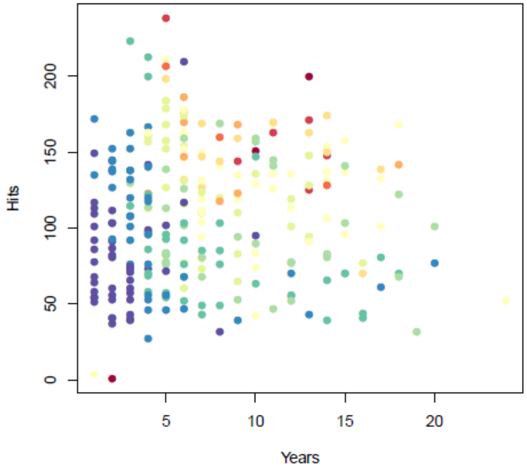
- Decision trees can be applied to both regression and classification problems.
- We start with regression, then move on to classification.

8.1.1 Regression Trees

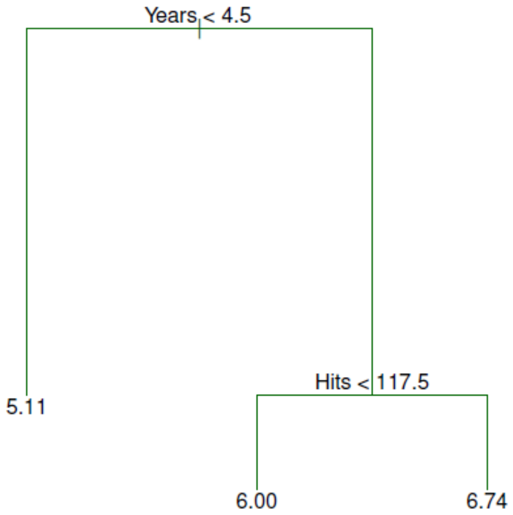
We will start with a simple example.

Baseball Salary Data

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



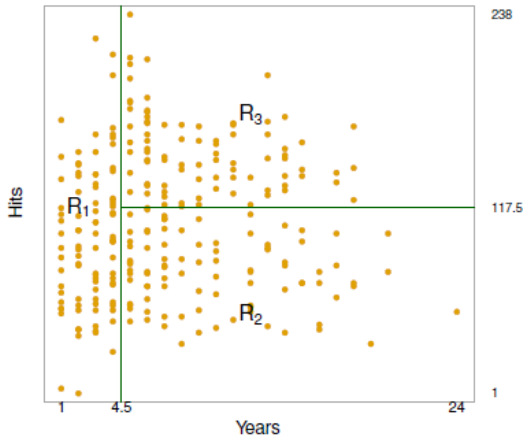
An initial decision tree



Tree details

- `Hitters` data used to predict log salary based on number of years in major leagues and number of hits made in the previous year.
- The label $X_j < t_k$ indicates the left-hand brank emanating from that split
    - For the first split, left indicates `Years`< 4.5
    - Right indicates `Years`≥ 4.5
- The tree has two internal nodes (the two splits) and three terminal nodes, or leaves.
    - The number for each leaf is the mean of the observations in that region.

Results



The tree stratifies the players into three regions: $R_1 = \{X | \text{Years} < 45\}$; $R_1 = \{X | \text{Years} \geq 45, \text{Hits} < 117.5\}$; and $R_3 = \{X | \text{Years} \geq 45, \text{Hits} \geq 117.5\}$

Tree Terminology

- The regions R_1, R_2, and R_3 are known as *terminal nodes*.
- As far as the tree analogy goes, trees are typically drawn upside down (the leaves are at the bottom).
- The splits are referred to as *internal nodes*

Interpretation of Results

- `Years` is the most important factor in determining `Salary`: players with less experience earn lower salaries than more experienced players.
- Given that a player is less experienced, the number of `Hits` doesn't seem to matter much in his `Salary`.
- For more experienced players, number of `Hits` does seem to matter, with players who make more `Hits` generally having higher salaries.
- Obviously a gross oversimplification, but we do have ways to make trees better!

