Machine Learning Using Tree-based Methods: Decision Trees, Random Forest

ISLR Chapter 08

Outline

The Basics of Decision Trees

- Regression Trees
- Classification Trees
- Pruning Trees
- Trees vs. Linear Models
- Advantages and Disadvantages of Trees

Partitioning Up the Predictor Space

One way to make predictions in a regression problem

- is to divide the predictor 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 every X

- that falls in a particular region (say R_j)
- we make the same prediction,

Regression Trees

Regression Trees

Suppose for example

- we have two regions R_1 and R_2
- with $\hat{Y}_1 = 10, \hat{Y}_2 = 20$

Then for any value of X

- such that $X \in R_2$
  we would predict 10,
- otherwise if $X \in R_1$
  we would predict 20.

The General View

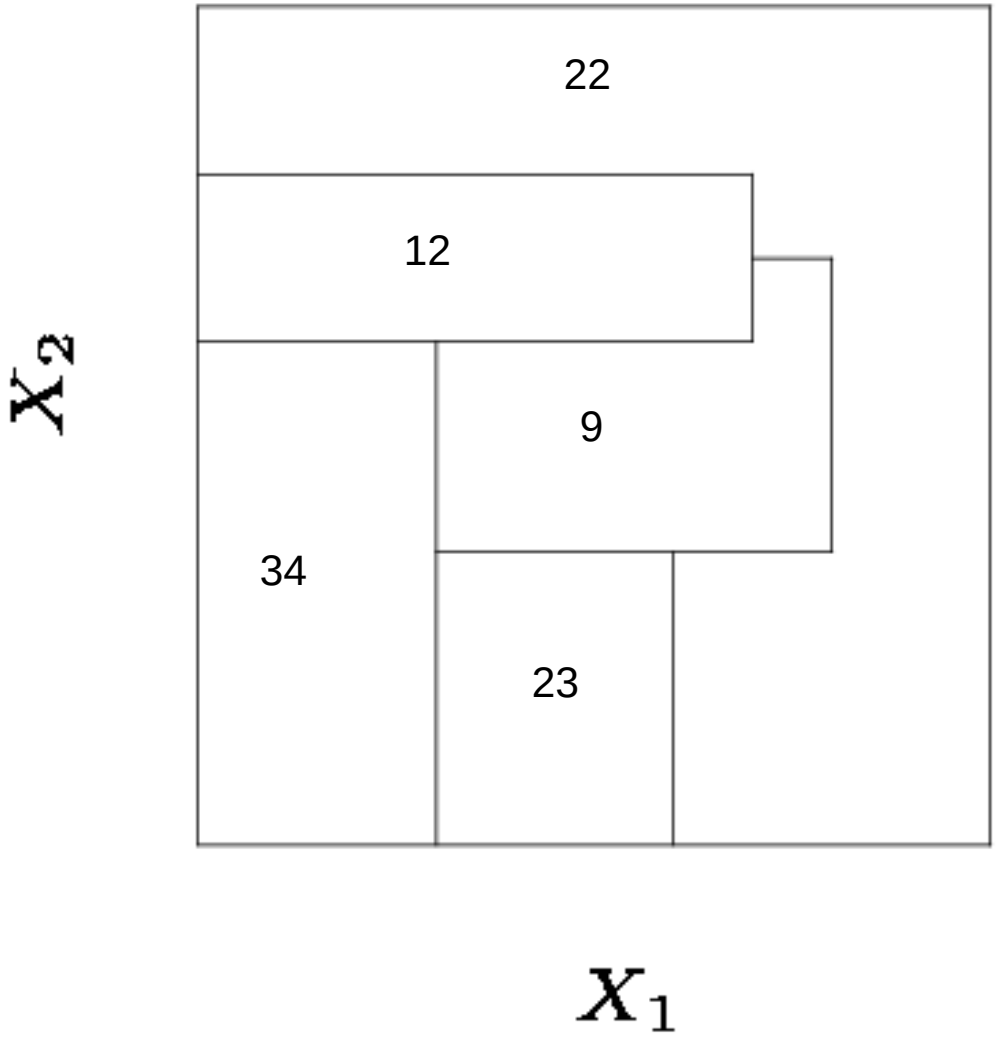
Here we have two predictors and five distinct regions

Depending on which region

- our new X comes from

we would make one of five

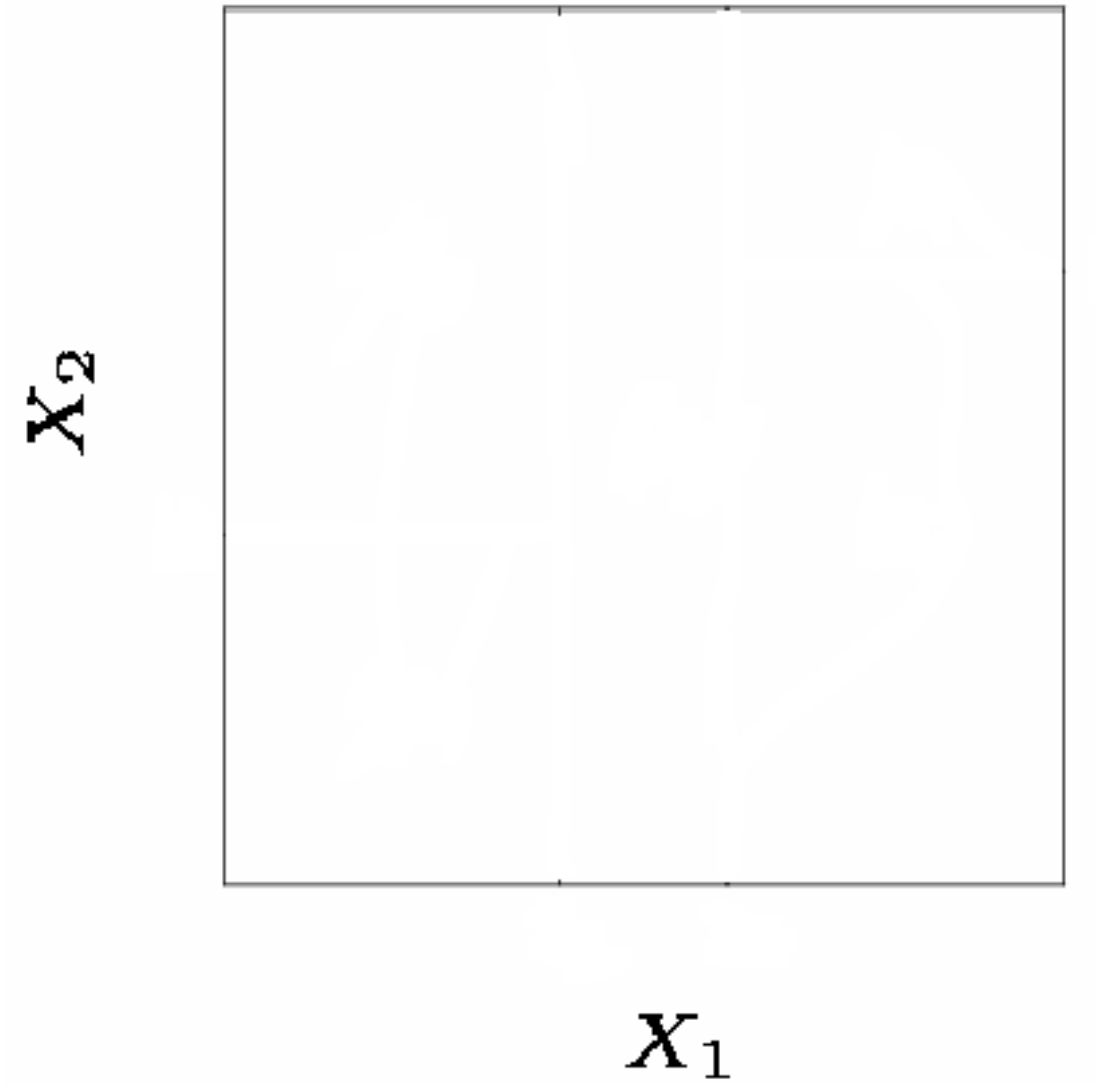
- possible predictions for Y.



