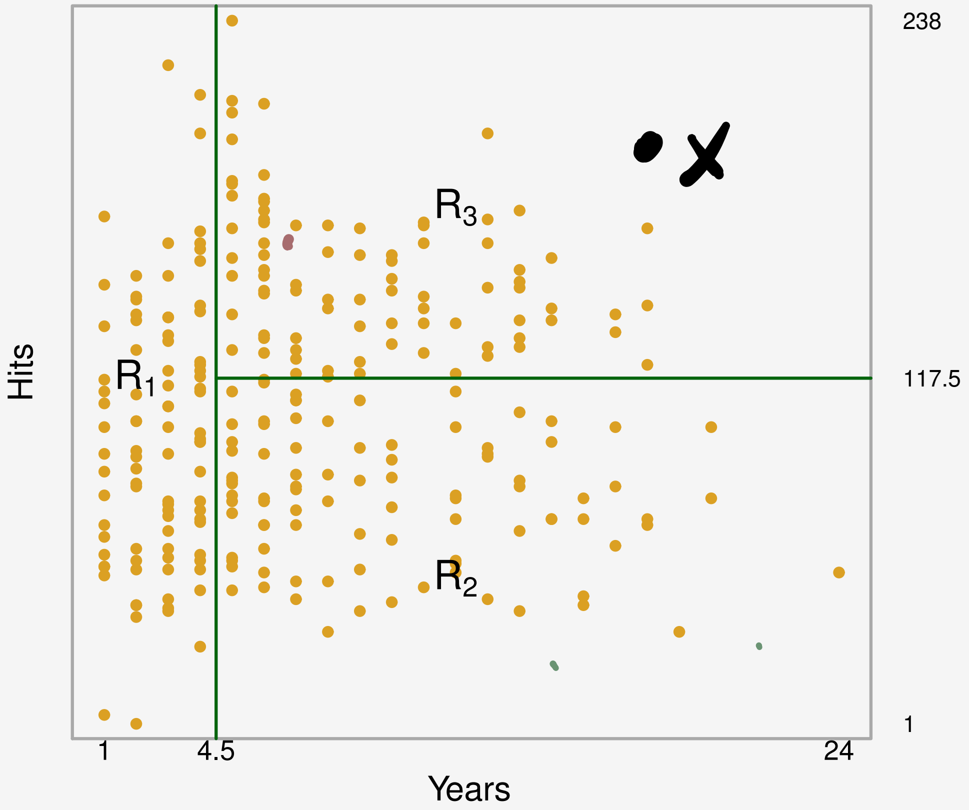
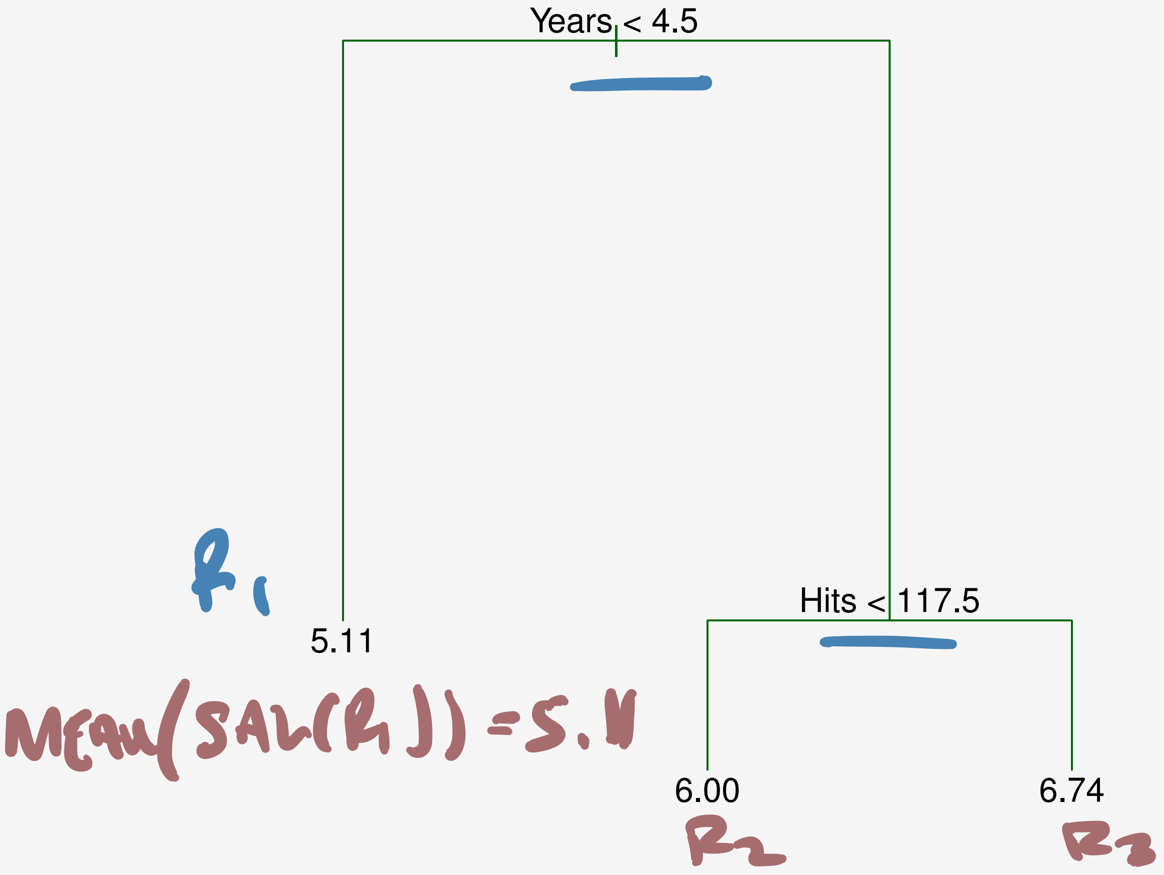
Bootstrapped Aggregation and Random Forests

Bonus: Regression Trees

- Suppose you want to PREDICT the salary of a MLB player based on two features: how many hits they average a year and how many years they've been in the league





