Classification IV: Decision Trees

-Applied Multivariate Analysis-

Lecturer: Darren Homrighausen, PhD

An Introductory Example

Use macroeconomic data to predict recessions

Use handful of national-level variables – Federal Funds Rate, Term Spread, Industrial Production, Payroll Employment, S&P500

Also include state-level Payroll Employment

In this example, we code $Y = 1$ as a recession and $Y = 0$ as growth.

We will use data from 1960 through 1999 as training data

We will use data from 2000 through 2011 as testing data

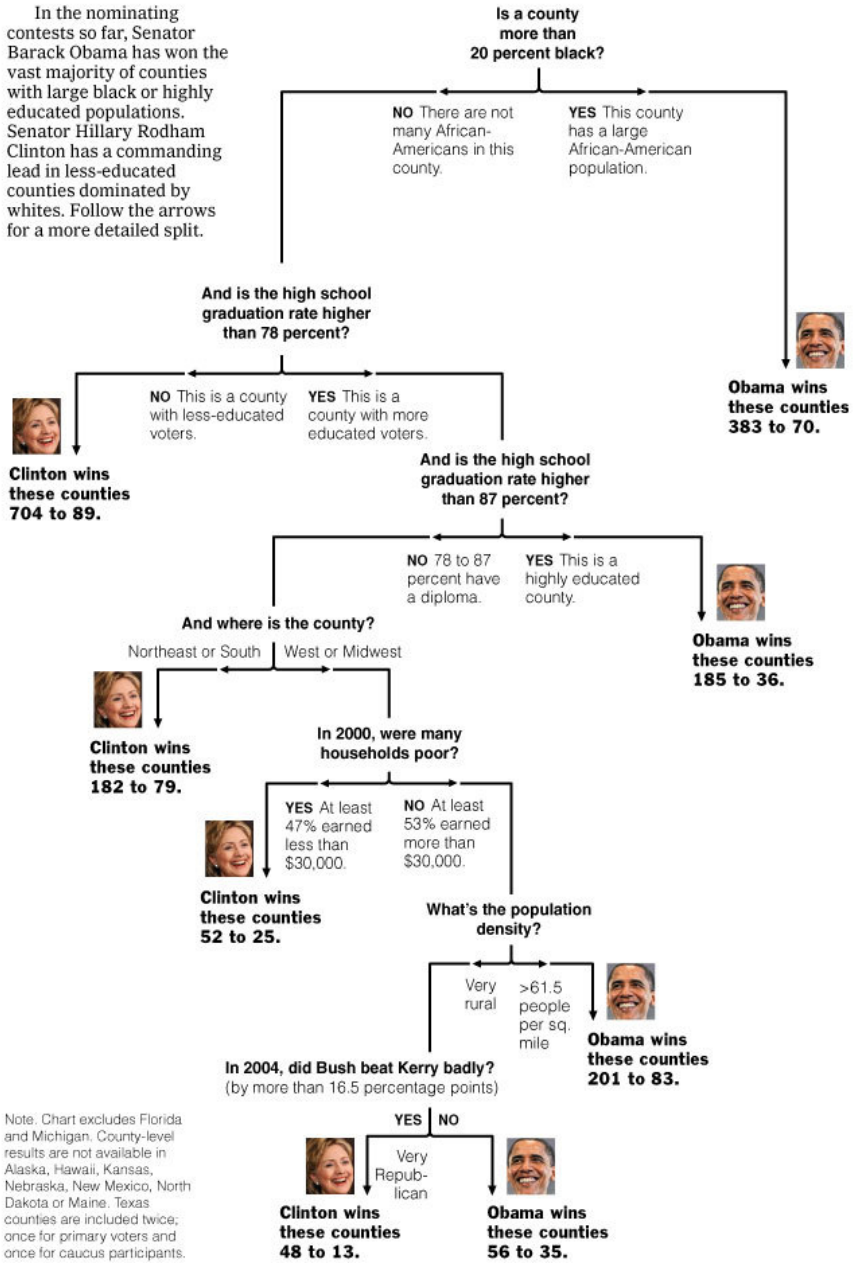
See Owyang, Piger, Wall (2012)

Trees

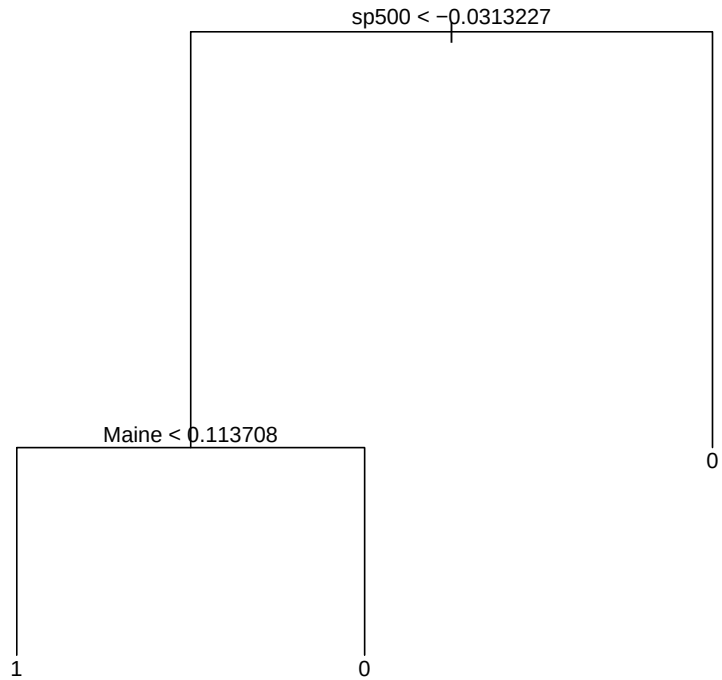
What is a (decision) tree?

- Trees involve stratifying or segmenting the predictor space into a number of simple regions.
- Trees are simple and useful for interpretation.
- Basic trees are not great at prediction.
- More modern methods that use trees are much better.

Example tree



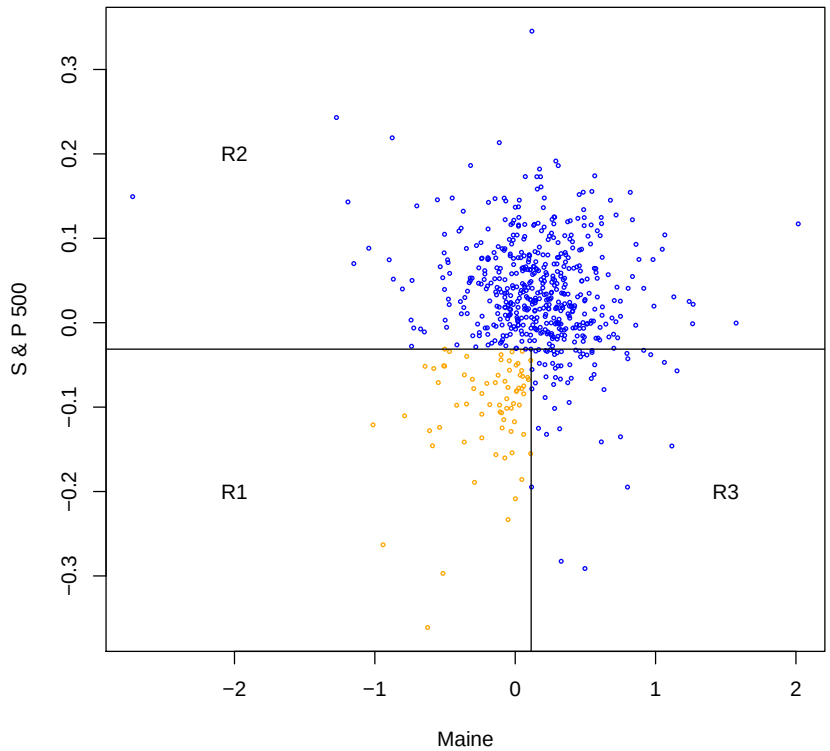
Dendrogram view



Terminology

- We call each split or end point a node. Each terminal node is referred to as a leaf
  - This tree has 2 interior nodes and 3 terminal nodes.
- The interior nodes lead to branches.
  - This graph has two main branches (the S&P 500 split).

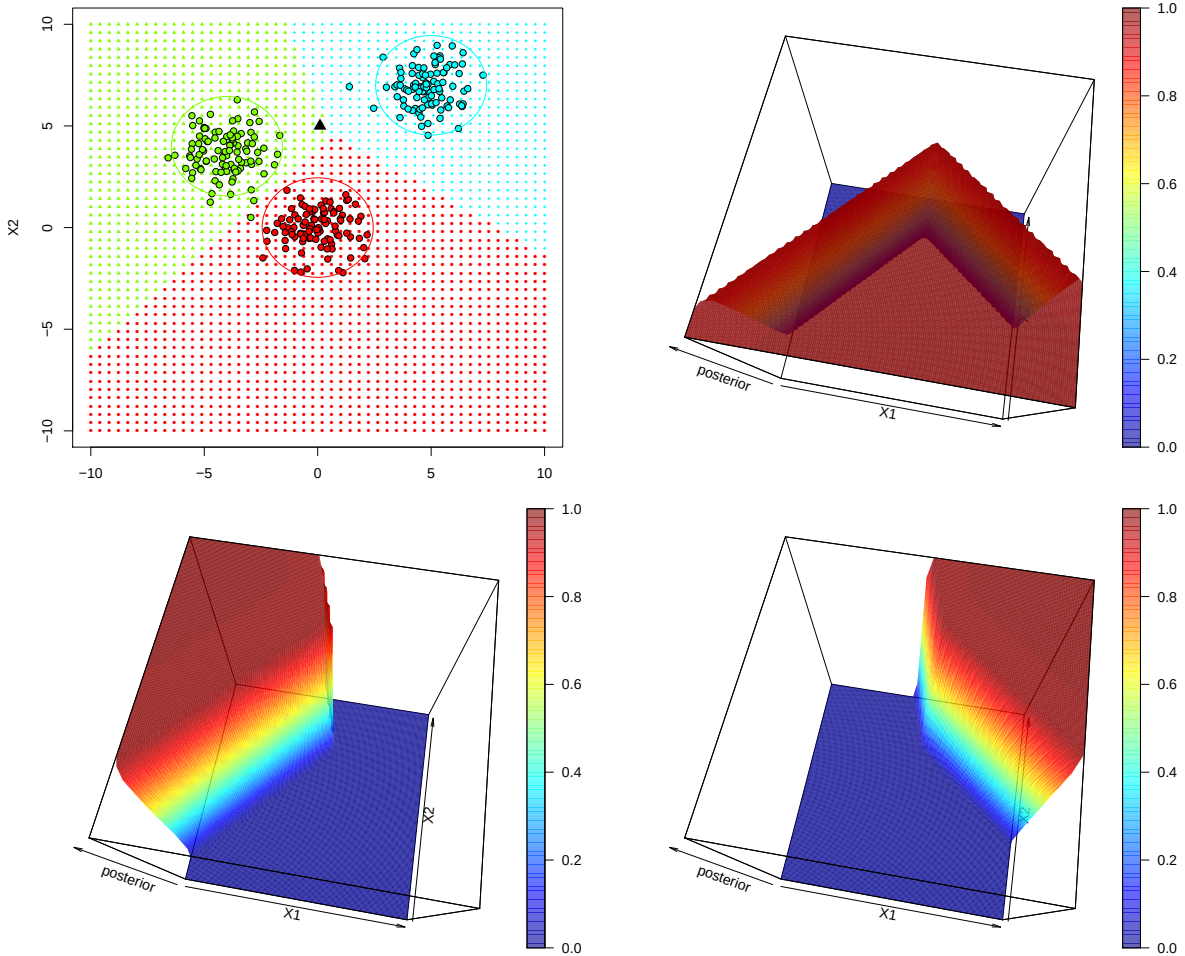
Partitioning view



Notes

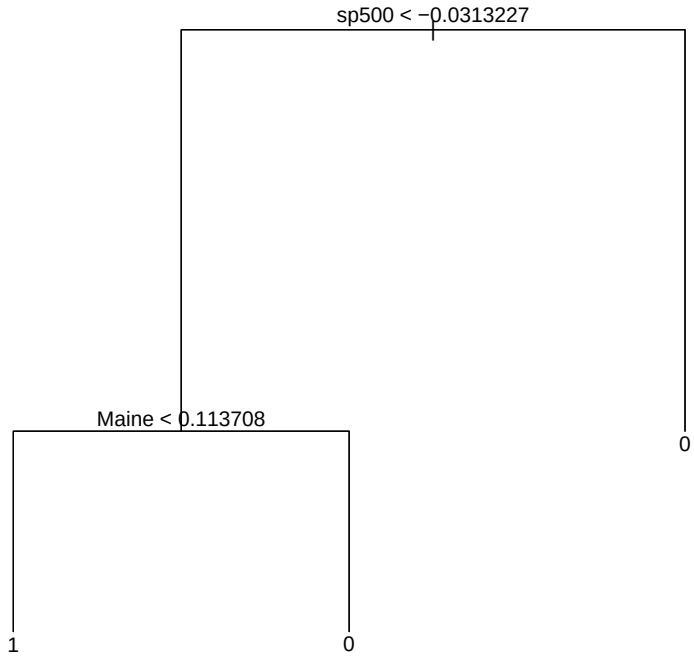
- We classify all observations in a region the same.
- The three regions R1, R2, and R3 are the leaves of the tree.

