DECISION TREES

Chapter 08 (part 01)

Outline

- The Basics of Decision Trees
  - Regression Trees
  - Classification Trees
  - Regularization via Pruning
  - Trees vs. Linear Models
  - Advantages and Disadvantages of Trees

REGRESSION TREES

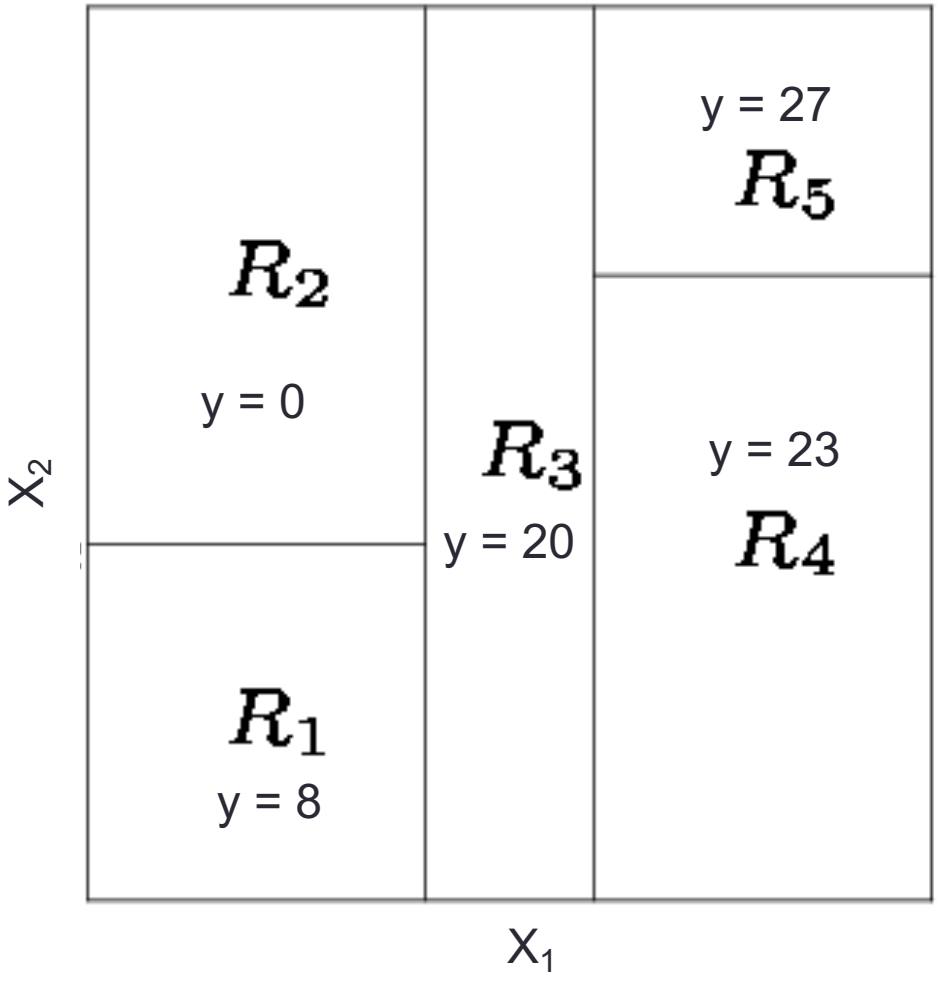
Prediction as Partitioning the feature space

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

- Then for each observation that falls in a particular region (say R_j) we make the same prediction
  - The value of the prediction should be influenced by the response variables of the observations which are members of the region

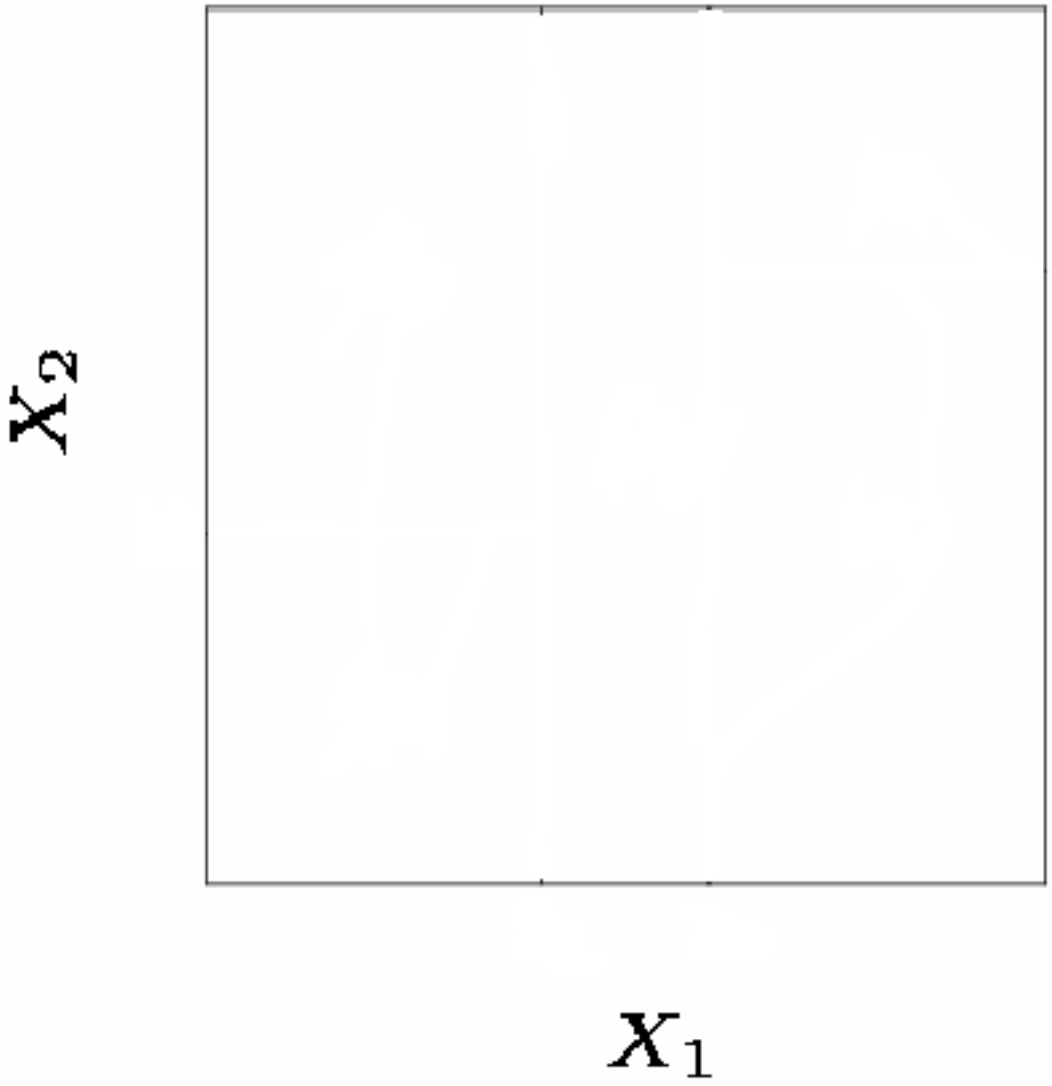
Multi-feature example

- two predictors and five distinct regions
- Depending on which region our new X comes from we would make one of five possible predictions for Y.



