# DSC5103 Statistics

Session 7. Tree-based Methods

# Last time

- Subset Selection
  - Subset Selection and Stepwise Selection revisited
  - Choosing the optimal model using Cross-Validation

- Shrinkage Methods (Regularization)
  - Ridge Regression
  - The Lasso
  - Elastic Net

# Linear Model Selection unified

- Best subset selection

$$\min RSS; \text{subject to } ||\beta||_0 \leq A$$

- LASSO

$$\min RSS; \text{subject to } ||\beta||_1 \leq A$$

- Ridge Regression

$$\min RSS; \text{subject to } ||\beta||_2 \leq A$$

# Plan for today

- The Basics of Decision Trees
  - Regression Trees
  - Building and Pruning Trees
  - Classification Trees

  - Trees vs. Linear Models
  - Advantages and Disadvantages of Trees

# Tree-based Methods

- The idea: to segment/partition the predictor (X) space into a number of simple regions and predict Y based on the regions

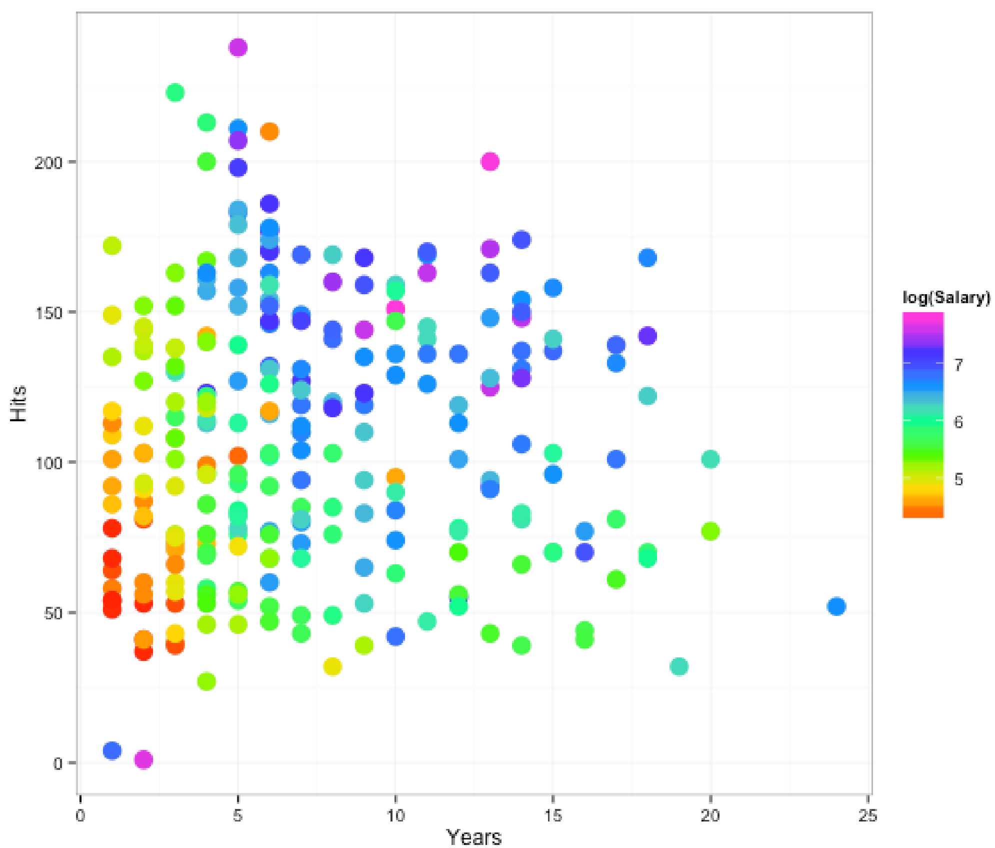
- The set of splitting rules used to segment the predictor space can be summarized in a tree, so these types of approaches are known as **decision-tree methods**
  – Regression trees
  – Classification trees

# Example: Hitter’s Salary

$X_1$ $X_2$ $y$

- Segment on Years and Hits based on log(Salary)