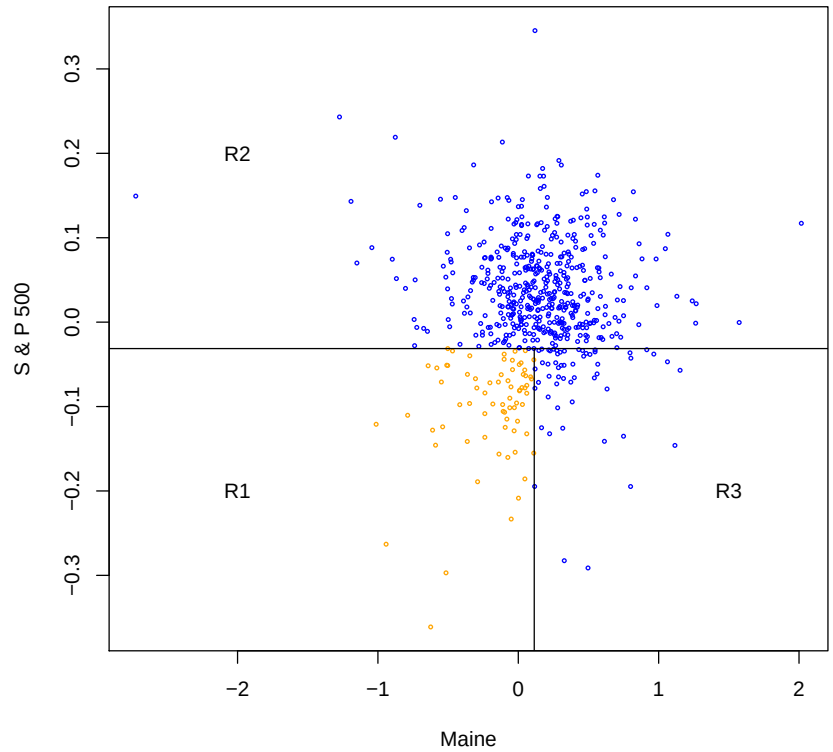
We've seen this before...



LDA also partitions into regions

Tree





We can interpret this as

- S&P 500 is the most important variable.
- If S&P 500 is large enough, then we predict no recession.
- If S&P 500 is small enough, then we need to know the change in the employment level of Maine.

How do we build a tree?

1. Divide the predictor space into M non-overlapping regions $R_1, \ldots, R_M$

   (this is done via greedy, recursive, binary splitting)
2. Every observation that falls into a given region R_m is given the same prediction
   - Regression: The average of the responses for a region
   - Classification: Determined by majority (or plurality) vote in that region

Important:

- Trees can only make rectangular regions that are aligned with the coordinate axis.
- The fit is greedy, which means that after a split is made, all further decisions are conditional on that split.
- The tree stops splitting when there are too few observations in a terminal node

How do we measure quality of fit?

Let $\hat{p}_{mk}$ be the proportion of training observations

- in the m^{th} region

  (Reminder: This corresponds to the m^{th} terminal node)

- from the k^{th} class

There are many possibilities:

classification error rate:	$E = 1 - \max_k(\hat{p}_{mk})$
Gini index:	$G = \sum_k \hat{p}_{mk}(1 - \hat{p}_{mk})$
cross-entropy:	$D = -\sum_k \hat{p}_{mk} \log(\hat{p}_{mk})$

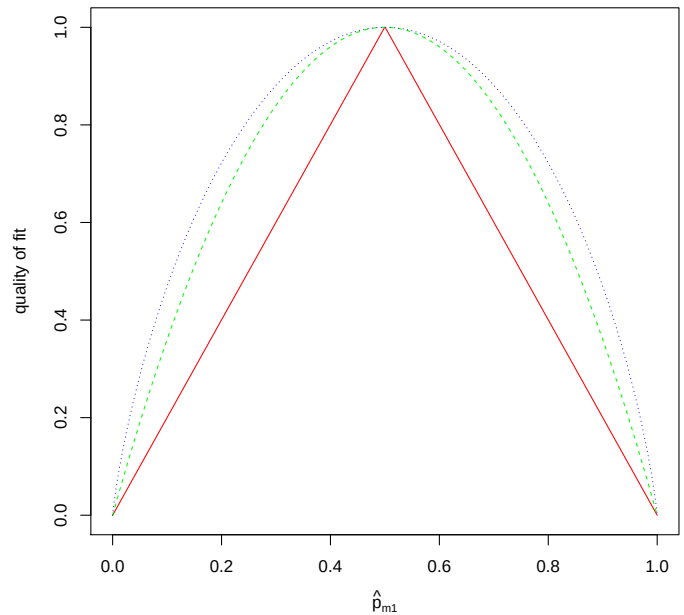
We build a classifier by growing a tree that minimizes G or D.

How do we measure quality of fit?

Example: Suppose $K = 2$. Then $\hat{p} = \hat{p}_{m1} = 1 - \hat{p}_{m2}$

The m^{th} node is made by minimizing either E, G, or D over all

- Covariates
- split points of that covariate



Generally, Gini index or Cross-entropy is preferred

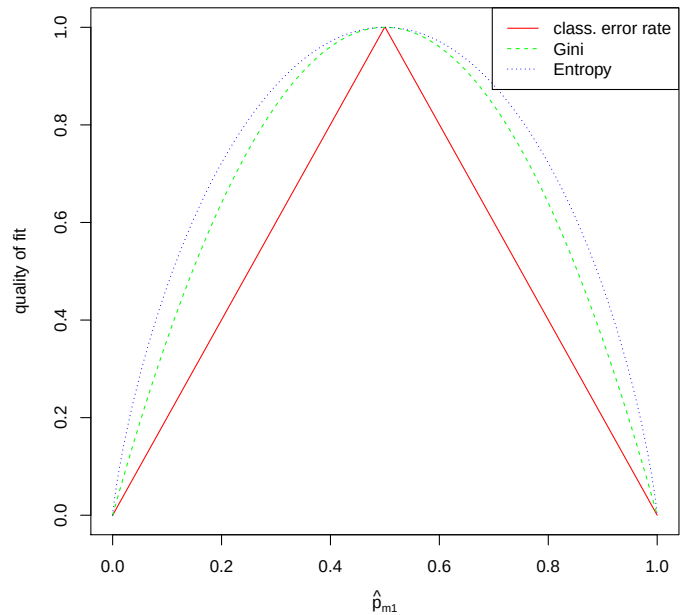
(They penalize values of $\hat{p}$ far from 0 or 1 more severely)

How do we measure quality of fit?

Example: Suppose $K = 2$. Then $\hat{p} = \hat{p}_{m1} = 1 - \hat{p}_{m2}$

The m^{th} node is made by minimizing either E, G, or D over all

- Covariates
- split points of that covariate



Generally, Gini index or Cross-entropy is preferred

(They penalize values of $\hat{p}$ far from 0 or 1 more severely)

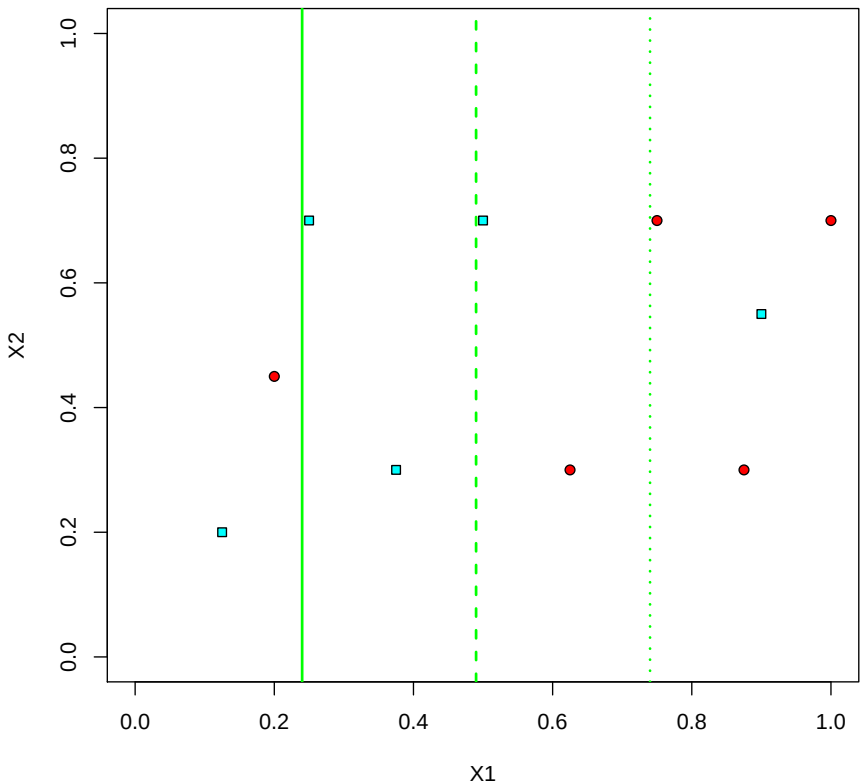
How do we measure quality of fit?