The Tree-Building Process

1. Divide the predictor space X into J distinct and non-overlappng regions $R_1, R_2, \ldots, R_J$
2. For every observation that falls into region R_j, predict the mean of the response values (from the training data) in R_j

The Tree-Building Process: Generating the Regions

- In theory, the regions could take any shape, but we choose to divide the predictor space into (high-dimensional) boxes.
- The goal is to find boxes that minimize RSS:

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

where $\hat{y}_{R_j}$ is the mean response for the training observations in the jth box.

The Tree-Building Process

- It's not feasible to consider every possible partition of the feature space into J boxes.
- Instead, we take a *town-down, greedy* approach known as recursive binary splitting.
- *Top down* because it starts at the top of the tree and successively splits the predictor space.
- *Greedy* because it makes the best split at that particular step, rather than looking ahead and picking a split that will lead to a better tree overall.

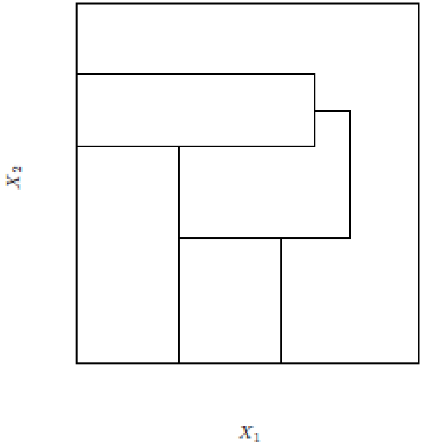
The Tree-Building Process

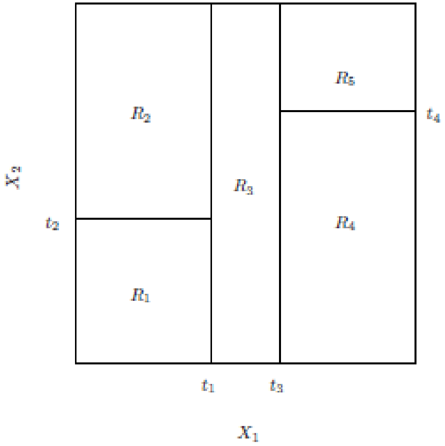
1. Select X_j and the cutpoint s such that splitting the predictor space into regions $\{X|X_j < s\}$ and $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ leads to the greatest possible reduction in RSS.
2. Repeat the process, looking for the best predictor and cutpoint in order to split the data further and minimize the RSS within each of the resulting regions.
   - Instead of splitting the entire predictor space, we split one of the previously identified regions, increasing our number of regions by 1.
3. Continue to repeat Step 2 until some stopping criterion is reached.
   - Ex: until no region contains more than 5 observations.

Predictions

- We predict the response for a given test observation using the mean of the training observations in the region to which the test observation belongs.

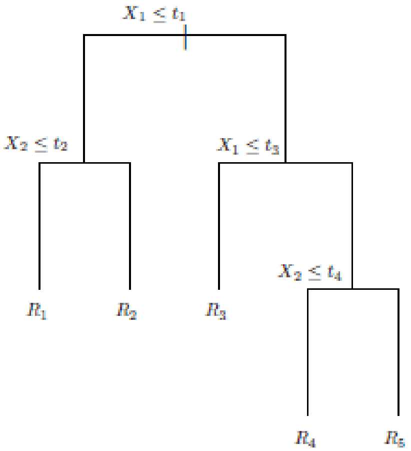
Example

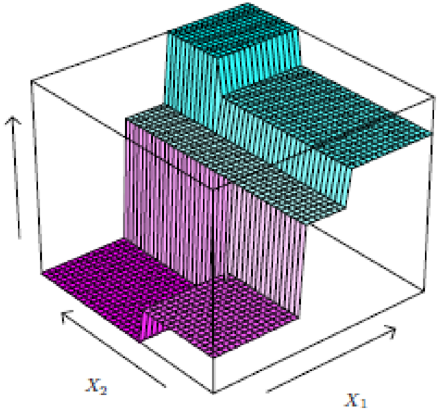




- Left: a partition that could *not* result from recursive binary splitting.
- Right: The output of recursive binary splitting on a two-dimensional example.

