QTM 347 Machine Learning

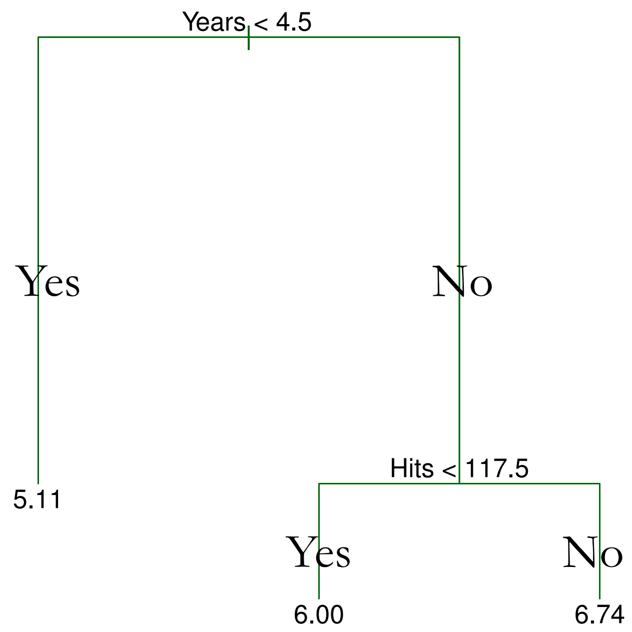
Lecture 12: Decision tree

Ruoxuan Xiong

Suggested reading: ISL Chapter 8

Example: Predicting a baseball player's salary

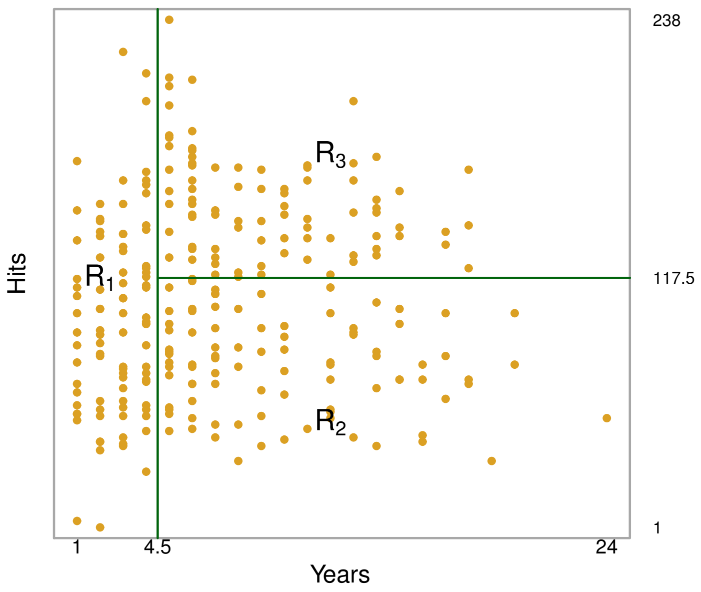
- Predict a baseball player's **log salary** (Y_i) based on
  - **Years**: The number of years that he has played in the major leagues
  - **Hit**: The number of hits that he made in the previous year
- **Regression tree** consists of a series of splitting rules
  - ➢ $\text{Years}_i < 4.5$: $\hat{Y}_i = 5.11$
  - ➢ $\text{Years}_i \geq 4.5$ & $\text{Hits}_i < 117.5$: $\hat{Y}_i = 6.00$
  - ➢ $\text{Years}_i \geq 4.5$ & $\text{Hits}_i \geq 117.5$: $\hat{Y}_i = 6.74$

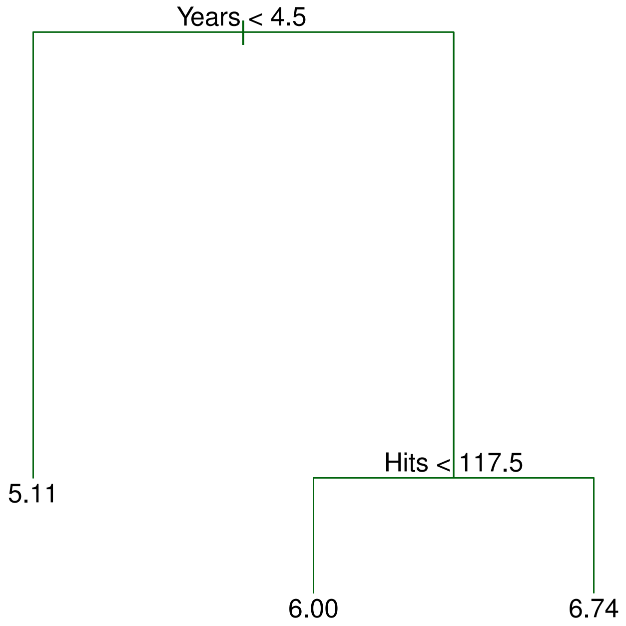


Example: Predicting a baseball player's salary

- Predict a baseball player's **log salary** (Y_i) based on
  - **Years**: The number of years that he has played in the major leagues
  - **Hit**: The number of hits that he made in the previous year
- **Regression tree** segments the feature space to disjoint regions