Bonus: Regression Trees

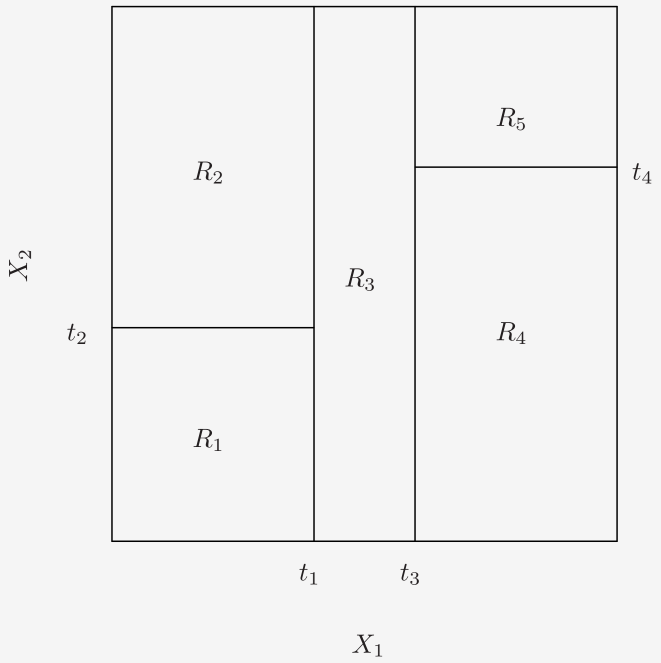
- Suppose you want to PREDICT the salary of a MLB player based on two features: how many hits they average a year and how many years they've been in the league

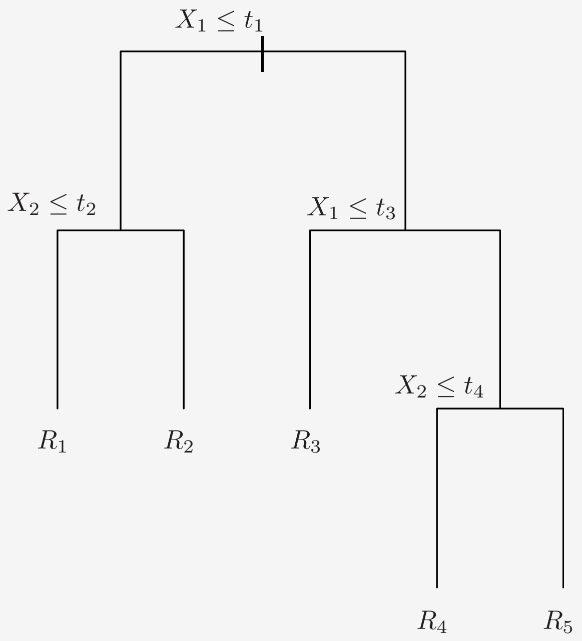
- Perform the usual binary splitting by feature

- Instead of classification, predict the response based on average response in leaf node

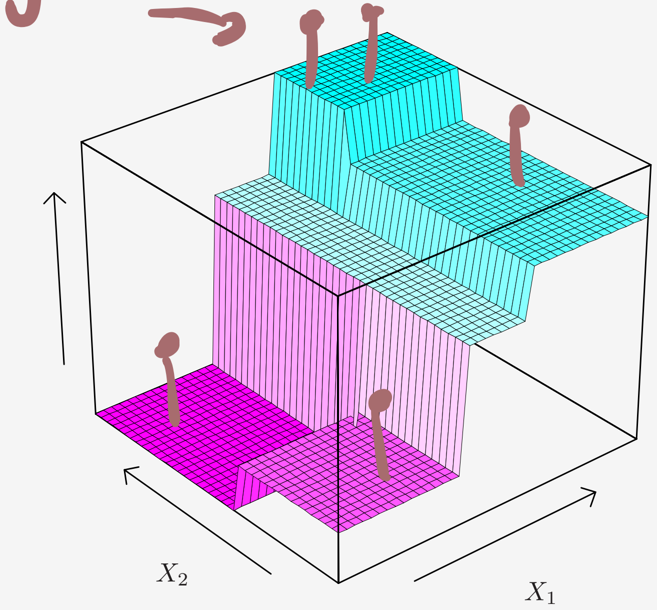
Bonus: Regression Trees

- Instead of classification, predict the response based on average response in leaf node





$(y_i - \bar{y})^2$



Bonus: Regression Trees

#REGIONS

- What measure do we use to decide which feature to split on?

- The goal is to find boxes that minimize the $RSS = \sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j}\right)^2$

Draw

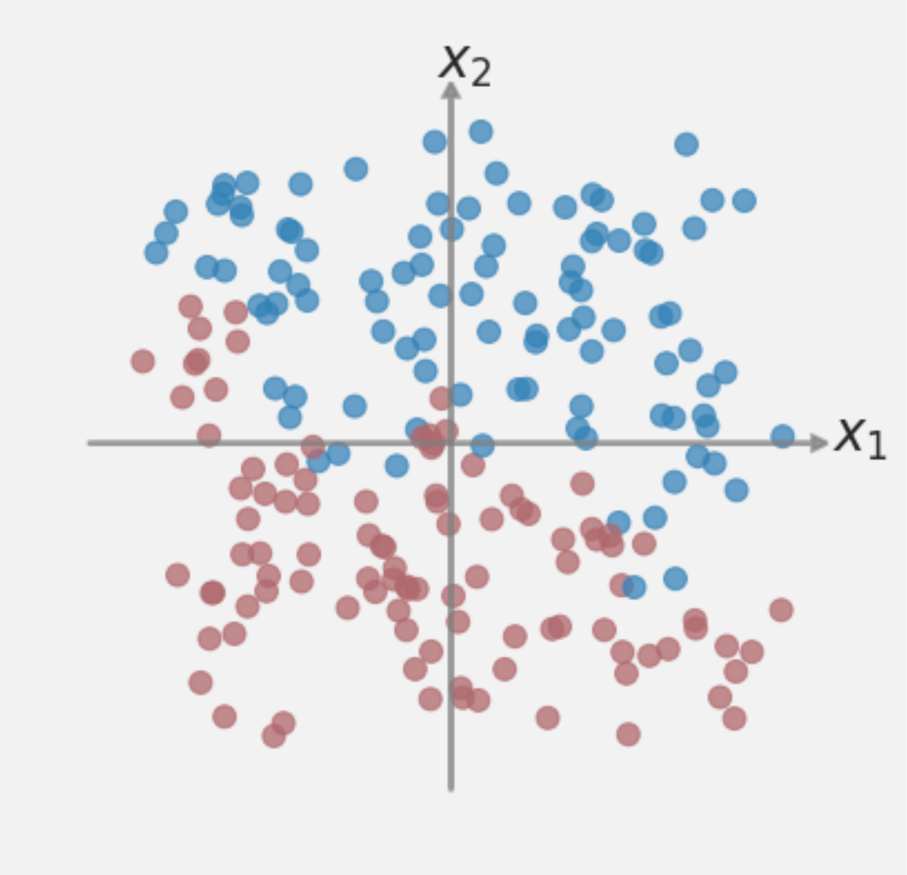
- Consider splitting the data on a node into two boxes. Choose feature and split to minimize