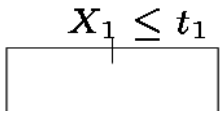
Splitting the X Variables

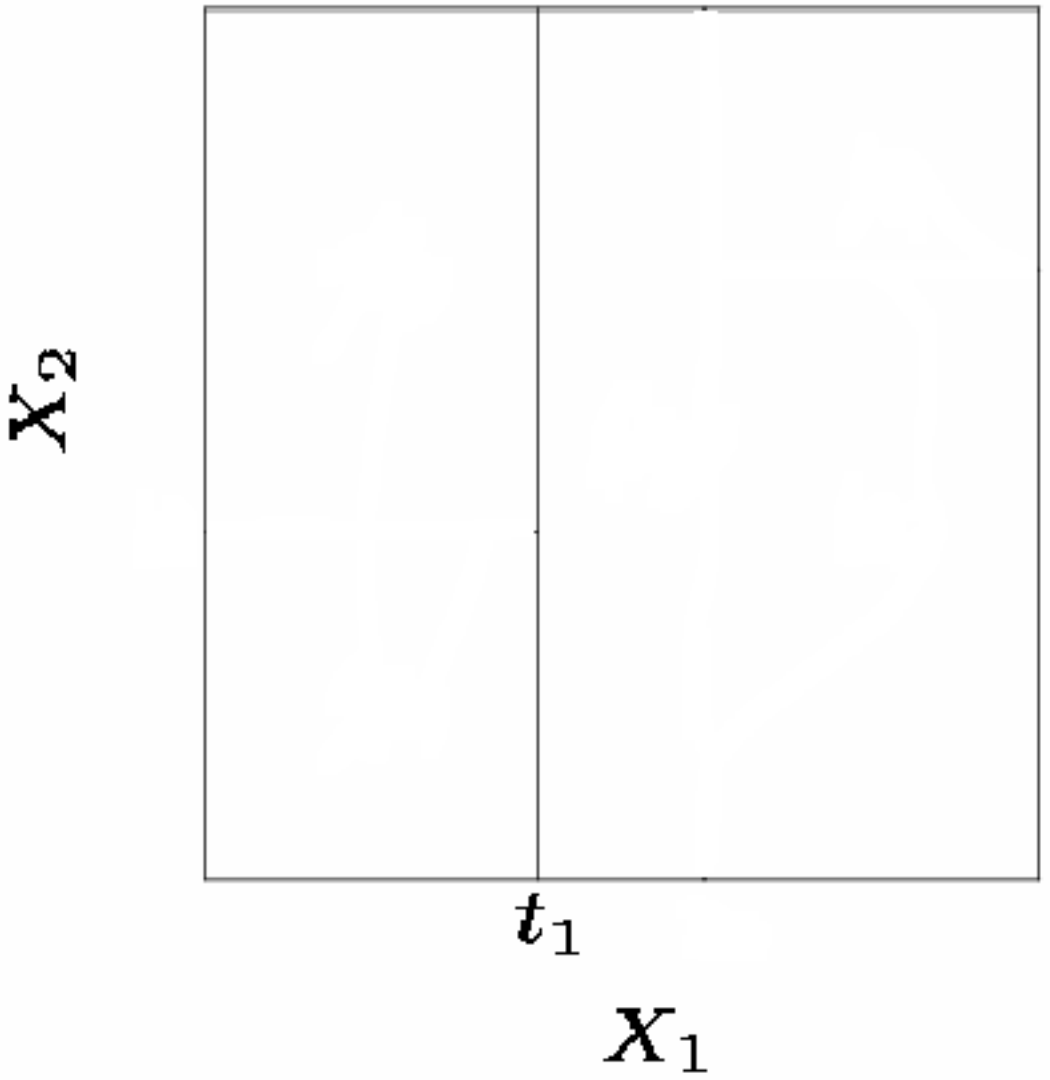
- Create the partitions by *iteratively* splitting one of the existing regions into two regions
- After the initial split, must decide which subregion to split next



Splitting the X Variable

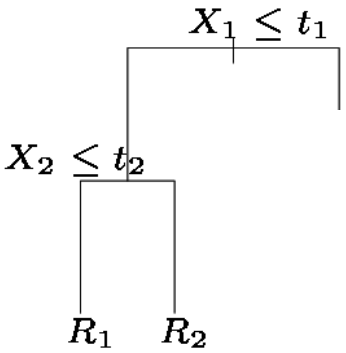
1. First split on $X_1=t_1$

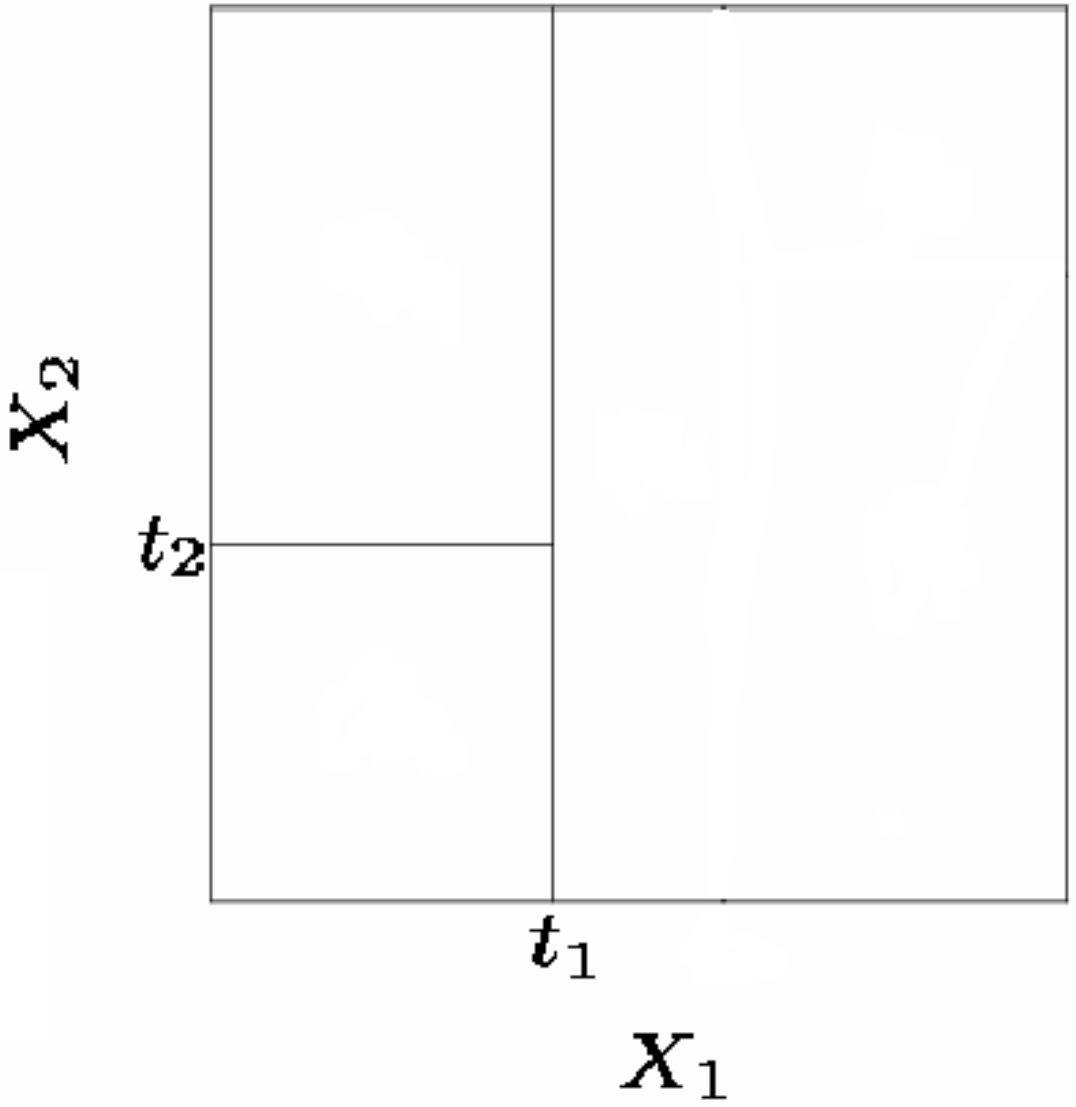




Splitting the X Variable

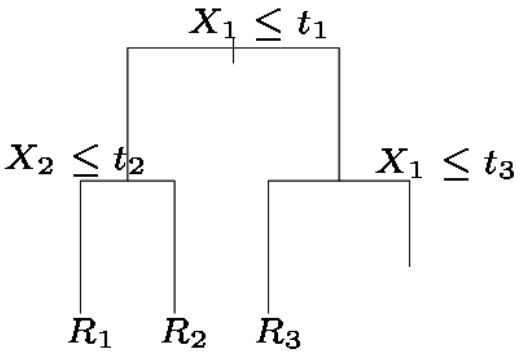
1. First split on $X_1=t_1$
2. If $X_1<t_1$, split on $X_2=t_2$