Splitting the X Variables

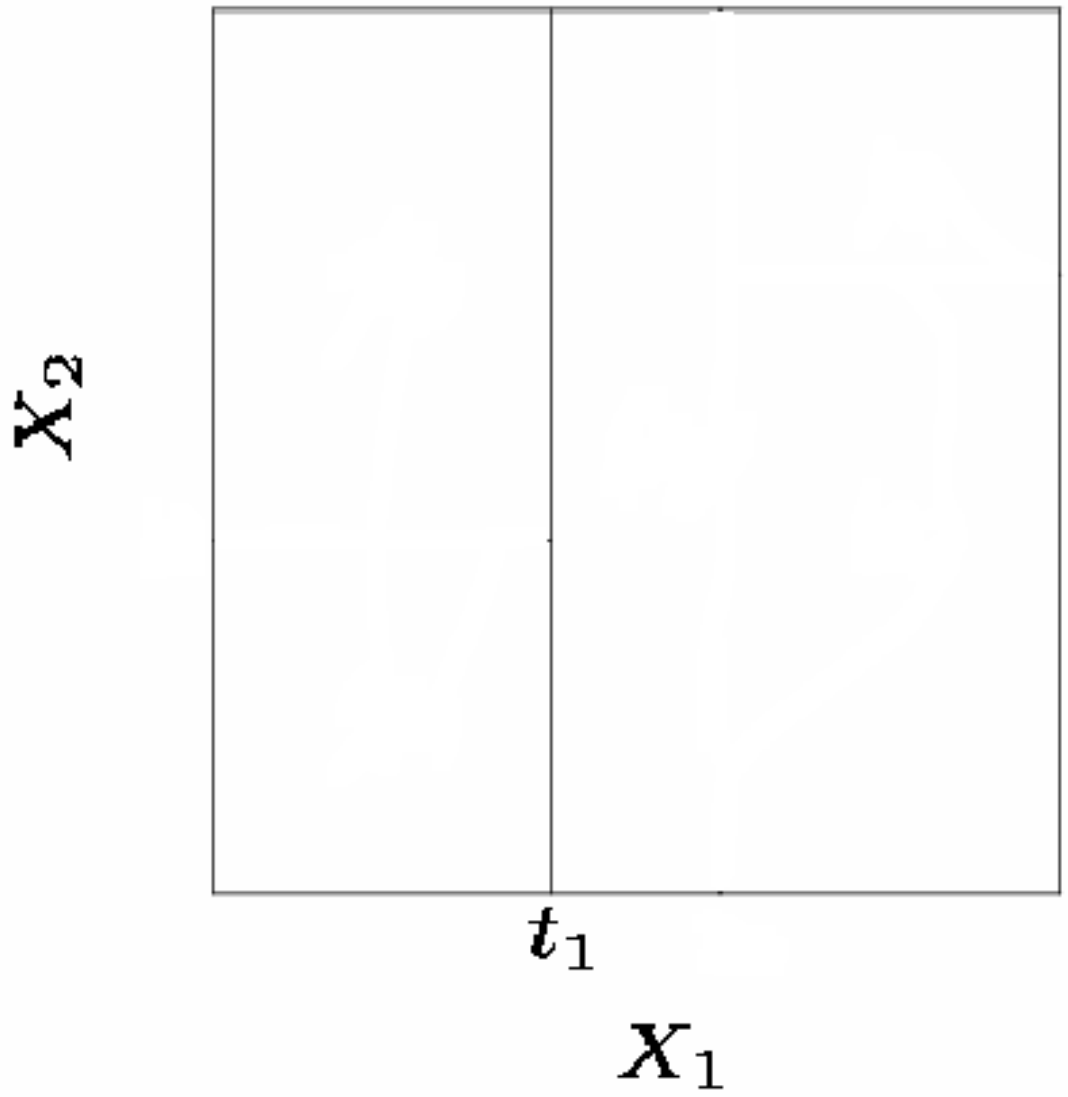
Generally we create the partitions

- by iteratively splitting
- one of the X variables
  into two regions



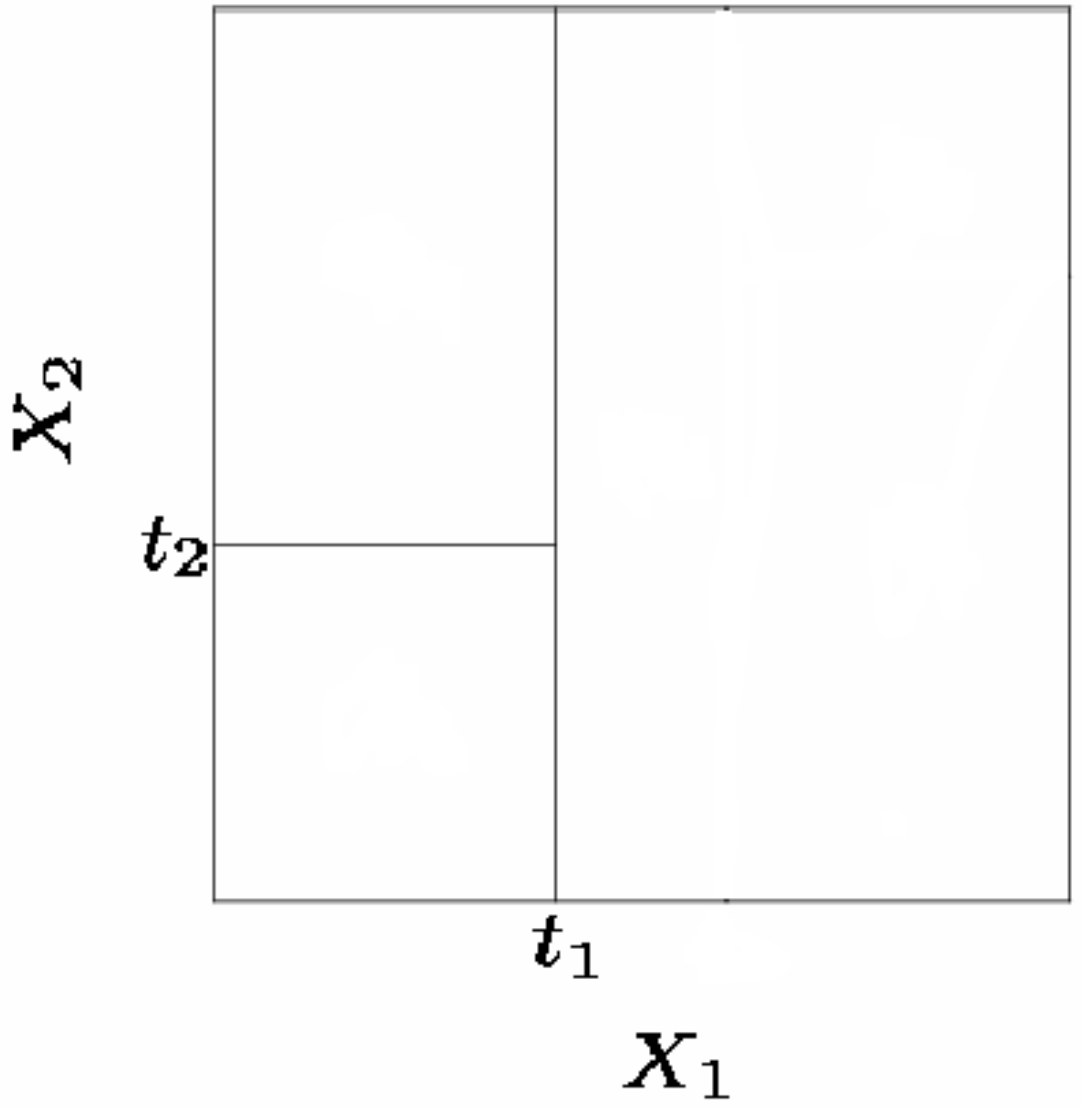
Splitting the X Variable

1. First split on $X_1 = t_1$



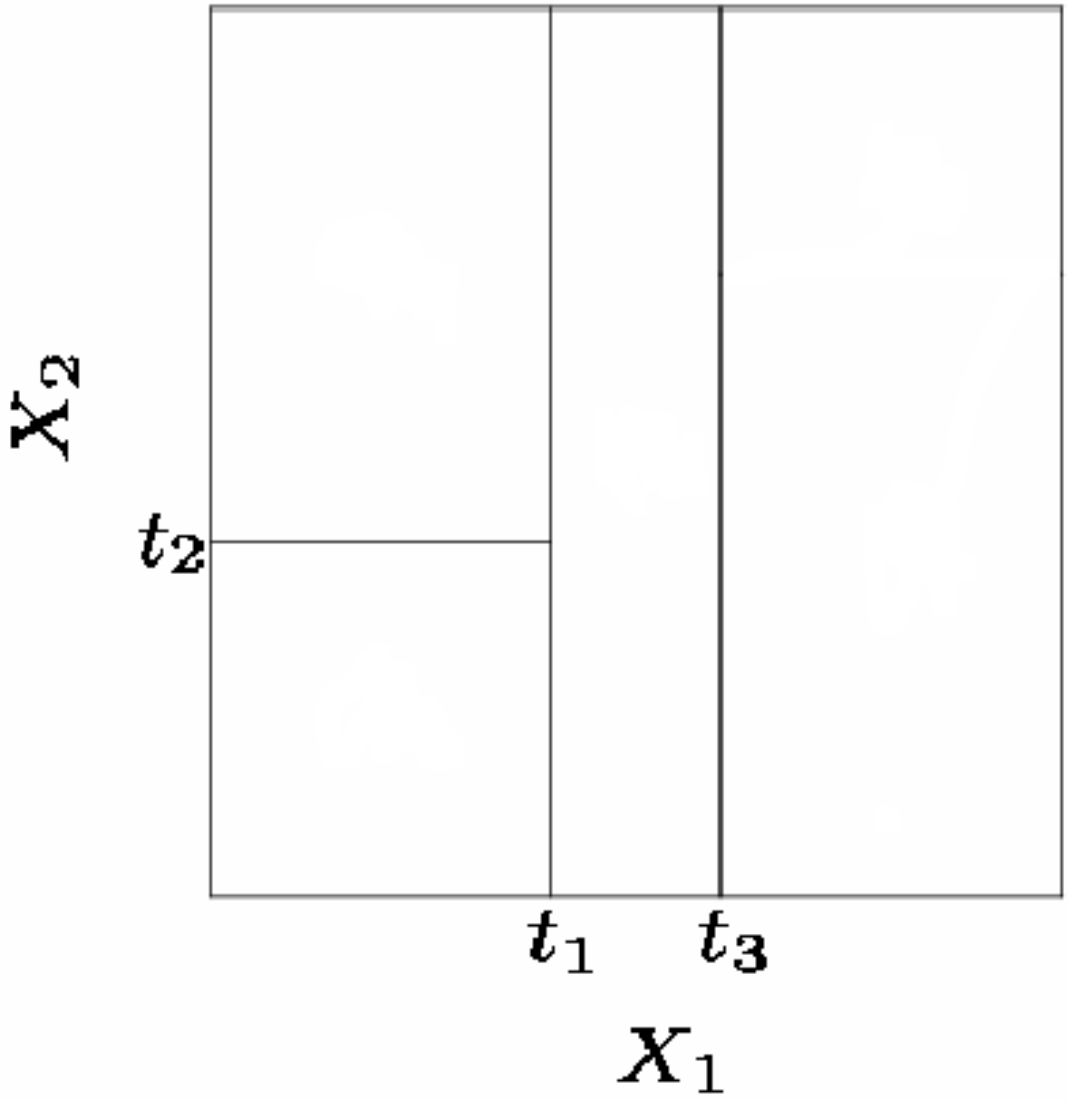
Splitting the X Variable

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



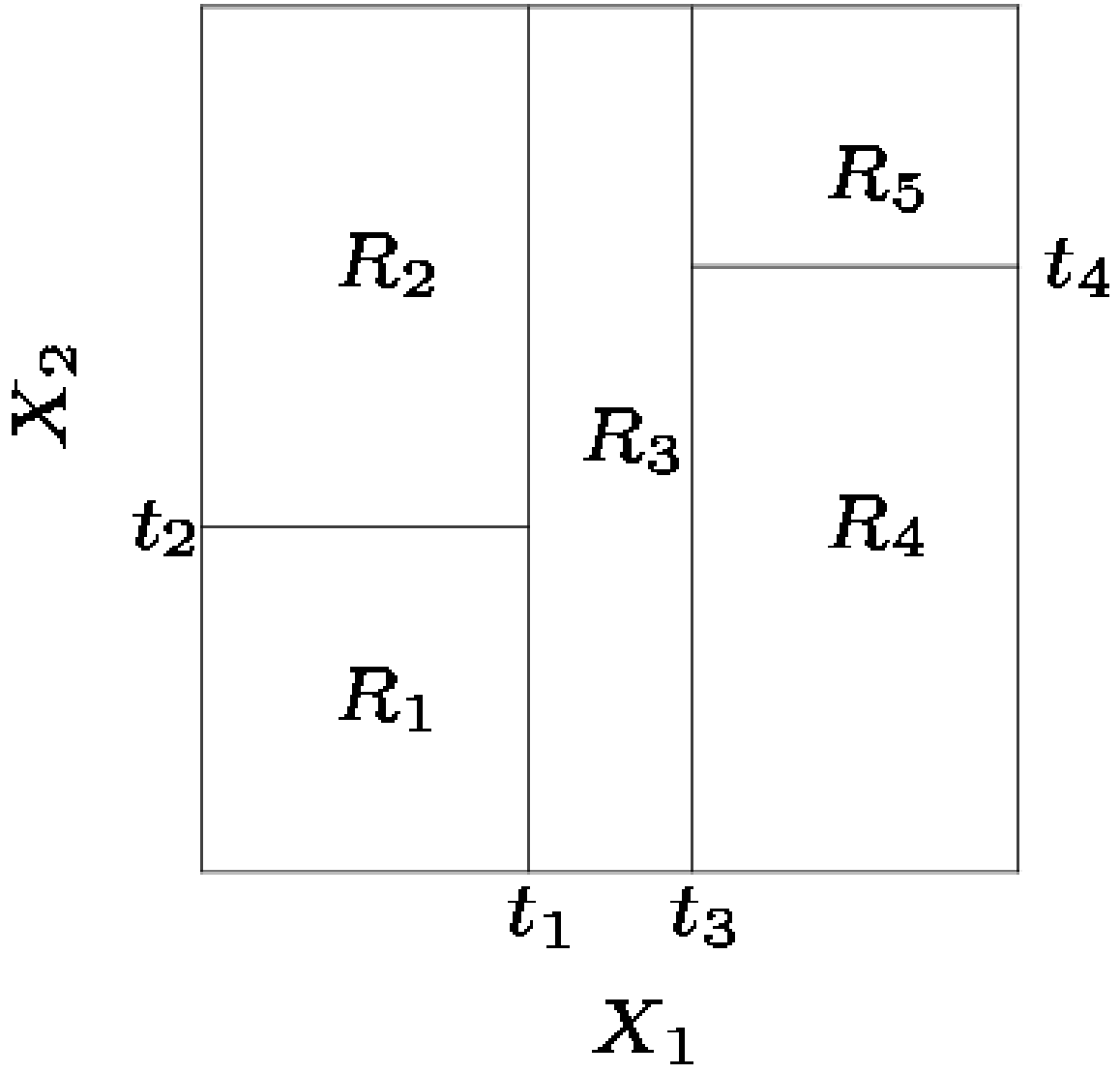
Splitting the X Variable

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 $X_1 = t_3$

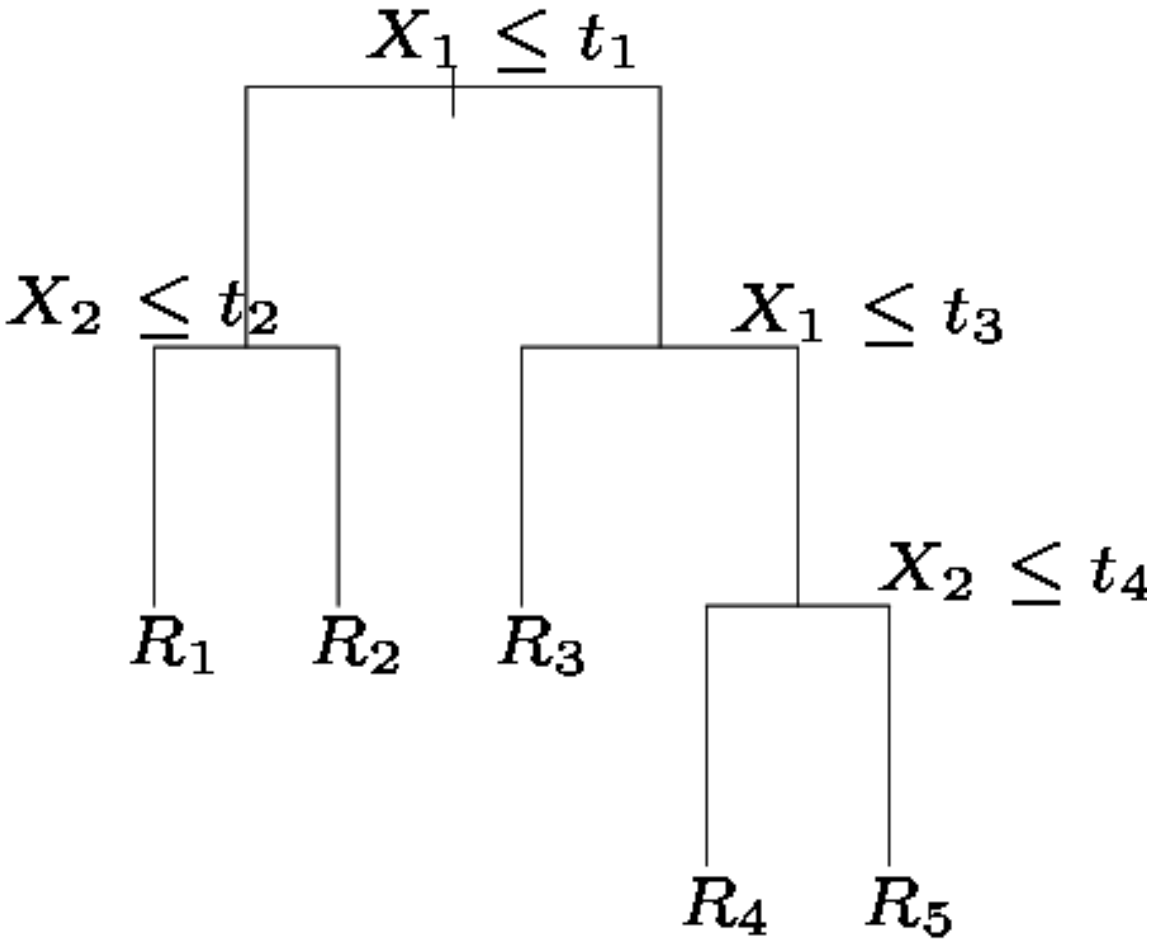


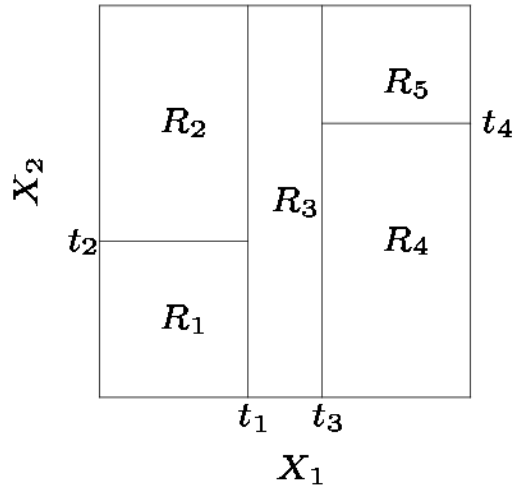
Splitting the X Variable

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





- 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