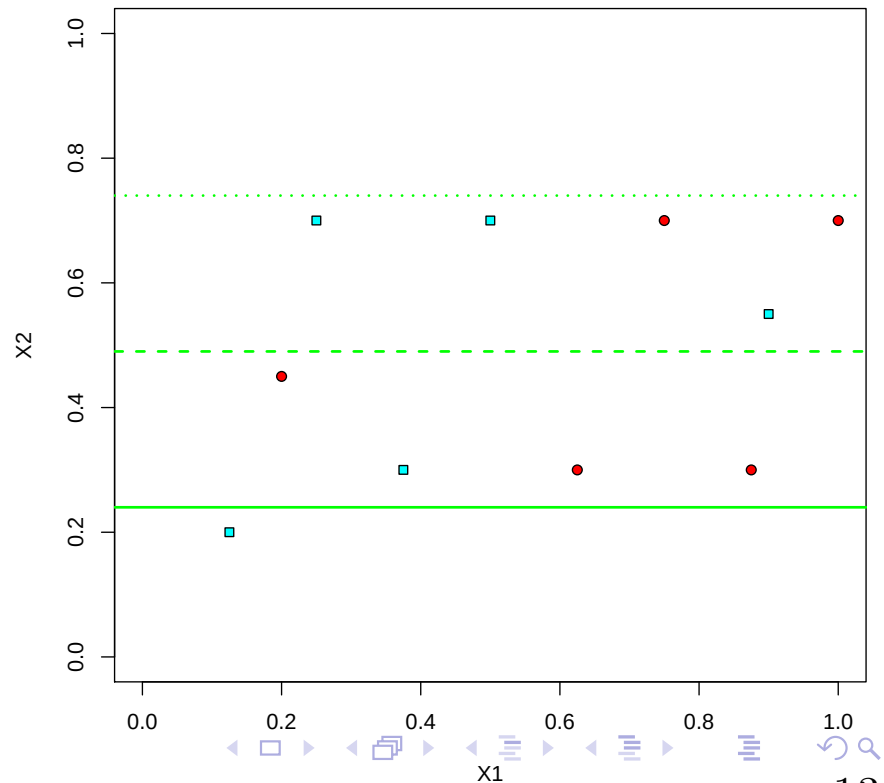
Example: Suppose $K = 2$ and we want to make the first split

Then $\hat{p}_{11} = 1 - \hat{p}_{12}$

(Define the 'left' or 'bottom' region as R_1)

Let's look at some possible splits:





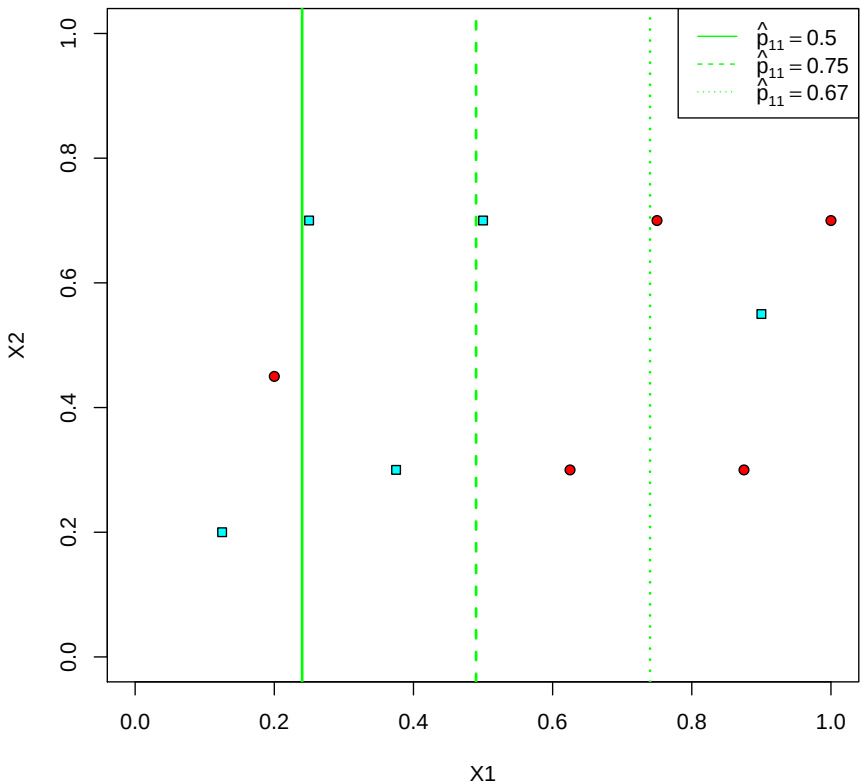
How do we measure quality of fit?

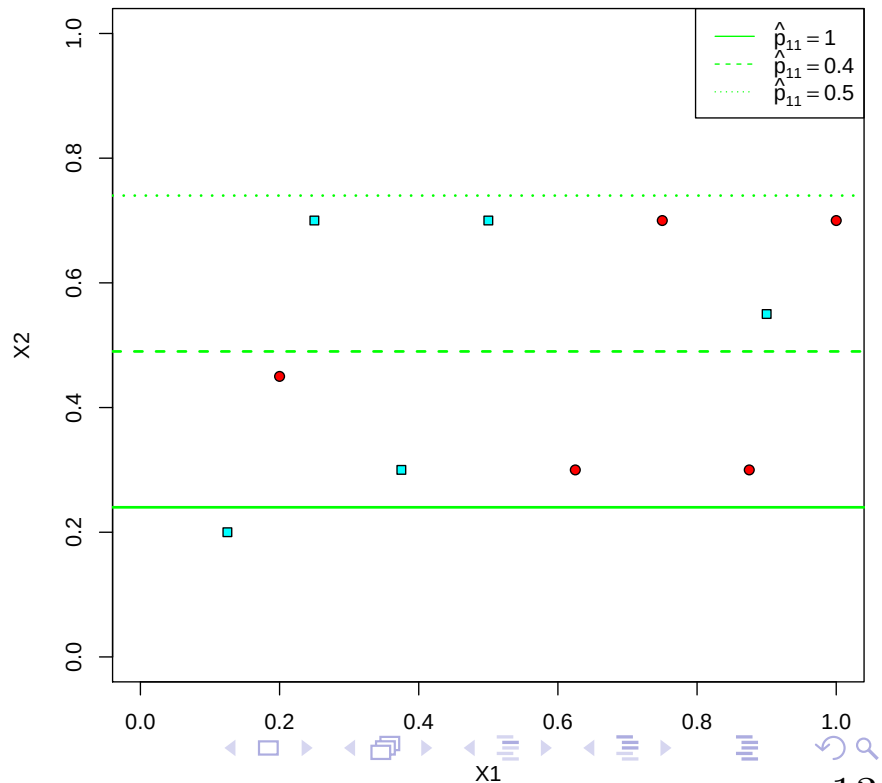
Example: Suppose $K = 2$ and we want to make the first split

Then $\hat{p}_{11} = 1 - \hat{p}_{12}$

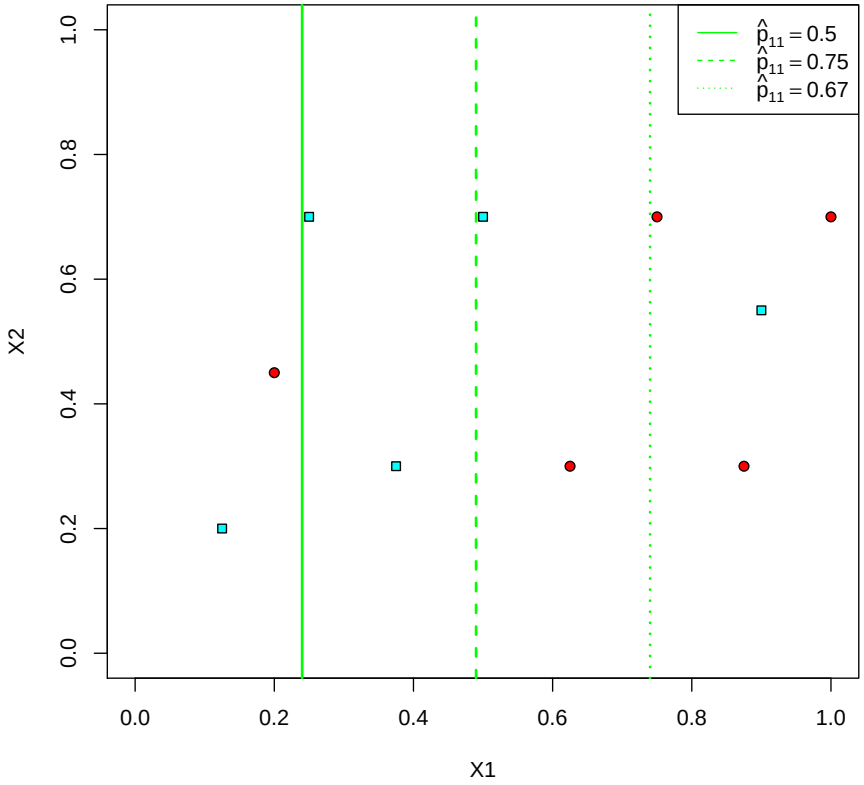
(Define the 'left' or 'bottom' region as R_1)

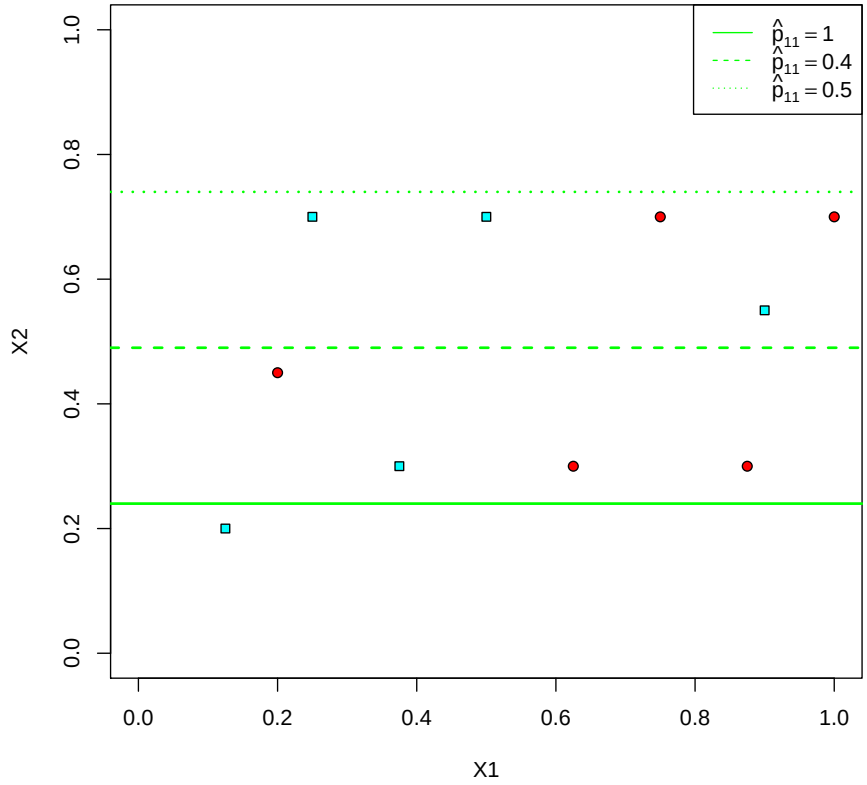
Let's look at some possible splits:





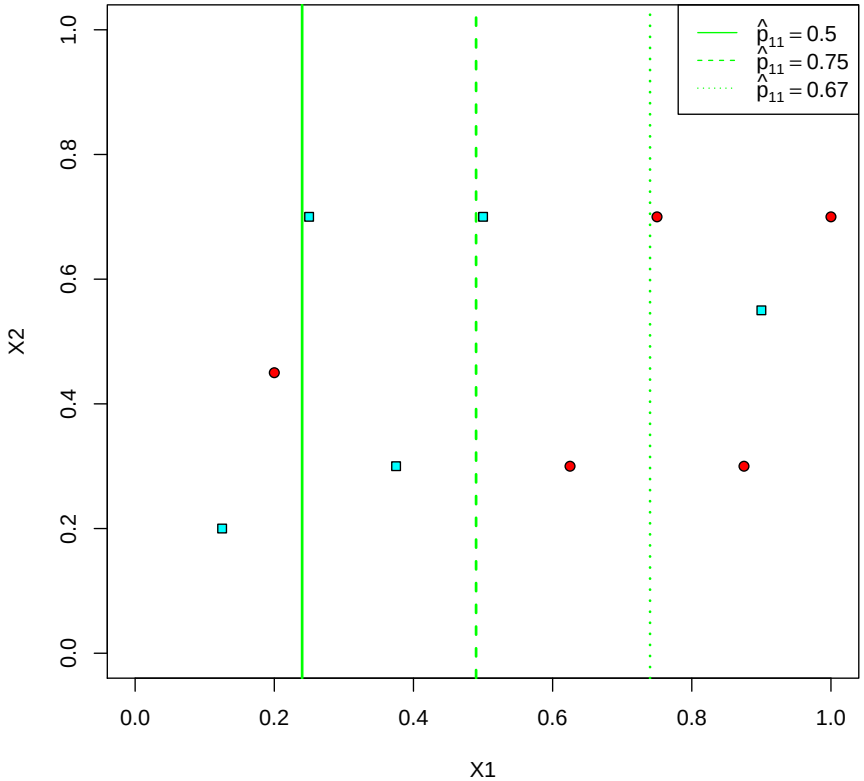
How do we measure quality of fit?