$$R_1 = \{X \mid \text{Years}_i < 4.5\}$$
$$R_2 = \{X \mid \text{Years}_i \geq 4.5, \text{Hits}_i < 117.5\}$$
$$R_3 = \{X \mid \text{Years}_i \geq 4.5, \text{Hits}_i \geq 117.5\}$$



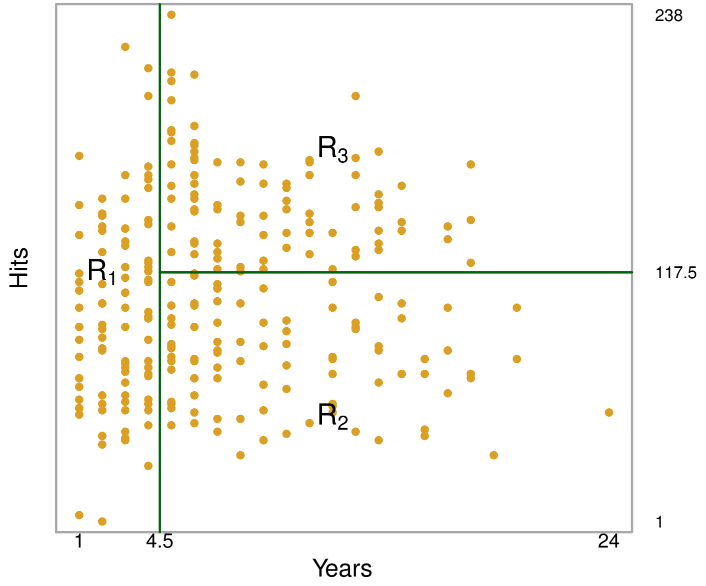


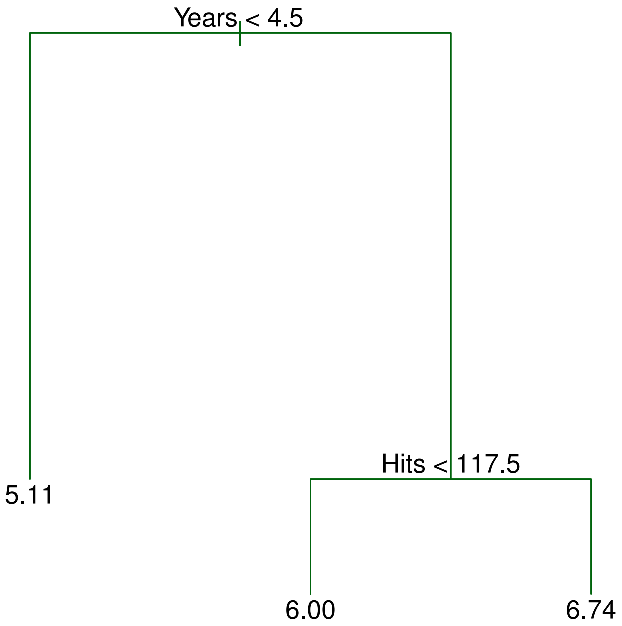
Advantage of decision trees

- Easy to interpret
- Closer to human decision-making
- Easy to visualize graphically

How is a decision tree built?

- **Two main steps**
  1. Partition the feature space into J **distinct and non-overlapping** regions, $R_1, R_2, \cdots, R_J$
  2. Make the **same** prediction for every observation in region R_j: Mean of the training observations in R_j

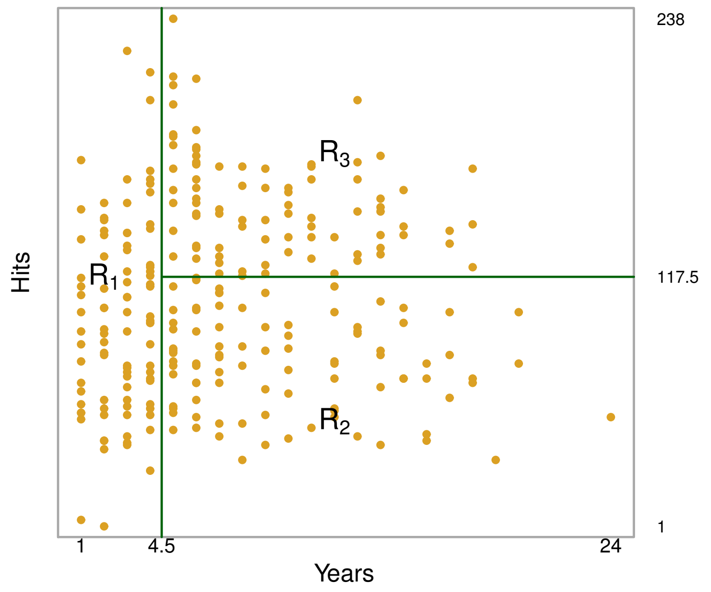




How is a decision tree built?