Example: Baseball Players' Salaries

The predicted Salary

- is the number in each leaf node.

It is the <u>mean</u> of the response

- for the observations that fall there

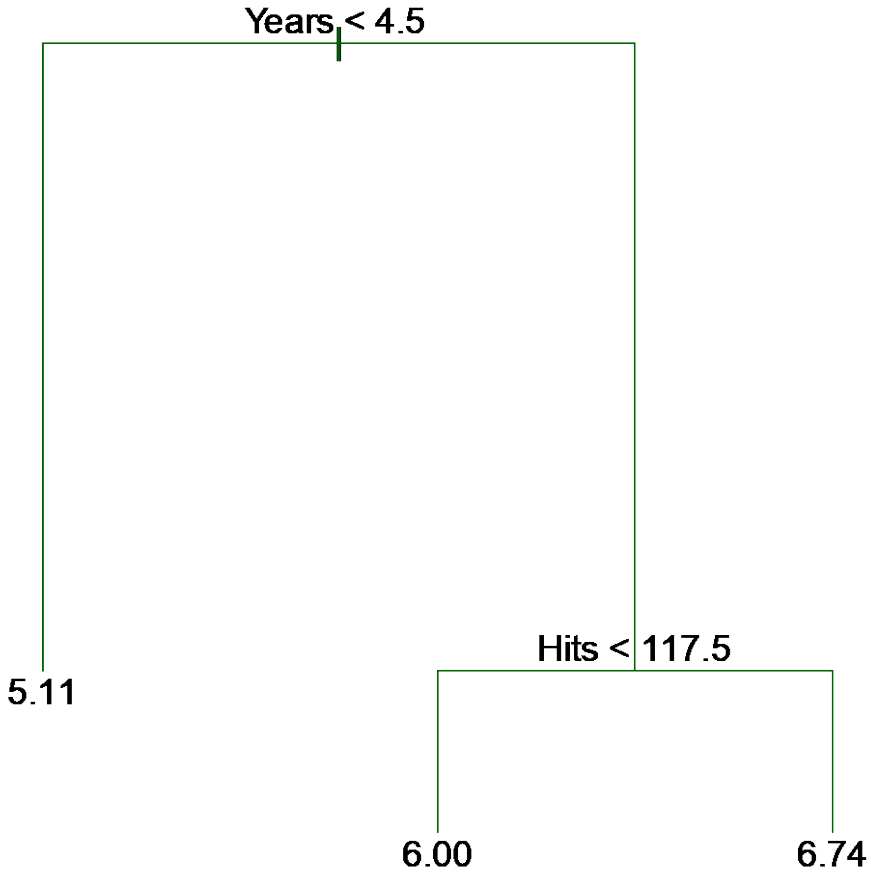
Note that Salary is

- measured in 1000s,
- and log-transformed

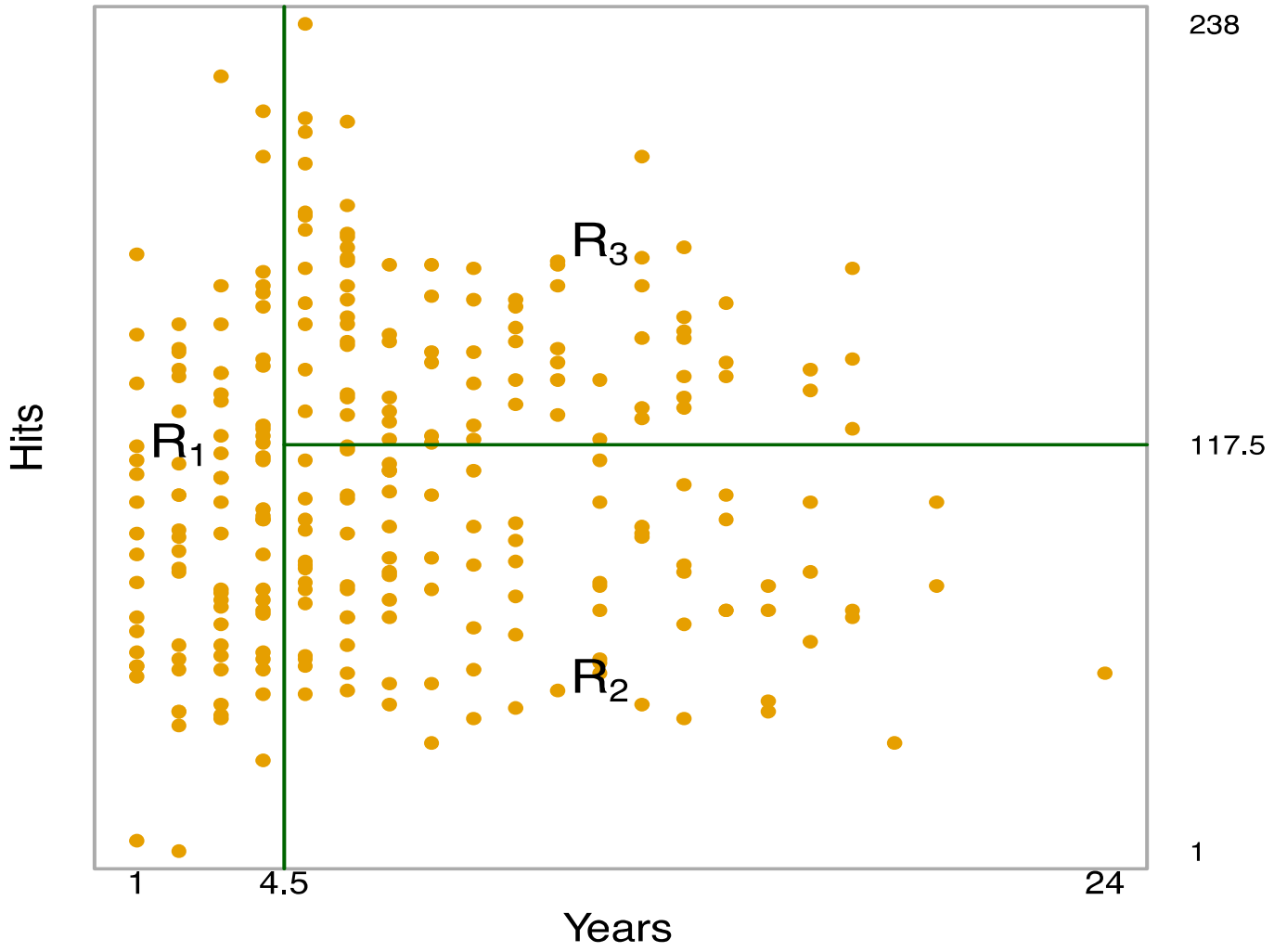
The predicted salary for a player

- 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…



Some Natural Questions

1. **Where to split?**
   - i.e. how do we decide on what regions to use
   - i.e. R_1, R_2,…,R_k
   - or equivalently what tree structure should we use?

2. **What values should we use for $\hat{Y}_1, \hat{Y}_2, \ldots, \hat{Y}_k$?**

1. What values should we use for $\hat{Y}_1, \hat{Y}_2, \ldots, \hat{Y}_k$?

Simple!

