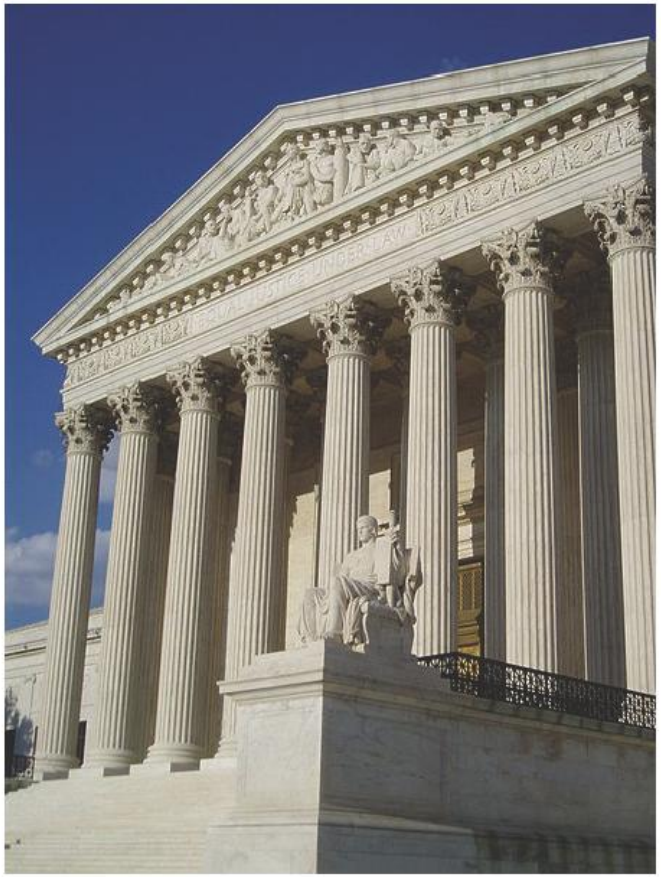
DIMENSIONLESS
TECHNOLOGY

Judge, Jury & Classifier

The American Legal System

- The legal system of the United States operates at the state level and at the federal level
- Federal courts hear cases beyond the scope of state law
- Federal courts are divided into:
  - **District Courts**
    - Makes initial decision
  - **Circuit Courts**
    - Hears appeals from the district courts
  - **Supreme Court**
    - Highest level – makes final decision



The Supreme Court of United States

- Consists of nine judges ("justices"), appointed by the President
  - Justices are distinguished judges, professors of law, state and federal attorneys
- The Supreme Court of the United States (SCOTUS) decides on most difficult and controversial cases
  - Often involve interpretation of Constitution
  - Significant social, political and economic consequences

Notable SCOTUS Decisions

- Wickard v. Filburn (1942)
  - Congress allowed to intervene in industrial/economic activity
- Roe v. Wade (1973)
  - Legalized abortion
- Bush v. Gore (2000)
  - Decided outcome of presidential election!
- National Federation of Independent Business v. Sebelius (2012)
  - Patient Protection and Affordable Care Act ("ObamaCare") upheld the requirement that individuals must buy health insurance

Predicting Supreme Court Cases

- Legal academics and political scientists regularly make predictions of SCOTUS decisions from detailed studies of cases and individual justices
- In 2002, Andrew Martin, a professor of political science at Washington University in St. Louis, decided to instead predict decisions using a statistical model built from data
- Together with his colleagues, he decided to test this model against a panel of experts

Predicting Supreme Court Cases

- Martin used a method called Classification and Regression Trees (CART)
- Why not logistic regression?
  - Logistic regression models are generally not *interpretable*
  - Model coefficients indicate importance and relative effect of variables, but do not give a simple explanation of how decision is made

Quick Question

How much data do you think Andrew Martin should use to build his model?

- Information from all cases with the same set of justices as those he is trying to predict. Data from cases where the justices were different might just add noise to our problem.
- Only information from the most recent year. Since the justices change every year, only this information would be useful.