- **Two main steps**
  1. Partition the feature space into J **distinct and non-overlapping** regions, $R_1, R_2, \cdots, R_J$
  2. Make the **same** prediction for every observation in region R_j: Mean of the training observations in R_j

- Example of step 2: $(\text{Years}_i, \text{Hits}_i, Y_i)$
  - Alan: (14, 81, 6.16)
  - Al: (2, 37, 4.25)
  - Andres: (2, 81, 4.32)
  - Bill: (18, 168, 6.66)
  - Brian: (14, 137, 6.80)
  - Bob: (7, 49, 5.70)



EMORY

How is a decision tree built?

- **Two main steps**
  1. Partition the feature space into J **distinct and non-overlapping** regions, $R_1, R_2, \cdots, R_J$
  2. Make the **same** prediction for every observation in region R_j: Mean of the training observations in R_j

- Example of step 2: $(\text{Years}_i, \text{Hits}_i, Y_i)$
  - Alan: (14, 81, 6.16)
  - Al: (2, 37, 4.25)
  - Andres: (2, 81, 4.32)
  - Bill: (18, 168, 6.66)
  - Brian: (14, 137, 6.80)
  - Bob: (7, 49, 5.70)

$R_1 = \{X \mid \text{Years}_i < 4.5\}$ $\quad \hat{Y}_{R_1} = \frac{4.25 + 4.32}{2}$

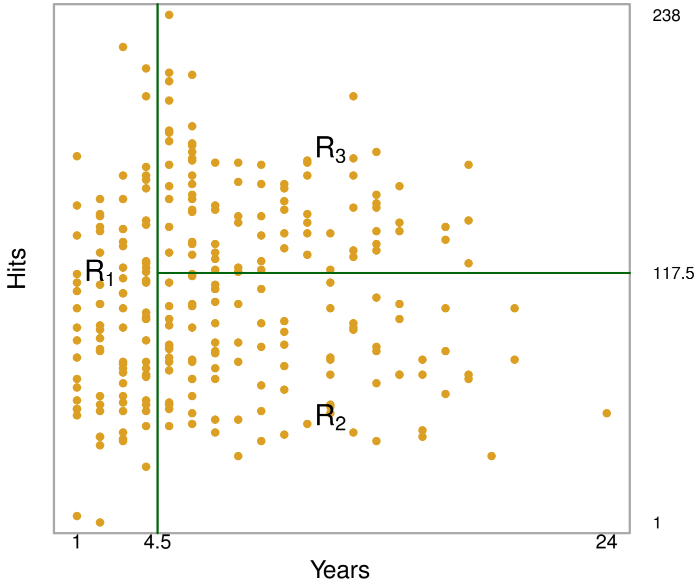
$R_2 = \{X \mid \text{Years}_i \geq 4.5, \text{Hits}_i < 117.5\}$ $\quad \hat{Y}_{R_2} = \frac{6.16 + 5.70}{2}$

$R_3 = \{X \mid \text{Years}_i \geq 4.5, \text{Hits}_i \geq 117.5\}$ $\quad \hat{Y}_{R_3} = \frac{6.66 + 6.80}{2}$

How is a decision tree built?

- **Two main steps**
  1. Partition the feature space into J **distinct and non-overlapping** regions, $R_1, R_2, \cdots, R_J$.

     - Find boxes that minimize the RSS $\sum_{j=1}^{J} \sum_{i \in R_j} \left(Y_i - \hat{Y}_{R_j}\right)^2$.
     - $\hat{Y}_{R_j}$ is the mean label value for the training observations in R_j.
     - **Next:** *Top-down, greedy approach*, begins at the top of the tree.