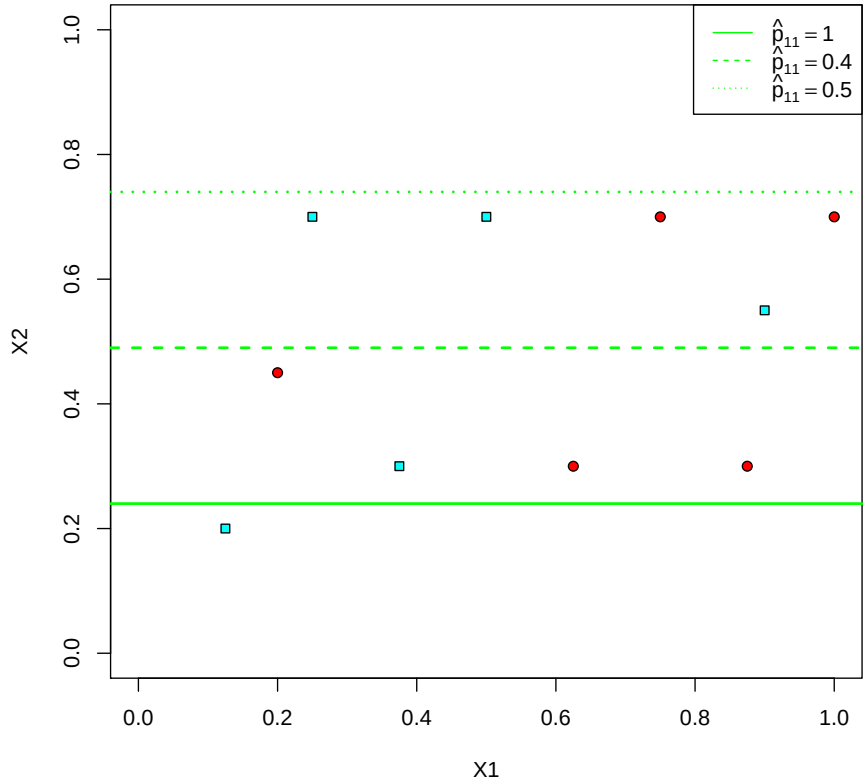




Where would we split?

How do we measure quality of fit?



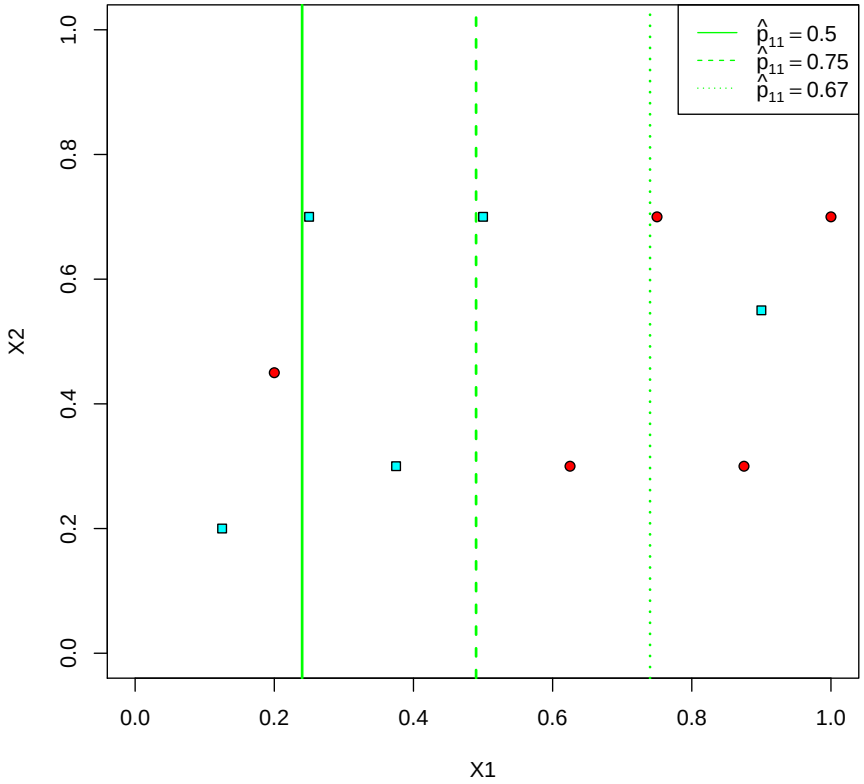


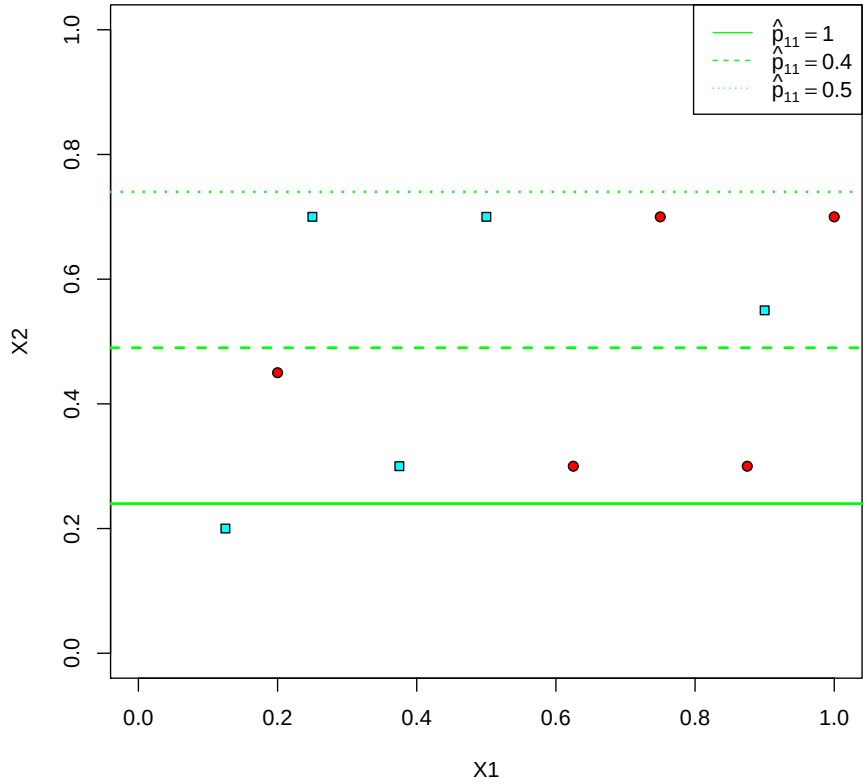
Where would we split?

At the solid, horizontal line

($\hat{p}_{11} = 1 \Rightarrow E, G, D = 0$)

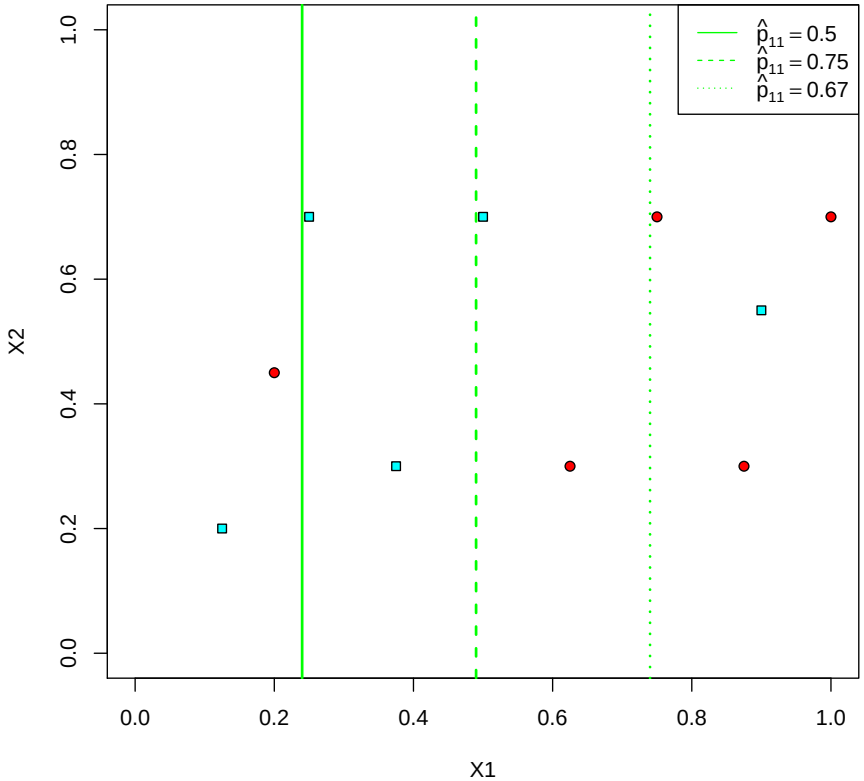
How do we measure quality of fit?

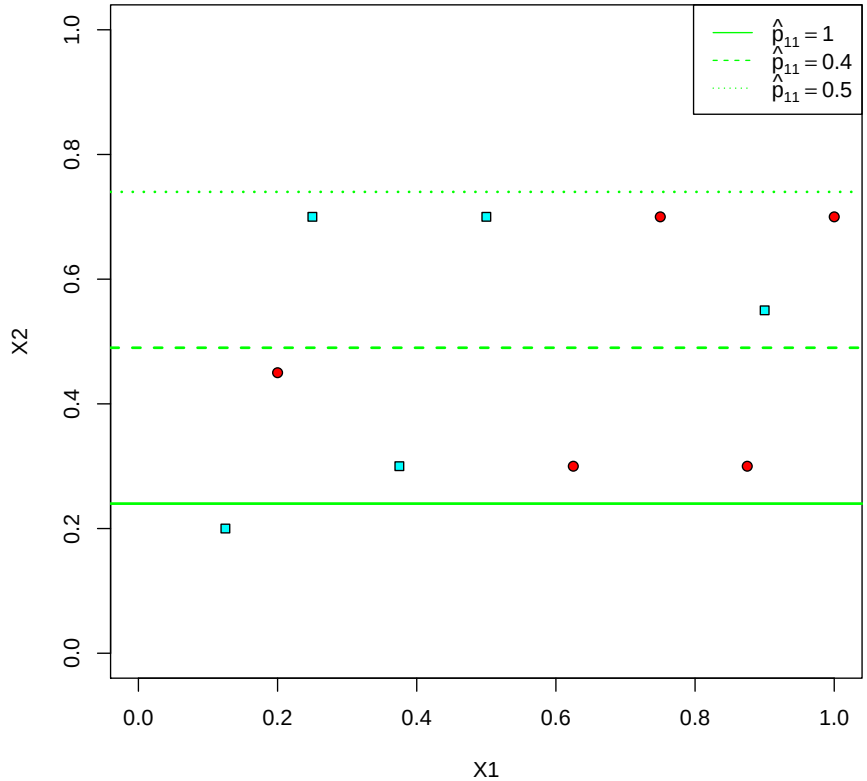




Where would we split if we required ≥ 2 observations in a node?

How do we measure quality of fit?





Where would we split if we required ≥ 2 observations in a node?

At the dashed, vertical line

($\hat{p}_{11} = .75 \Rightarrow E = 0.25, G = .375, D = .562$)

There's a problem

Following this procedure overfits!

- The process described so far will fit overly complex trees, leading to poor predictive performance.
- Overfit trees mean they have too many leaves.
- To stretch the analogy further, trees with too many leaves must be pruned.