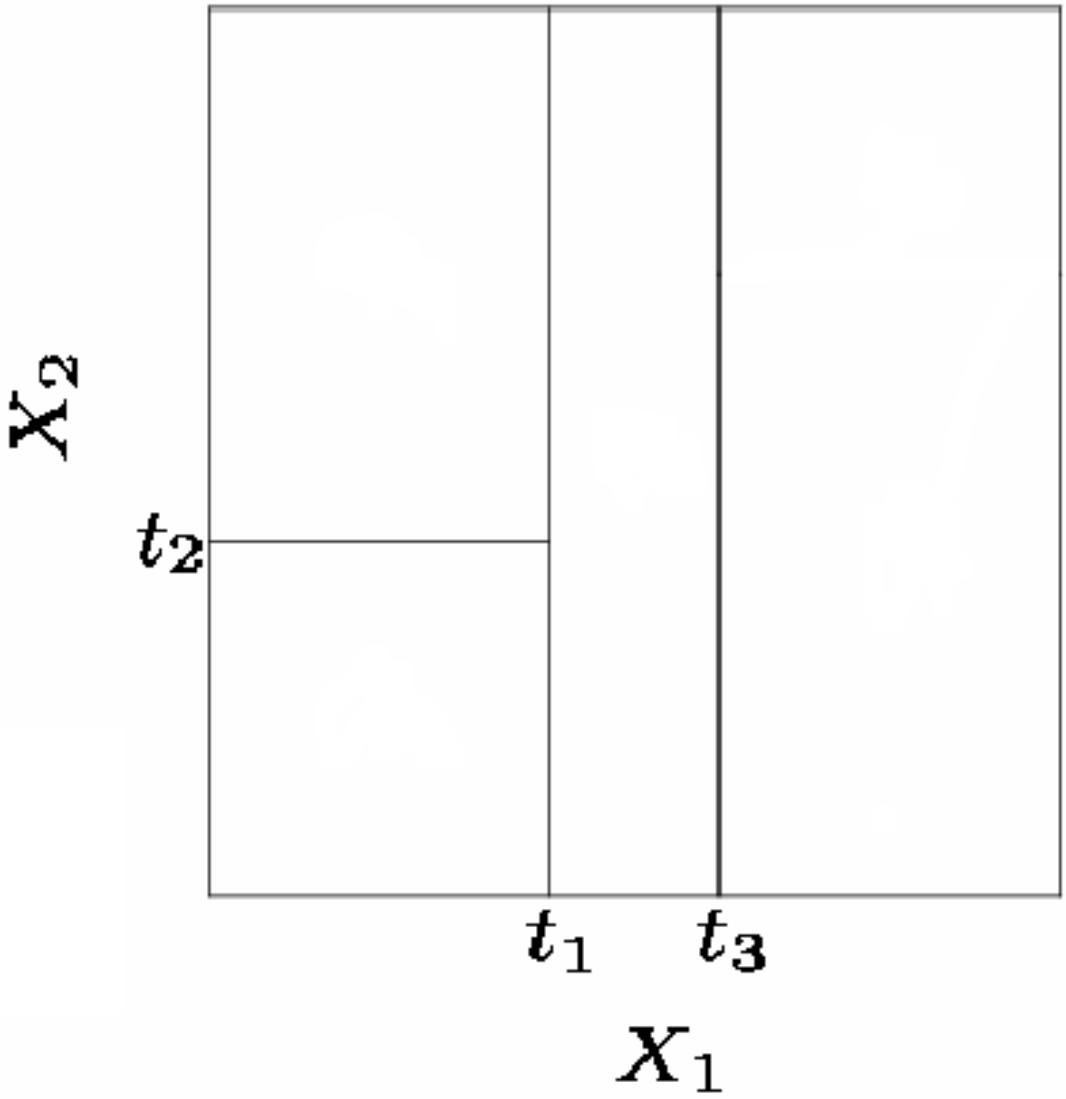




Splitting the X Variable

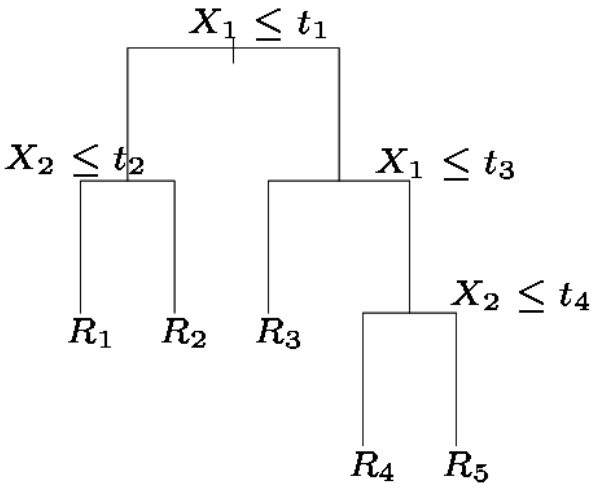
1. First split on $X_1=t_1$
2. If $X_1<t_1$, split on $X_2=t_2$
3. If $X_1>t_1$, split on $X_1=t_3$
4. If $X_1>t_3$, split on $X_2=t_4$