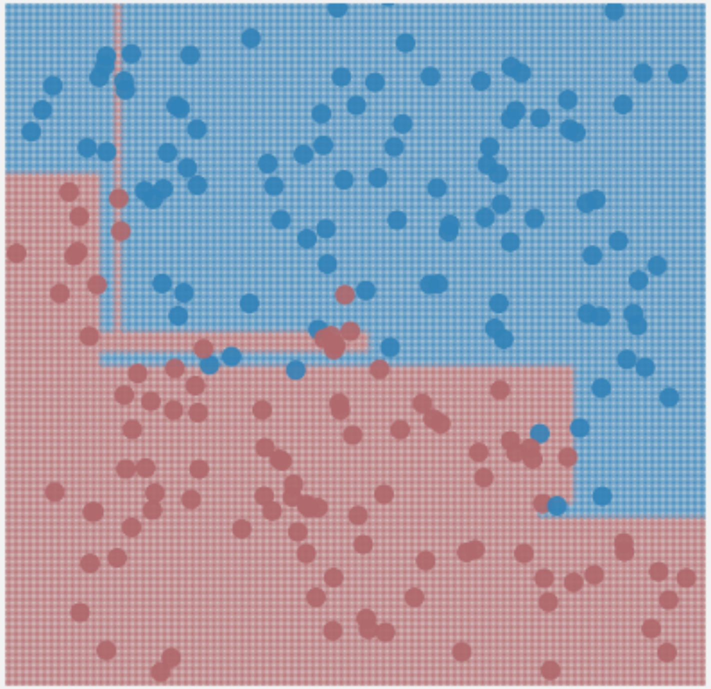
$$\sum_{i:\ x_i \in R_1(j,s)} \left(y_i - \hat{y}_{R_1}\right)^2 + \sum_{i:\ x_i \in R_2(j,s)} \left(y_i - \hat{y}_{R_2}\right)^2$$

where $R_1(j,s) = \{X \mid X_j < s\}$ and $R_2(j,s) = \{X \mid X_j \geq s\}$

Previously on CSCI 4622

Last time we saw that full Decision Tree Classifiers tended to overfit





Previously on CSCI 4622

Last time we saw that full Decision Tree Classifiers tended to overfit

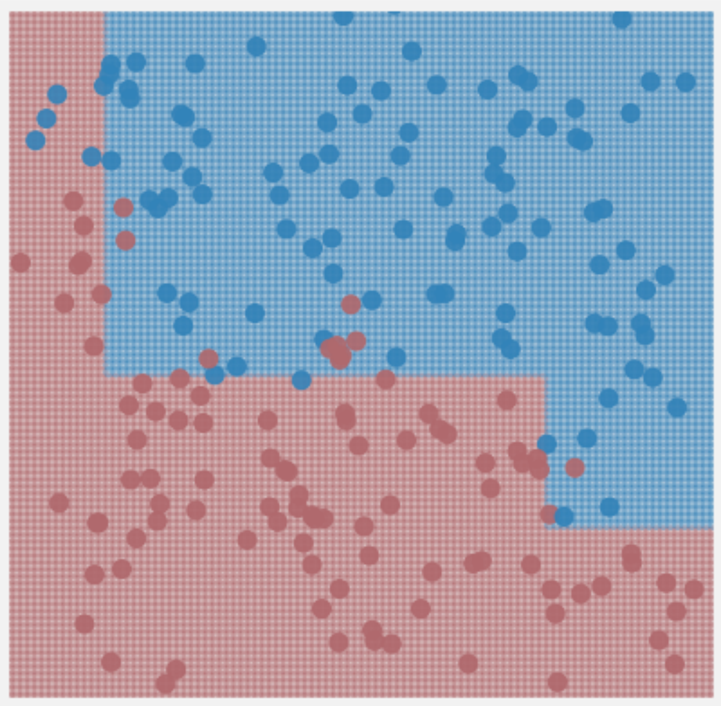
We could alleviate this a bit with pruning

Prepruning:

- Don't let the tree have too many levels
- Stopping conditions

Postpruning:

- Build the complete tree
- Go back and remove vertices that don't affect performance too much



Ensemble Methods

Today we'll see how we can use many decision trees to do even better

Ensemble methods are very popular in Machine Learning of late

General Idea:

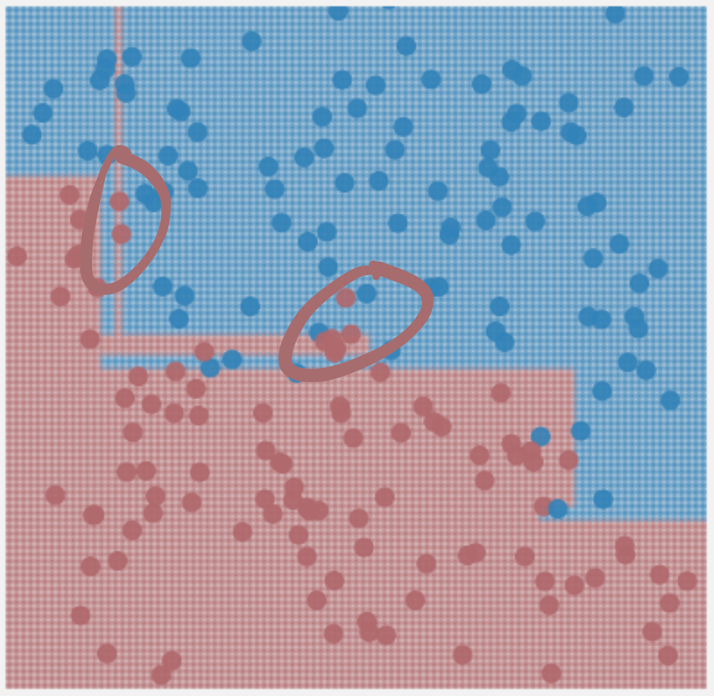
- Train numerous slightly different classifiers
- Use each classifier to make a prediction on a query point
- Make the ensembled prediction by taking majority vote from individual classifiers

Bagging

This week we'll look at three different types of Ensembled Decision Tree Classifiers