Classification & Regression Trees

CART

Data

- Cases from 1994 through 2001
- In this period, same nine justices presided SCOTUS
  - Breyer, Ginsburg, Kennedy, O'Connor, Rehnquist (Chief Justice), Scalia, Souter, Stevens, Thomas
  - Rare data set – longest period of time with the same set of justices in over 180 years
- We will focus on predicting Justice Stevens' decisions
  - Started out moderate, but became more liberal
  - Self-proclaimmed conservative

Variables

- **Dependent Variable**: Did Justice Stevens vote to reverse the lower court decision? 1 = reverse, 0 = affirm
- **Independent Variables**: Properties of the case
  - Circuit court of origin (1st – 11th, DC, FED)
  - Issue area of case (e.g., civil rights, federal taxation)
  - Type of petitioner, type of respondent (e.g., US, an employer)
  - Ideological direction of lower court decision (conservative or liberal)
  - Whether petitioner argued that a law/practice was unconstitutional

Collection of Data

•Martin and his colleagues read through all of the cases and coded the information.

•Some of it, like the **circuit court**, is straightforward.

•But other information required a judgment call, like the **ideological direction of the lower court**.

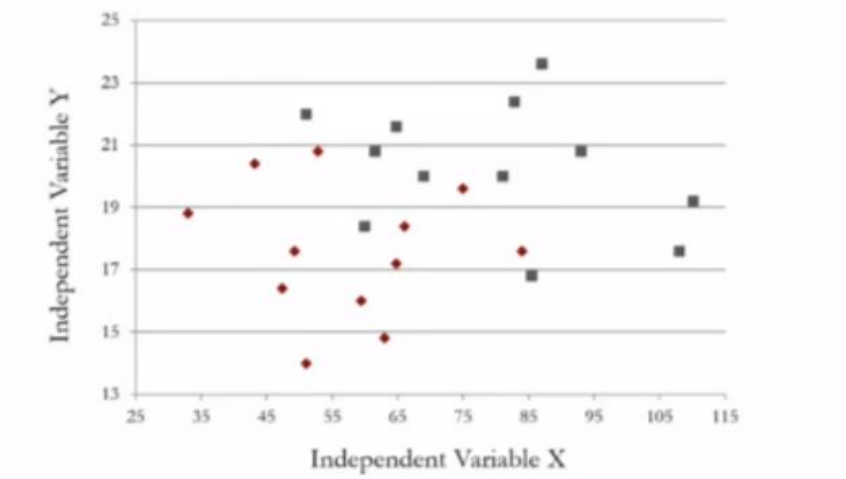
Logistic Regression For Justice Stevens

- Some significant variables and their coefficients:
  - Case is from 2nd circuit court: +1.66
  - Case is from 4th circuit court: +2.82
  - Lower court decision is liberal: -1.22
- This is complicated…
  - Difficult to understand which factors are more important
  - Difficult to quickly evaluate what prediction is for a new case