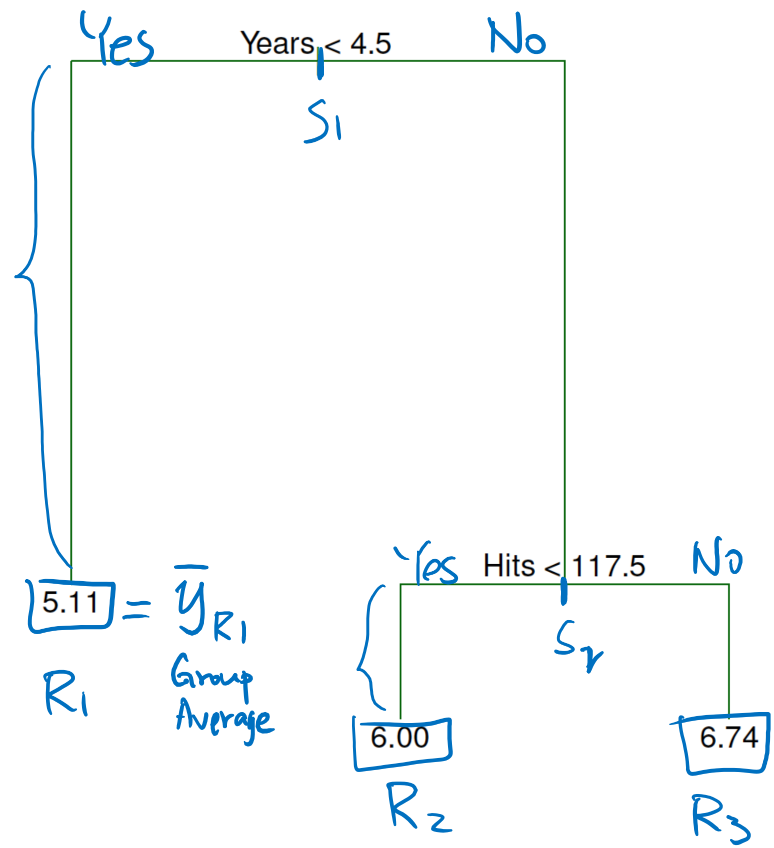
$X_2$




$X_1$

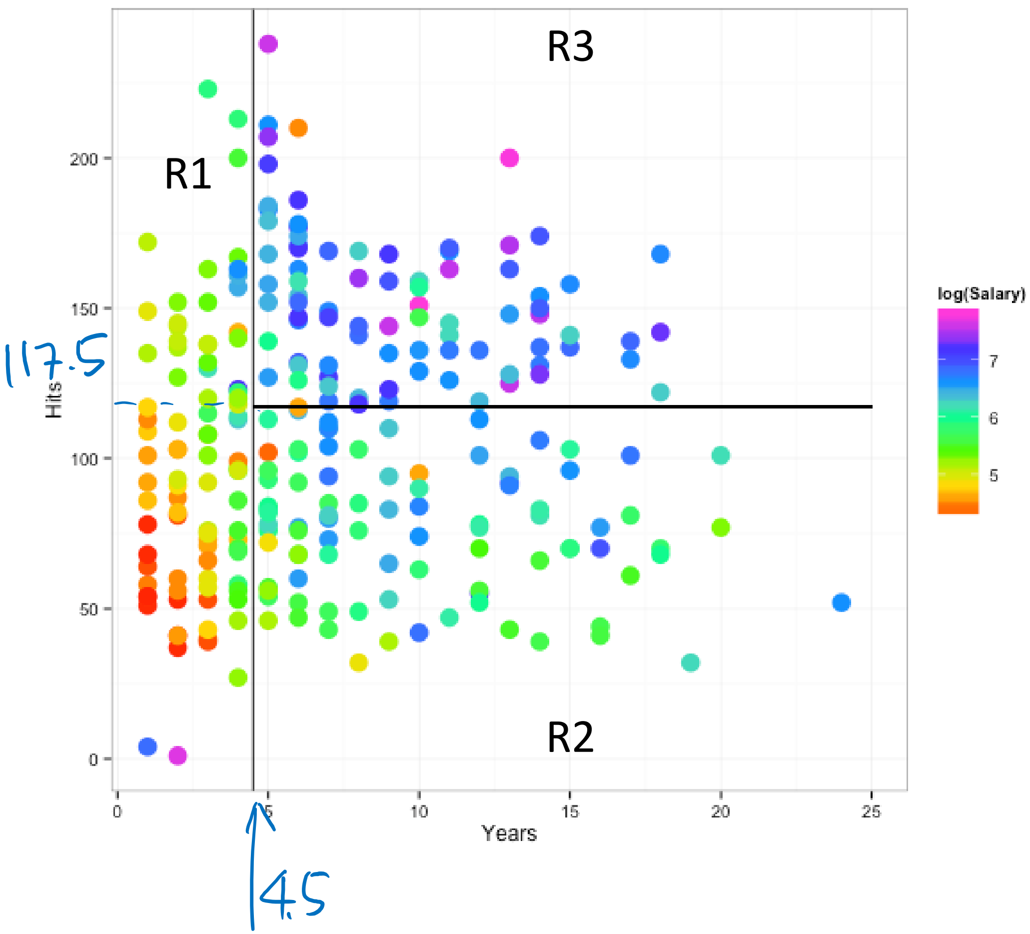
# Hitter's Salary --- fit the decision tree

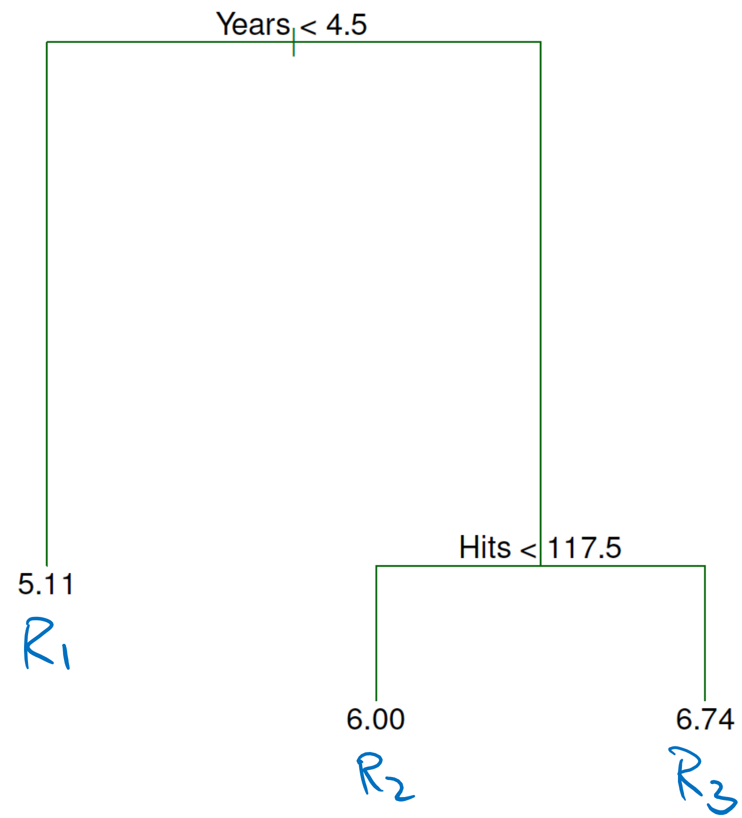
- A regression tree: log(salary) ~ Years + Hits.
- The tree has two internal nodes and three terminal nodes (leaves).
- At a given internal node, the label indicates the splitting rule.
  - For instance, the split at the top of the tree results in two large branches. The left-hand branch corresponds to Years < 4.5, and the right-hand branch corresponds to Years ≥ 4.5.
- The number in each leaf is the mean of the response for the observations that fall there.



# Hitter's Salary --- visualize the tree

- The tree segments the data into three regions of X space.



# Hitter's Salary --- interpret the tree

- *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*.

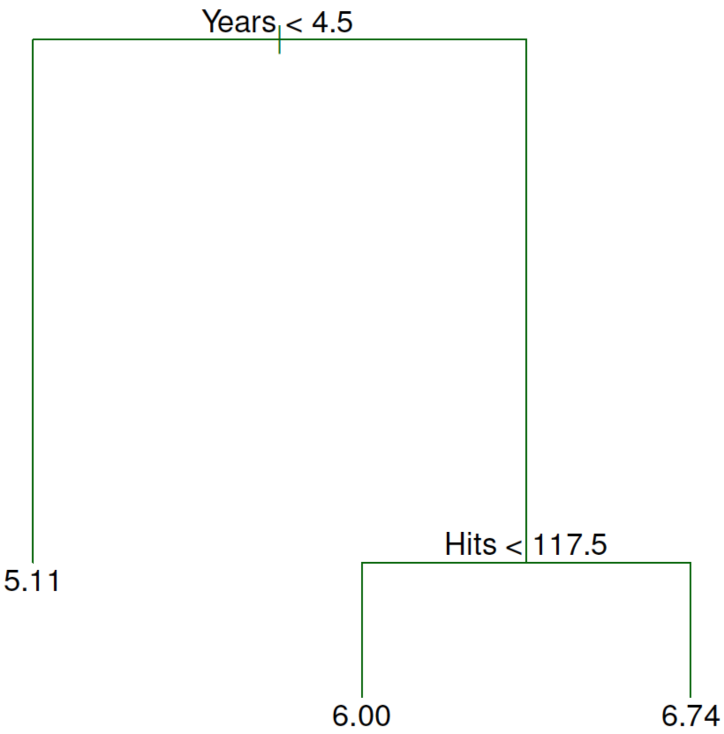
- But among players who have been in the major leagues for five or more years, the number of *Hits* made in the previous year does affect *Salary*, and players who made more *Hits* last year tend to have higher salaries.