Example





- Left: A tree corresponding to the splitting from the previous figure.
- Right: A perspective plot of the prediction surface corresponding to the tree.

Pruning a Tree

The proces described previously may produce good predictions on the training set, but is likely to *overfit* the data, leading to poor test set performance.

Why?

Pruning a Tree

- A smaller tree with fewer splits might lead to lower variance and easier interpretation at the cost of a little bias.
- One possible fix is to use the stopping criteria that the decrease in RSS at each split must exceed some treshold.
- But this strategy is short-sighted because the algorithm is greedy.
    - A seemingly worthless split might be followed by a very good split.

Pruning a Tree

- A better strategy is to grow a very large tree T_0 and then *prune* is to obtain a *subtree*
- *Cost complexity pruning*, or *weakest link pruning*, is used for this.

Cost Complexity Pruning

- Consider a sequence of tree indexed by some nonnegative tuning parameter α.
- For each value of α, there is a subtree $T \subset T_0$ such that

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

  is as small as possible.
  - $|T|$ is the number of terminal nodes in the tree.
  - R_m is the box corresponding to the mth terminal node.
  - $\hat{y}_{R_m}$ is the mean of the training observations in R_m.
- Notice that this is just another minimization of (loss) + (flexibility penalty)!

Choosing the Best Subtree

- The tuning parameter α controls a trade-off between the subtree's complexity and its fit to the training data.
- we select an optimal value of $\hat{\alpha}$ using cross-validation.
- We then return to the full data set and obtain the subtree corresponding to $\hat{\alpha}$

Summary: Tree Algorithm

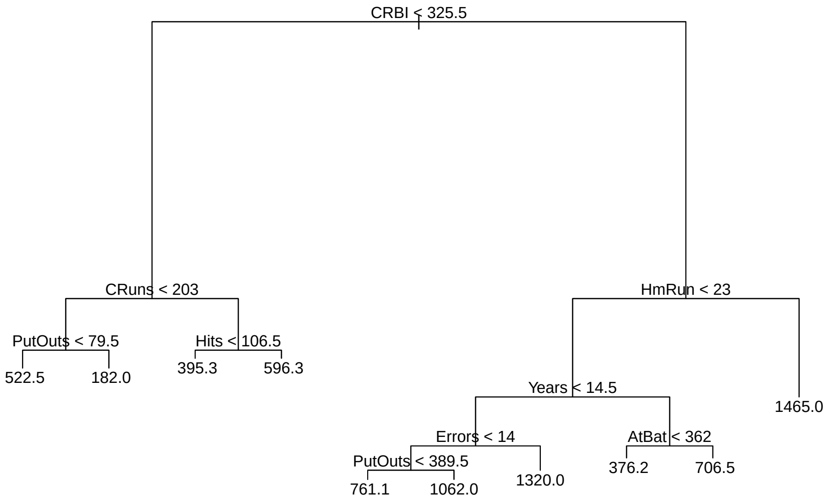
1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold cross-validation to choose α, selecting α to minimize error.
4. Return the subtree from Step 2 that corresponds to the chosen value of α.