Classification & Regression Trees

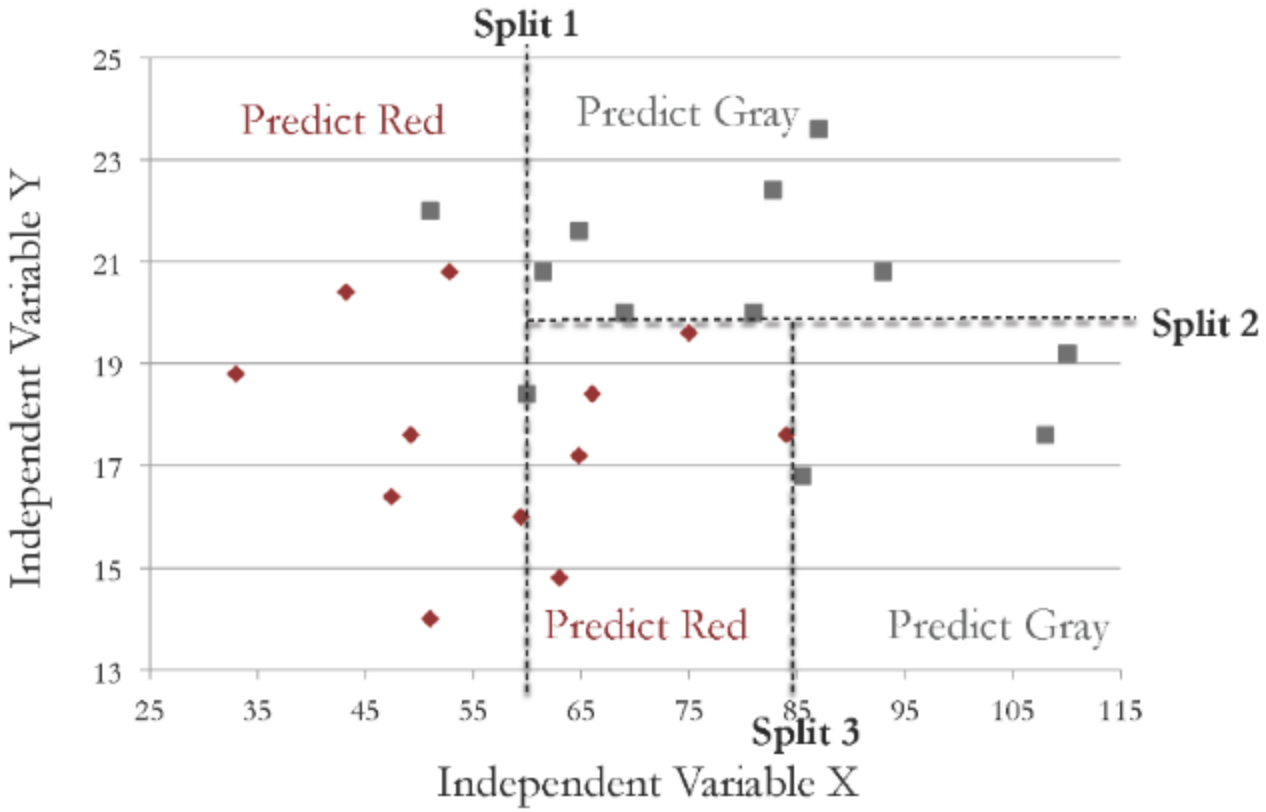
- Build a tree by splitting on variables
- To predict the outcome for an observation, follow the splits and at the end, predict the most frequent outcome
- Does not assume a linear model
- Interpretable