Top-down, greedy approach to partition feature space

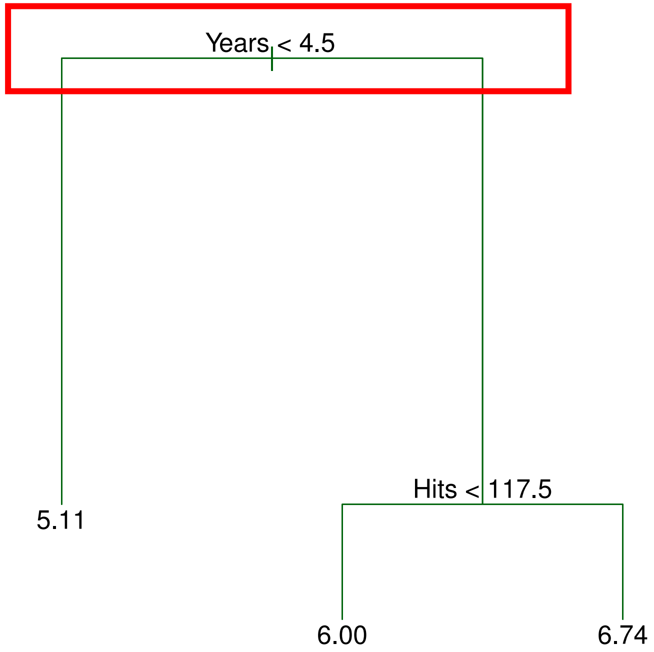
- **Select 1st cut point:** Select the predictor X_j and the cut point s
  - Define the pair of half-planes $R_1(j,s) = \{X|X_j < s\}$ and $R_2(j,s) = \{X|X_j \geq s\}$ that minimize

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

- Example:

$$\sum_i (Y_i - \bar{Y})^2 = 207.15$$

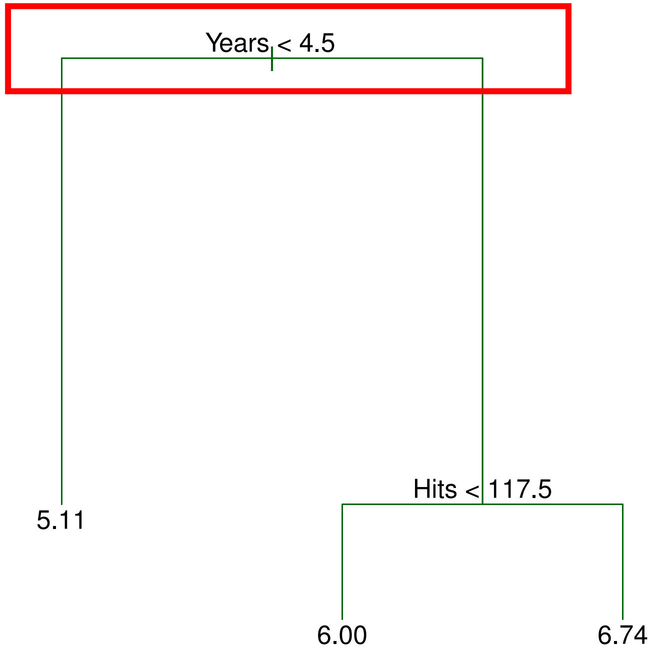
X_j	s	RSS
Year	4	120.18
Year	4.5	115.06
Year	6	133.30
Hits	110	163.75
Hits	120	164.53



Top-down, greedy approach to partition feature space

- **Select 1st cut point**: Select the predictor X_j and the cut point s
  - Define the pair of half-planes $R_1(j,s) = \{X|X_j < s\}$ and $R_2(j,s) = \{X|X_j \geq s\}$ that minimize

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

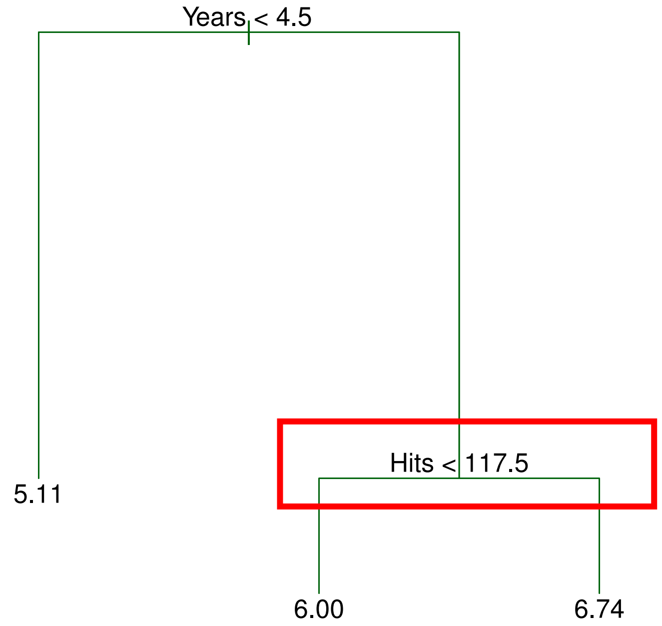


- Example: $R_1 = \{X|\text{Years}_i < 4.5\}$
  $R_2 = \{X|\text{Years}_i \geq 4.5\}$