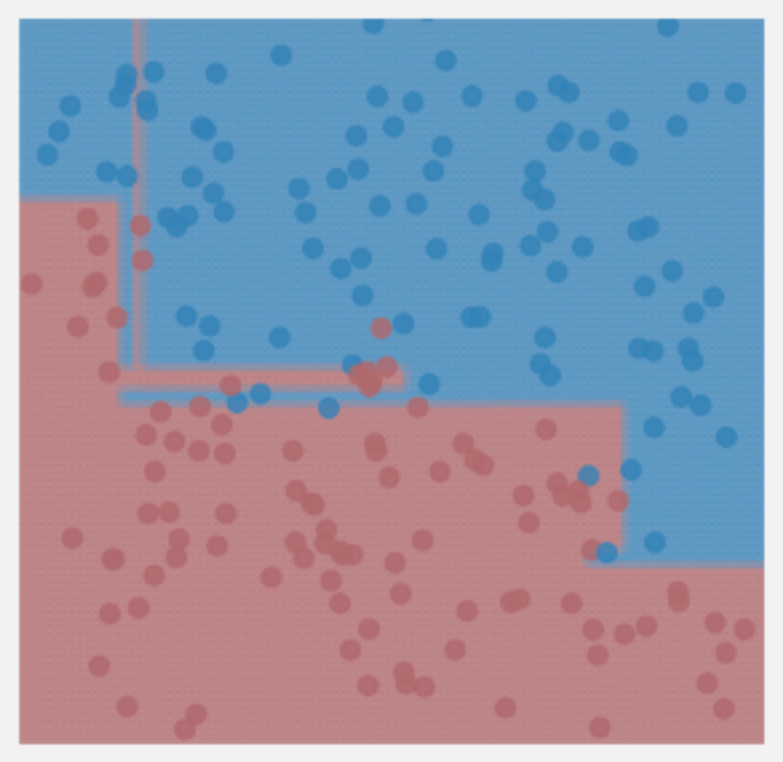
The simplest is called **Bootstrapped Aggregation** (or Bagging)

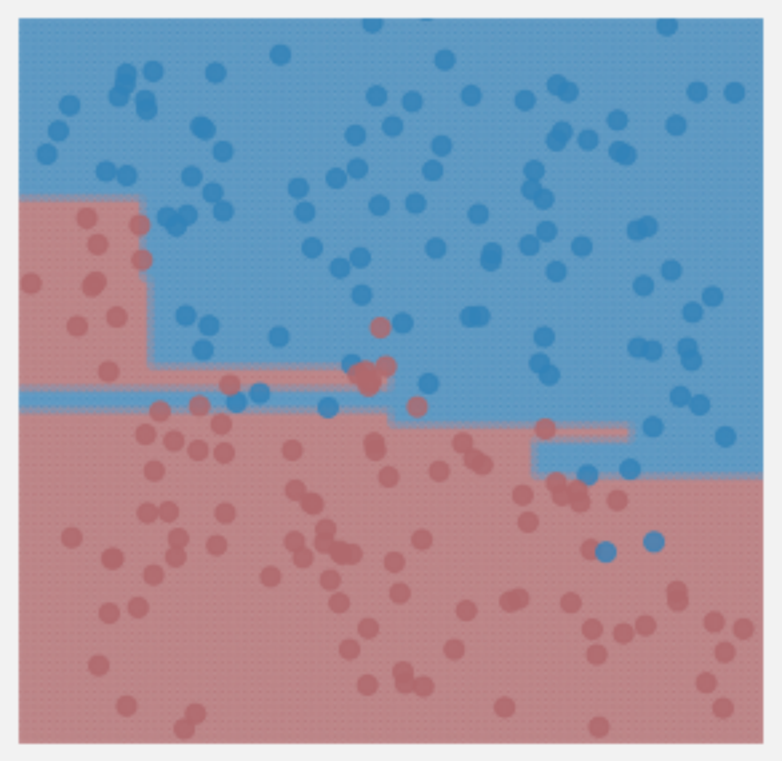
Idea: What would have happened if those training examples that we clearly overfit to weren't there?



Bagging

The simplest is called **Bootstrapped Aggregation** (or Bagging)



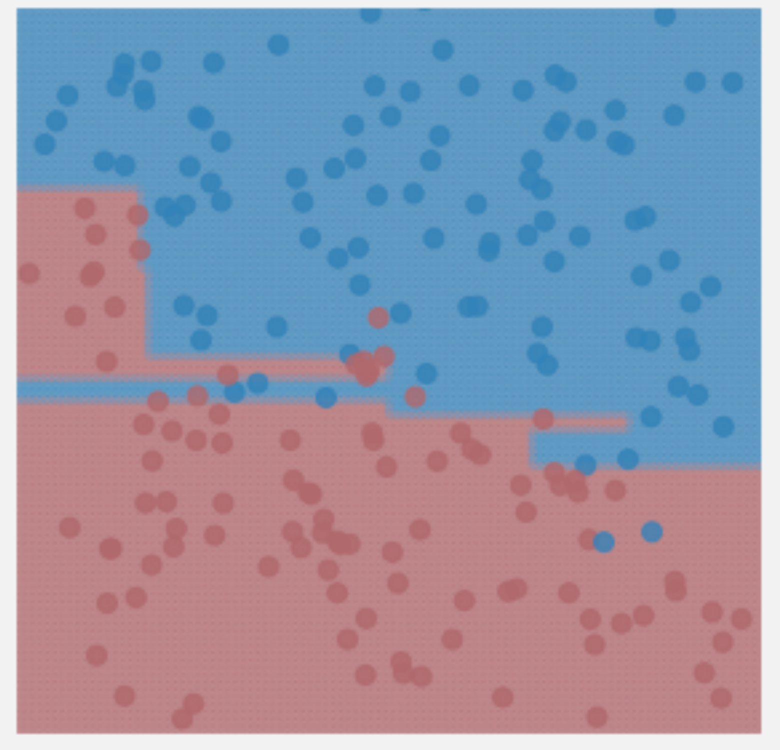


Bagging

n TRAINING EXS

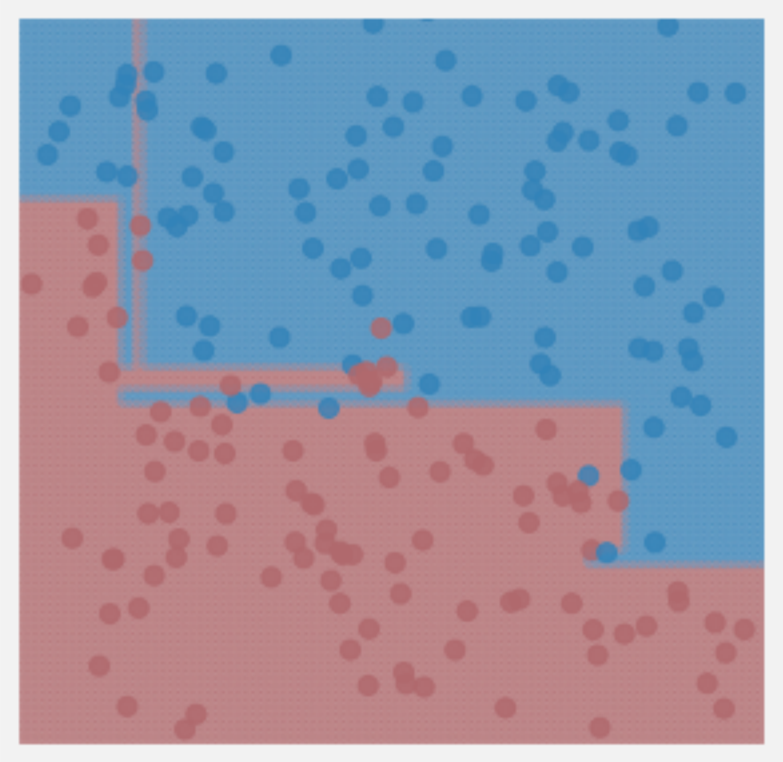
The simplest is called **Bootstrapped Aggregation** (or Bagging)