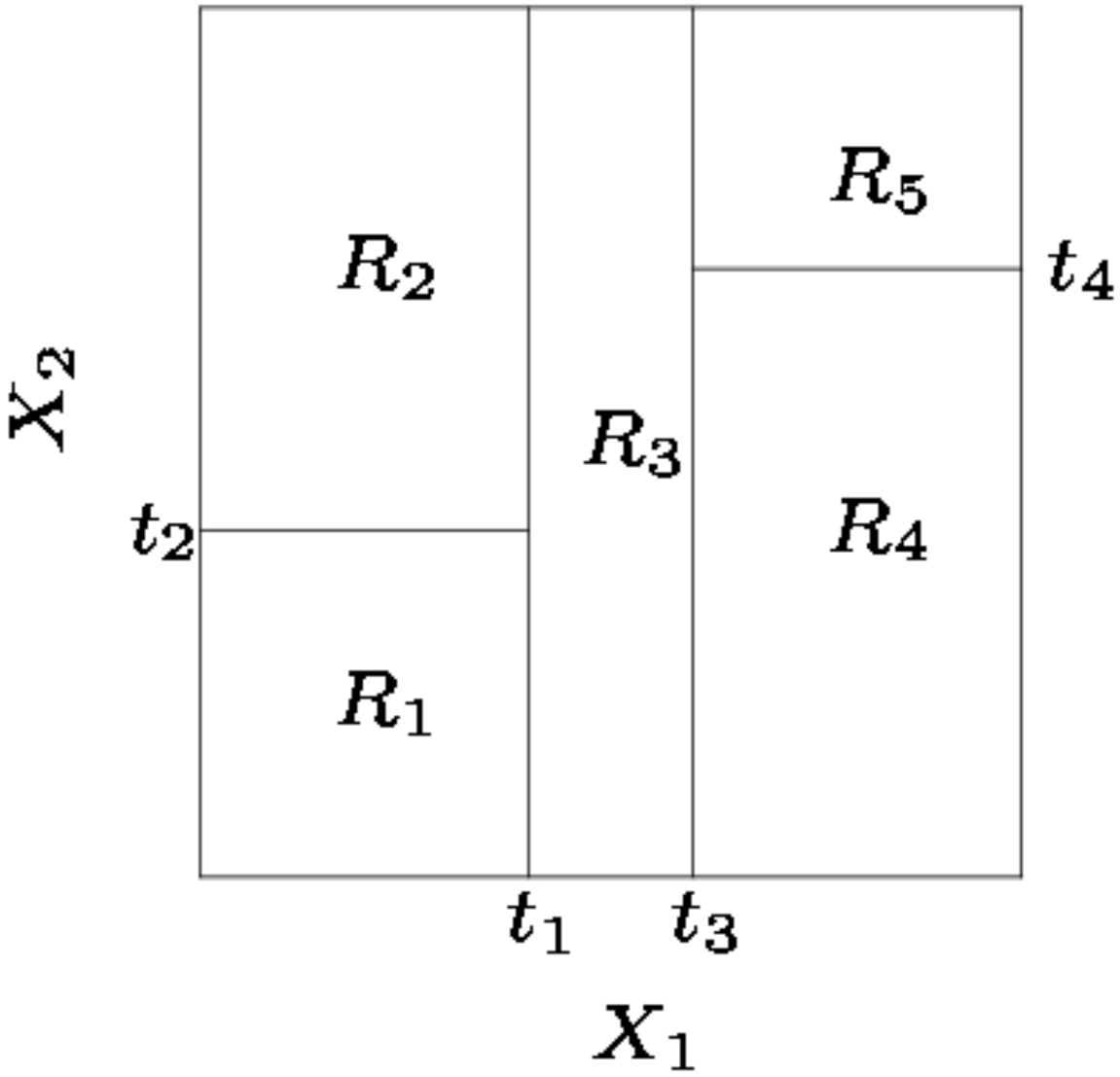




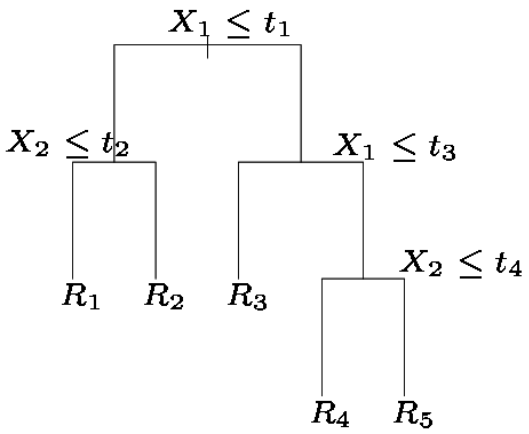
Splitting the X Variable

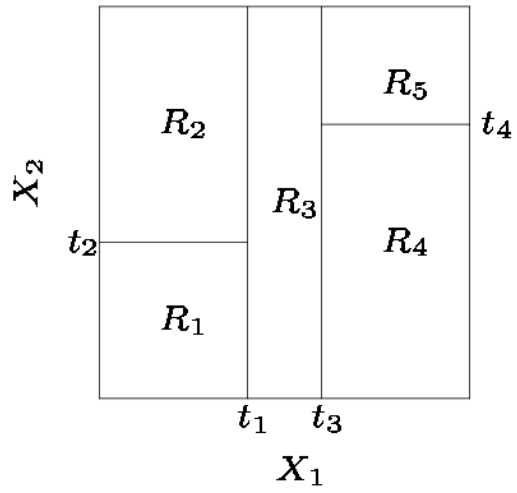
1. First split on $X_1=t_1$
2. If $X_1<t_1$, split on $X_2=t_2$
3. If $X_1>t_1$, split on $X_1=t_3$
4. If $X_1>t_3$, split on $X_2=t_4$

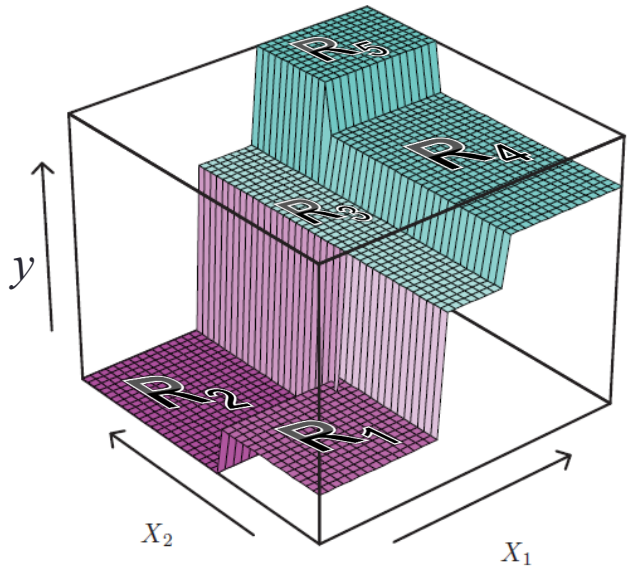




Splitting the X Variable





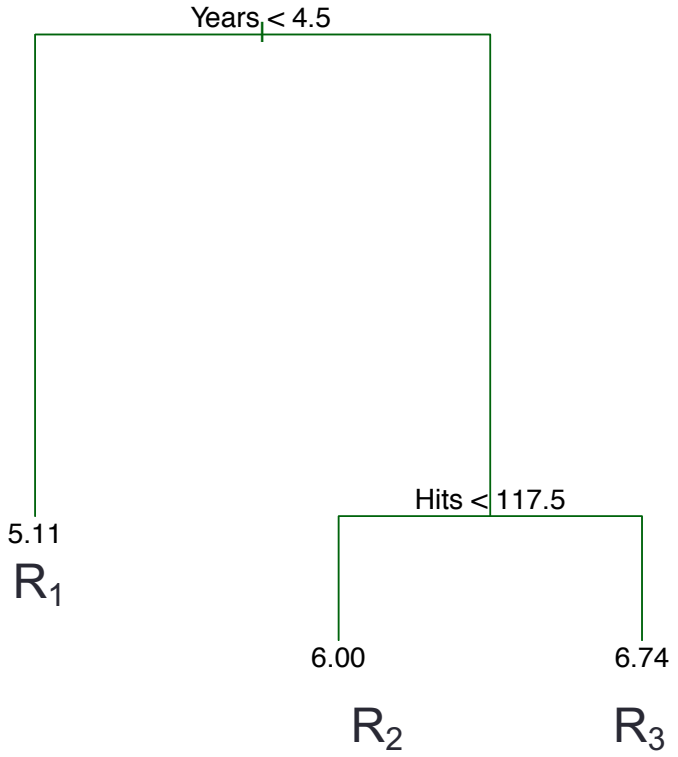


- Partitions can be represented with a tree structure.
- This provides a very simple way to interpret the model