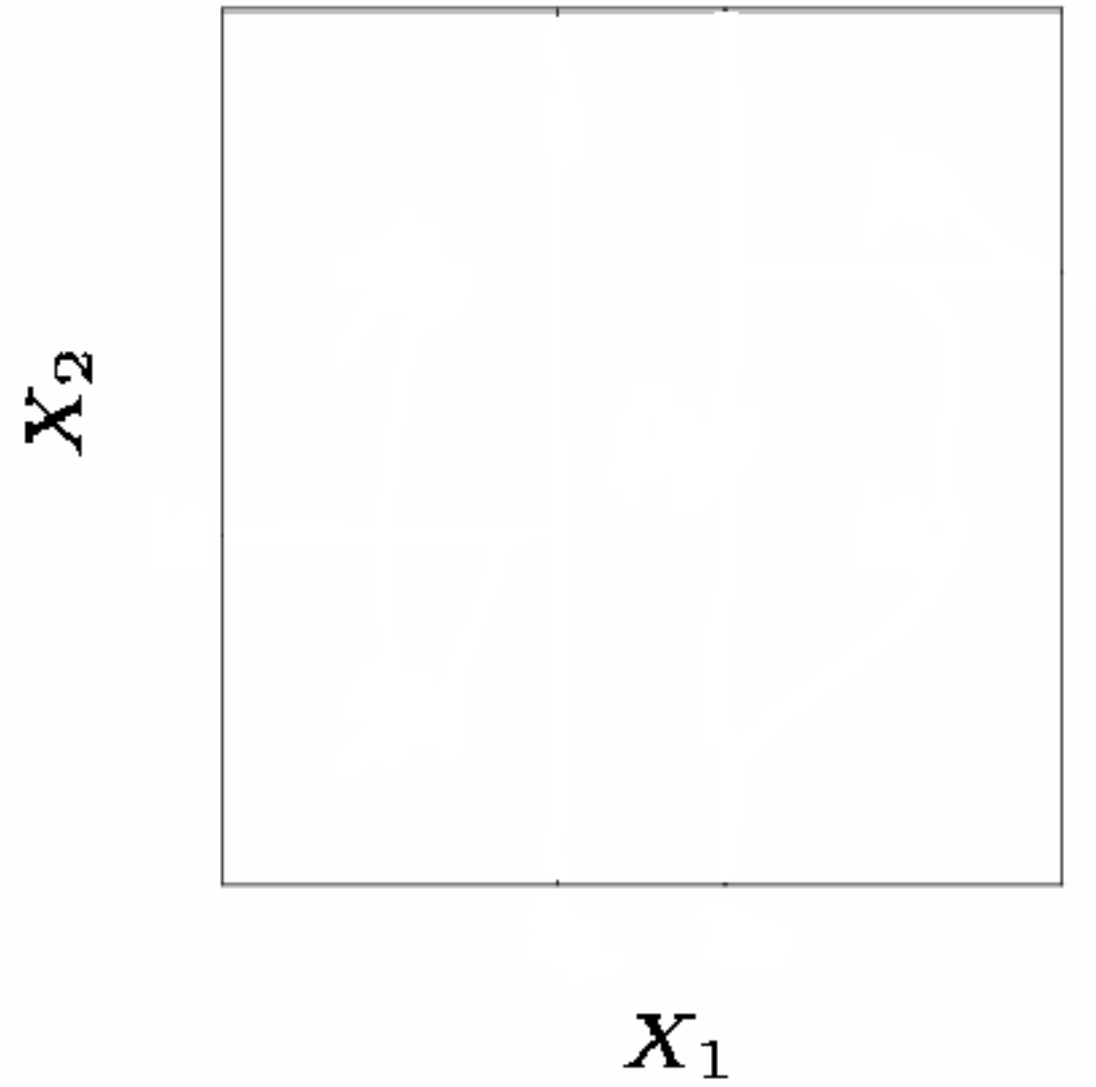
For region R_j,

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

2. Where to Split?

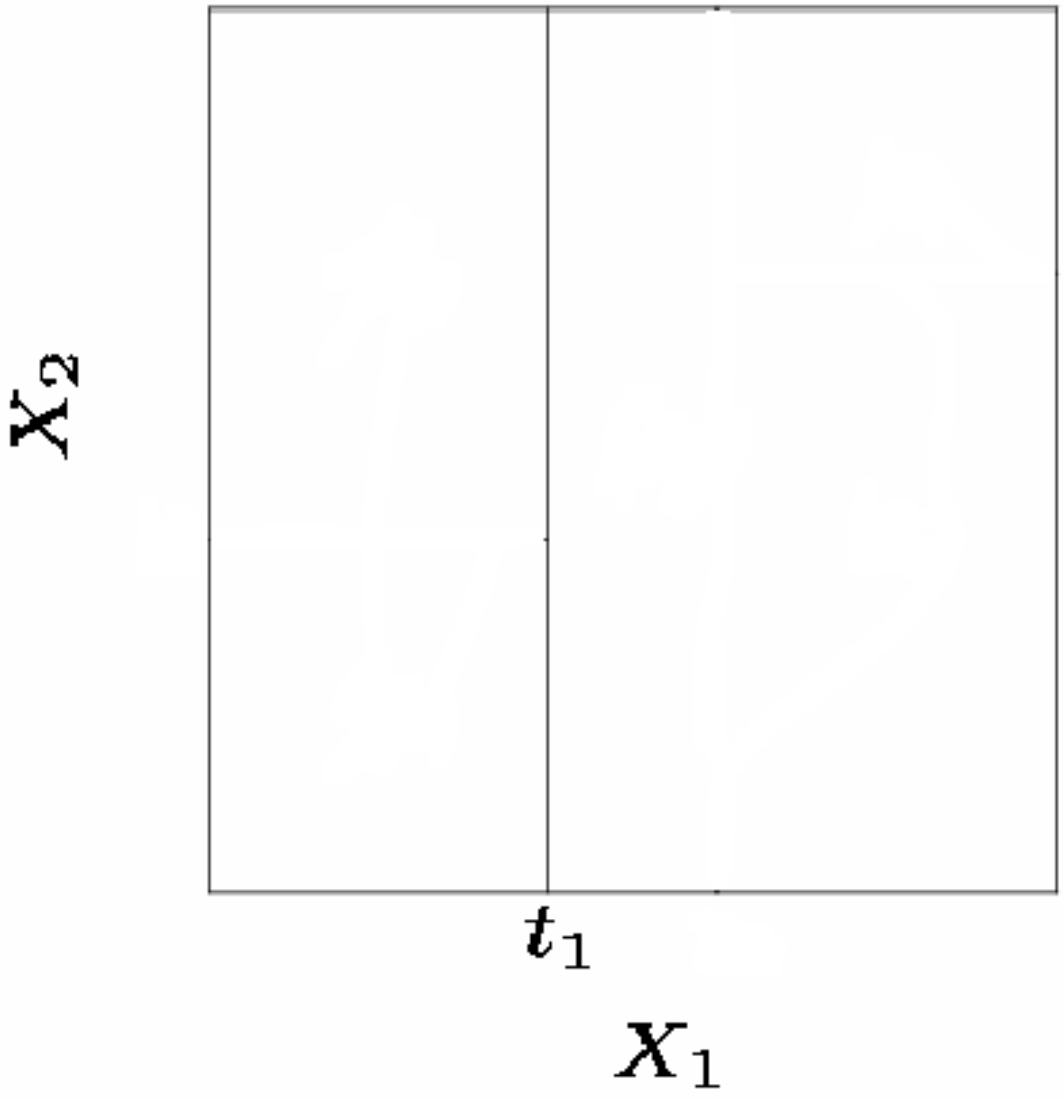
- We consider splitting
  - into two regions, X_j>s and X_j<s
  - for all possible values of s and j.

- We then choose
  - the s and j
  - that results in the lowest MSE
    - on the training data.



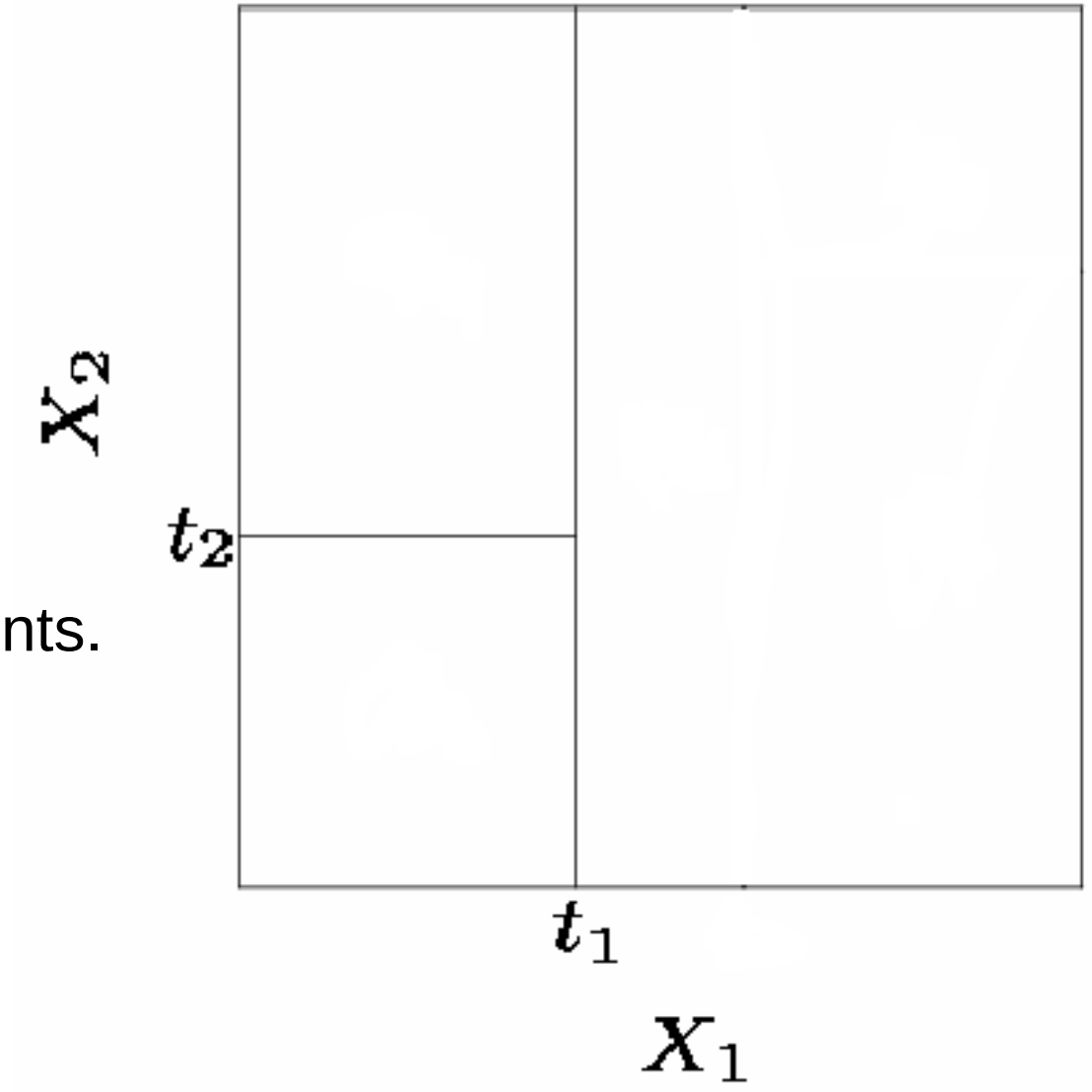
Where to Split?

- Here 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.

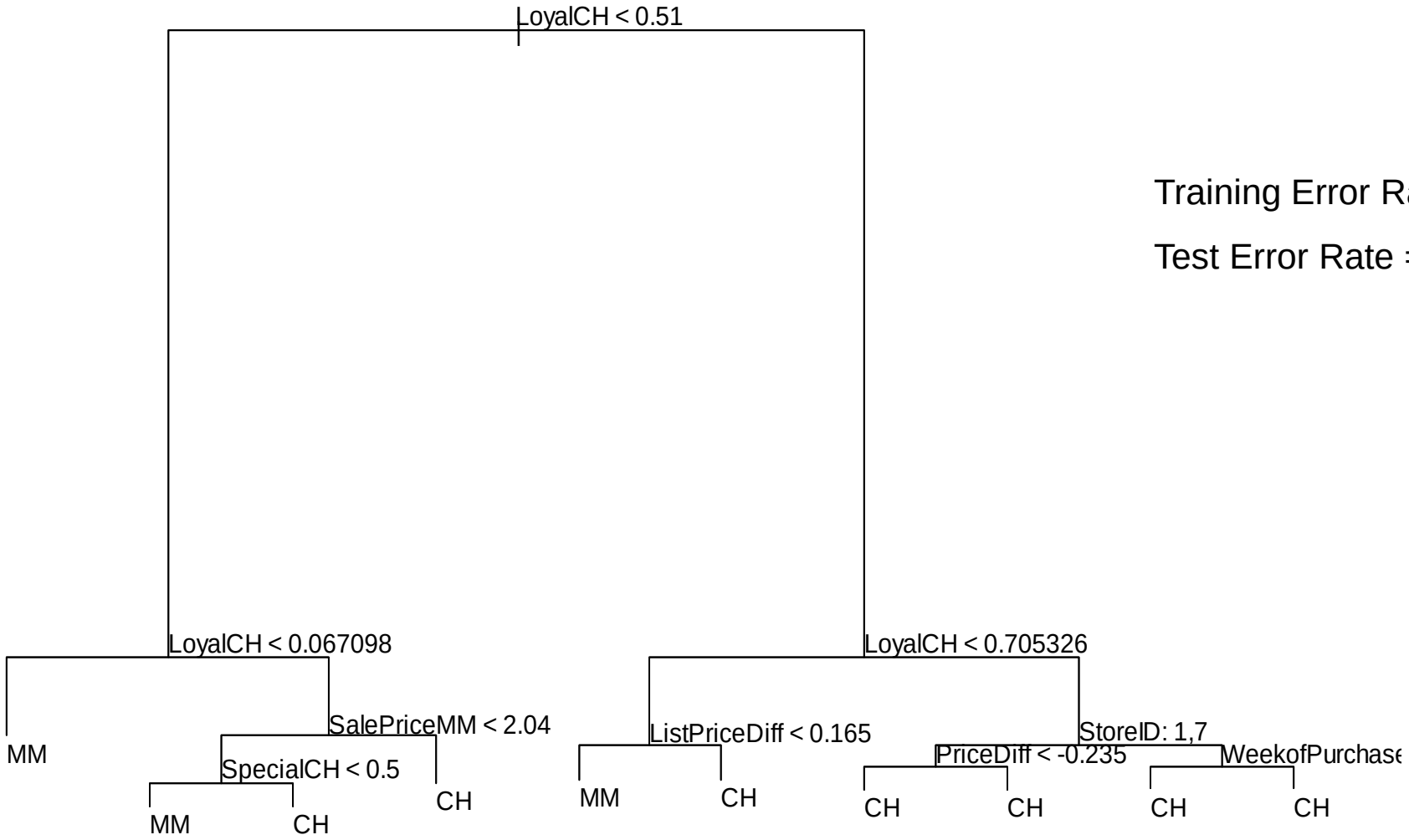


Classification Trees

Growing a Classification Tree

- A classification tree is very similar to a regression tree
  - except that we try to make a prediction for a categorical rather than continuous 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 that minimizing MSE no longer makes sense.
- 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%