Pruning the tree

- Cross-validation can be used to directly prune the tree, but it is far too expensive (computationally) to use in practice (combinatorial complexity)
- Instead, we use weakest link pruning

$$\sum_{m=1}^{|T|} \sum_{i \in R_m} \mathbf{1}(Y_i \neq \hat{Y}_{R_m}) + \alpha |T|$$

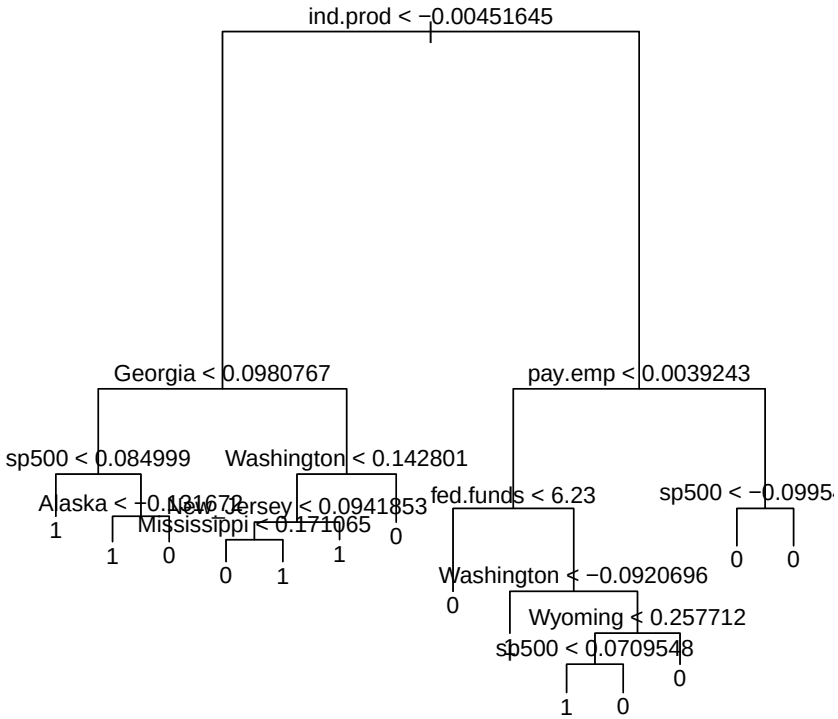
where $|T|$ is the number of terminal nodes.

Essentially, we are trading training fit (first term) with model complexity (second term)

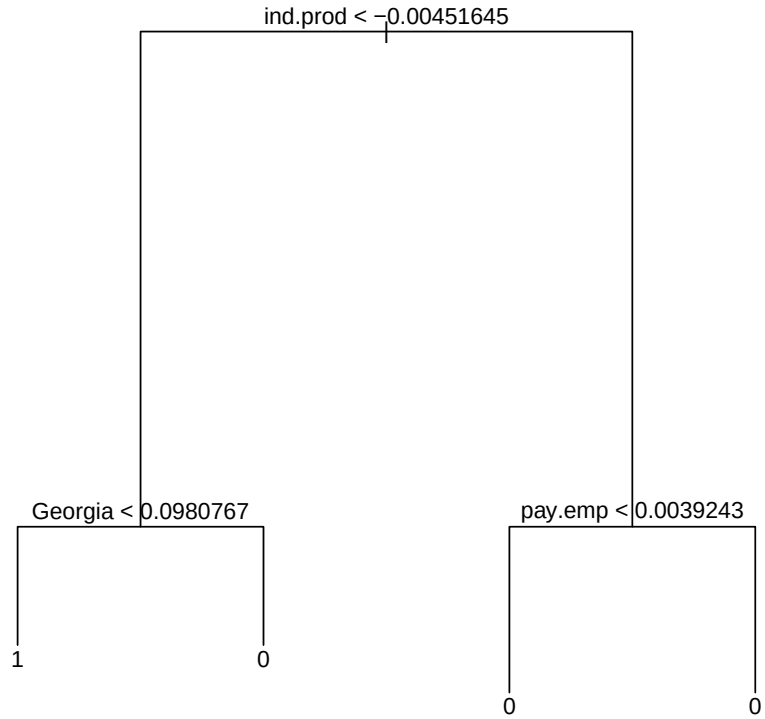
(compare to lasso)

- Now, cross-validation can be used to pick α.

Results of trees on recession data

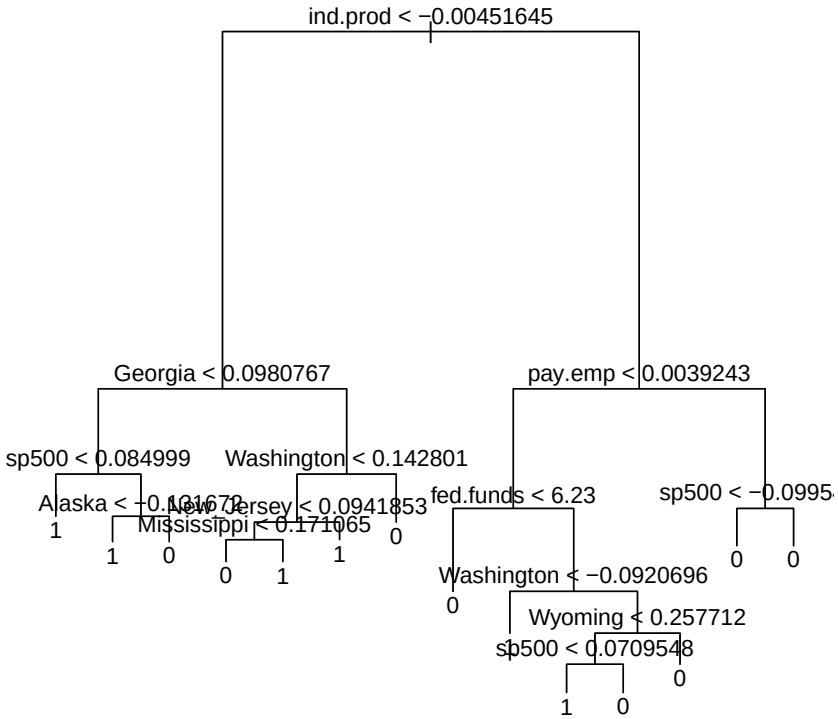


Unpruned tree

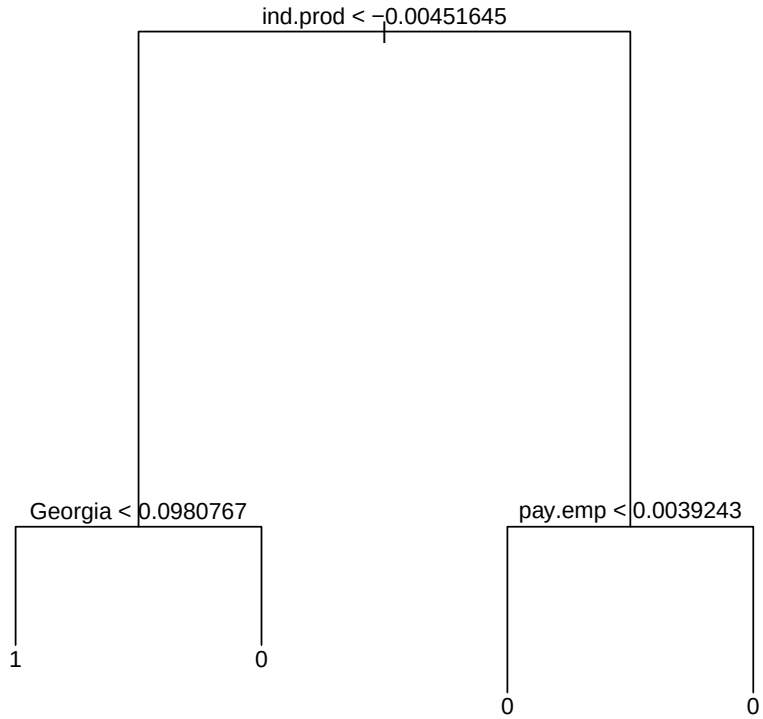


Pruned Tree

Results of trees on recession data



Unpruned tree

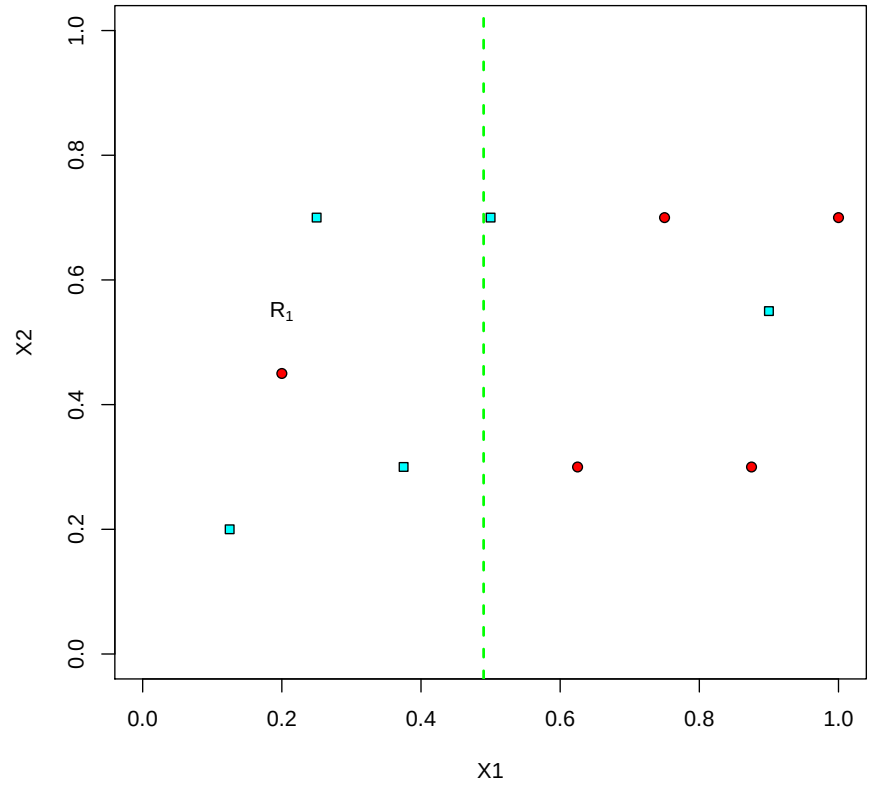


Pruned Tree

The pruned tree is a subset of the unpruned tree (nested)

There are splits that result in having the same prediction. Why?

Splits with same prediction



Suppose we split at verticle, dashed line. Then $\hat{p}_{11} = 0.75$.

What happens if we were to now split R_1 at $X2 = 0.5$?

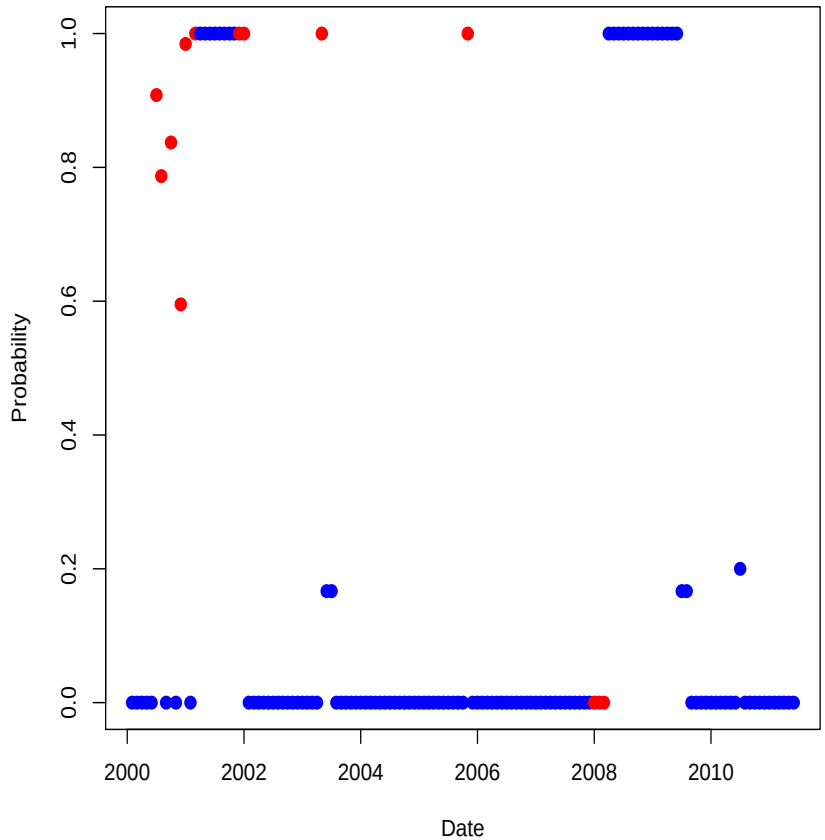
What happens if we were to now split R_1 at $X2 = 0.4$?