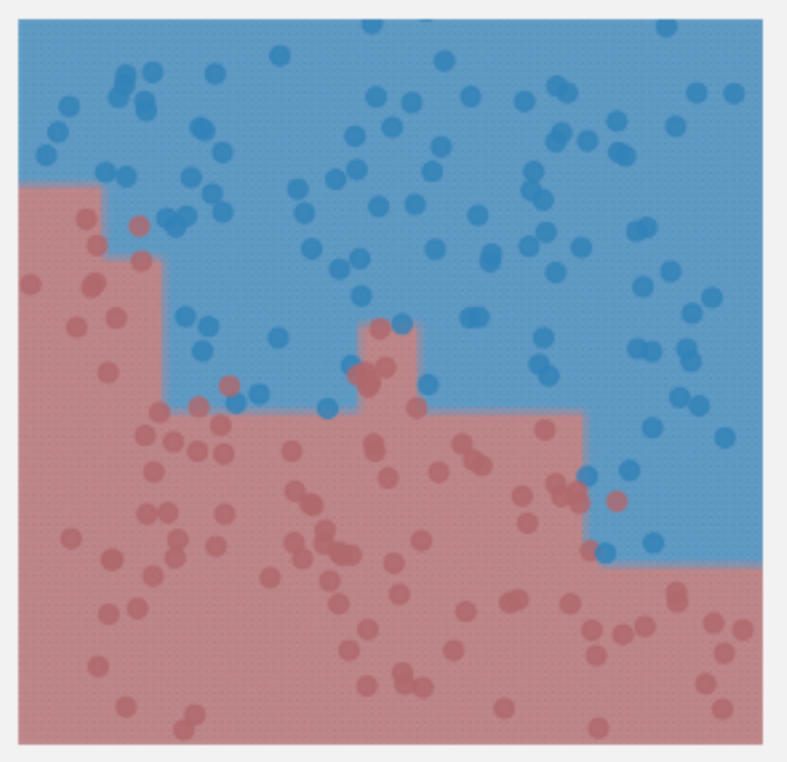
- Sample n training examples **with replacement**
- Fit full-depth Decision Tree Classifier
- Repeat many times
- Aggregate results by majority vote



Bagging

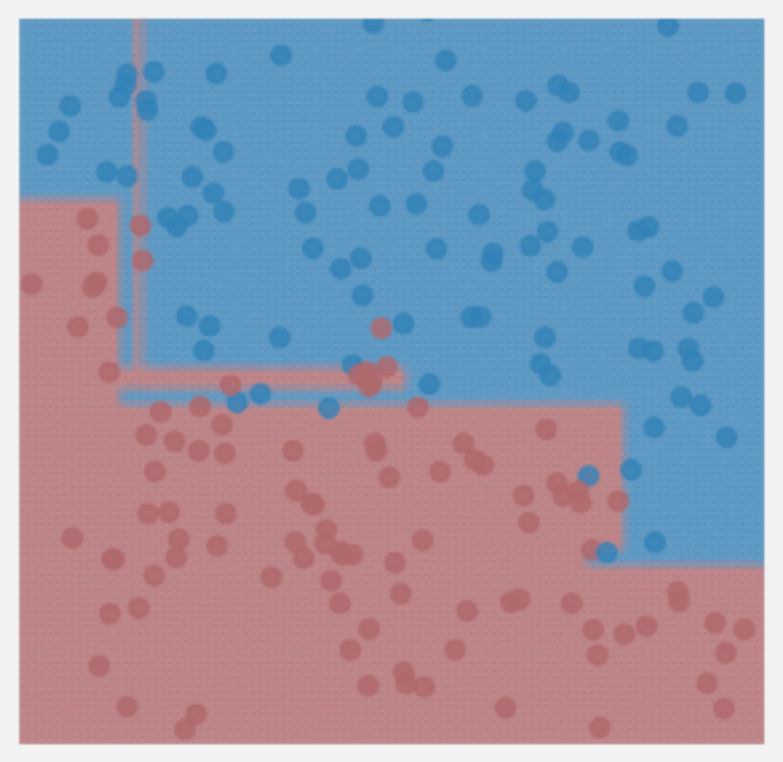
The simplest is called **Bootstrapped Aggregation** (or Bagging)

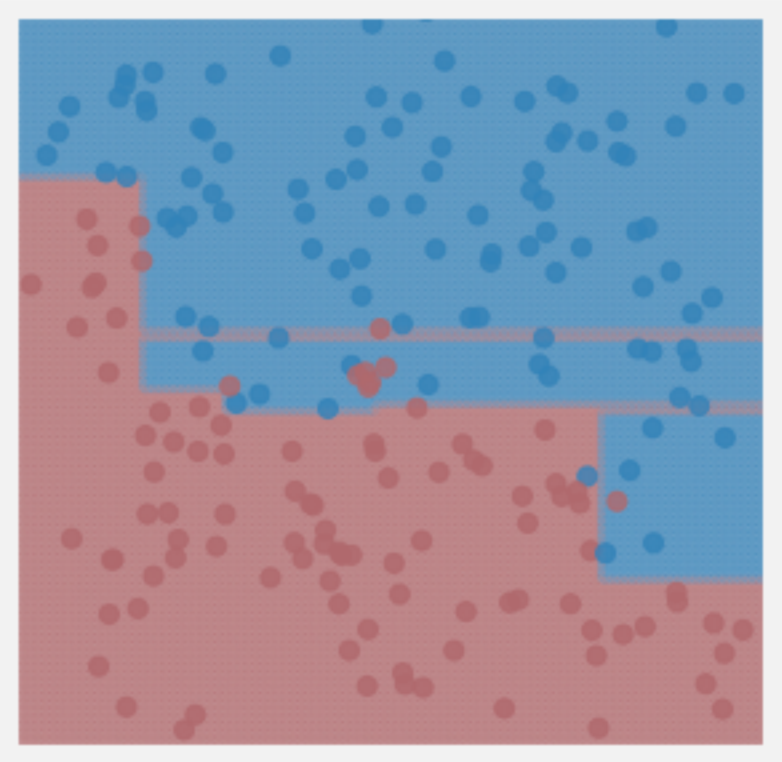




Bagging

The simplest is called **Bootstrapped Aggregation** (or Bagging)