Top-down, greedy approach to partition feature space

- **Select 2nd cut point**: 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
- Example: $R_1 = \{X|\text{Years}_i < 4.5\}$ and $R_2 = \{X|\text{Years}_i \geq 4.5\}$

R_k	X_j	s	RSS
R_1	Year	3.5	105.85
R_1	Hits	110	107.66
R_1	Hits	120	108.88
R_2	Year	5.5	107.65
R_2	Hits	110	95.91
R_2	Hits	117.5	95.18
R_2	Hits	120	96.23



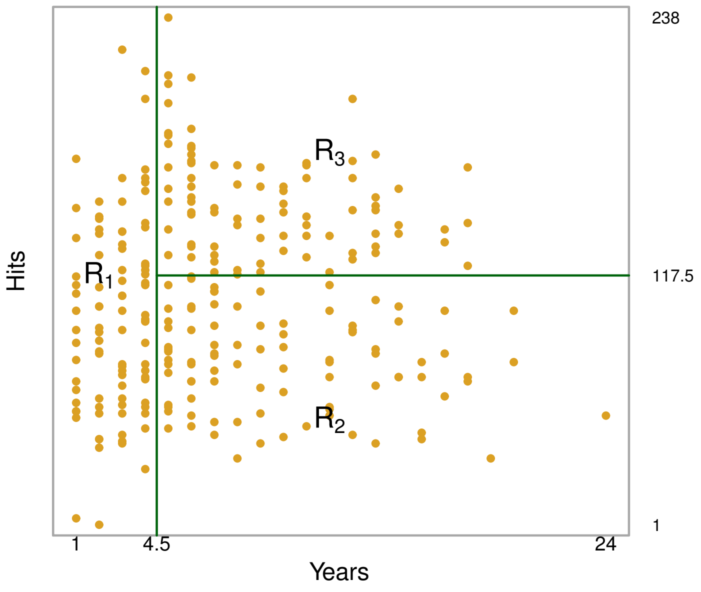
Top-down, greedy approach to partition feature space

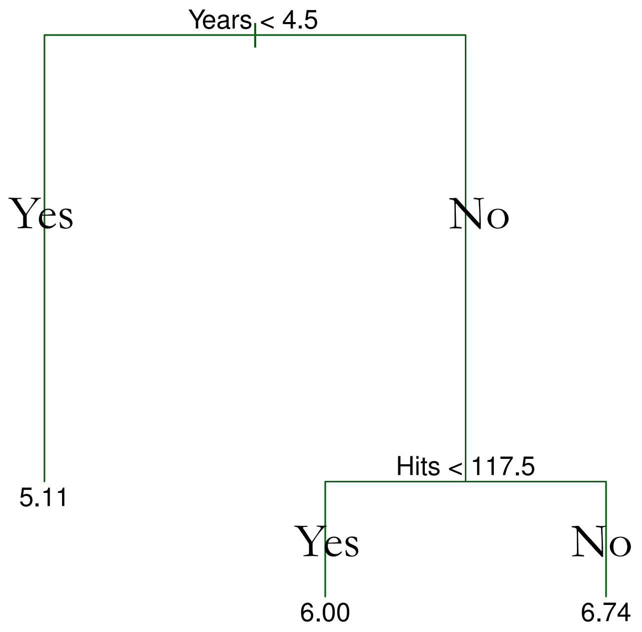
- **Illustration:** Combining both cut points,