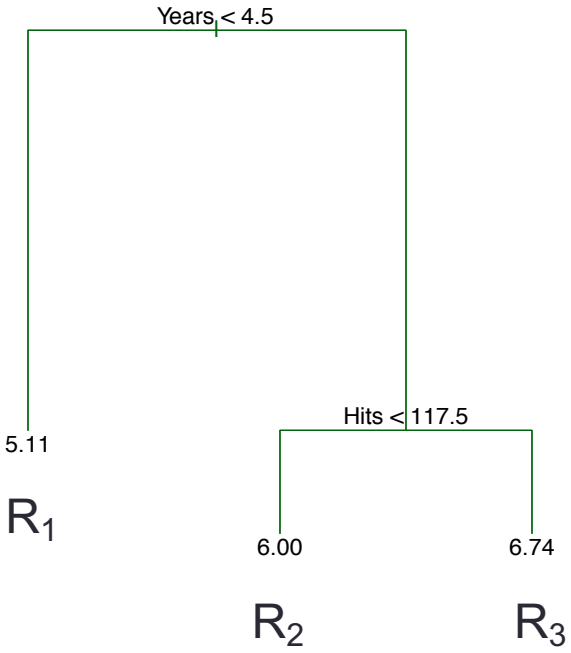
Example: Baseball Players' Salaries

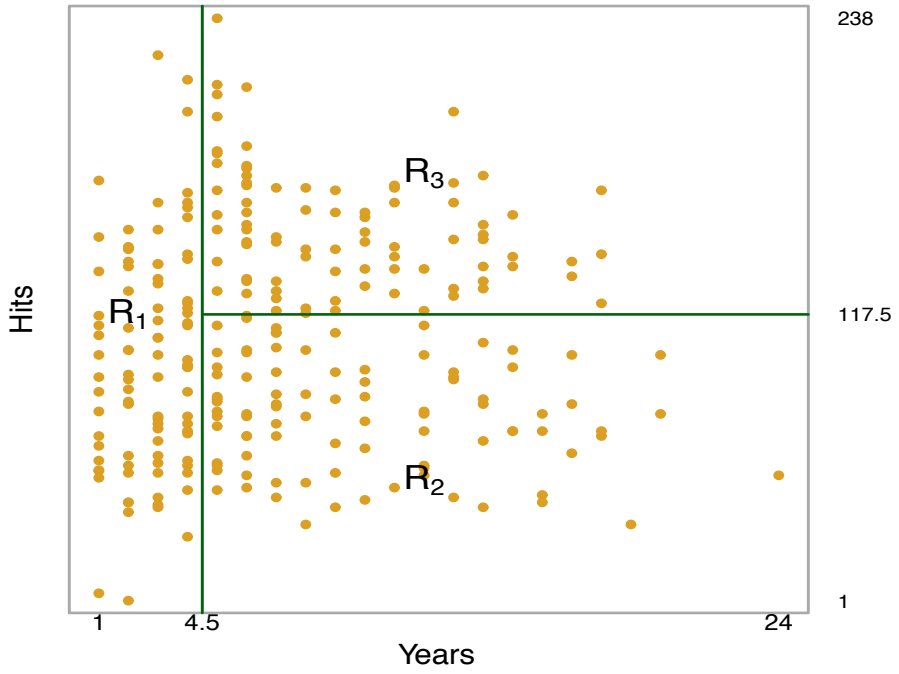
- The predicted Salary is the number in each leaf node. It is the mean of the response for the observations that fall there
- Note that Salary is measured in 1000s, and log-transformed
- The predicted salary for a player who played in the league for more than 4.5 years and had less than 117.5 hits last year is

$$\$1000 \times e^{6.00} = \$402,834$$



Another way of visualizing the decision tree…





Design Decisions

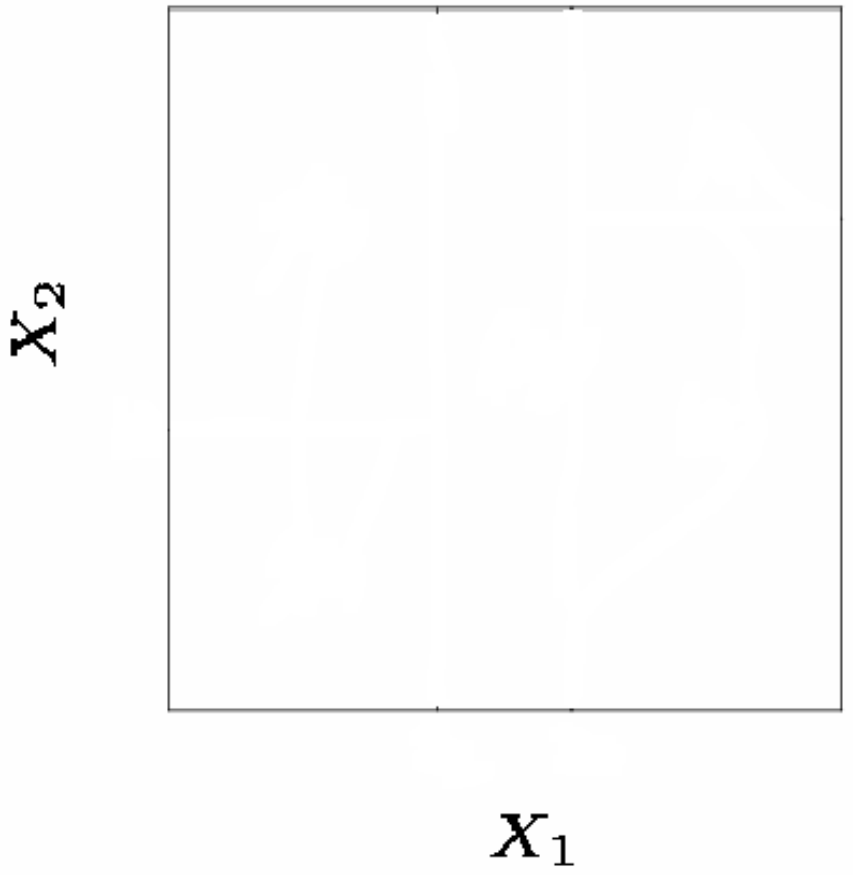
1. Where to split?

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

2. What values should we use for $\hat{y}_{R_1}, \hat{y}_{R_2}, \ldots, \hat{y}_{R_k}$?

Where to Split?

- Consider splitting a region into two regions, $X_j > s$ and $X_j \leq s$ for all possible real values of s and feature indices j.
- One option: Choose the s and j that results in the lowest MSE on the training data.

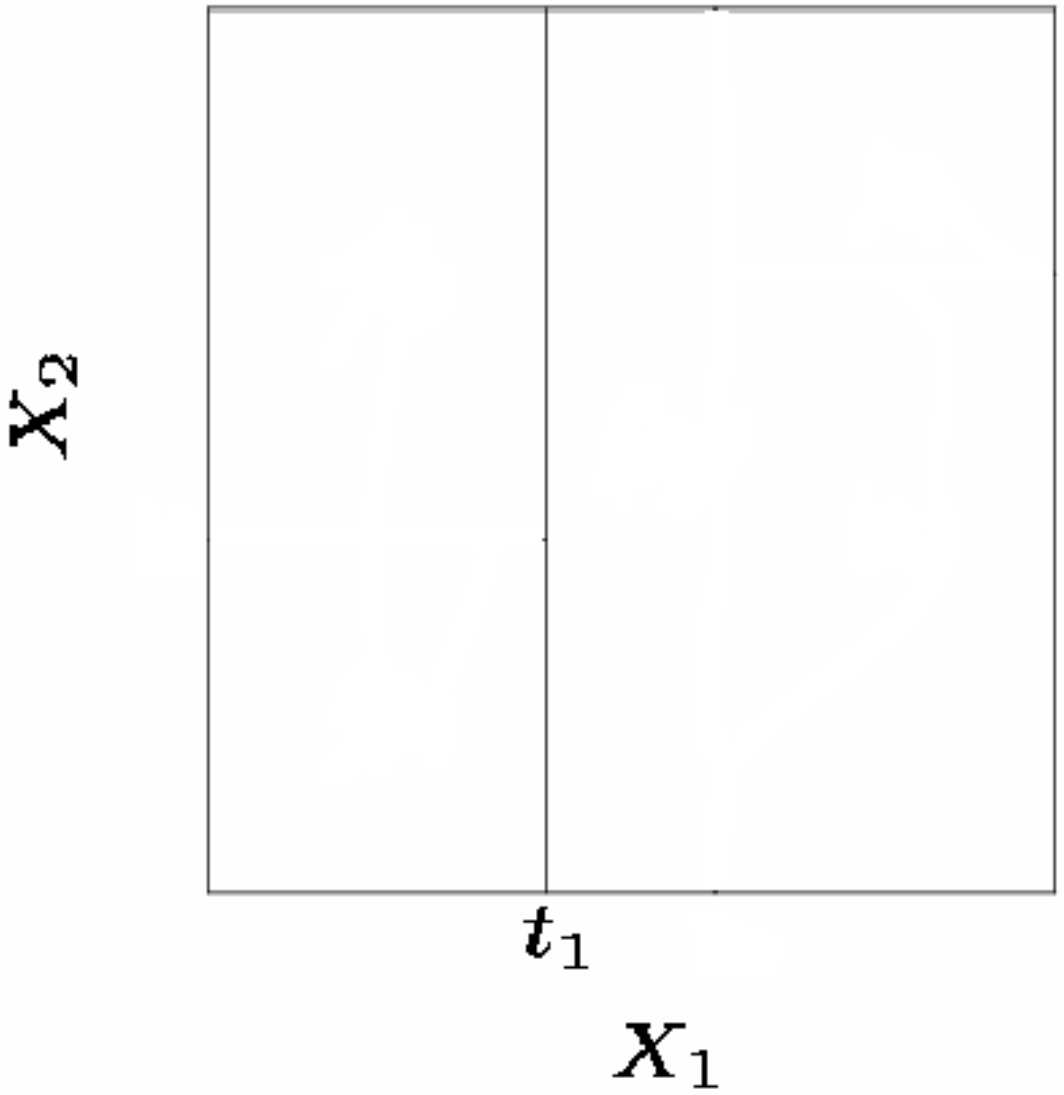


What values should we use for $\hat{y}_{R_1}, \hat{y}_{R_2}, \ldots, \hat{y}_{R_k}$?