- Surely an over-simplification, but compared to a regression model, it is easy to display, interpret and explain.

# Hitter's Salary --- predict with the tree

- The tree divides the predictor space into J distinct and non-overlapping regions, $R_1$, $R_2$, …, $R_J$.

- For every new 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 $R_j$.



# The Tree Building Process

- The goal: find regions $R_1, R_2, \ldots, R_J$ to minimize RSS:

$$\min_{R_1, R_2, \ldots, R_J} \underbrace{\sum_{j=1}^{J} \underbrace{\sum_{i \in R_j} \underbrace{\left(y_i - \underbrace{\hat{y}_{R_j}}_{\text{Group Average in } R_j}\right)^2}_{\text{squared error of } i}}_{\text{total squared error in } R_j}}_{\text{total squared error}}$$

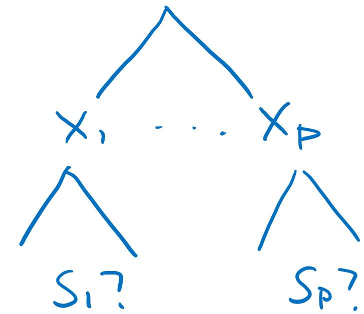
- In theory, the regions could have any shape. However, we choose to divide the predictor space into high-dimensional rectangles, or boxes, for simplicity and for ease of interpretation of the resulting predictive model.

# The Tree Building Process

- How to find the optimal partition $R_1, R_2, \ldots, R_J$ ?
  - Computationally infeasible to optimize

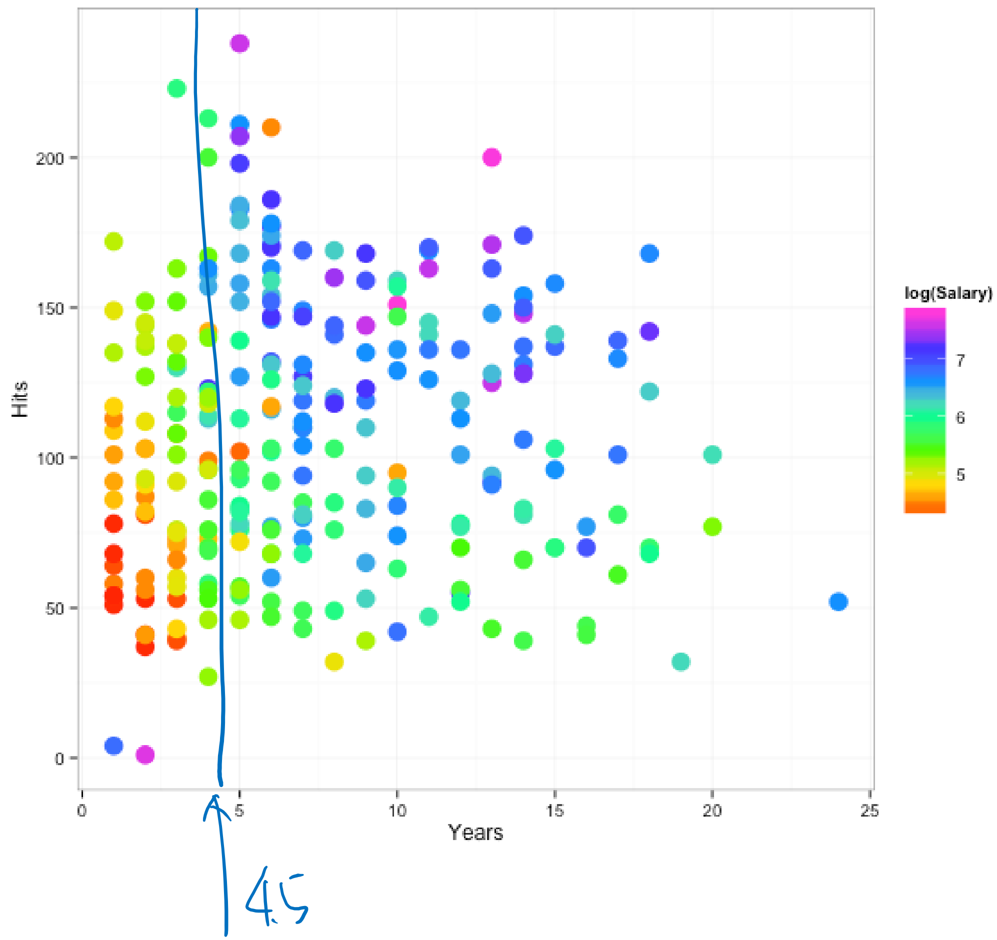
- CART (Classification And Regression Tree): a top-down greedy approach ("forward stepwise")
  - it begins at the top of the tree
  - successively splits the one of the regions along one of the X's each time
  - each split is indicated via two new branches further down on the tree

# Where to Split?

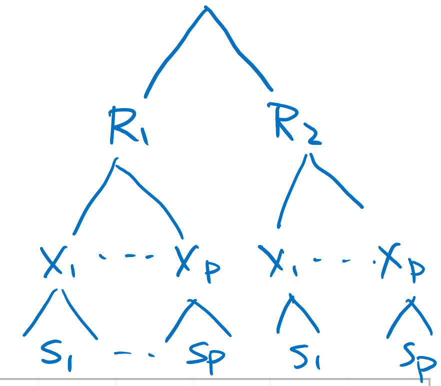



- Find one predictor $X_j$ and a cut point $\boldsymbol{s}$ among all possible values of $\boldsymbol{j}$ and $\boldsymbol{s}$ to minimize the RSS.