Tree Pruning

Improving Tree Accuracy

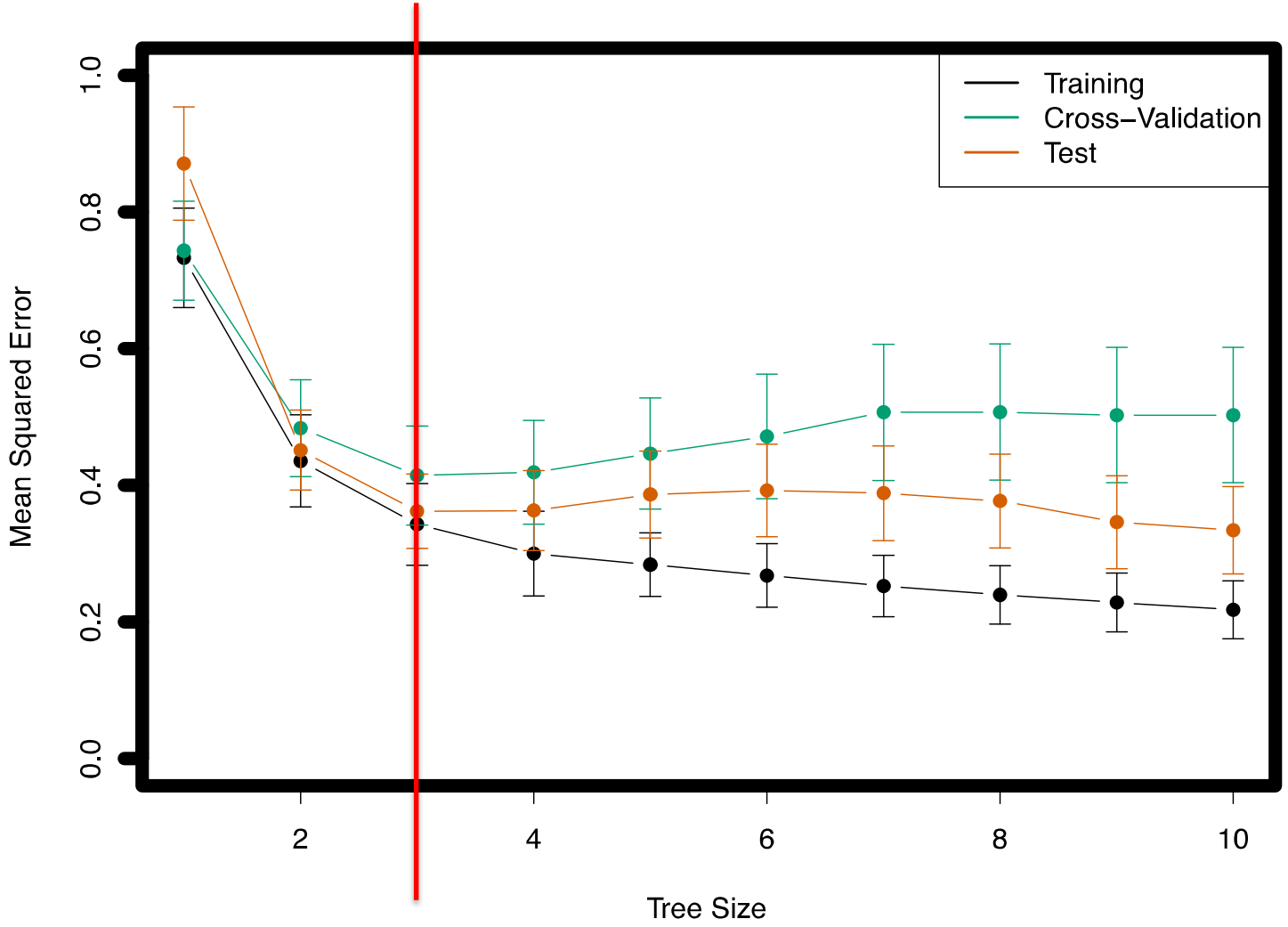
- A large tree (i.e. one with many terminal nodes)
  - may tend to over fit the training data
  - in a similar way
    - to linear regression with too many degrees of freedom or predictors
    - or to neural networks without a weight decay.