More Baseball

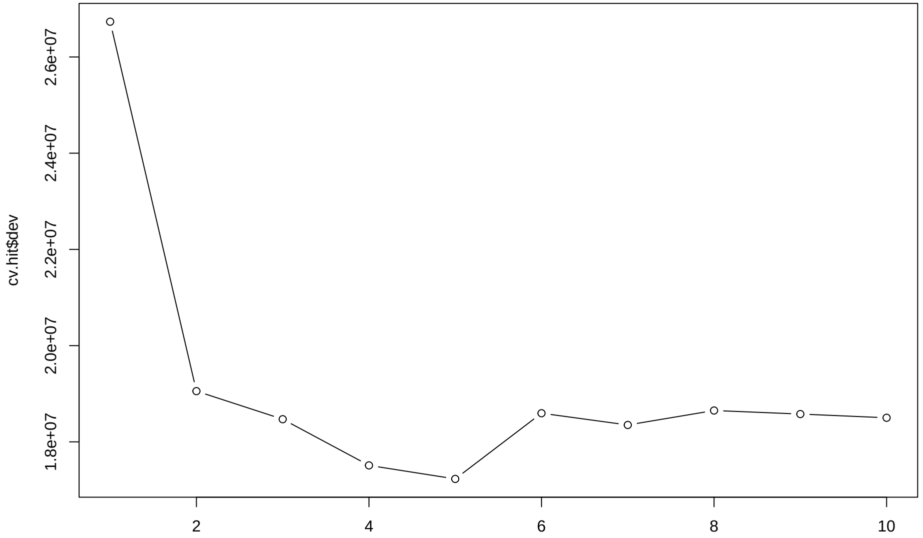
- Now, we create training and test data by randomly divide the `Hitters` data in half.
- Then we build a large regression tree on the training data and varied α to create subtrees with different numbers of terminal nodes.
- Finally, we perform 6-fold cross-validation to estimate cross-validated MSE of the trees as a function of α.

```r
set.seed(0)
ind <- sample(nrow(Hitters), nrow(Hitters)/2)
hit.trn <- Hitters[ind,]
hit.tst <- Hitters[-ind,]

library(tree)
hit.tree <- tree(Salary ~ ., Hitters, subset=ind)
plot(hit.tree); text(hit.tree, pretty = 0)

cv.hit <- cv.tree(hit.tree, K=6)
plot(cv.hit$size, cv.hit$dev, type="b")

prune.hit <- prune.tree(hit.tree, best = 5)
plot(prune.hit); text(prune.hit, pretty = 0)
```

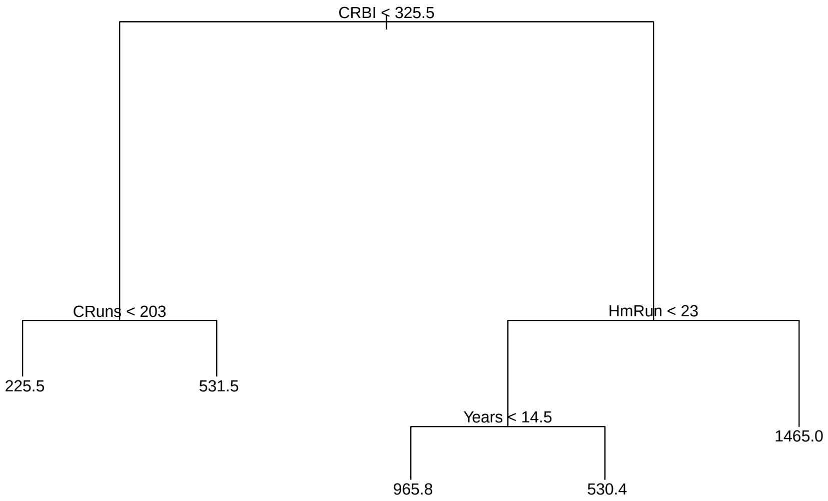
The Unpruned Tree



Cross-Validation Results



The Pruned Tree



8.1.2 Classification Trees

- These are very similar to regression trees, just with a qualitative response.
- In this case, we predict that each observation belongs to the *most commonly occurring class* of training observations in the region to which it belongs.

Details of Classification Trees

- We again use recursive binary splitting to grow the tree.
- However, in this setting RSS doesn't make sense.
- We might think to use the *classification error rate*, the proportion of training observations in that region which do not belong to the most common class

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

  - $\hat{p}_{mk}$ is the proportion of training observations in the mth region from the kth class.
  - However, classification error is not sensitive enough for tree-growing, so we need other approaches.
  - We will consider two other measures.

Gini Index

The *Gini index* is defined by

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

- This is a measure of the total variance across all K classes.
- Takes on a small value if all of the $\hat{p}_{mk}$ are close to 0 or 1.
- The Gini index is referred to as a measure of *node purity*
  - Small values suggest nodes contain mostly observations from a single class.