$R_1 = \{X|\text{Years}_i \ < \ 4.5\}$

$R_2 = \{X|\text{Years}_i \geq \ 4.5, \text{Hits}_i < 117.5\}$

$R_3 = \{X|\text{Years}_i \geq \ 4.5, \text{Hits}_i \geq 117.5\}$





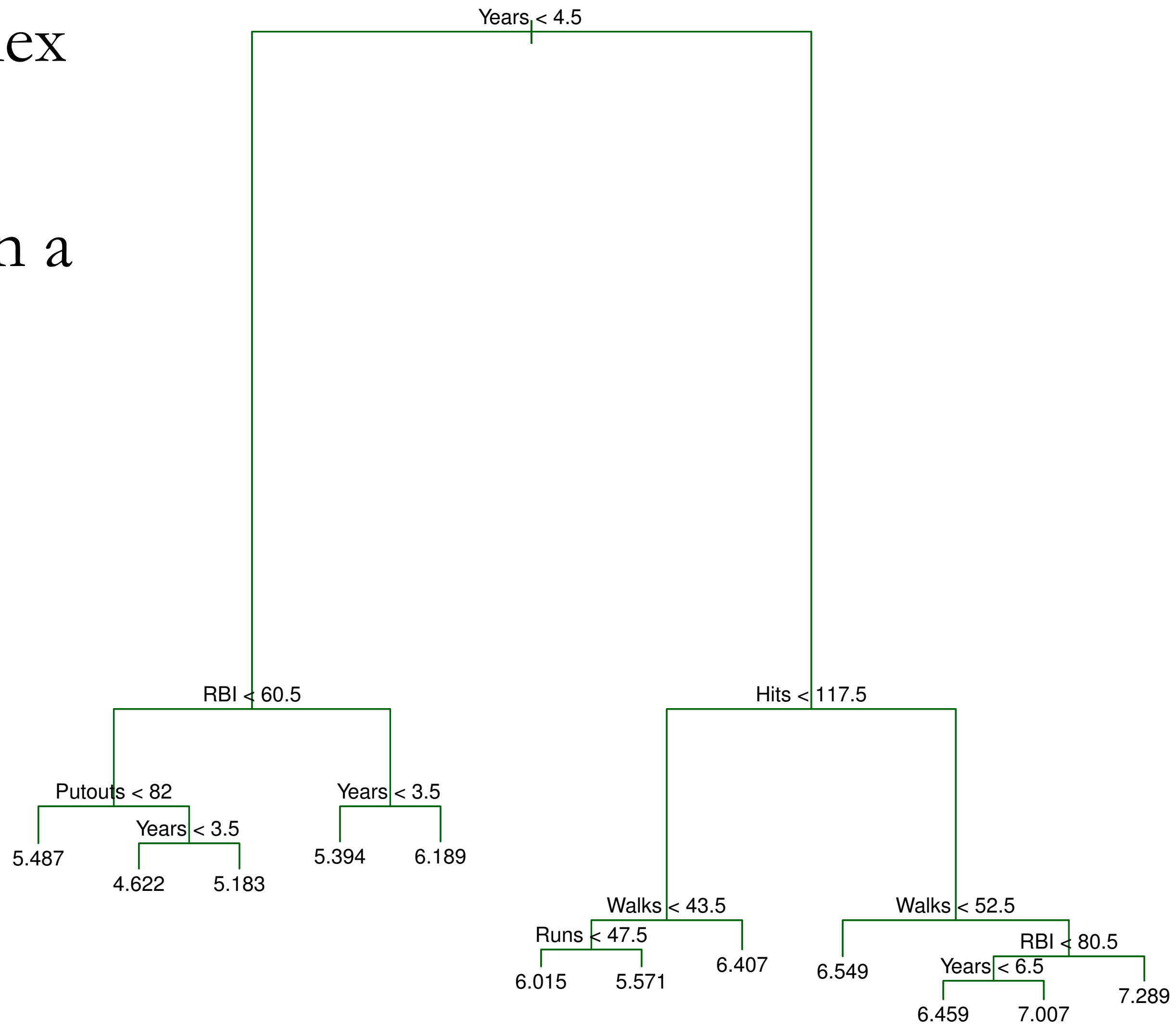
Top-down, greedy approach to partition feature space

- **Select 3rd cut point**: Repeat the same
    - 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.
- …
- **Stopping rule**: Terminate when there are five or fewer observations in each region

EMORY

Decision tree can be overfitted

- The tree might be too complex
- A leaf node may only contain a handful of data points



How to alleviate overfitting?

- **Idea 1**: Find the optimal subtree by cross validation
    - ➢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
    - ➢In our greedy algorithm, it is possible to find good cuts after bad ones

How do we control overfitting?

- **Possible solution**: Prune a large tree T_0 from leaves to the root

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

$$\min \sum_{j=1}^{|T|} \sum_{i \in R_j} \left(Y_i - \hat{Y}_{R_j}\right)^2 + \alpha |T|$$

Similar to lasso

  - $|T|$: number of terminal nodes of the tree T

  - If α is larger, then $|T|$ tends to be ______
    A. larger
    B. smaller

How do we control overfitting?

- **Possible solution**: Prune a large tree T_0 from leaves to the root

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

$$\min \sum_{j=1}^{|T|} \sum_{i \in R_j} \left(Y_i - \hat{Y}_{R_j}\right)^2 + \alpha |T|$$

Similar to lasso

  - $|T|$: number of terminal nodes of the tree T

  - If α is larger, then $|T|$ tends to be smaller

How do we control overfitting?

- **Possible solution**: Prune a large tree T_0 from leaves to the root

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

$$\min \sum_{j=1}^{|T|} \sum_{i \in R_j} \left(Y_i - \hat{Y}_{R_j}\right)^2 + \alpha|T|$$

Similar to lasso

  - When $\alpha = 0$, we select the full tree ($T = T_0$)
  - When $\alpha = \infty$, we select the null tree ($|T| = 0$)

How do we control overfitting?