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

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

Example: Baseball Players' Salaries

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



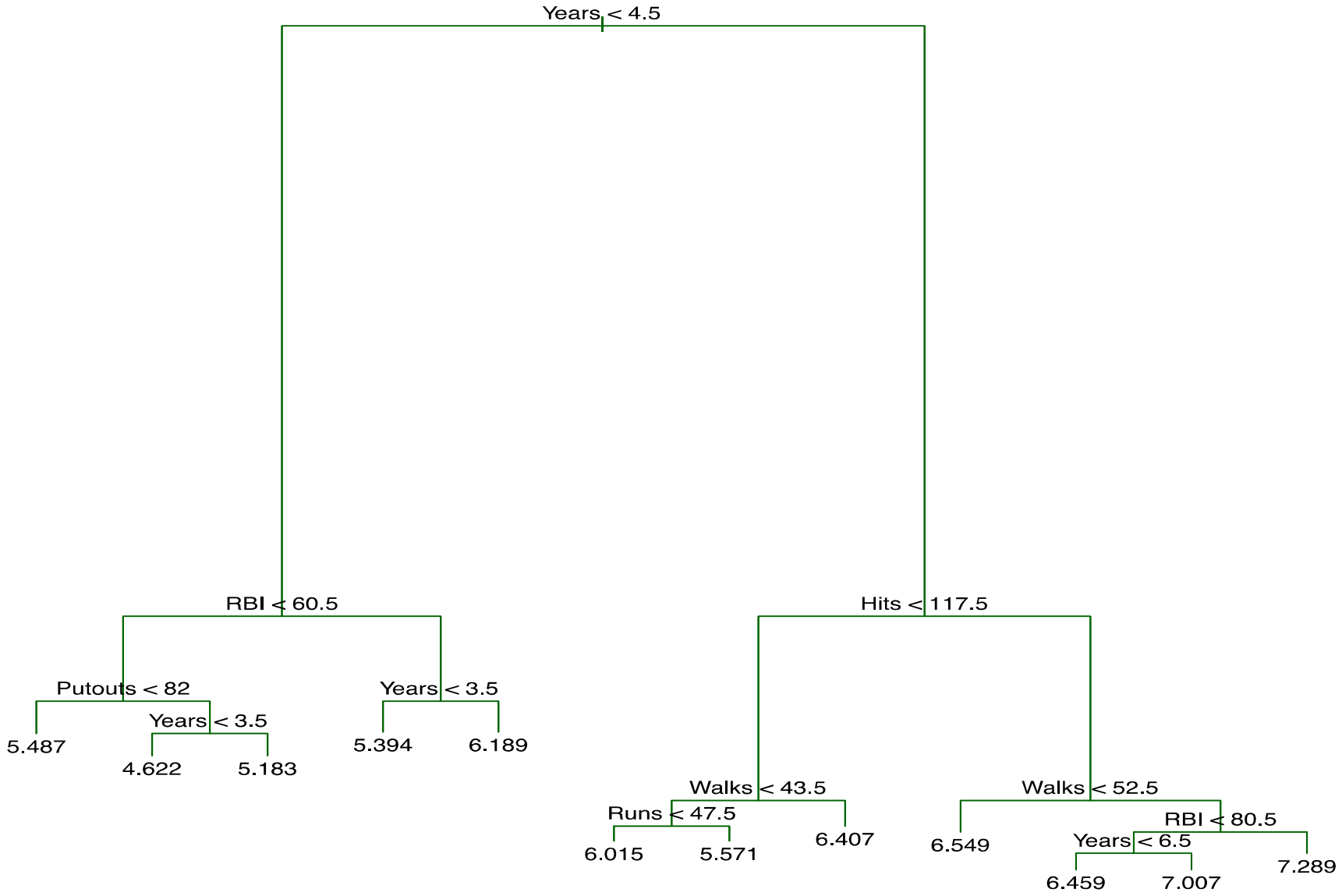
So once again we are looking

<u>At Learning Curves</u>

For lots of models

To choose the best model

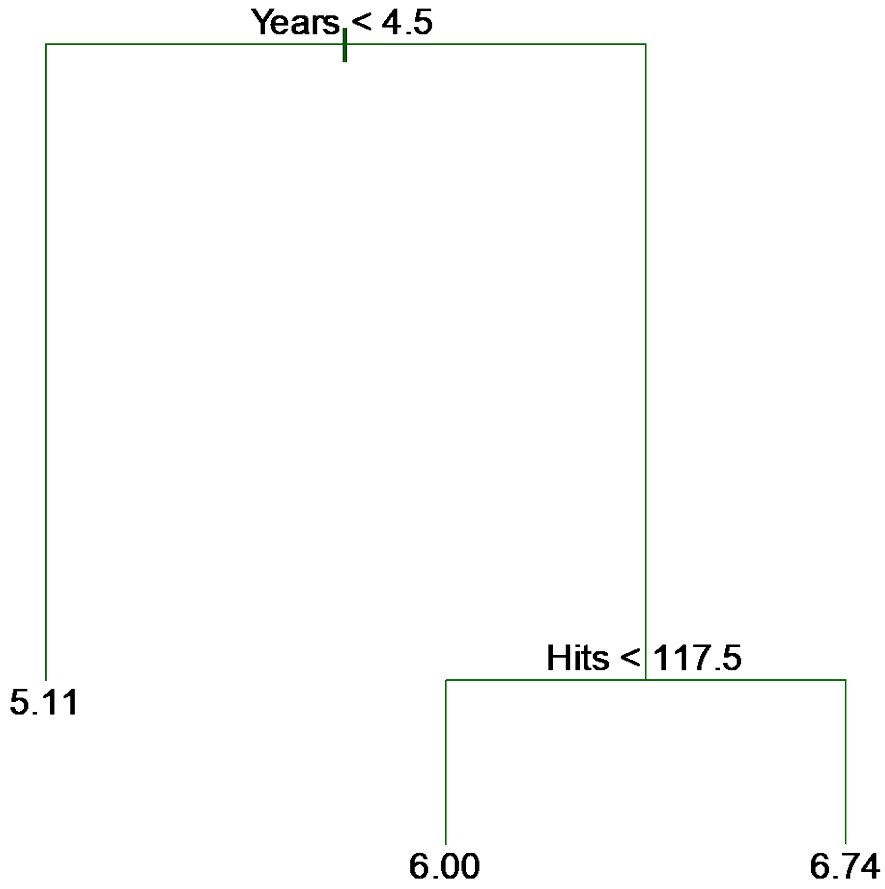
Example: Baseball Players' Salaries



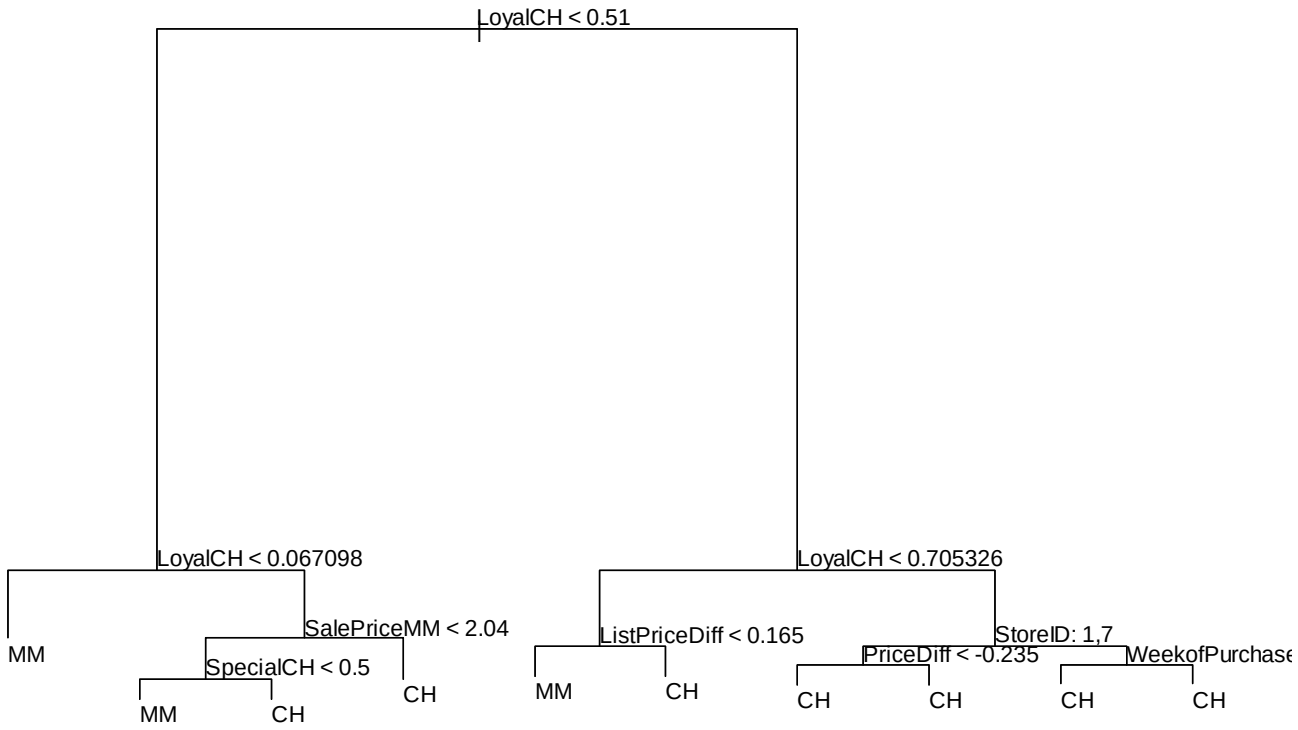
Example: Baseball Players' Salaries

Cross Validation indicated

- that the minimum MSE
- is when the tree size is three
- (i.e. the number of leaf nodes is 3)



Example: Orange Juice Preference

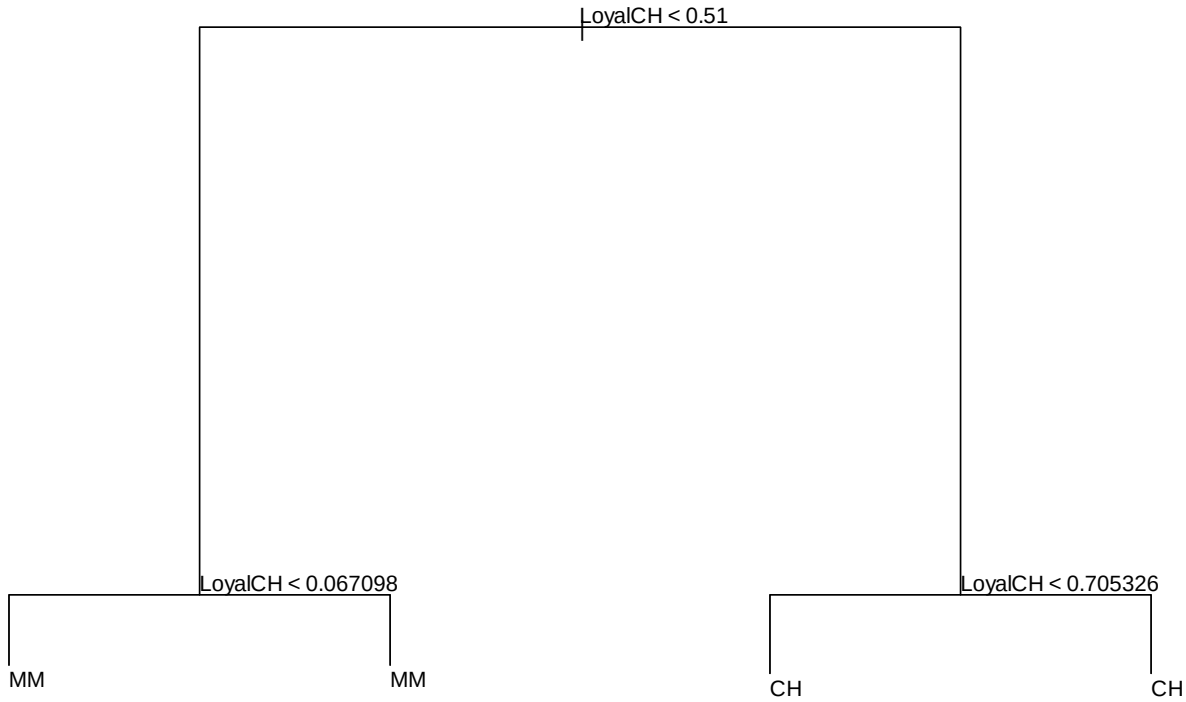


Full Tree Training Error Rate = 14.75%

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
- would outperform classical approaches

Trees vs. Linear Model: Classification Example

Top row:

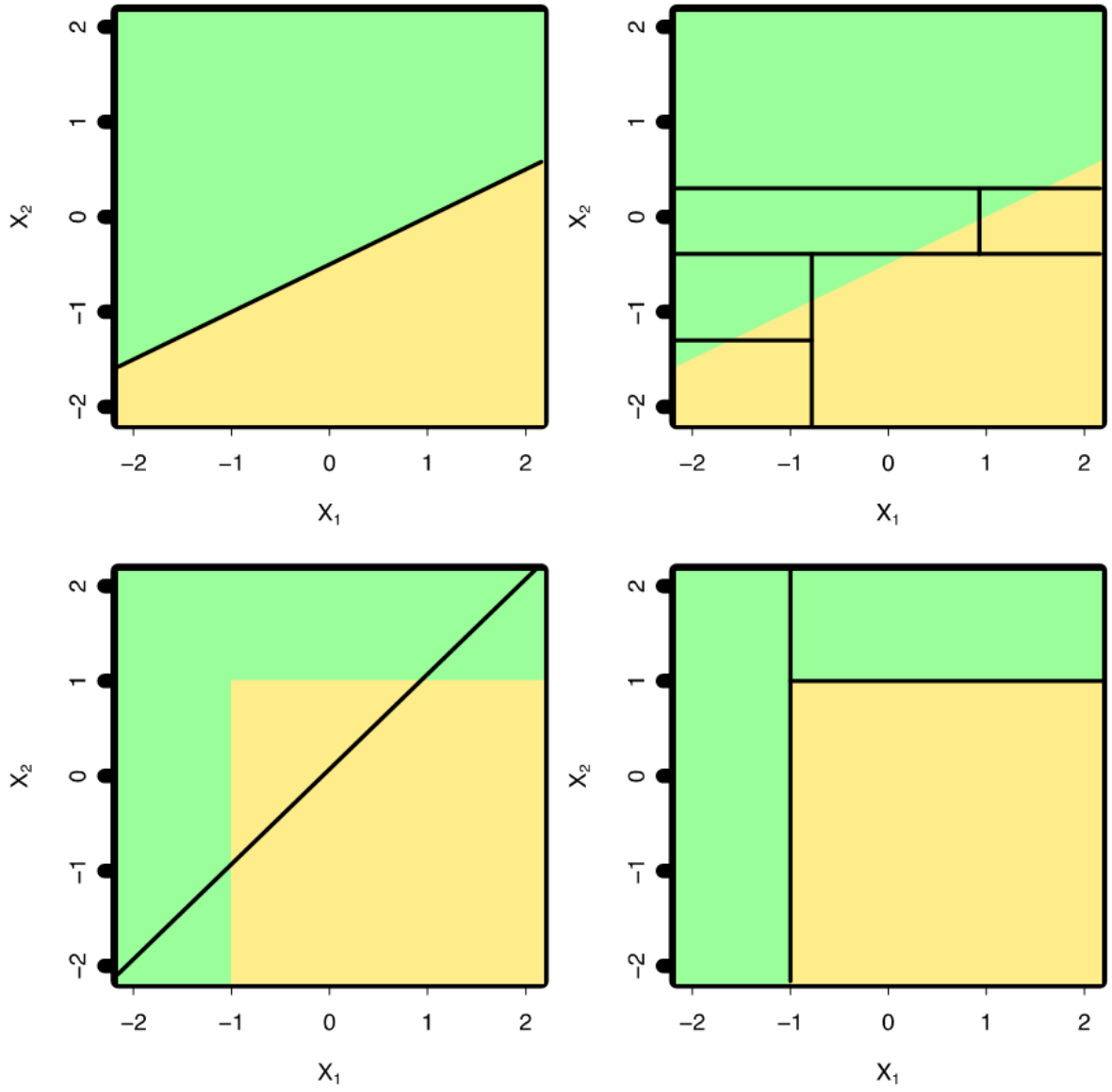
the true decision boundary is linear

- 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 explain to people
  (probably even easier than linear regression)
- Trees can be plotted graphically,
  and are easily interpreted even by non-expert
- They work fine on both
  classification and regression problems

Cons:

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

So Next We'll look into Bagging, Random Forest and Boosting

Bagging and Random Forests

Outline

Bagging

- Bootstrapping
- Bagging for Regression Trees
- Bagging for Classification Trees
- Out-of-Bag Error Estimation
- Variable Importance: Relative Influence Plots

Random Forests

Bagging

Problem!