- For region R_j, the best prediction is simply the average of all the responses from our training data that fell in region R_j.

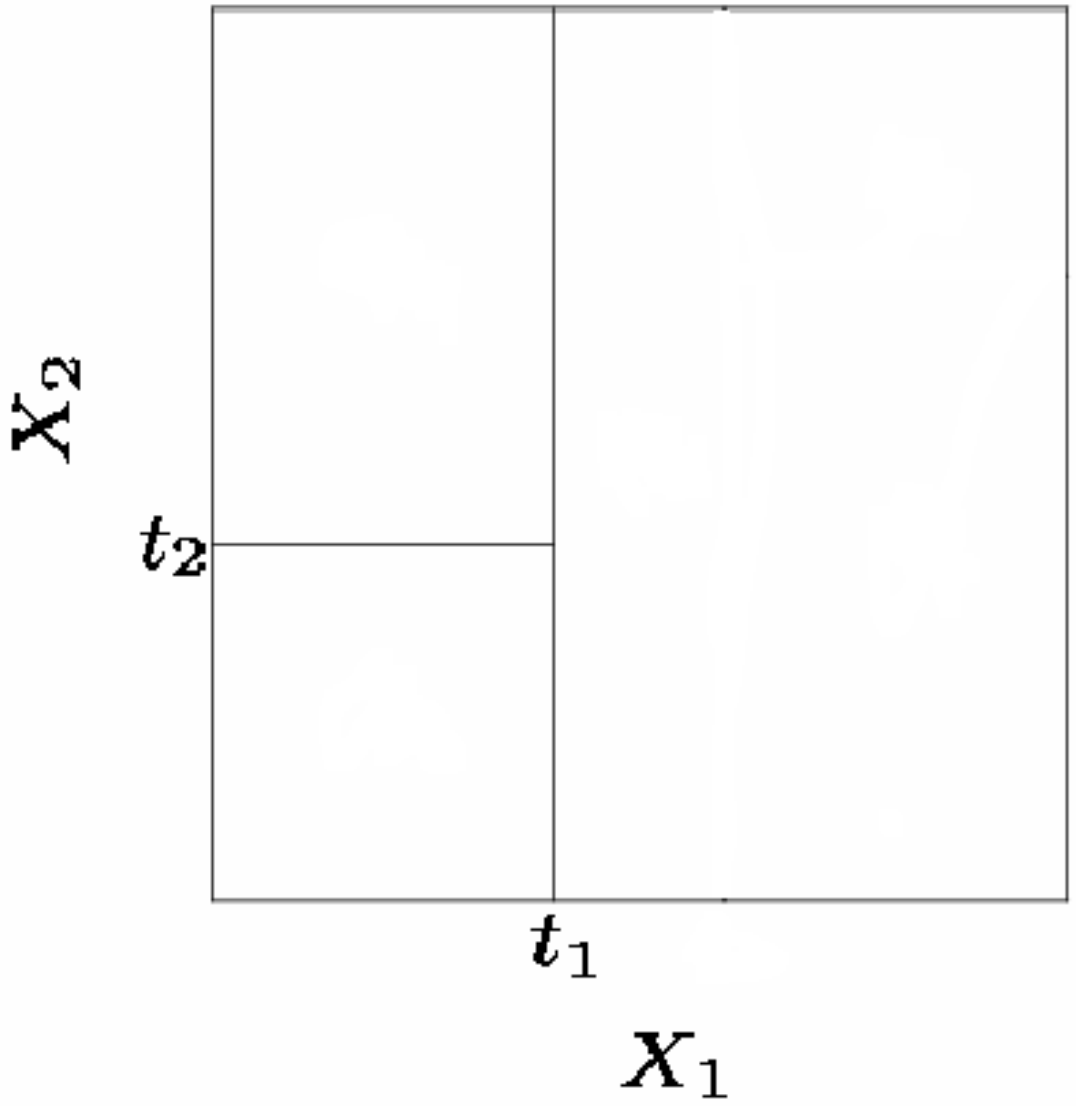
Where to Split Next?

- Suppose the optimal split was on X_1 at point t_1.
- 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 up.
- Again the criteria is smallest MSE.



Where to Split?

- Here the optimal split was the left region on X_2 at point t_2.
- This process continues until our regions have too few observations to continue e.g. all regions have 5 or fewer points.

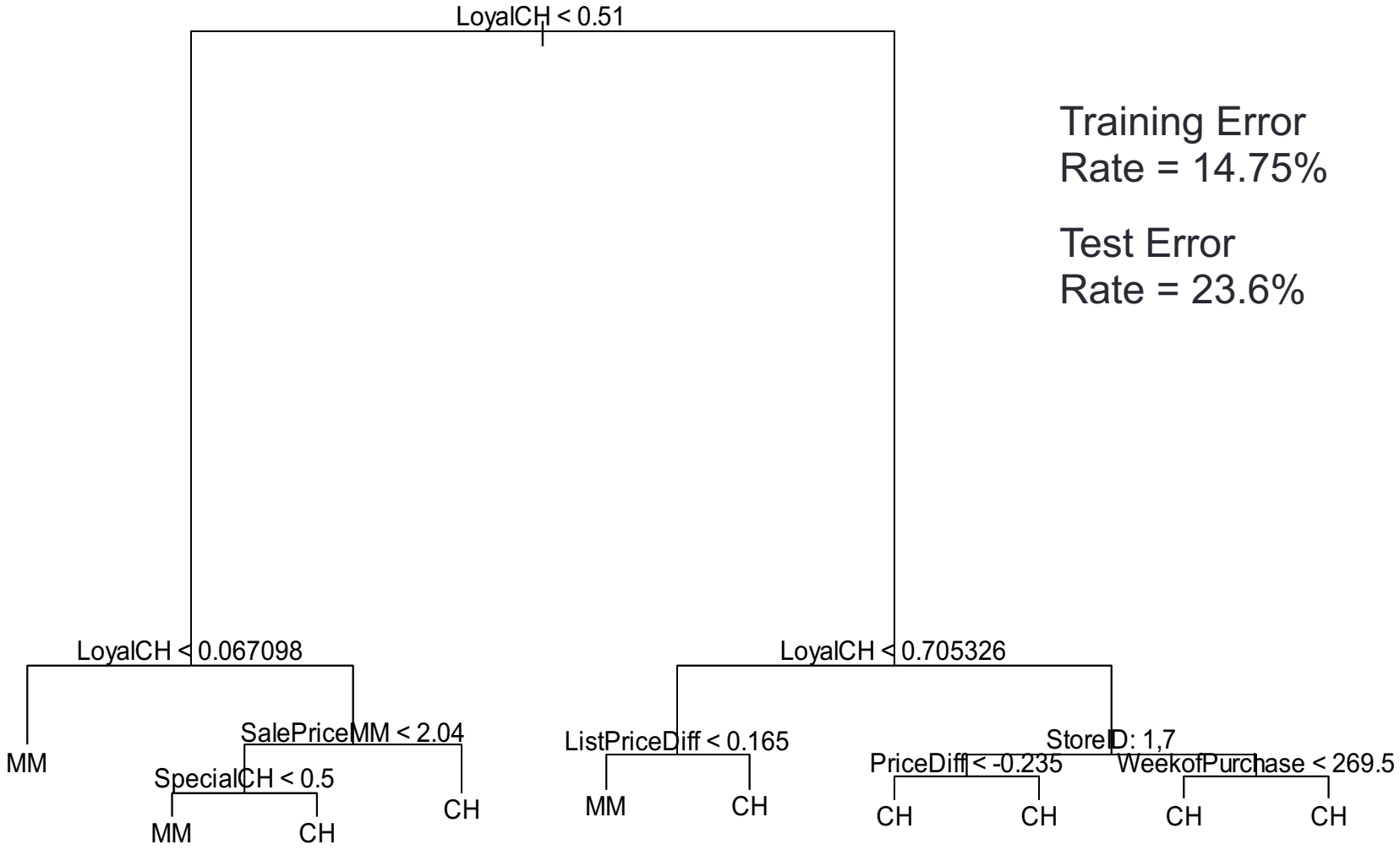


CLASSIFICATION TREES

Growing a Classification Tree

- A classification makes a prediction for a categorical Y
- For each region (or node) we predict the most common category among the training data within that region
- The tree is grown (i.e. the splits are chosen) in exactly the same way as with a regression tree except we need a different criteria to optimize
- There are several possible different criteria to use such as the “gini index” and “cross-entropy” but the easiest one to think about is to minimize the error rate.