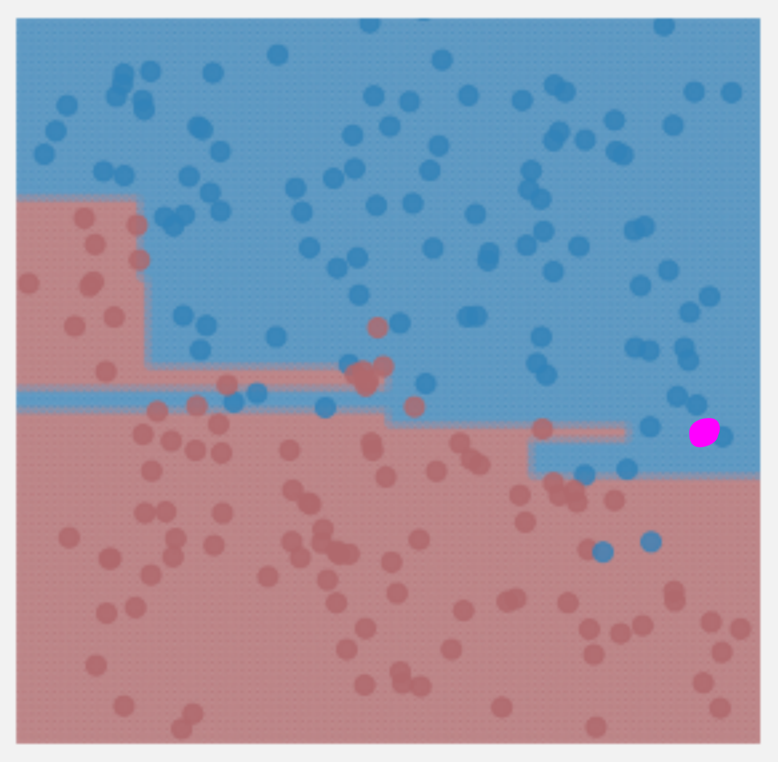


Bagging

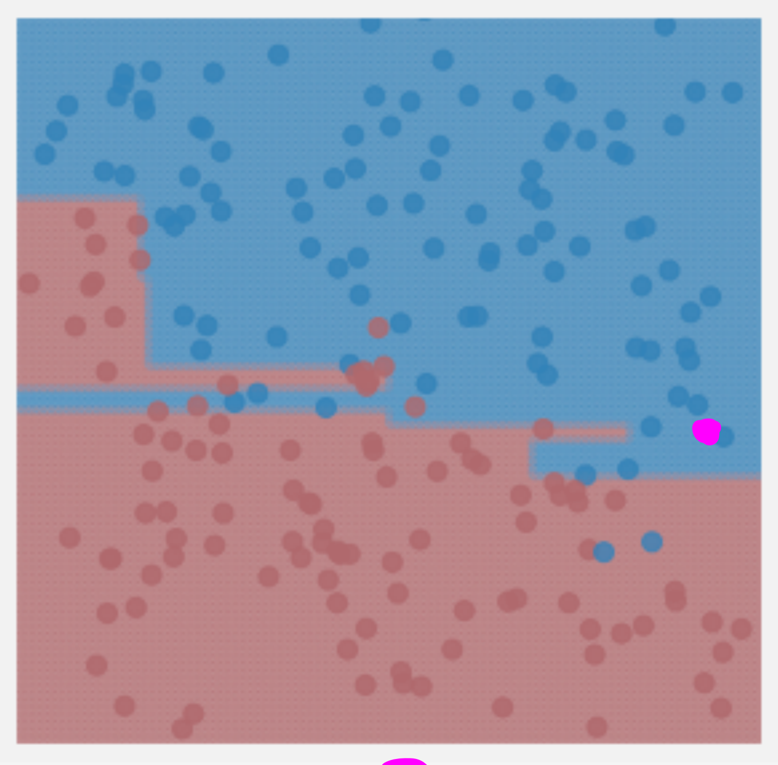
2B's 1R

⇒ MV ⇒ B

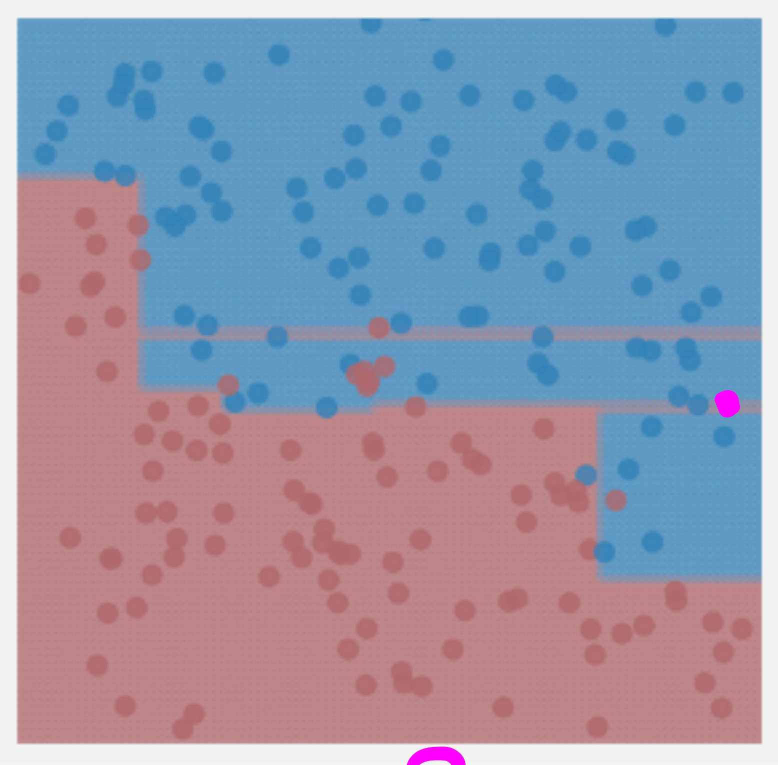
The simplest is called **Bootstrapped Aggregation** (or Bagging)