Splits in CART

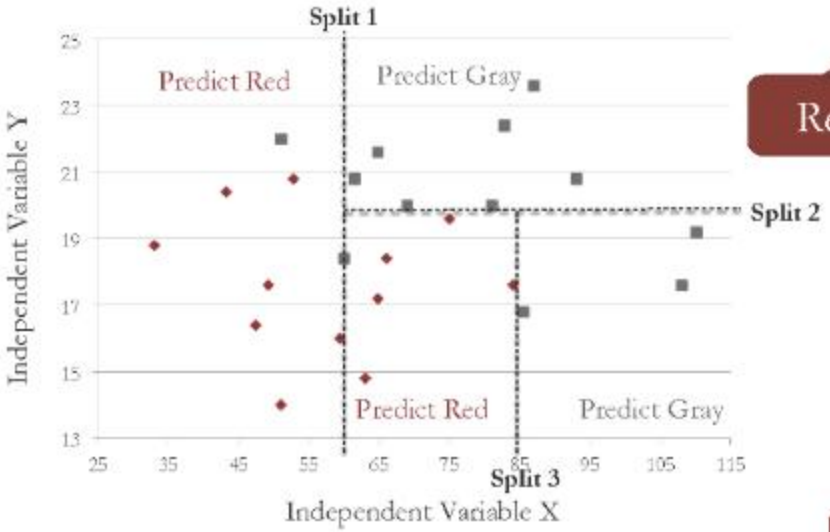


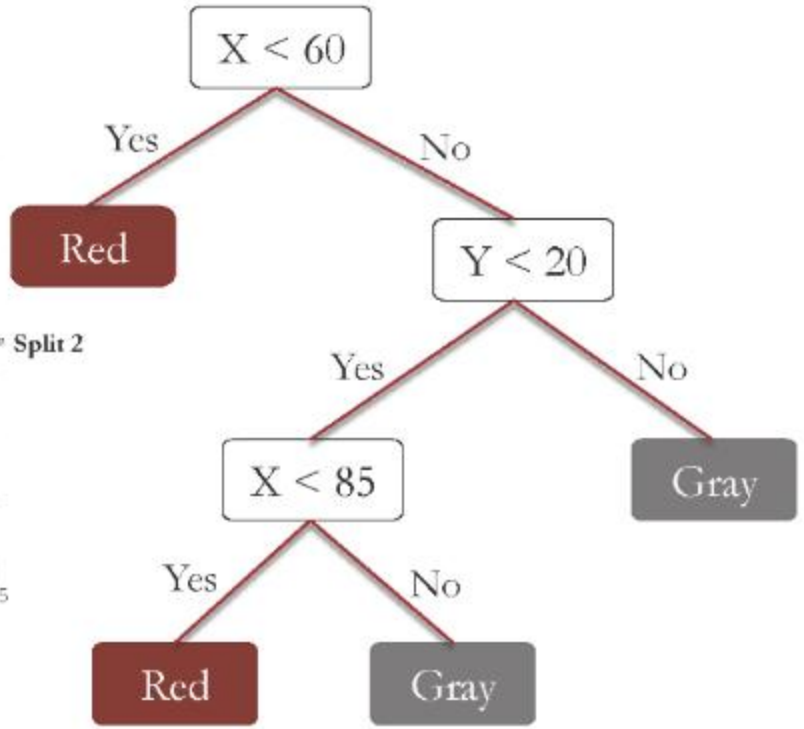
- The plot shows sample data for two independent variables, x and y.
- Each data point is colored by the outcome variable, red or gray.
- CART tries to split this data into subsets
- Each subset is as pure or homogeneous as possible .

Splits in CART

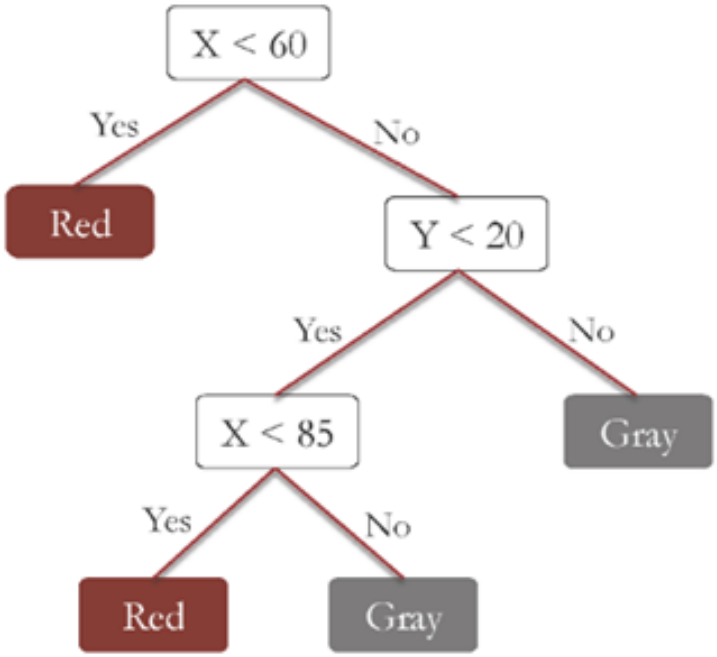


Final Tree


Split 1
Predict Red
Predict Gray
Split 2
Predict Red
Predict Gray
Split 3
Independent Variable Y
Independent Variable X
25
23
21
19
17
15
13
25
35
45
55
65
75
85
95
105
115


X < 60
Yes
No
Red
Y < 20
Yes
No
X < 85
Gray
Yes
No
Red
Gray

Tree -Details



- The first split tests whether the variable x is less than 60.
- If yes, the model says to predict red, and if no, the model moves on to the next split.
- Then, the second split checks whether or not the variable y is less than 20.
- If no, the model says to predict gray, but if yes, the model moves on to the next split.
- The third split checks whether or not the variable x is less than 85.
- If yes, then the model says to predict red, and if no, the model says to predict gray.