Results of trees on recession data

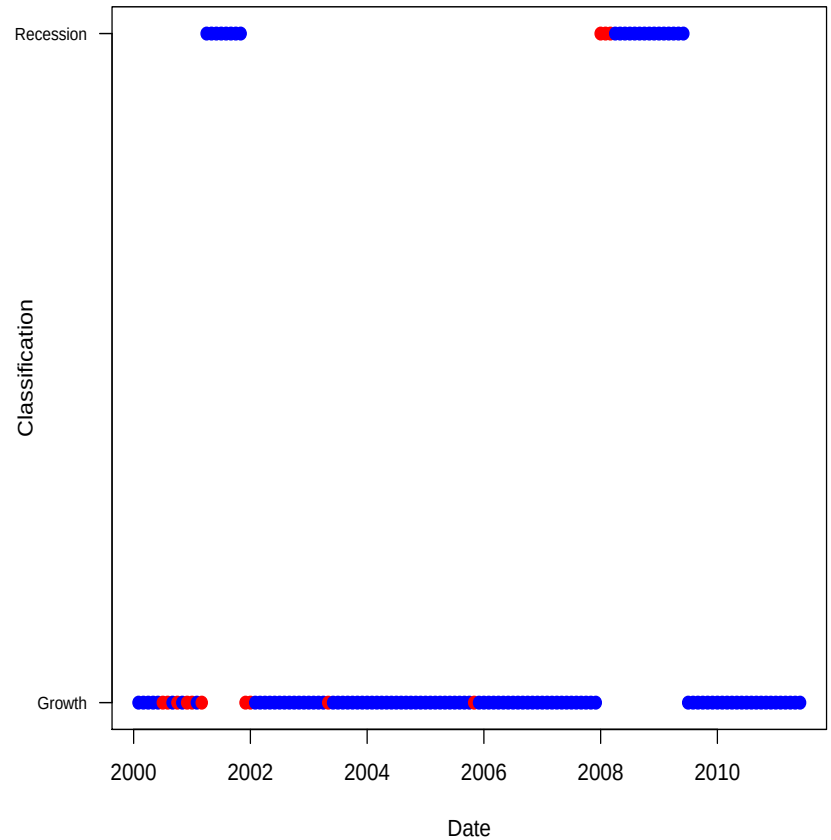
The trees on the previous slide were grown on the training data (from 1960 to 2000)

Now, we use them to predict on the test data (from 2000 to 2011)

Results of trees on recession data



Posterior probability of prediction



Predictions

Advantages and disadvantages of trees

+ Trees are very easy to explain (much easier than even linear regression).
+ Some people believe that decision trees mirror human decision.
+ Trees can easily be displayed graphically no matter the dimension of the data.
+ Trees can easily handle qualitative predictors without the need to create dummy variables.
– Trees aren't very good at prediction.

To fix this last one, we can try to grow many trees and average their performance.

Trees in R

Create a basic, unpruned tree:

```
require(tree)
out.tree = tree(Y~.,data=X,split='gini')
plot(tmp.tree)
text(tmp.tree)
```

Trees in R

Prune the tree via cross-validation

```
out.tree.orig   = tree(Y~.,data=X)
out.tree.cv     = cv.tree(out.tree.orig,FUN=prune.misclass)
> names(out.tree.cv)
[1] "size"   "dev"    "k"      "method"
```

Trees in R

Prune the tree via cross-validation

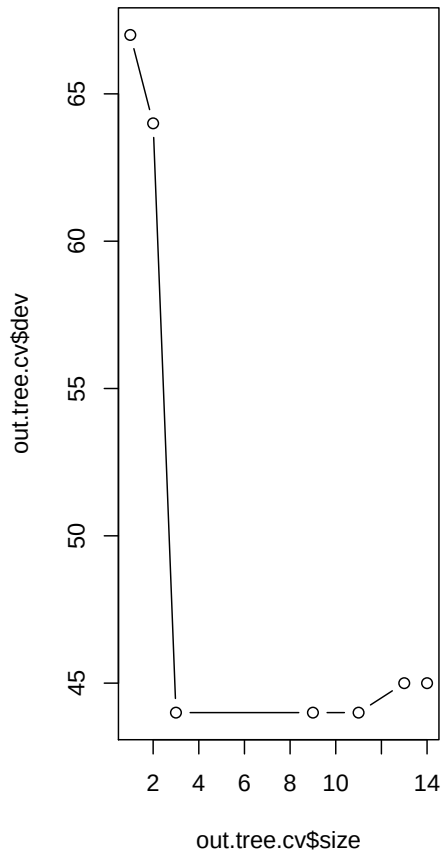
```
> out.tree.cv
$size
[1] 14 13 11  9  3  2  1

$dev
[1] 45 45 44 44 44 64 67

$k
[1] -Inf  0.0  2.0  2.5  3.0 15.0 20.0

$method
[1] "misclass"

attr(,"class")
[1] "prune"         "tree.sequence"
```

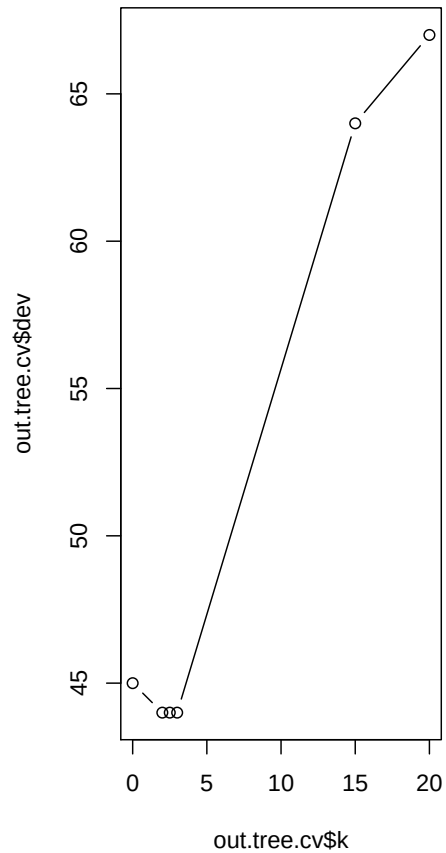
Note: k corresponds to α in weakest-link pruning

(Editors note: I had α as λ in posted notes)

Cross validation plots

```
plot(out.tree.cv$size,out.tree.cv$dev,type="b")
plot(out.tree.cv$k,out.tree.cv$dev,type="b")
```





Trees in R

Prune the tree via cross-validation

```
best.size       = out.tree.cv$size[which.min(out.tree.cv$dev)]
> best.size
[1] 11
out.tree        = prune.misclass(out.tree.orig,best=best.size)
class.tree      = predict(out.tree,X_0,type='class')
```

Bagging

Notation

Reminder: For either classification or regression, we produce predictions for a given covariate vector X

That is, we form

$$\hat{Y} = \hat{f}(X)$$

where

- $\hat{f}$ is some procedure formed with the training data
  (Examples: $\hat{\beta}$ formed by least squares, or the discriminant function δ_k for LDA)
- The prediction $\hat{Y}$ formed at a desired covariate vector X
  (Examples: $\hat{Y} = X^\top \hat{\beta}$ formed by least squares, or $\hat{Y} = \arg\max_k \delta_k(X)$ for LDA)

Bagging

Many methods (trees included) tend to be designed to have lower bias but high variance

This means that if we split the training data into two parts at random and fit a decision tree to each part, the results could be quite different

A low variance estimator would yield similar results if applied repeatedly to distinct data sets
(consider $\hat{f}(X) = 0$ for all X)

Bagging, also known as Bootstrap AGgregation, is a general purpose procedure for reducing variance.

We'll use it specifically in the context of trees, but it can be applied more broadly.

Bagging: The main idea

Suppose we have n uncorrelated observations $Z_1, \ldots, Z_n$, each with variance σ^2.

What is the variance of

$$\overline{Z} = \frac{1}{n} \sum_{i=1}^{n} Z_i ?$$

Bagging: The main idea

Suppose we have n uncorrelated observations $Z_1, \ldots, Z_n$, each with variance σ^2.

What is the variance of

$$\overline{Z} = \frac{1}{n} \sum_{i=1}^{n} Z_i ?$$

Answer: σ^2/n.

More generally, if we have B separate (uncorrelated) training sets, we could form B separate model fits,

$$\hat{f}^1(X), \ldots, \hat{f}^B(X)$$

Then average them:

$$\hat{f}_B(X) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^b(X)$$

Bagging: The bootstrap part

Of course, this isn't practical as having access to many training sets is unlikely.

We therefore turn to the bootstrap to simulate having many training sets.

The bootstrap is a widely applicable statistical tool that can be used to quantify uncertainty without Gaussian approximations.

Let's look at an example.

Bootstrap detour

Bootstrap detour

Suppose we are looking to invest in two financial instruments, X and Y. The return on these investments is random, but we still want to allocate our money in a risk minimizing way.

That is, for some $\alpha \in (0, 1)$, we want to minimize

$$\mathrm{Var}(\alpha X + (1 - \alpha) Y)$$

The minimizing α is:

$$\alpha_* = \frac{\sigma_Y^2 - \sigma_{XY}^2}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}^2}$$

(Here, σ_{XY}^2 is the covariance between X and Y)

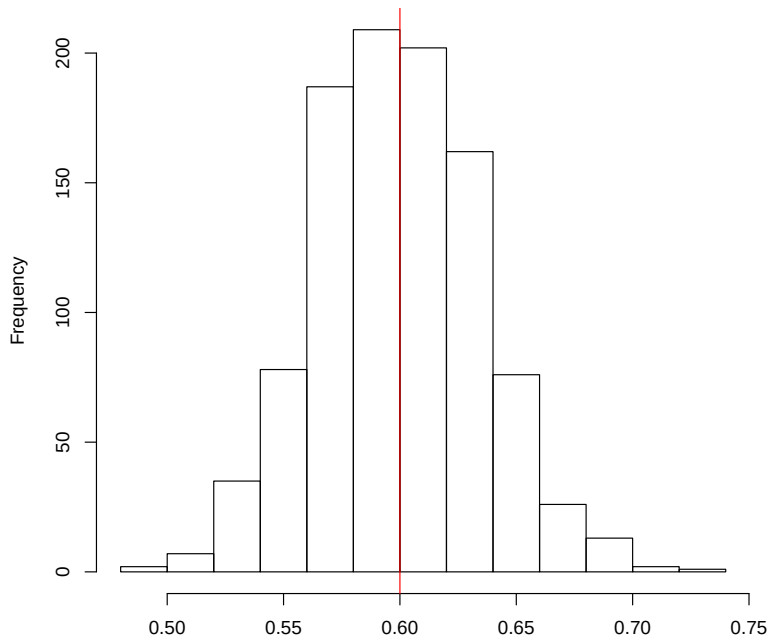
which we can estimate via

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}^2}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}^2}$$

Bootstrap detour

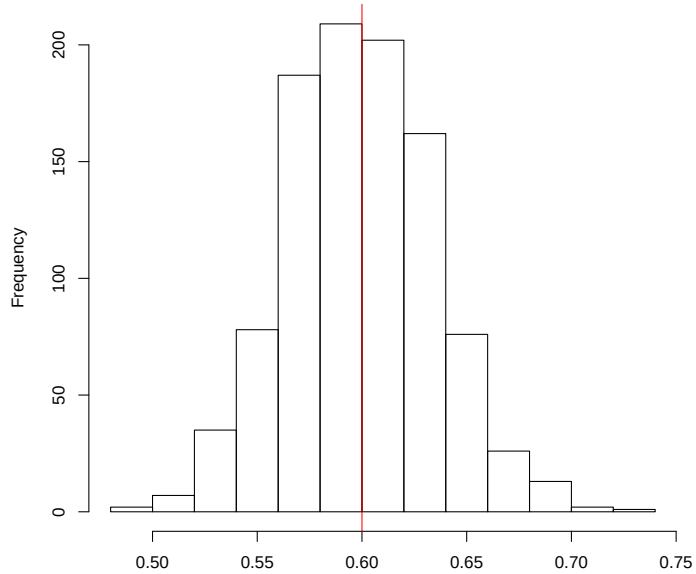
Now that we have an estimator of α, it would be nice to have an estimator of its variability. In this case, computing a standard error is difficult.