- **Possible solution**: Prune a large tree T_0 from leaves to the root

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

$$\min \sum_{j=1}^{|T|} \sum_{i \in R_j} \left(Y_i - \hat{Y}_{R_j}\right)^2 + \alpha|T|$$

Similar to lasso

  - For $0 < \alpha_1 < \alpha_2 < \cdots < \alpha_m$,
    When $\alpha = 0$, solution is T_0
    When $\alpha = \alpha_1$, solution is T_1
    When $\alpha = \alpha_2$, solution is T_2
    $\vdots$
    When $\alpha = \alpha_m$, solution is T_m
  - Tree size: $|T_m| < |T_{m-1}| < \cdots < |T_2| < |T_1| < |T_0|$

How do we control overfitting?

- **Possible solution**: Prune a large tree T_0 from leaves to the root

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

$$\min \sum_{j=1}^{|T|} \sum_{i \in R_j} \left(Y_i - \hat{Y}_{R_j}\right)^2 + \alpha|T|$$

Similar to lasso

  - For $0 < \alpha_1 < \alpha_2 < \cdots < \alpha_m$ (the corresponding trees are $T_0, T_1, T_2, \cdots, T_m$), choose the optimal α (the optimal T_i) by cross validation

Cross validation to select α

- **Cross-validation**: Split the training observations into 10 folds
  - For $k = 1, \cdots, 10$, using every fold except the kth:
    - For a range of values $\alpha_1, \alpha_2, \cdots, \alpha_m$, construct the corresponding sequence of trees $T_1^{(k)}, T_2^{(k)}, \cdots, T_m^{(k)}$
      - The sequence of trees vary with the hold-out fold
    - Make prediction for each region in each tree $T_i^{(k)}$
    - For each tree $T_i^{(k)}$, calculate the RSS on the hold-out fold k
  - Select the parameter α that minimizes the average error across 10 folds

Possible variation of cross-validation

- **Cross-validation**: Split the training observations into 10 folds
  - For a range of values $\alpha_1, \alpha_2, \cdots, \alpha_m$, construct the corresponding sequence of trees are $T_1, T_2, \cdots, T_m$
    - Tree structures are fixed in the cross validation
  - For $k = 1, \cdots, 10$, using every fold except the kth
    - Make prediction for each region in each tree T_i
      - Prediction for each region vary with the hold-out fold k
    - For each tree T_i, calculate the RSS on the hold-out fold k
  - Select the optimal tree T_i that minimizes the average error across 10 folds
- **Is this correct?**

EMORY

Possible variation of cross-validation

- **Cross-validation**: Split the training observations into 10 folds
  - For a range of values $\alpha_1, \alpha_2, \cdots, \alpha_m$, construct the corresponding sequence of trees are $T_1, T_2, \cdots, T_m$
    - Tree structures are fixed in the cross validation
  - Select the optimal tree T_i that minimizes the average error across 10 folds

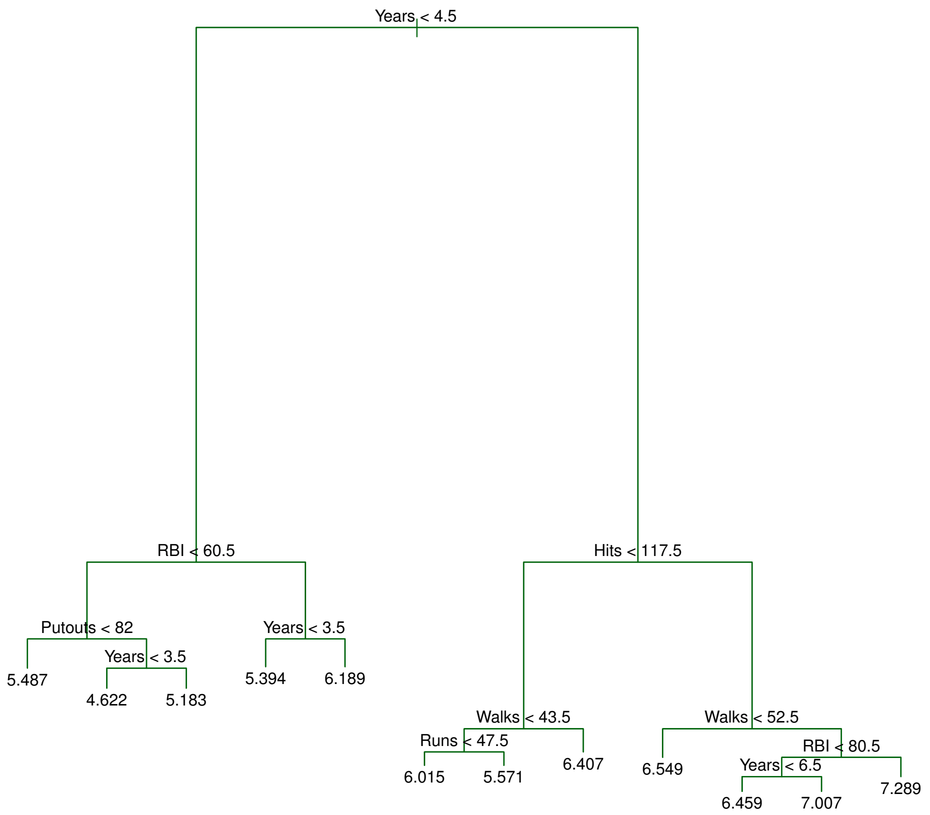
- **Is this correct?**
  - Nope ☹ We need to include **the construction of trees**, using only the training data
  - The construction of trees can overfit