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

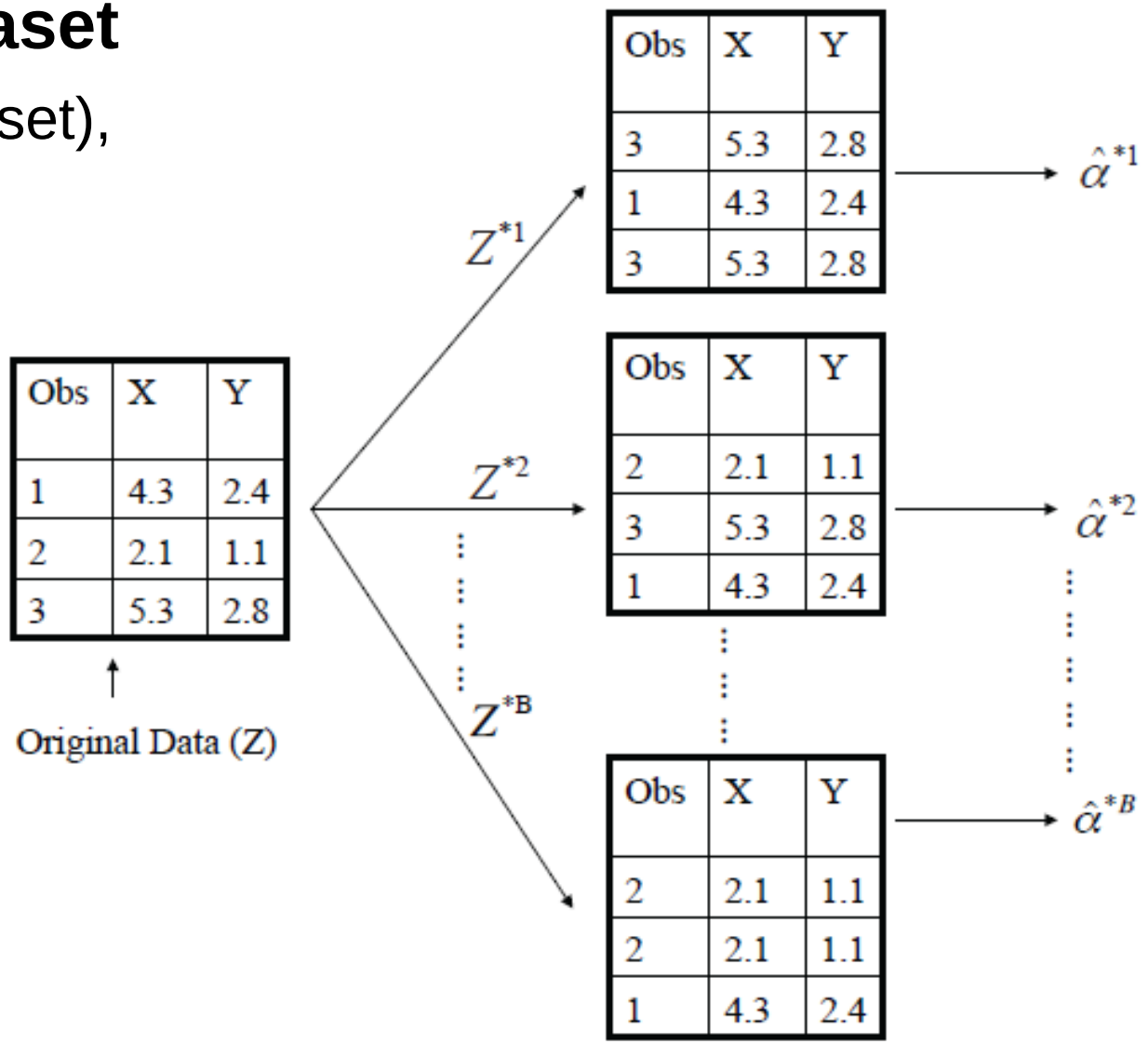
To solve this problem,

we can use bagging (*b*ootstrap *agg*regat*ing*).

Bootstrapping is simple!

Resampling of the observed dataset

- (and of equal size to the observed dataset),
- each of which is obtained
  - by random sampling
  - with replacement
- from the original dataset.



What is bagging?

Bagging is an extremely powerful idea based on two things:

- Averaging: reduces variance!
- Bootstrapping: plenty of training datasets!

Why does averaging reduces variance?

- Averaging a set of observations reduces variance.

Recall that given a set of n independent observations Z_1, …, Z_n,

- each with variance σ^2,
- the variance of the mean $\overline{Z}$ of the observations is given by σ^2/n

How does bagging work?

Generate B different bootstrapped training datasets

Train the statistical learning method

- on each of the B training datasets,
- and obtain the prediction

For prediction:

- Regression:
  average all predictions from all B trees
- Classification:
  majority vote among all B trees

Bagging for Regression Trees

Construct B regression trees

- using B bootstrapped training datasets

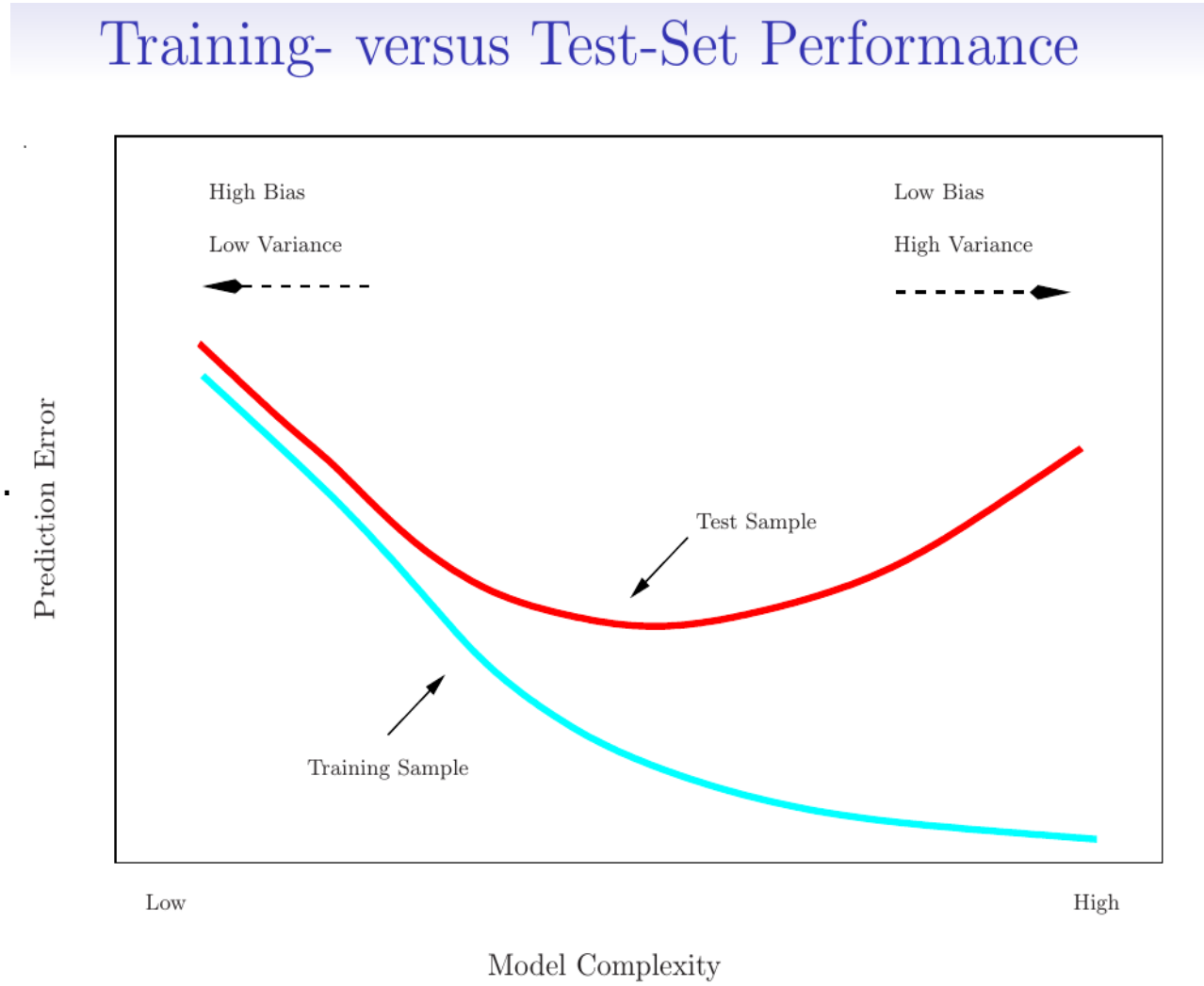
Average the resulting predictions

Note: These trees are not pruned,

- so each individual tree has high variance but low bias.

Averaging these trees reduces variance,

- and thus we end up
- lowering both variance and bias ☺



Bagging for Classification Trees

Construct B regression trees

- using B bootstrapped 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

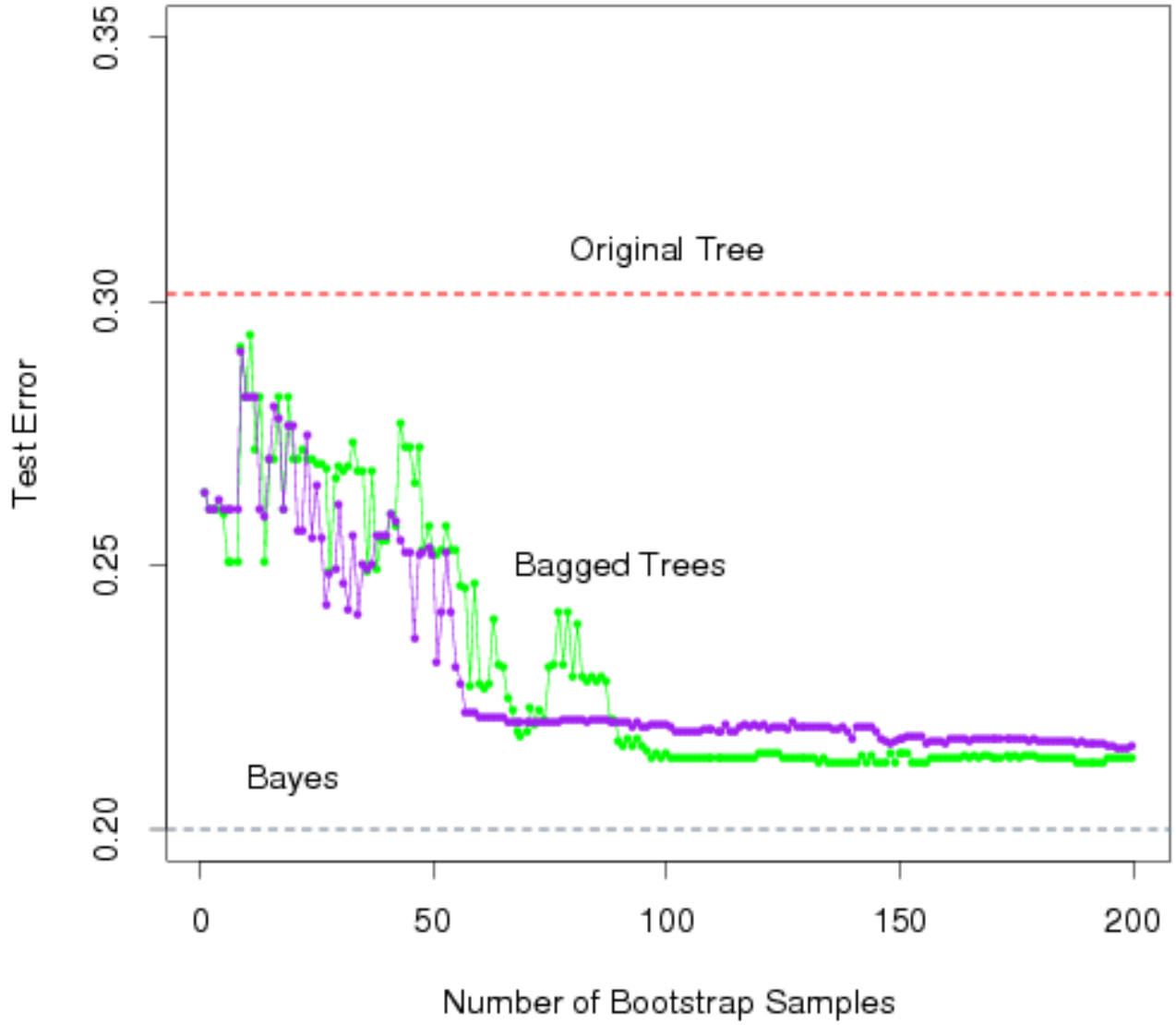
- averaging the probability estimates.

Both do far better

- than a single tree (dashed red)

And get close to

- the Bayes error rate (dashed grey).