Suppose for a moment that we can simulate a large number of draws (say 1000) of the data, which has actual value $\alpha = 0.6$. Then we could get estimates $\hat{\alpha}_1, \ldots, \hat{\alpha}_{1000}$:



This is the sampling distribution of $\hat{\alpha}$

Bootstrap detour



The mean of all of these is:

$$\overline{\alpha} = \frac{1}{1000} \sum_{r=1}^{1000} \hat{\alpha}_r = 0.599,$$

which is very close to 0.6 (red line), and the standard error is

$$\sqrt{\frac{1}{1000-1} \sum_{r=1}^{1000} (\hat{\alpha}_r - \overline{\alpha})^2} = 0.035.$$

Bootstrap detour

The standard error of 0.035 gives a very good idea of the accuracy of $\hat{\alpha}$ for a single sample. Roughly speaking, for a new random sample, we expect $\hat{\alpha} \in (\alpha - 2 * 0.035, \alpha + 2 * 0.035)$.

In practice, of course, we cannot use this procedure as it relies on being able to draw a large number of (independent) samples from the same distribution as our data.

This is where the bootstrap comes in.

We instead draw a large number of samples directly from our observed data. This sampling is done with replacement, which means that the same data point can be drawn multiple times.

Bootstrap detour: Small example

Suppose we have data $\mathcal{D} = (4.3, 3, 7.2, 6.9, 5.5)$.

Then we can draw bootstrap samples, which might look like:

$$\begin{aligned}
\mathcal{D}_1^* &= (7.2, 4.3, 7.2, 5.5, 6.9) \\
\mathcal{D}_2^* &= (6.9, 4.3, 3.0, 4.3, 6.9) \\
&\vdots \\
\mathcal{D}_B^* &= (4.3, 3.0, 3.0, 5.5, 6.9)
\end{aligned}$$

It turns out each of these $\mathcal{D}_b^*$ have very similar properties as $\mathcal{D}$

Bootstrap detour: Small example

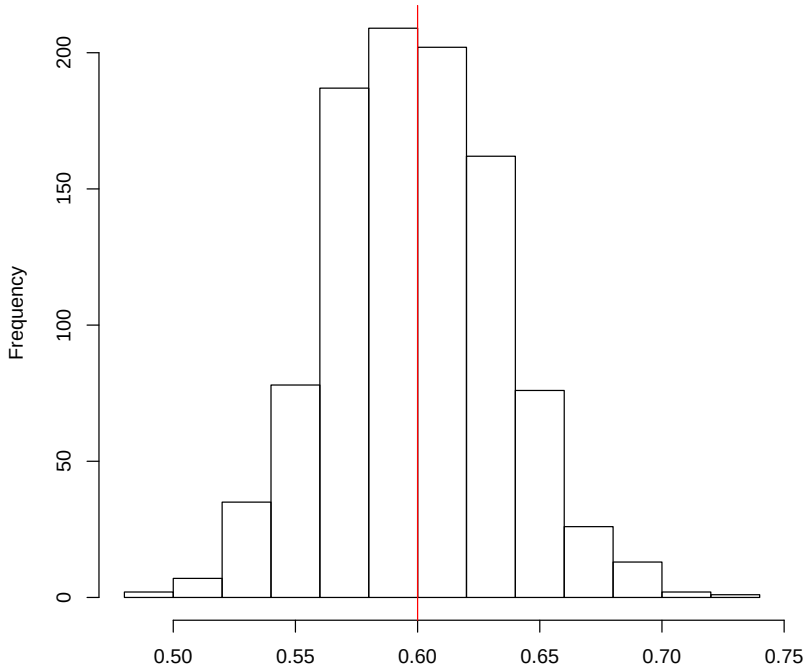
Now, we form the bootstrap mean:

$$\text{mean}_B = \frac{1}{B}\sum_{b=1}^{B}\hat{\alpha}_b^*$$

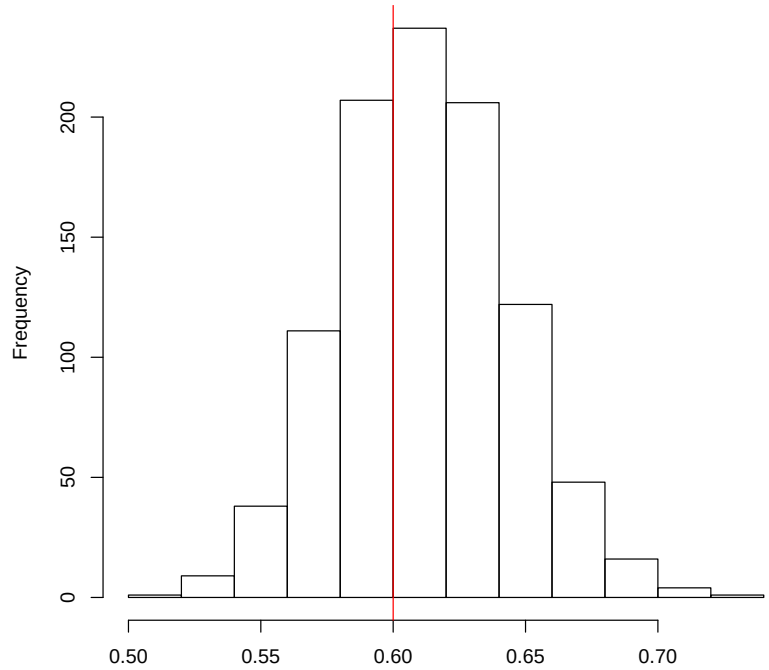
The bootstrap estimator of the standard error is:

$$\text{SE}_B = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}\left(\hat{\alpha}_b^* - \text{mean}_B\right)^2}$$

Bootstrap detour



Sampling distribution of $\hat{\alpha}$ (impossible to form)



Bootstrap sampling distribution of $\hat{\alpha}$ (possible to form)

Bootstrap: End detour

Summary:

Suppose we have data $\mathcal{D} = (Z_1, \ldots, Z_n)$ and we want to get an idea of the sampling distribution of some statistic $\hat{f}$ trained on $\mathcal{D}$.

Then we do the following: For a large number (call it B) of samples:

(B could be, say, 1000)

Then for each $b = 1, \ldots, B$

1. Form a new bootstrap draw from $\mathcal{D}$, call it $\mathcal{D}^*$

Bootstrap: End detour

Summary:

Suppose we have data $\mathcal{D} = (Z_1, \ldots, Z_n)$ and we want to get an idea of the sampling distribution of some statistic $\hat{f}$ trained on $\mathcal{D}$.

Then we do the following: For a large number (call it B) of samples:

(B could be, say, 1000)

Then for each $b = 1, \ldots, B$

1. Form a new bootstrap draw from $\mathcal{D}$, call it $\mathcal{D}^*$
2. Compute $\hat{f}_b^*$ from $\mathcal{D}^*$

Bootstrap: End detour

Summary:

Suppose we have data $\mathcal{D} = (Z_1, \ldots, Z_n)$ and we want to get an idea of the sampling distribution of some statistic $\hat{f}$ trained on $\mathcal{D}$.

Then we do the following: For a large number (call it B) of samples:

(B could be, say, 1000)

Then for each $b = 1, \ldots, B$

1. Form a new bootstrap draw from $\mathcal{D}$, call it $\mathcal{D}^*$
2. Compute $\hat{f}_b^*$ from $\mathcal{D}^*$

Now, we can estimate the distribution of $\hat{f}$ trained on $\mathcal{D}$ by looking at the distribution of the B draws, $\hat{f}_b^*$.

End detour

Bagging: The bootstrap part

Now, instead of having B separate training sets, we train on B bootstrap draws:

$$\hat{f}_1^*(X), \ldots, \hat{f}_B^*(X)$$

and then average them:

$$\hat{f}_{\text{bag}}(X) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b^*(X)$$

This process is known as Bagging

Bagging trees

Bagging trees

The procedure for trees is the following

1. Choose a large number B.
2. For each $b = 1, \ldots, B$, grow an unpruned tree on the b^{th} bootstrap draw from the data.
3. Average all these trees together.

Each tree, since it is unpruned, will have (low/high) variance and (low/high) bias

Bagging trees

The procedure for trees is the following

1. Choose a large number B.
2. For each $b = 1, \ldots, B$, grow an unpruned tree on the b^{th} bootstrap draw from the data.
3. Average all these trees together.

Each tree, since it is unpruned, will have (low/high) variance and (low/high) bias

Therefore averaging many trees results in an estimator that has lower variance and still low bias.

Additional tree bagging topics

Bagging trees

Now that we are growing a large number (B) of random trees, we can't directly look at the dendrogram

This means we have sacrificed some interpretability for better performance

However, we do get some helpful information instead

- Mean decrease variable importance
- Out-of-Bag error estimation (OOB)
- Permutation variable importance

Mean decrease variable importance

To recover some information, we can do the following:

1. For each of the B trees and each of the p variables, we record the amount that the Gini index (or cross-entropy) is reduced by the addition of that variable
2. Report the average reduction over all B trees

This gives us an indication of the importance of a variable

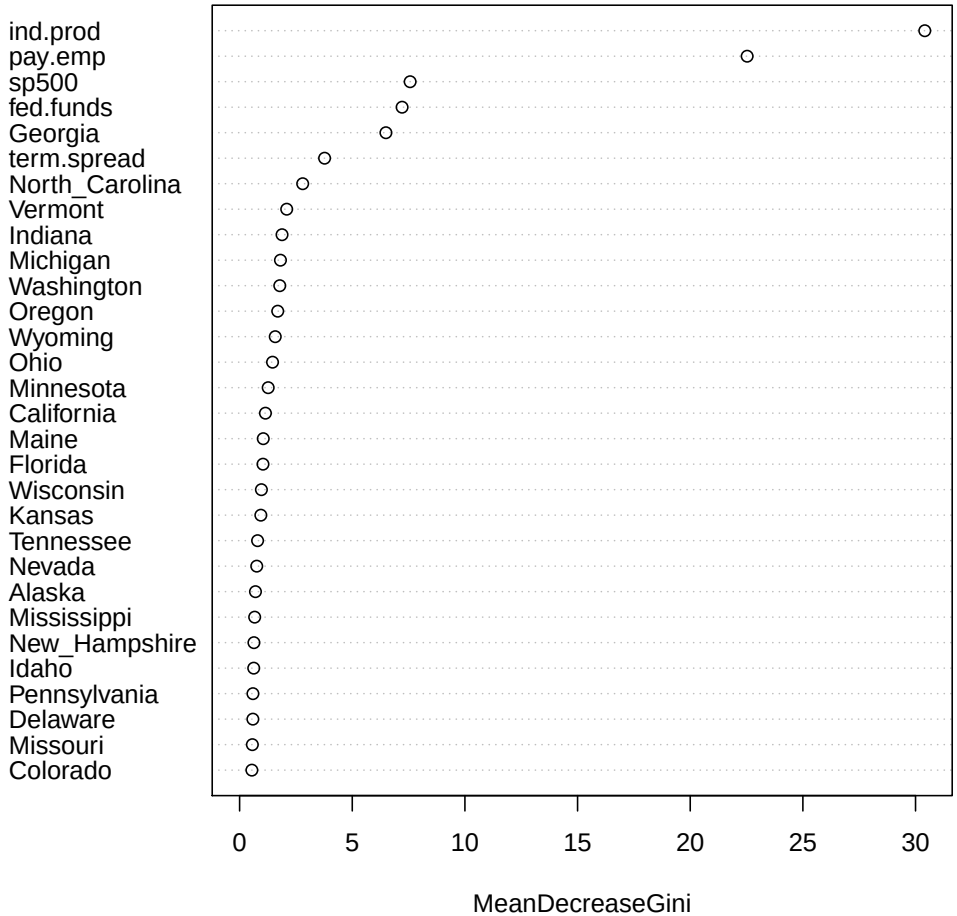
Mean decrease variable importance

Observation: Every time a split of a node is made on a covariate, the gini impurity criterion for the two descendent nodes is less than the parent node