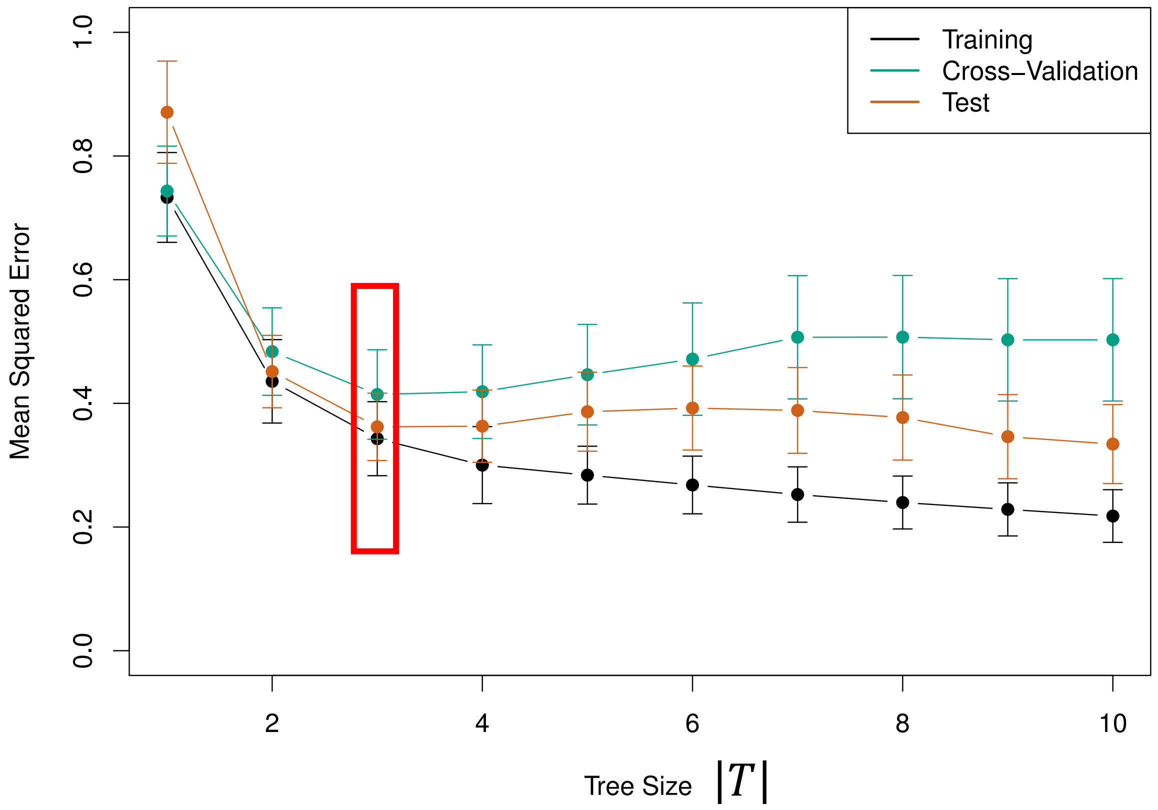
Cross-validation to select α

- **Cross-validation**: Split the training observations into K folds
  - Hold out the kth fold, for a range of values $\alpha_1, \alpha_2, \cdots, \alpha_m$, construct the corresponding sequence of trees are $T_1^{(k)}, T_2^{(k)}, \cdots, T_m^{(k)}$
  - The sequence of trees vary with which fold is held out
  - Use tree $T_i^{(k)}$ to make prediction and calculate RSS on the hold-out fold k
  - Select the optimal parameter α that minimizes the average error across ten folds

Example: a baseball player's salary

- **Unpruned tree** (many terminal nodes) without any cost complexity tuning





Example: a baseball player's salary

- **Pruned tree** (three terminal nodes, $|T| = 3$) with the cost complexity tuning