- The first split is at Years < 4.5
  - Left branch: Years < 4.5
  - Right branch: Years ≥ 4.5

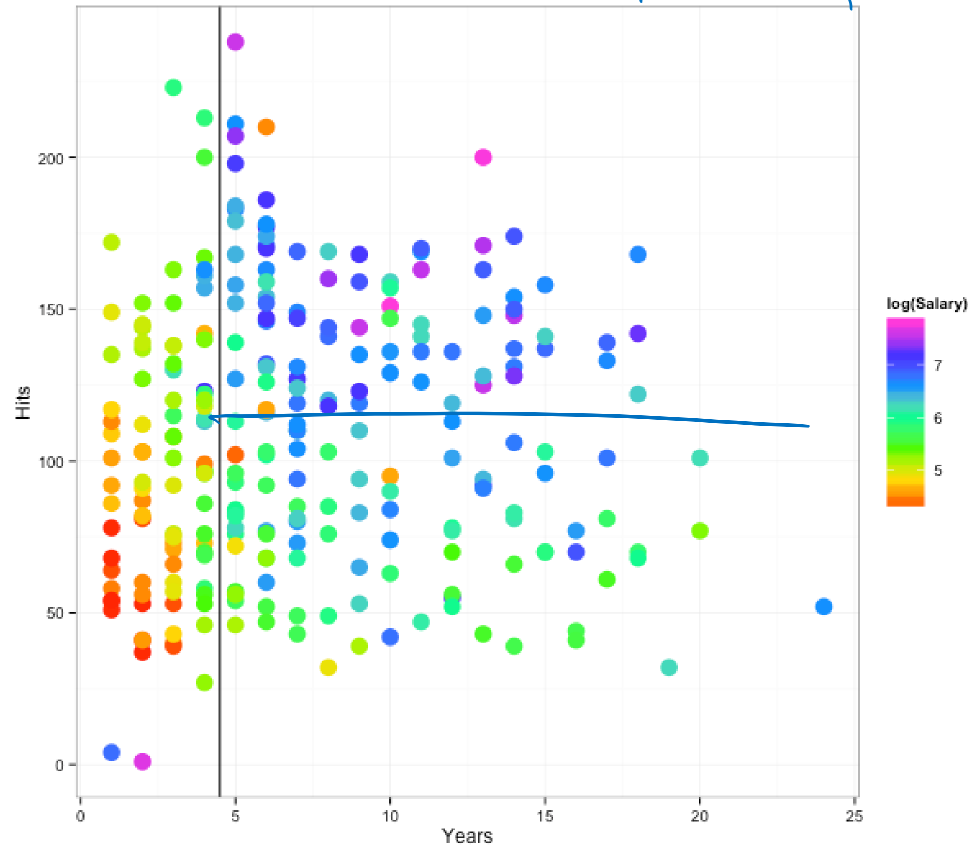


# Where to Split?



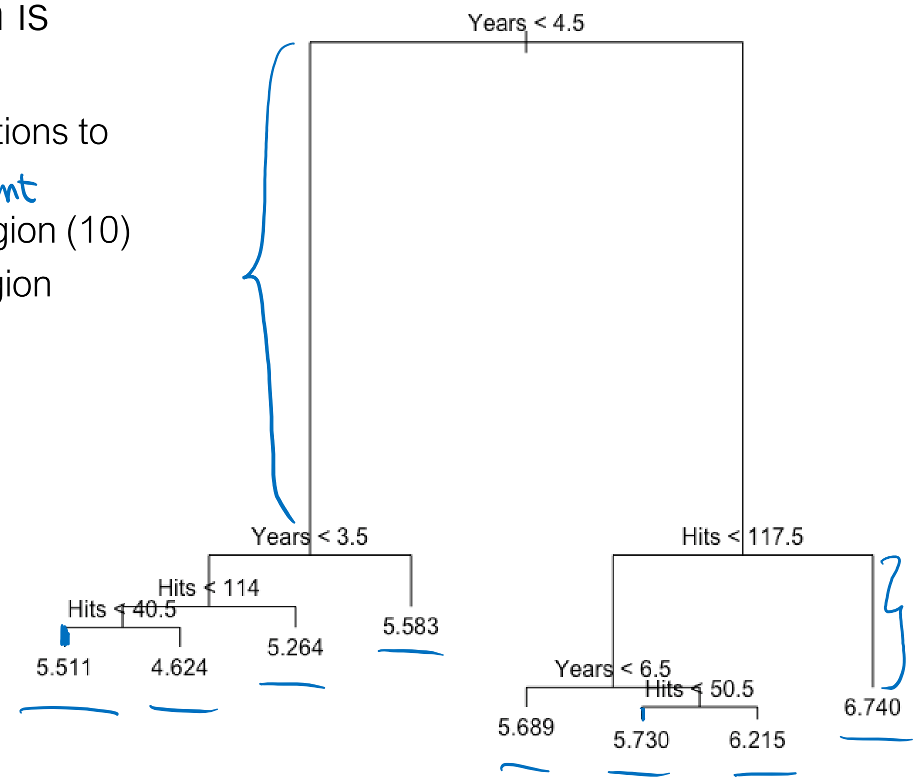

- Repeat the process, looking for the best predictor and best cut point in order to split the data further so as to minimize the total RSS across all regions

- Instead of splitting the entire predictor space, we split one of the previously identified regions