Example: Orange Juice Preference



Training Error Rate = 14.75%

Test Error Rate = 23.6%

REGULARIZATION VIA PRUNING

Improving Tree Accuracy

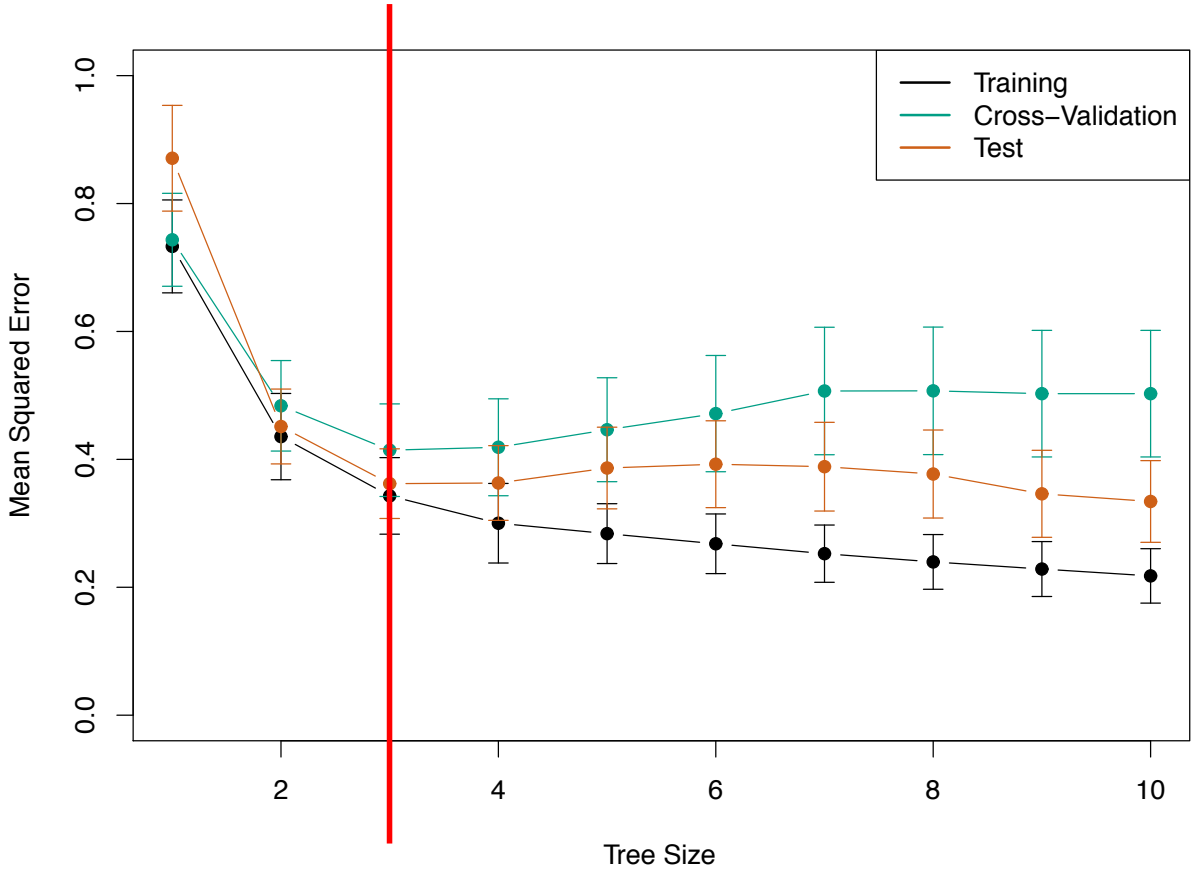
- A large tree (i.e. one with many terminal nodes) may tend to overfit the training data since you could continue to split until each observation had its own leaf node.

- Generally, we can improve estimated test set accuracy by “pruning” the tree i.e. cutting off some of the terminal nodes.

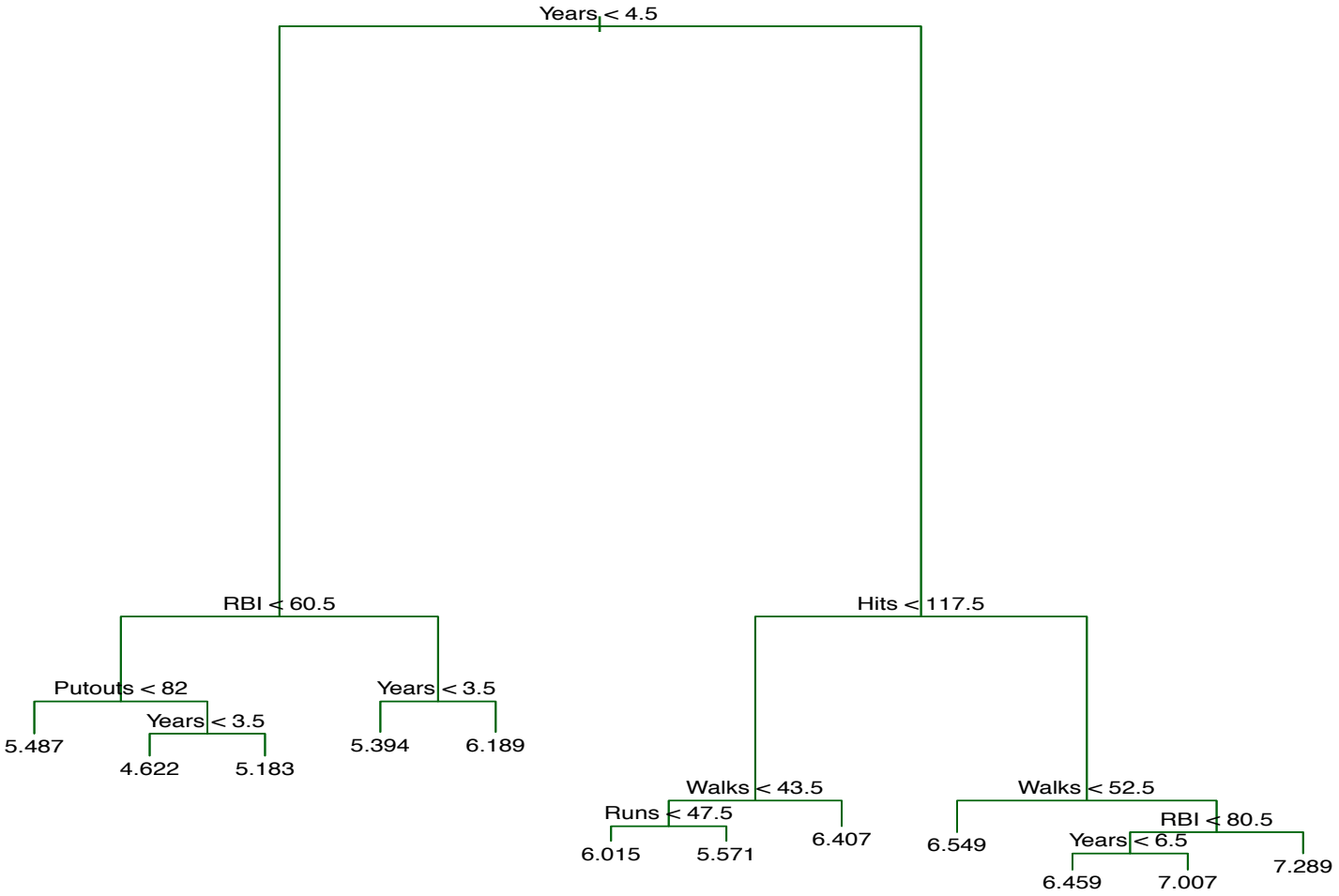
- How do we know how far back to prune the tree? We use **cross validation** to see which subtree has the lowest validation error rate.