Entropy

Entropy is given by

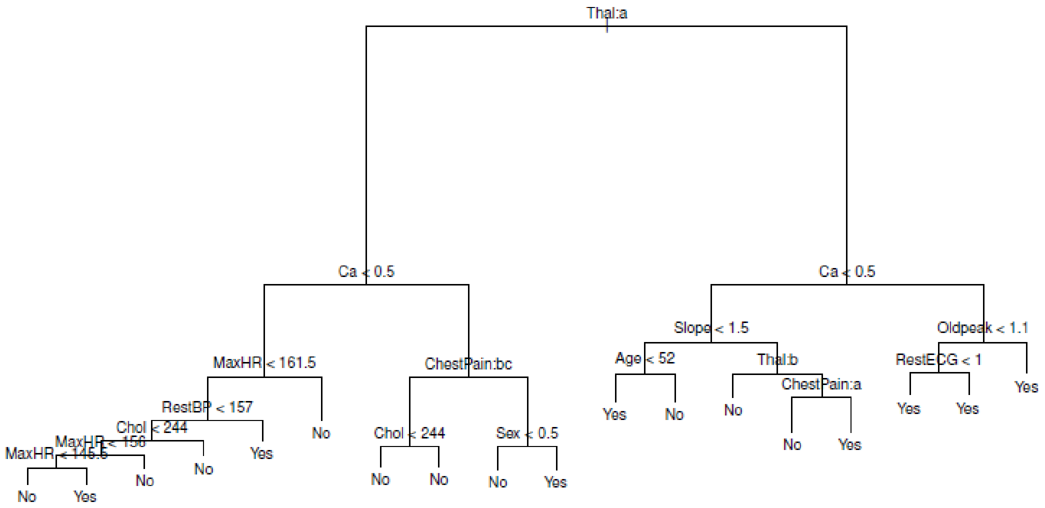
$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

- It turns out the Gini index and entropy are very similar numerically.
- We will use one of these two measures in place of RSS when building classification trees.
- We may use either of these, or the classification error rate, when pruning trees.
  - However, we often prefer classification error rate for pruning.

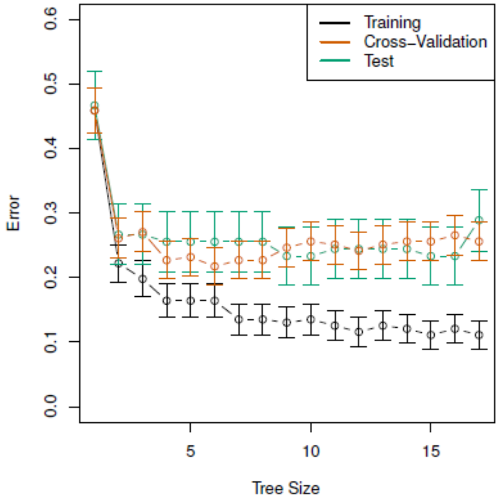
Example: `Heart` Data

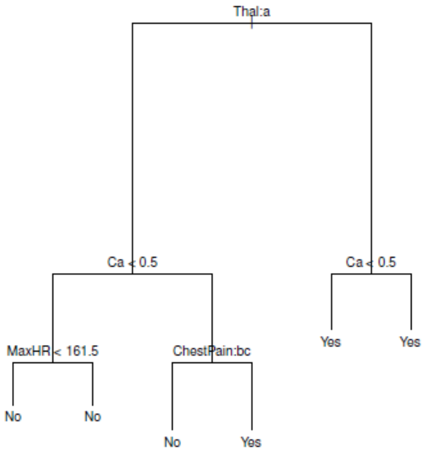
- Binary outcome `HD` with 303 patients who presented with chest pain.
    - Outcome of `yes` indicates heart disease; `no` indicates absence of heart disease
- There are 13 predictors, including age, sex, cholesterol, and more
- Cross-validation yields a tree with six terminal nodes

Example: Heart Data (big tree)



Example: Heart Data (C-V tree)