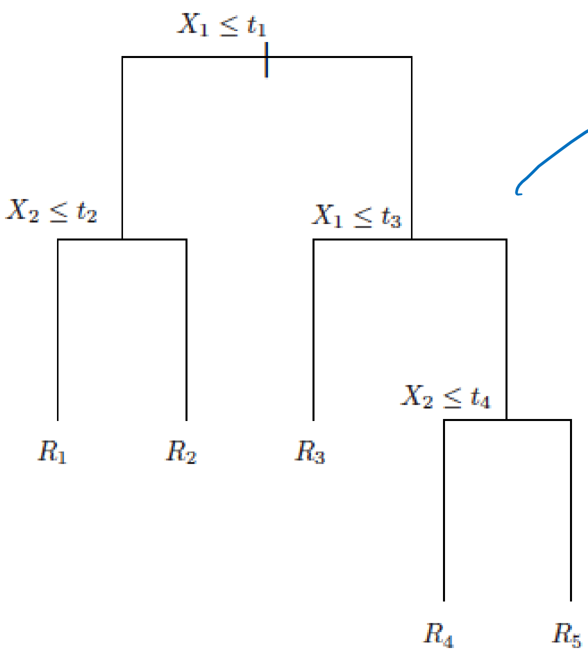


# Where to Split?

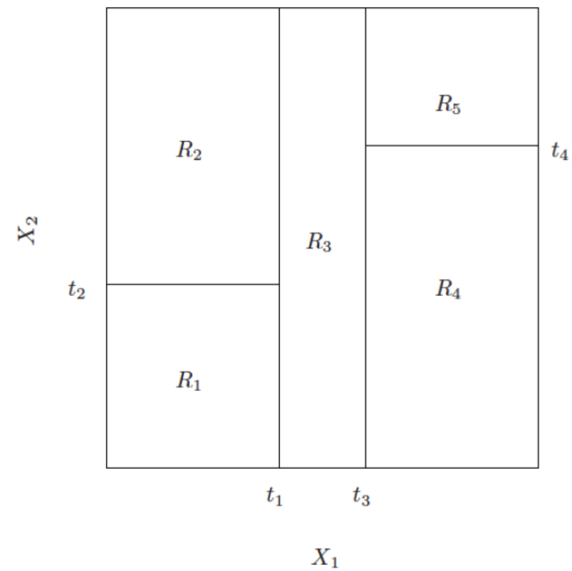
- Continues until a stopping criterion is reached
  - mincut: the minimum number of observations to include in either child node (5) parent
  - minsize: too few data points in a node/region (10)
  - mindev: too small differences within a region
  - …
- Obtain a tree
  - Internal nodes ⇔ splits
  - Terminal notes ⇔ regions
  - Length of branch ⇔ reduction in RSS



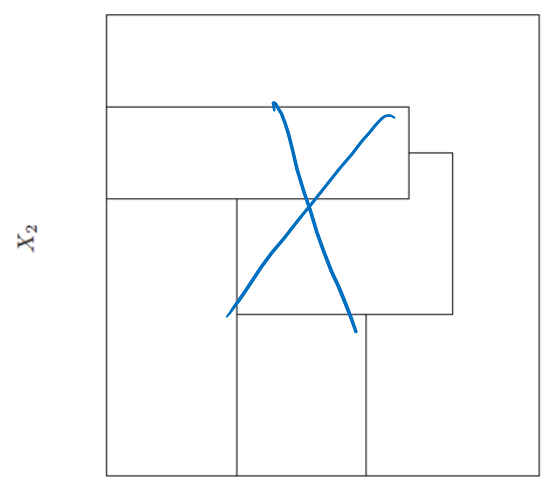
# Tree Output



$X_1 \leq t_1$
$X_2 \leq t_2$
$X_1 \leq t_3$
$X_2 \leq t_4$
$R_1$
$R_2$
$R_3$
$R_4$
$R_5$

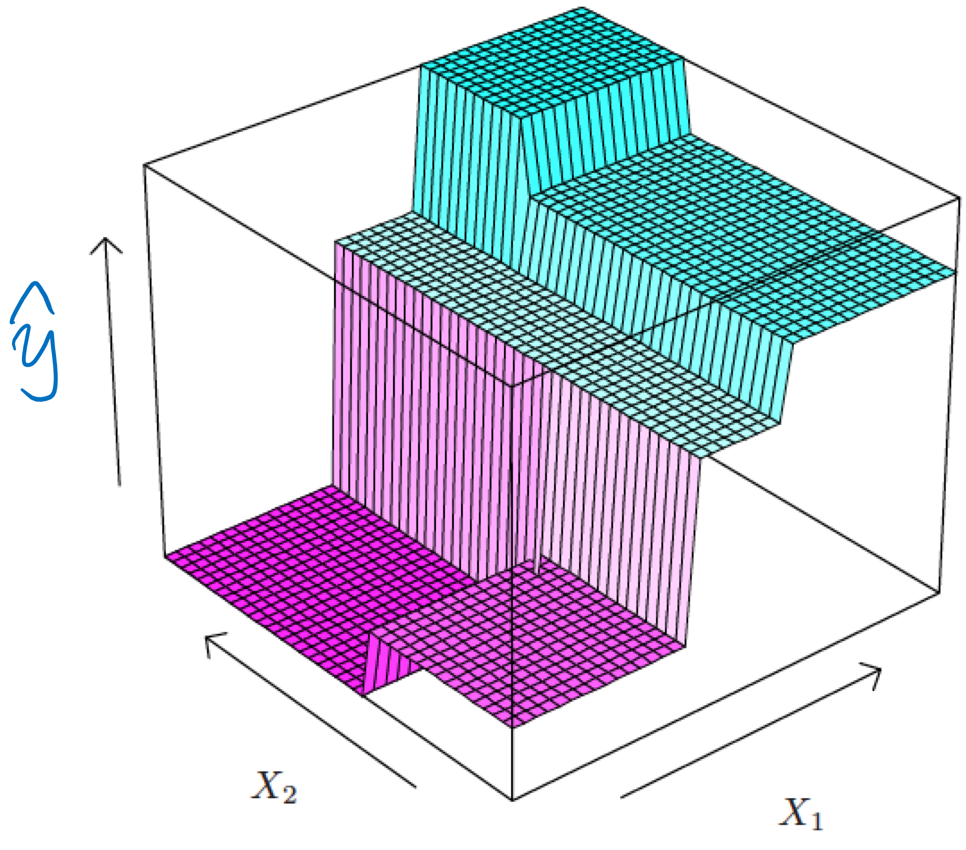



$R_5$
$R_2$
$t_4$
$X_2$
$R_3$
$t_2$
$R_4$
$R_1$
$t_1$
$t_3$
$X_1$

$X_2$

# Tree Prediction



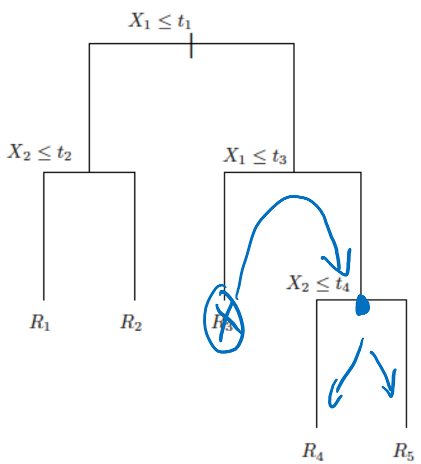
# Tree Pruning

- The tree-building process is forward stepwise expansion
  - As the model complexity (**number of leaves**) increases, it tends to overfit the data