B

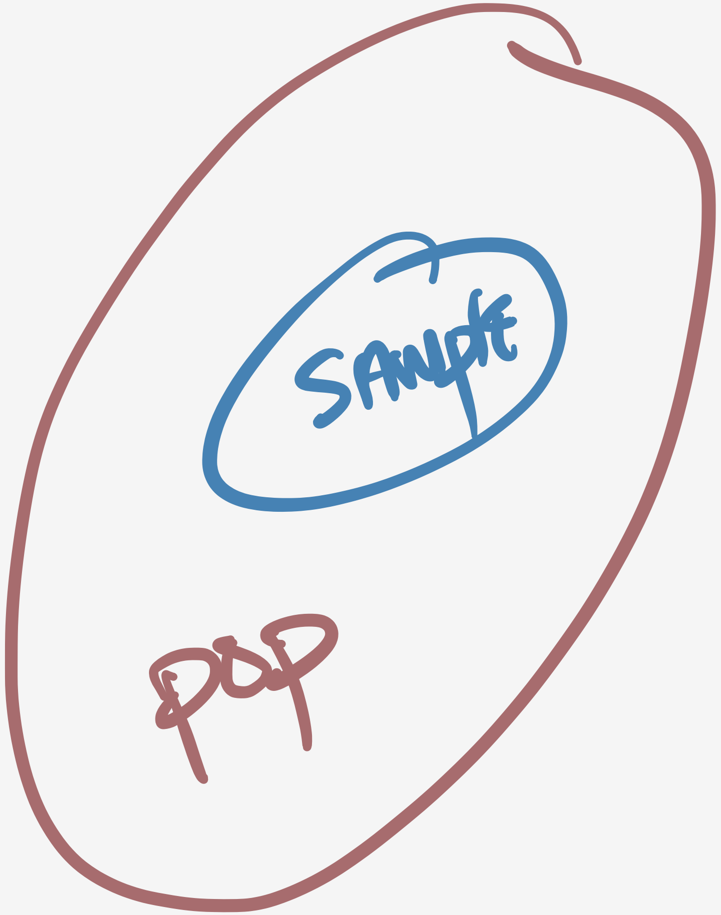


B



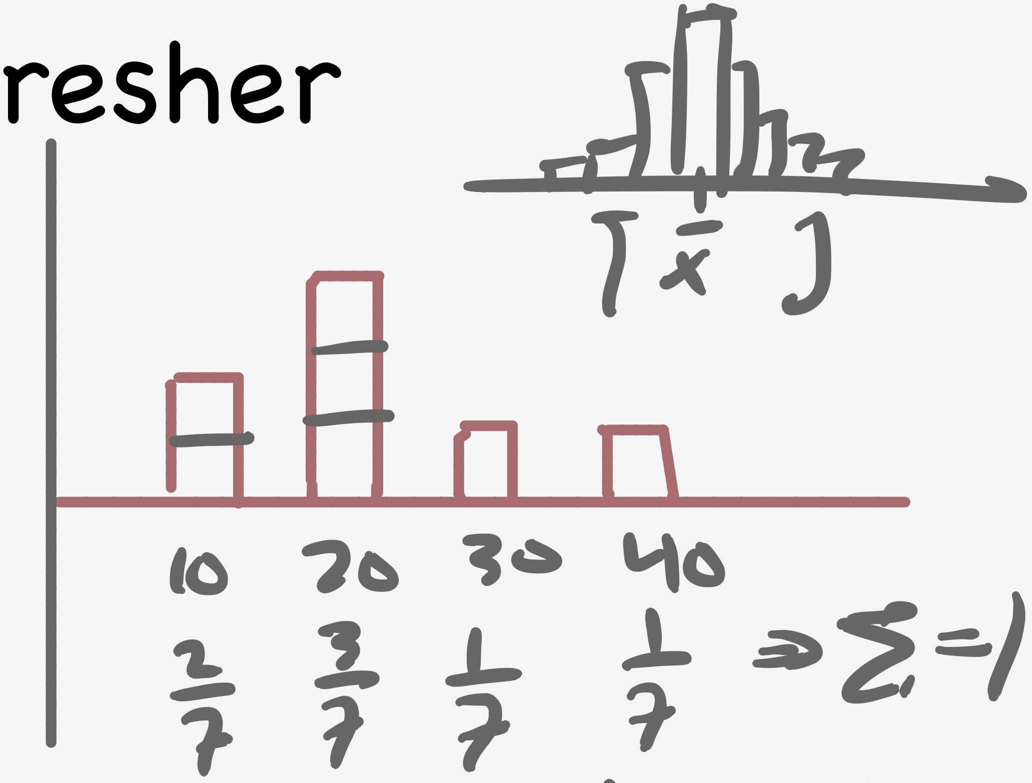
R

Bootstrap Refresher



INCOMES

10k
10k
20k
20k
20k
30k
40k



EMPIRICAL DIST.

$R_1 =$ 10, 20, 20, 20, 20, 20

$R_2 =$ 10, 10, 20, 20, 10, 2

Bagging

In general, assume that we define $y \in \{-1, 1\}$ and each individual DCT has $h_k(\mathbf{x})$

Then we can define the Bagged Classifier as

$$H(\mathbf{x}) = \text{sign}\left[\sum_{k=1}^{K} h_k(\mathbf{x})\right]$$

$1, 1, -1, -1, 1, 1$

↓

$\Sigma = 2$

$\text{sign}(2) = 1$

How do we Validate the Bagged model?

- Could use cross-validation or a held-out validation set, as usual
- But there is a better way.

Out-of-Bag Error Estimation

- Trees are repeatedly fit on bootstrapped re-samples of the training set
- Can be shown that around 2/3 of training examples present in each bootstrapped re-sample
- OOB Error Estimation makes use of the training examples NOT in the re-sample

Here's How:

$$x_i \Rightarrow \text{sign}\left[\sum_{\substack{k=1 \\ k \notin R_k}}^{500} h_k(x_i)\right] \Rightarrow +1 \text{ or } -1$$

Bagging Pros and Cons

Pros:

- Results in a much lower variance classifier than a single Decision Tree

Cons:

- Results in a much less interpretable model

We essentially give up some interpretability in favor of better prediction accuracy