Hence, adding up the gini decreases for each covariate over all trees gives an indication of variable importance

Intuitively an important covariate is one that if split upon, it leads to a large drop in the Gini index

Mean decrease variable importance



Out-of-Bag samples (OOB)

One can show that, on average, drawing n samples from n observations with replacement results in about $2/3$ of the observations being selected.

The remaining one-third of the observations not used are referred to as out-of-bag (OOB)

Out-of-Bag samples (OOB)

We can think of it as a for-free cross-validation

Let $\tilde{\mathcal{D}}^*$ be the observations in $\mathcal{D}$ that aren't in $\mathcal{D}^*$

The observations in $\tilde{\mathcal{D}}^*$ serve as test data

This provides a free estimate of pred for each tree

Permutation variable importance

Consider the b^{th} tree T_b

1. The OOB prediction accuracy of T_b is recorded
2. Then, the j^{th} variable is randomly permuted in the OOB samples
3. The prediction error is recomputed and the change in prediction error is recorded

Intuition: If a variable is highly important, then the OOB prediction error should decrease substantially after permuting the OOB values for the variable

Random forest

Random Forest

Random Forest is a small extension of Bagging, in which the bootstrap trees are decorrelated

The idea is, we draw a bootstrap sample and start to build a tree.

- At each split, we randomly select m of the possible p predictors as candidates for the split.
- A new sample of size m of the predictors is taken at each split.

Usually, we use about $m = \sqrt{p}$

(this would be 7 out of 56 predictors for GDP data)

In other words, at each split, we aren't even allowed to consider the majority of possible predictors!

Random Forest

What is going on here?

Suppose there is 1 really strong predictor and many mediocre ones.

- Then each tree will have this one predictor in it,
- Therefore, each tree will look very similar (i.e. highly correlated).
- Averaging highly correlated things leads to much less variance reduction than if they were uncorrelated.

If we don't allow some trees/splits to use this important variable, each of the trees will be much less similar and hence much less correlated.

Bagging is Random Forest when $m = p$, that is, when we can consider all the variables at each split.

Random forest

An average of B i.i.d random variables has variance

$$\frac{\sigma^2}{B}$$

An average of B random variables has variance

$$\rho\sigma^2 + \frac{(1-\rho)\sigma^2}{B}$$

for correlation ρ

As $B \to \infty$, the second term goes to zero, but the first term remains

Hence, correlation of the trees limits the benefit of averaging

Sensitivity and specificity for recessions

Sensitivity: The proportion of times we label recession, given that recession is the correct answer.

Specificity: The proportion of times we label no recession, given that no recession is the correct answer.

We can think of this in terms of hypothesis testing. If

$$H_0: \text{ no recession},$$

then

Sensitivity: $P(\text{reject } H_0 | H_0 \text{ is false})$, that is: 1 - P(Type II error)

Specificity: $P(\text{accept } H_0 | H_0 \text{ is true})$, that is: 1 - P(Type I error)

Confusion matrix

We can report our results in a matrix:

		Truth	
		Up	Down
Our Predictions	Up	(A)	(B)
	Down	(C)	(D)

For each observation in the test set, we compare our prediction to the truth.

The total number of each combination is recorded in the table.

The overall miss-classification rate is

$$\frac{(\mathrm{B}) + (\mathrm{C})}{(\mathrm{A}) + (\mathrm{B}) + (\mathrm{C}) + (\mathrm{D})} = \frac{(\mathrm{B}) + (\mathrm{C})}{\text{total observations}}$$

What is the sensitivity/specificity?

Confusion matrix

We can report our results in a matrix:

		Truth	
		Up	Down
Our Predictions	Up	(A)	(B)
	Down	(C)	(D)

For each observation in the test set, we compare our prediction to the truth.

The total number of each combination is recorded in the table.

The overall miss-classification rate is

$$\frac{(\mathrm{B})+(\mathrm{C})}{(\mathrm{A})+(\mathrm{B})+(\mathrm{C})+(\mathrm{D})}=\frac{(\mathrm{B})+(\mathrm{C})}{\text{total observations}}$$

What is the sensitivity/specificity?

(Sensitivity is (A)/[(A) + (C)], Specificity is (D)/[(B) + (D)])

Tree results: Confusion matrices

			Truth		
			Growth	Recession	Mis-Class
Our Predictions	Null	Growth	111	26	
		Recession	0	0	18.9%
	Tree	Growth	99	3	
		Recession	12	23	10.9%
	Random Forest	Growth	102	5	
		Recession	9	21	10.2%
	Bagging	Growth	104	3	
		Recession	7	23	7.3%

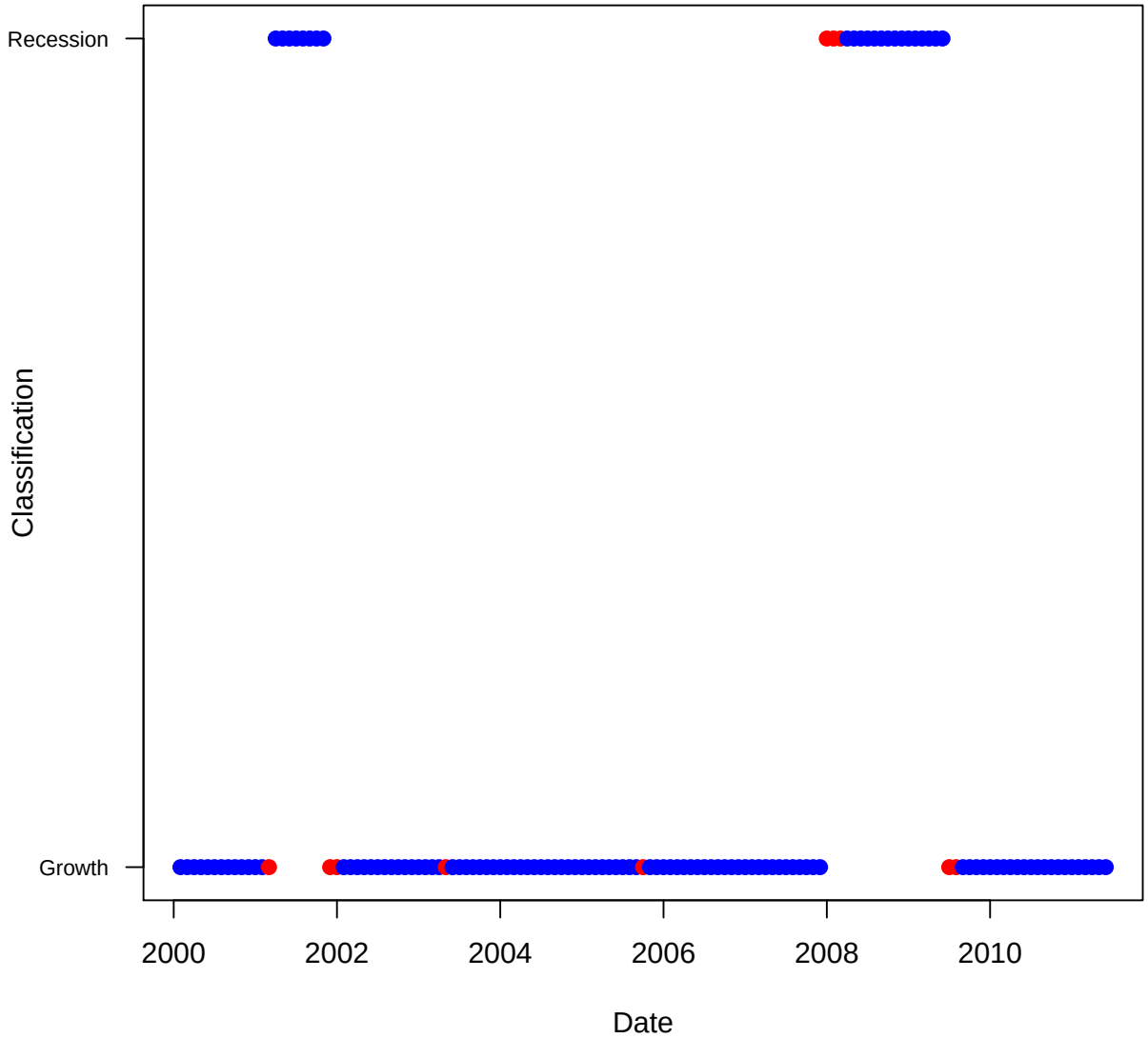
Tree results: Sensitivity & specificity

	Sensitivity	Specificity
Null	0.000	1.000
Tree	0.884	0.891
Random Forest	0.807	0.918
Bagging	0.884	0.936

Out-of-bag error estimation for bagging

		Truth		
		Growth	Recession	Miss-Class
OOB Bagging	Growth	400	10	
	Recession	21	46	6.5%
Test Bagging	Growth	104	3	
	Recession	7	23	7.3%

Results of Bagging on recession data



Random Forest in R

```
require(randomForest)
out.rf   = randomForest(Y~.,data=X,importance=T,mtry=ncol(X))
class.rf = predict(out.rf,X_0)
```

Notes:

- The importance statement tells it to produce the variable importance measures
- the mtry = ncol(X) tells randomForest to consider all the covariates at each split

  (This particular choice corresponds to ??)

Random Forest in R

```
require(randomForest)
out.rf   = randomForest(Y~.,data=X,importance=T,mtry=ncol(X))
class.rf = predict(out.rf,X_0)
```

Notes:

- The importance statement tells it to produce the variable importance measures
- the mtry = ncol(X) tells randomForest to consider all the covariates at each split

  (This particular choice corresponds to ?? (bagging))

Random Forest in R

```
> out.rf

Call:
randomForest(formula = Y ~ ., data = X, import = T, mtry = ncol(
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 56

        OOB estimate of  error rate: 7.33%
Confusion matrix:
    0  1 class.error
0 508 13  0.02495202
1  32 61  0.34408602
```

Random Forest in R

```
#Permutation variable importance
> head(importance(out.rf,type=1))
           MeanDecreaseAccuracy
Alabama                3.7277511
Alaska                 1.7941463
Arizona                2.9659623
Arkansas               0.8341577
California             7.2973572
#Mean decrease variable importance
> head(importance(out.rf,type=2))
           MeanDecreaseGini
Alabama           0.4551073
Alaska            1.6440170
Arizona           0.7025527
Arkansas          0.3503138
California        1.4616203

#variable importance plot:
varImpPlot(out.rf,type=2)
```