- The tree-pruning strategy is backward shrinkage (subset selection for trees)
  - Start with building a large tree $T_0$ => m regions $R_1, \ldots, R_m$
  - Prune it backward: for each value of α (a tuning parameter), find a subtree T to minimize

$$\sum_{m=1}^{|T|} \sum_{i:\, x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T| \qquad \min \sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} \left(y_i - \hat{y}_{R_m}\right)^2; \text{subject to } |T| \le A$$

    - |T| is the number of leaves/regions
  - Use cross-validation to find the optimal α (or tree size A)



# The Final Tree Algorithm

enough

1. Build a large tree using all the training data (tree() in R)

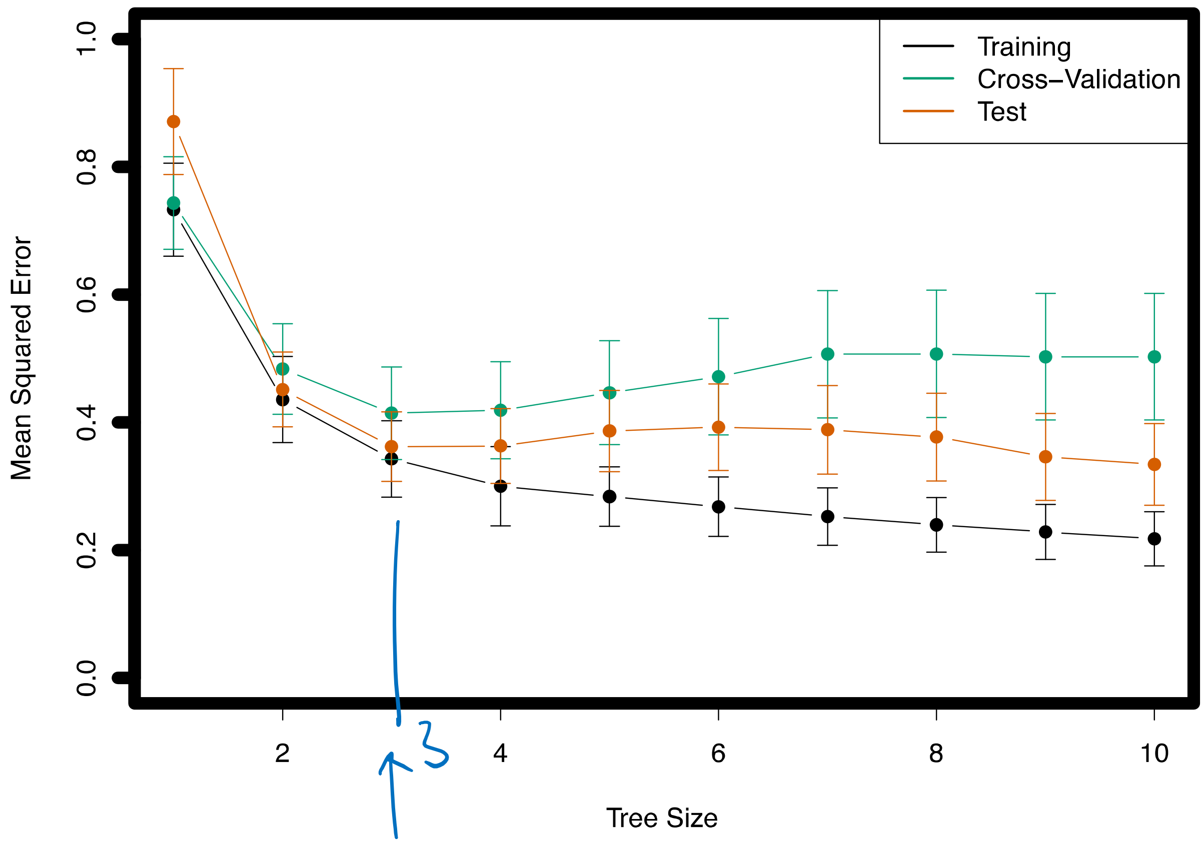
2. Run k-fold cross-validation to choose tree size A (or parameter α) (cv.tree() in R)
    1. In each iteration
        1. use the k-1 training folds to build a large tree
        2. Prune the tree for each tree size A (or α)
        3. Evaluate MSE on the validation fold for each tree size A (or α)
    2. Find the best tree size A (or α)

3. Prune the large tree to the optimal tree size A (prune.tree() in R)

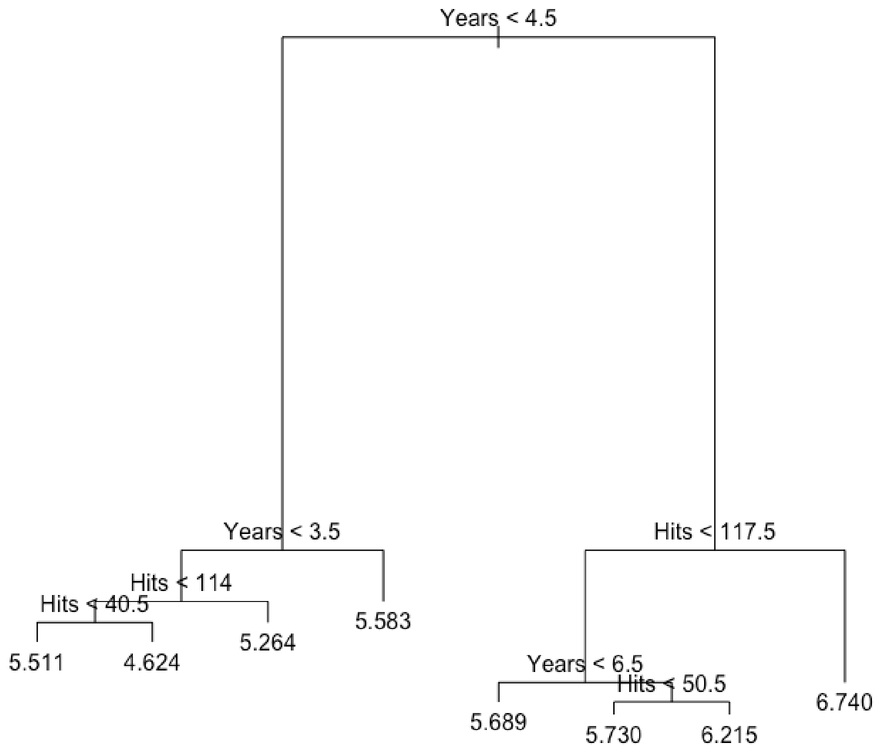
** How/why is this different from the subset selection of linear models? **