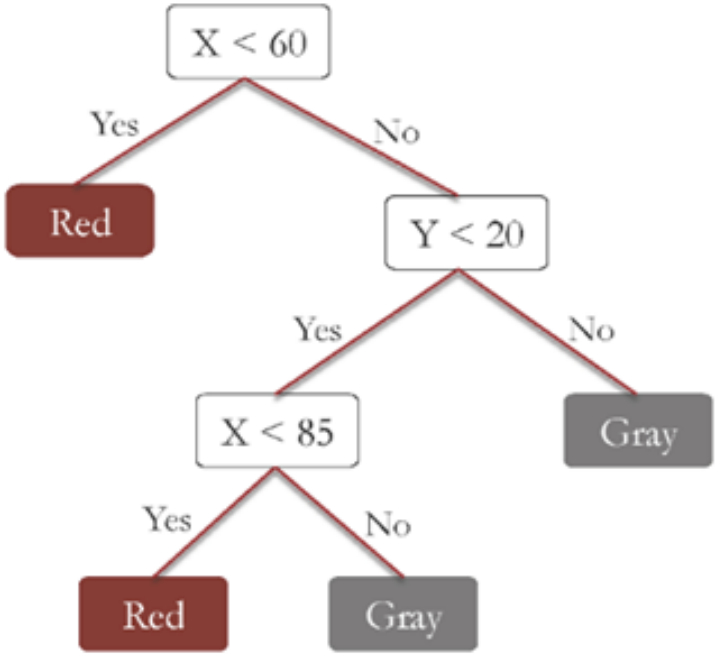
Word of Caution

- A yes response is always to the left and a no response is always to the right.
- Make sure you always start at the top of the tree.
- The x less than 85 split only counts for observations for which x is greater than 60 and y is less than 20.

Quick Question

Suppose that you have the following CART tree:

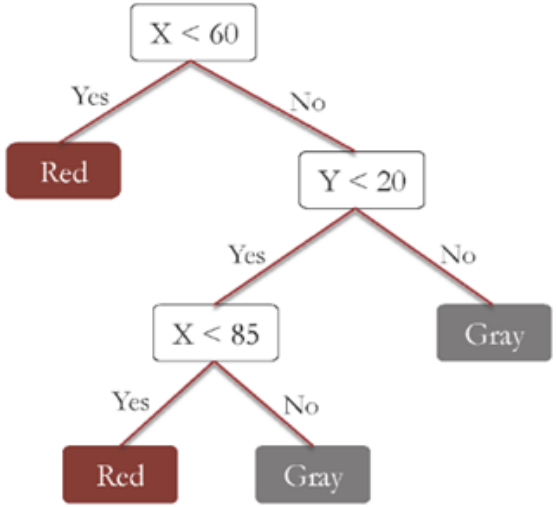
- How many splits are in this tree?



Quick Question

Suppose that you have the following CART tree:

- For which data observations should we predict "Red", according to this tree? Select all that apply.



- [ ] If X is less than 60, and Y is any value.
- [ ] If X is greater than or equal to 60, and Y is greater than or equal to 20.
- [ ] If X is greater than or equal to 85, and Y is less than 20.
- [ ] If X is greater than or equal to 60 and less than 85, and Y is less than 20.

Splitting and Predictions

When Does CART Stop Splitting ?