8.1.3 Trees vs. Linear Models

Linear regression assumes a model of the form

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

Regression trees assume a model of the form

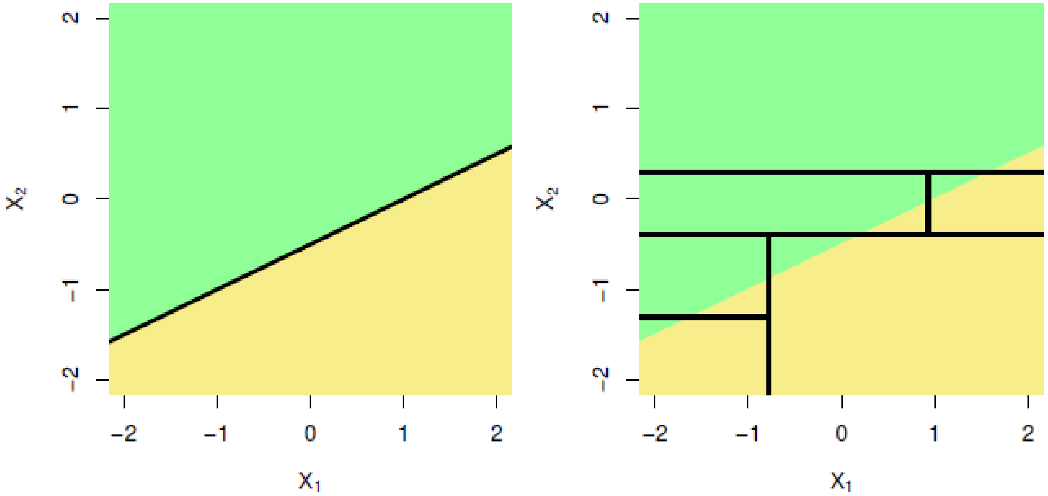
$$f(X) = \sum_{m=1}^{M} c_m I(X \in R_m)$$

Trees vs. Linear Models

Which model is better?

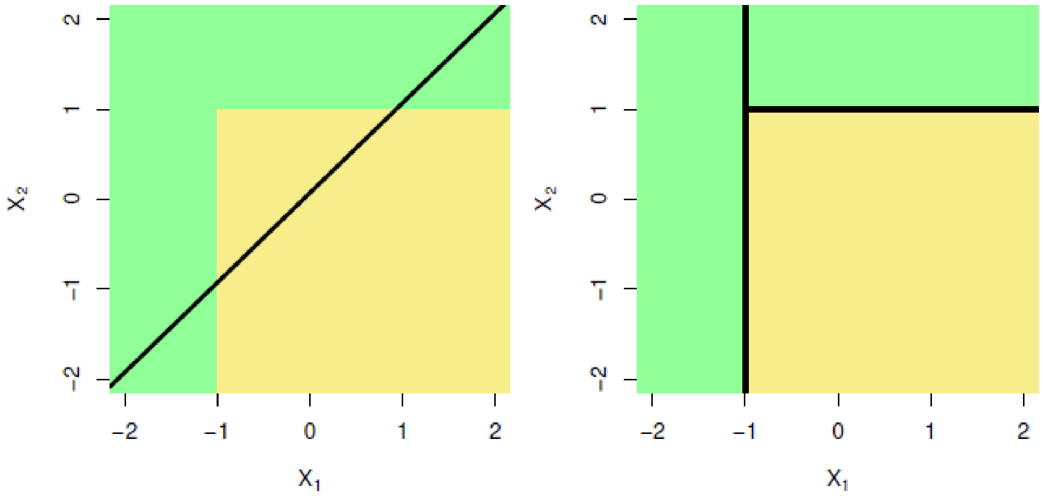
- If the relationship is well-approximated by a linear model, naturally linear regression will work well.
- If instead the relationship is highly nonlinear/complex, decision trees may outperform classical approaches.

Trees vs. Linear Models



- True decision boundary is linear.

Trees vs. Linear Models



- True decision boundary is nonlinear.

Advantages of Trees

- Very easy to explain.
- Very easy to display and displays are easy to interpret, even by non-experts.
- Handle qualitative predictors directly (without needing to create dummy variables).
- Some people believe decision trees more closely mirror human decision-making than do the regression and classification approaches seen previously.
  - Side note: I guess this is good because it means they're intuitive?

Disadvantages of Trees

- Generally lack the predictive accuracy as many of the other learning algorithms seen in the text.
- Trees can be very non-robust
  - That is, a small change in the data can cause a large change in the final tree.