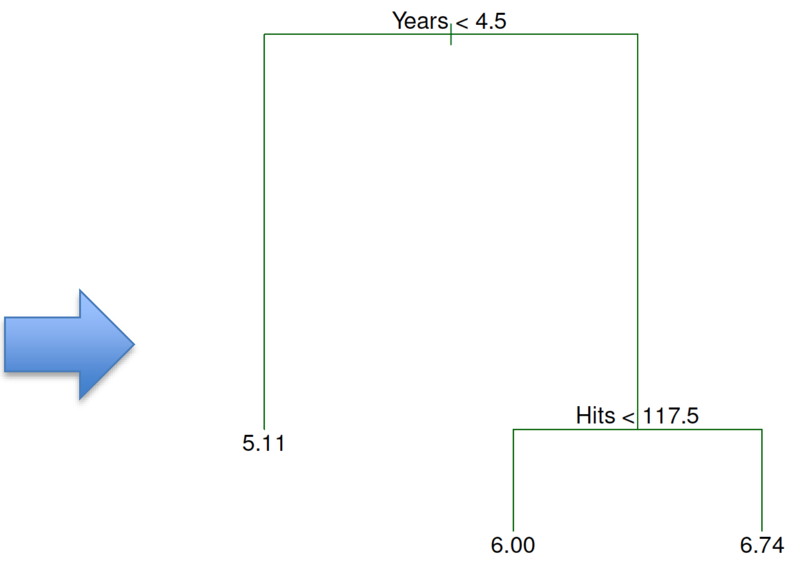
# Hitter's Salary --- Cross-Validation

- Size=3 seems good



# Hitter’s Salary --- Final Pruning



# Classification Trees

- Very similar to a regression tree, except for
  - Prediction is not based on mean but the proportion of each class in the region (similar to KNN)

  - When splitting a tree, RSS is no longer the right criterion
    - Classification error rate is the natural criterion, but not sensitive enough
      - Also cutoff dependent

    - Deviance

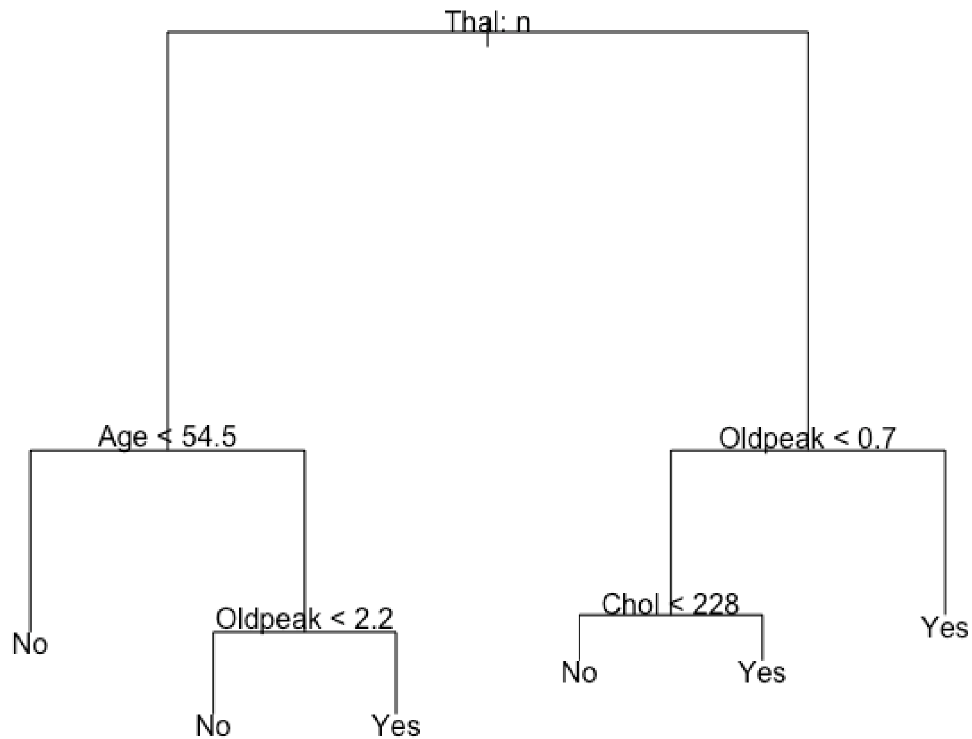
    - Gini index: a measure of total variance across the K classes ("purity")

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

    - Deviance and Gini index are very similar numerically

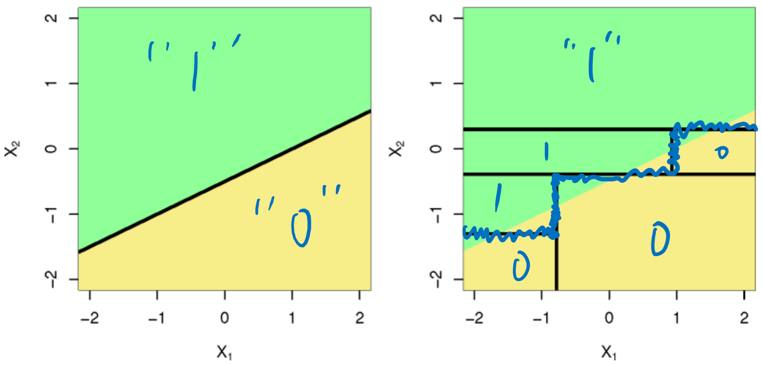
  - When pruning or optimizing tree size, use deviance / classification error / AUC as the criterion