But we can still get **insight** into what our model is doing using Bagging

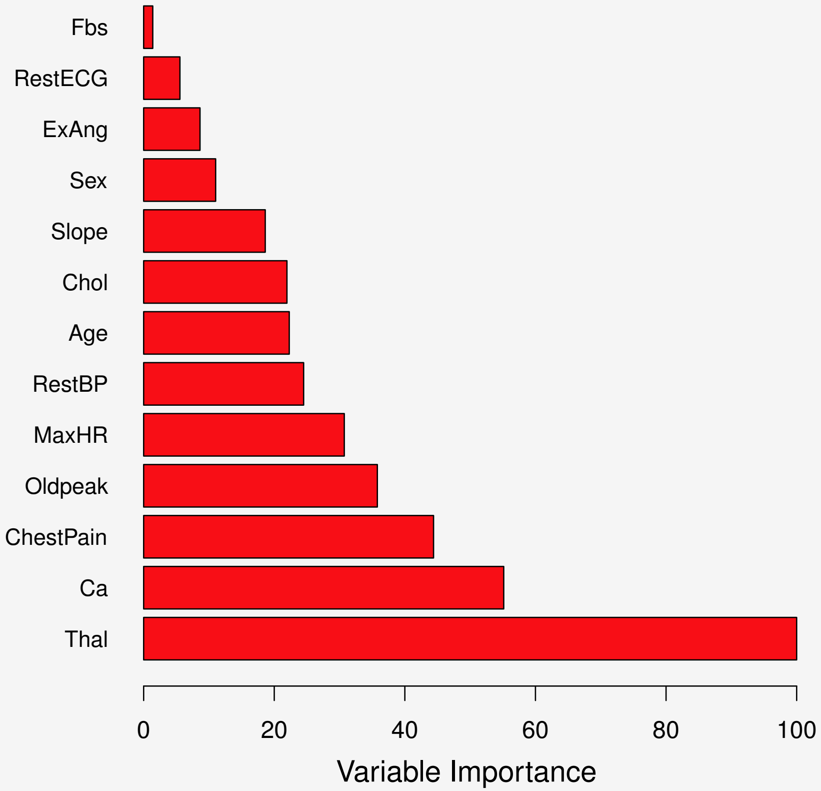
Bagging and Feature Importance

In a single Decision Tree Classifier we can get an idea of feature importance by

- ○ Measuring Information Gain / reduction in Impurity is achieved by splitting on feature
- ○ Compare these values at the end and rank features

With Bagging we can still do this.

We just compute these measures over all trees and average



An Even Better Approach

It turns out, we can take the Bagging idea and make it even better

Suppose you have a particular feature that is a very strong predictor for the response

So strong that in almost all Bagged Decision Trees, it’s the first feature that gets split on

When this happens, the trees can all look very similar. They’re too **correlated**

Maybe always splitting on the same feature isn’t the best idea

Maybe, like with bootstrapping training examples, we split on a random subset of features too

Random Forests

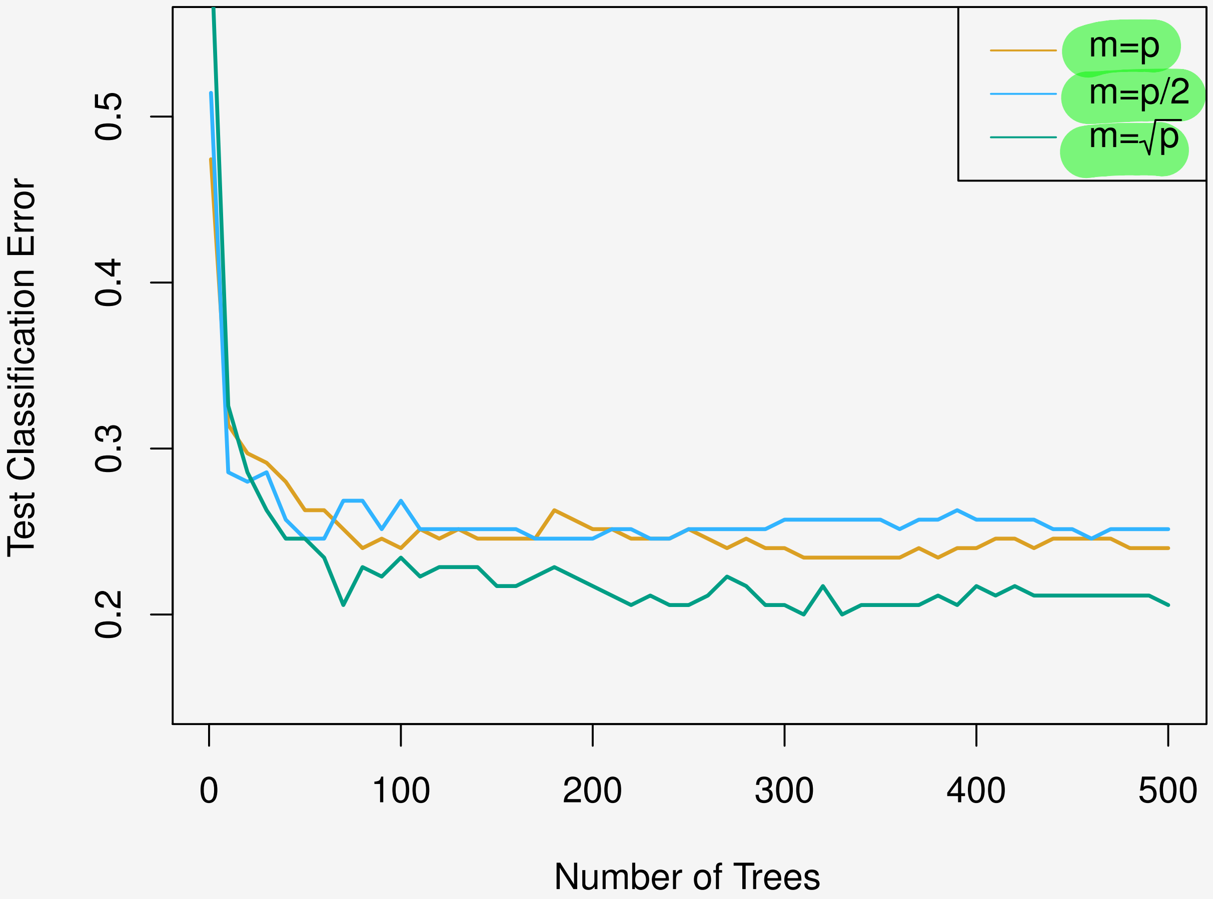
- Still doing Bagging, in the sense that we fit bootstrapped resamples of training data
- BUT, every time we have to choose a feature to split on, don't consider all p features
- Instead, consider only $\underline{m \approx \sqrt{p}}$ features to split on

Advantages:

- Since at each split, considering less than half of features, this gives very different trees
- Very different trees in your ensemble decreases correlation, and gives more robust results
- Also, slightly cheaper because you don't have to evaluate each feature to choose best split

Practical Results

- Look at expression of 500 different genes in tissue samples. Predict 15 cancer states



[James, et al]

Where Do We Go From Here?

- Unfortunately, Decision Tree Classifiers are prone to overfitting
- On their own, tend to do worse than other classifiers we've seen

But **WITH THEIR POWERS COMBINED**!

Next up, Ensemble Methods.

- Take various weaker classifiers, and combine them to get strong classifiers
- Bagging and Random Forests
- Boosted Decision Trees and the AdaBoost Algorithm