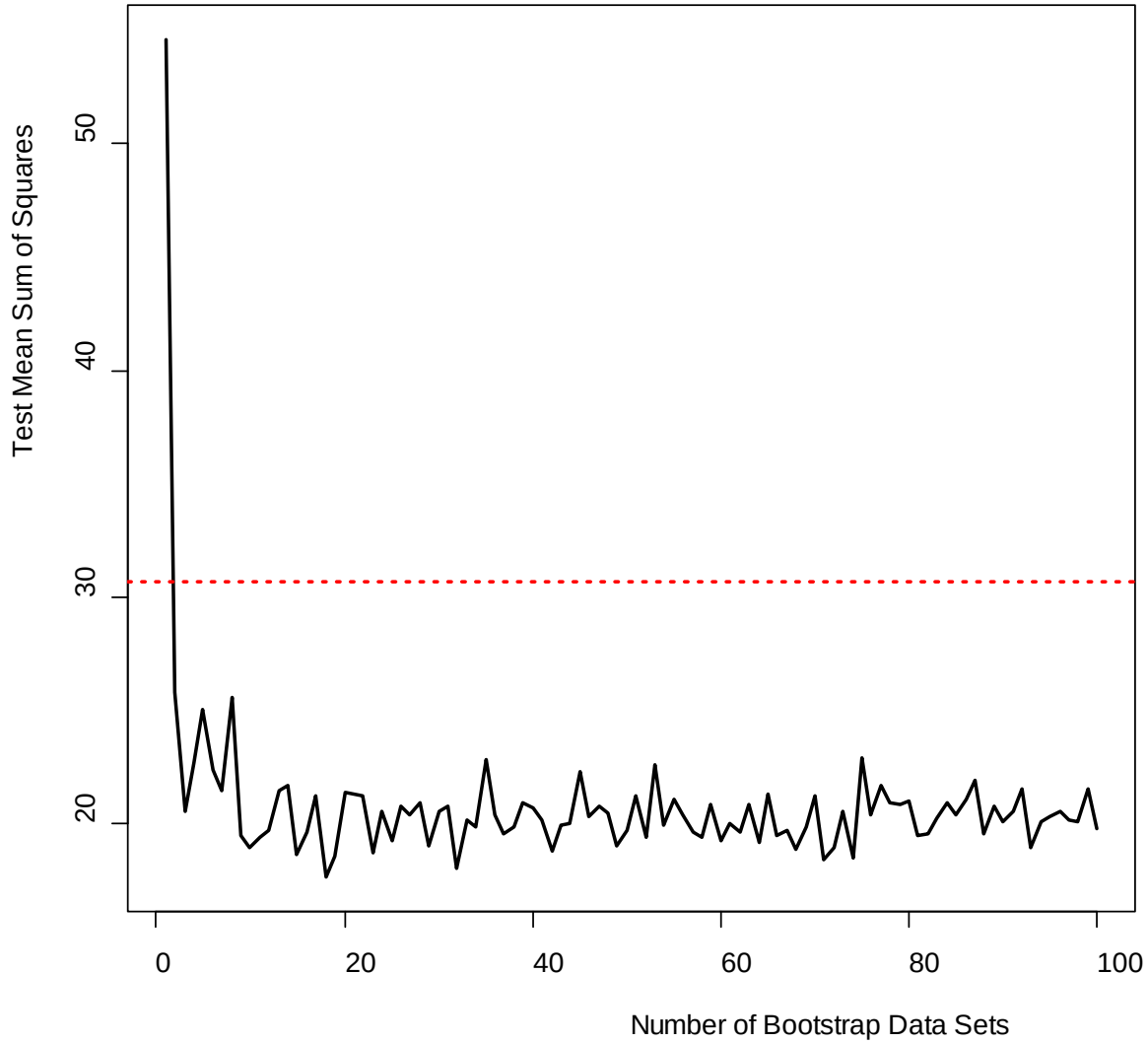
Example 1: Housing Data

The red line represents

- the test mean sum of squares
- using a single tree.

The black line corresponds to

- the bagging error rate



Example 2: Car Seat Data

The red line represents

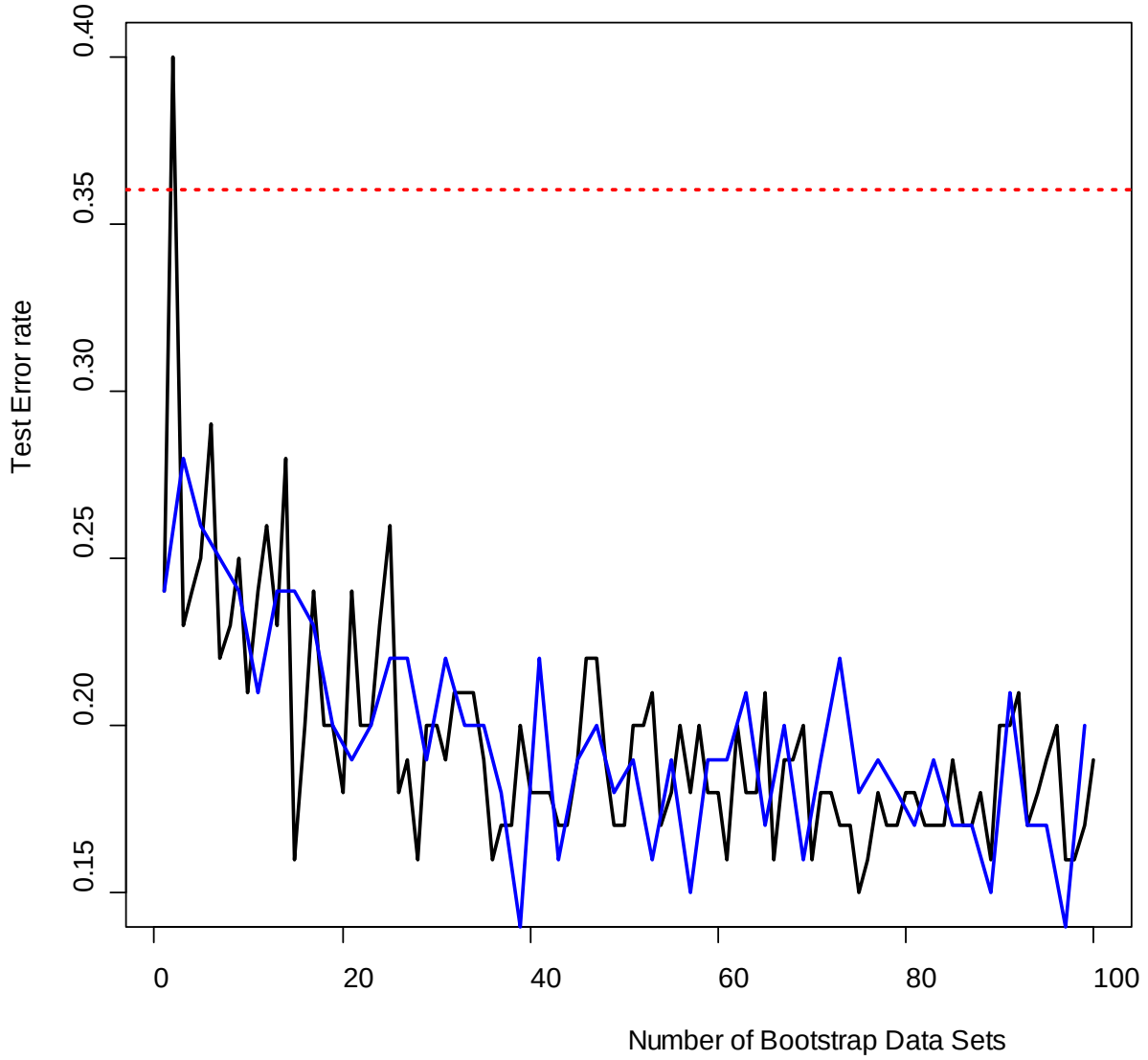
- the test error rate
- using a single tree.

The black line corresponds to

- the bagging error rate
- using majority vote

While the blue line

- averages the probabilities.



Out-of-Bag (oob) Error Estimation

Since bootstrapping involves

- random selection of subsets of observations
- to build a training data set,

then the remaining non-selected part

- could be the testing data.

On average, each bagged tree

- makes use of around 2/3 of the observations,

so we end up having 1/3 of the observations

- used for testing

Variable Importance Measure

Bagging typically improves

- the accuracy over prediction using a single tree,
- but it is now hard to interpret the model!

We have hundreds of trees, and

- it is no longer clear which variables
- are most important to the procedure

Thus bagging

- improves prediction accuracy
- at the expense of interpretability

But, we can still get an overall summary of

- the importance of each predictor
- using Relative Influence Plots

Relative Influence Plots

How do we decide which variables are most useful in predicting the response?

- We can compute something called relative influence plots.
- These plots give a score for each variable.
- These scores represents the decrease in MSE when splitting on a particular variable
- A number close to zero indicates the variable is not important and could be dropped.
- The larger the score the more influence the variable has.

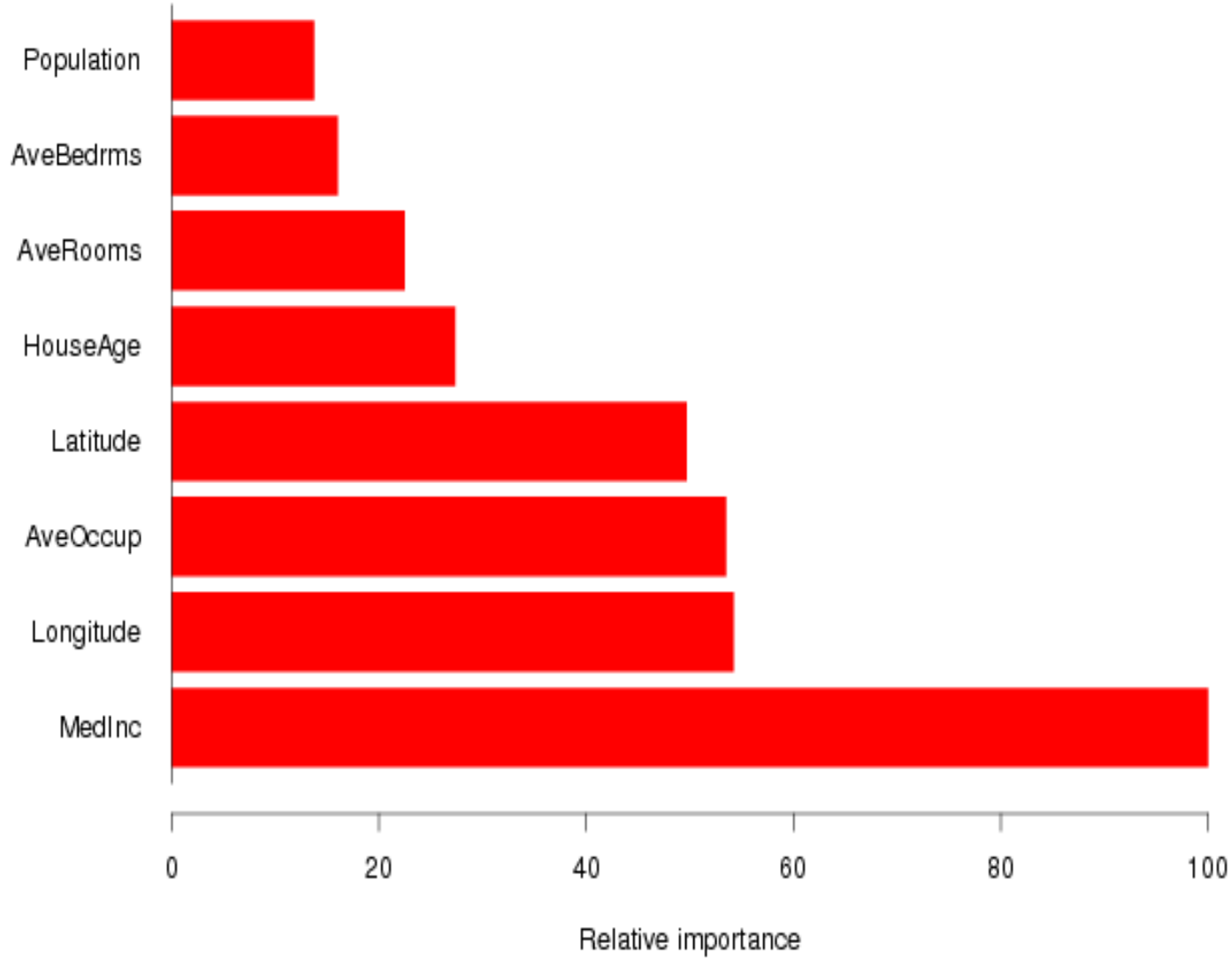
Example: Housing Data

Median Income is

- by far the most important variable.

Longitude, Latitude and Average occupancy

- are the next most important.



Random Forests

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,

but when building these trees,

- each time a split in a tree is considered,
- a random sample of *m* predictors is chosen as split candidates
- from the full set of *p* predictors (Usually $m \approx \sqrt{p}$)

Why are we considering a random sample of m predictors instead of all p predictors for 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.