Pruning Example: Baseball Players' Salaries

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



Pruning Example: Baseball Players' Salaries



Pruning Example: Baseball Players' Salaries

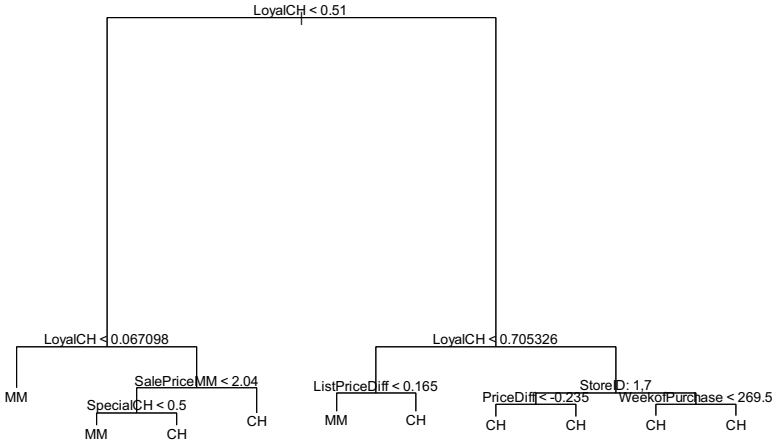
- Even though training MSE continues to decrease with additional leaf nodes…
- Cross Validation indicates that the minimum CV MSE is when the tree size is three (i.e. the number of leaf nodes is 3)

Concept Check:

Why does training MSE continue to decrease as trees gain additional leaf nodes?

Why does cross-val prefer shallower trees?

Pruning Example: Orange Juice Preference

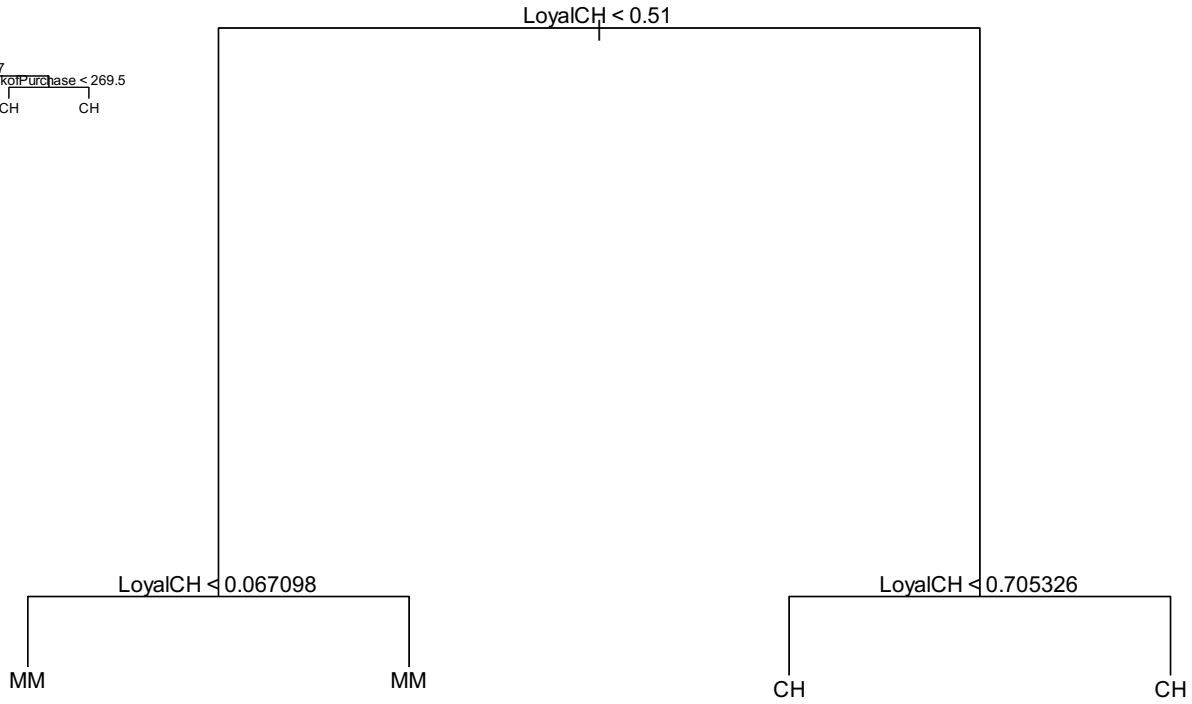


Full Tree Training Error Rate = 14.75%

Full Tree Test Error Rate = 23.6%

Pruned Tree

CV Tree Error Rate = 22.5%



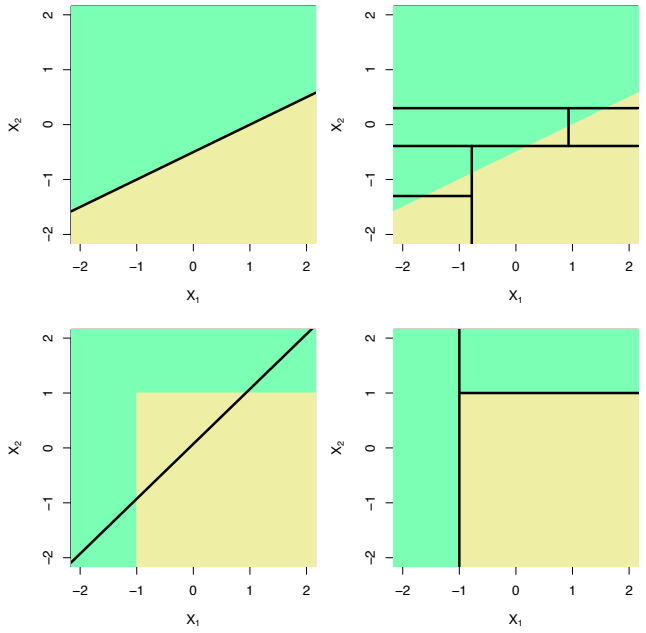
TREES VS. LINEAR MODELS

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 can outperform linear approaches

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)



ADVANTAGES AND DISADVANTAGES OF TREES

Pros and Cons of Decision Trees

- Pros:
  - Trees are very easy to interpret (probably even easier than linear regression)
  - Trees can be plotted graphically, and are easily interpreted even by non-expert
  - Tress handle both classification and regression problems

- Cons:
  - Trees don’t have the same prediction accuracy as some of the more complicated approaches in ML

In Class Exercise Part 1

- Given a set of datapoints in a region, write code for making a single splitting decision based on minimizing MSE. You should write this as a function call where you are provided
  - A matrix X of predictor values (*n* rows containing X_1 through X_P) which fall into the current region
  - vector of real valued labels (Y), one value per row
- Use *i* to index through the observations
- Use *j* to index through the predictors
- Use *s* to represent the splitting threshold for the chosen predictor
- Return the values for *j* and *s* which minimizes training set MSE
- Challenge: Can you vectorize it?
- Call your function `splitRegion(df)`
  - `Return(j,s,yLow,yHigh,dfLow,dfHigh)`

In Class Exercise Part 2

- Add code (to wrap your code from part 1) to allow consideration for determining which of the regions in the tree affords the best next split. You should write this as a function where you get:
  - The current tree which partitions the space into regions 1..r
  - The training set MSE for this tree
  - A function “region(T, k)” which returns the matrix X of observations belonging in region k of the tree T, along with a vector of true Y values for those observations.
- Return the region (k), split feature (j) and threshold (s) which minimizes the training set MSE over the choice of region, feature and threshold