# Example: Heart data

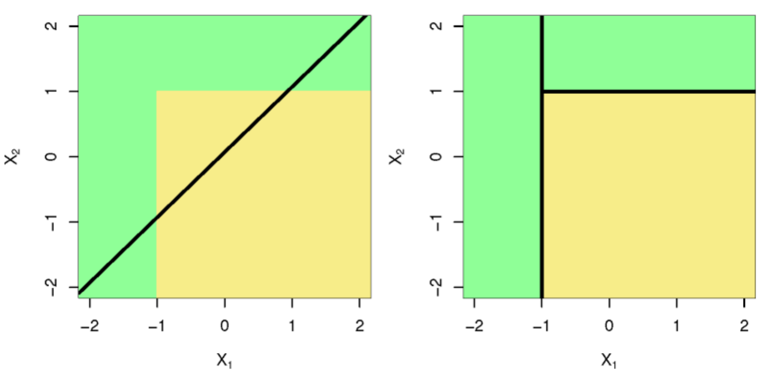


# Trees vs. Linear Model: Classification Example

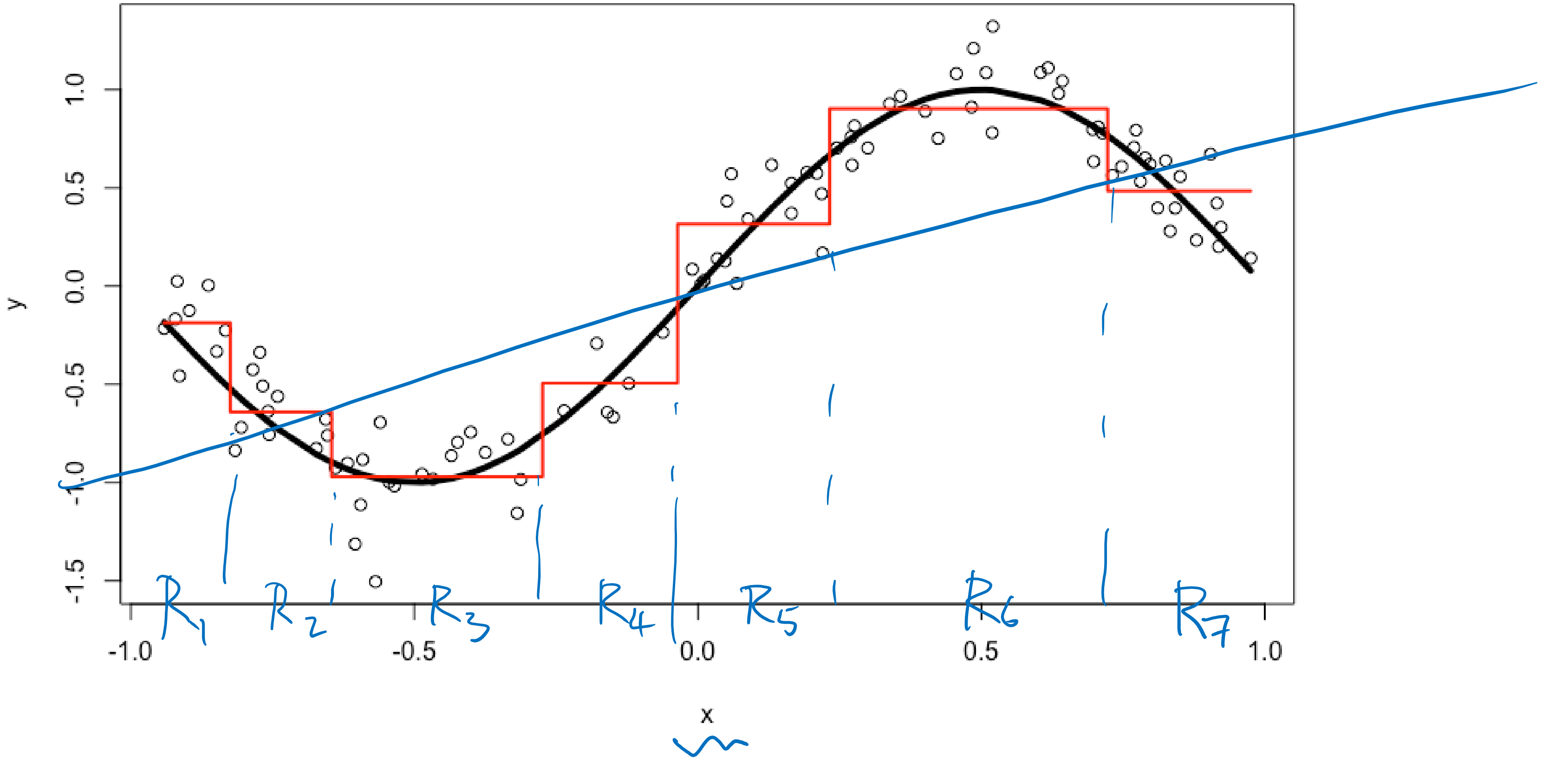
- Top row: the true decision boundary is linear
  - Left: linear model (good)
  - Right: decision tree

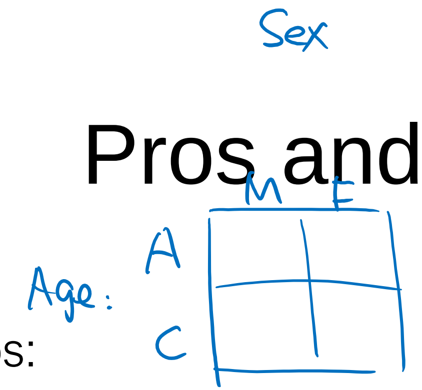



- Bottom row: the true decision boundary is non-linear
  - Left: linear model
  - Right: decision tree (good)

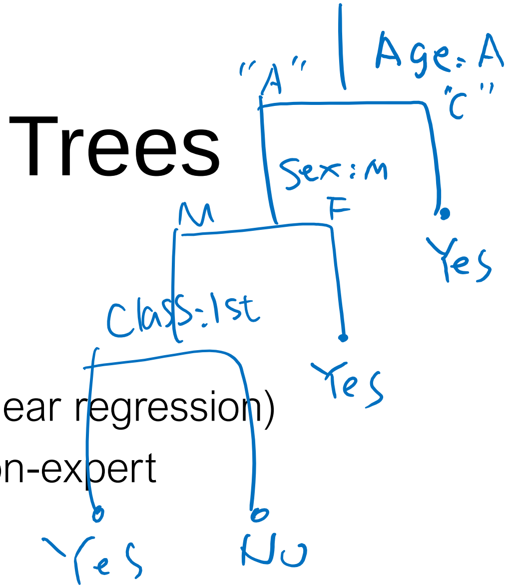


# Trees vs. Linear Model: Regression Example






# Pros and Cons of Decision Trees




- Pros:
  - Very easy to explain to people (probably even easier than linear regression)
  - Can be plotted graphically and easily interpreted even by non-expert
  - Work on both classification and regression problems

  - Capture nonlinear effect (no more transformations on X or Y)
  - Handle categorical predictors (no more dummy variables)
  - Handle interactions (no more * or : )
  - Handle missing data

- Cons:
  - Trees don't have the same prediction accuracy as some of the more complicated approaches that we examine in this course
  - Final tree is not very stable
  - Computational issue with big categorical variables