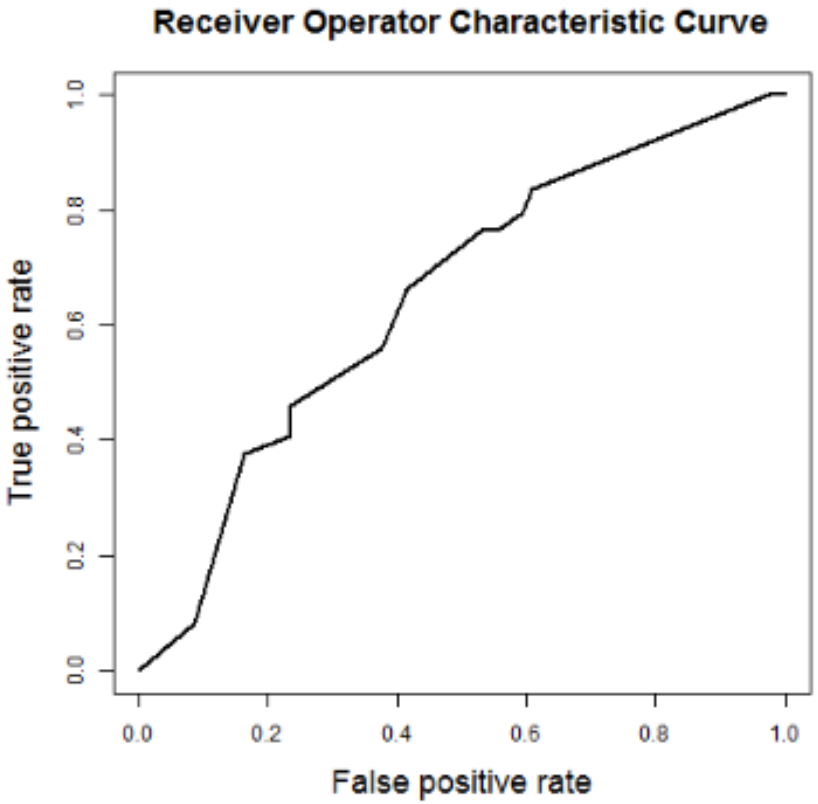
- There are different ways to control how many splits are generated
  - One way is by setting a lower bound for the number of points in each subset
- In R, a parameter that controls this is minbucket
  - The smaller it is, the more splits will be generated
  - If it is too small, overfitting will occur
  - If it is too large, model will be too simple and accuracy will be poor

Predictions From CART

- In each subset, we have a bucket of observations, which may contain both outcomes (i.e., affirm and reverse)
- Compute the percentage of data in a subset of each type
  - Example: 10 affirm, 2 reverse → 10/(10+2) = 0.83
- Just like in logistic regression, we can threshold to obtain a prediction
  - Threshold of 0.5 corresponds to picking most frequent outcome

ROC curve for CART

- Vary the threshold to obtain an ROC curve



Quick Question

Suppose you have a subset of 20 observations, where 14 have outcome A and 6 have outcome B.

- What proportion of observations have outcome A?
- If we set the threshold to 0.25 when computing predictions of outcome A, will we predict A or B for these observations?
- If we set the threshold to 0.5 when computing predictions of outcome A, will we predict A or B for these observations?
- If we set the threshold to 0.75 when computing predictions of outcome A, will we predict A or B for these observations?

Implementing in R

Dataset

- Read the data from the file https://storage.googleapis.com/dimensionless/Analytics/stevens.csv.
- We'll call our dataframe "stevens".
- Look at the structure of the dataset.
- We have 566 observations, or Supreme Court cases, and nine different variables.

Variables

- Docket is just a unique identifier for each case.
- Term is the year of the case.
- We have our six independent variables
  - The **circuit** court of origin.
  - The **issue** area of the case.
  - The type of **petitioner.**
  - The type of **respondent.**
  - The **lower court** direction.
  - Whether or not the petitioner argued that a law or practice was **unconstitutional**.
- The last variable is our dependent variable, whether or not Justice Stevens voted to **reverse** the case: 1 for reverse, and 0 for affirm.

Splitting the Data

- Split the data into a training set and a testing set using the sample.split function.
- Set the seed to 3000.
- We'll call our split "spl".
- Use the split ratio as 70%.
- We'll call our training set "Train".
- We'll call our testing set "Test".

- set.seed(3000)
- spl<-sample.split(stevens$Reverse,0.7)
- Train<-subset(stevens,spl==TRUE)
- Test<-subset(stevens,spl==FALSE)

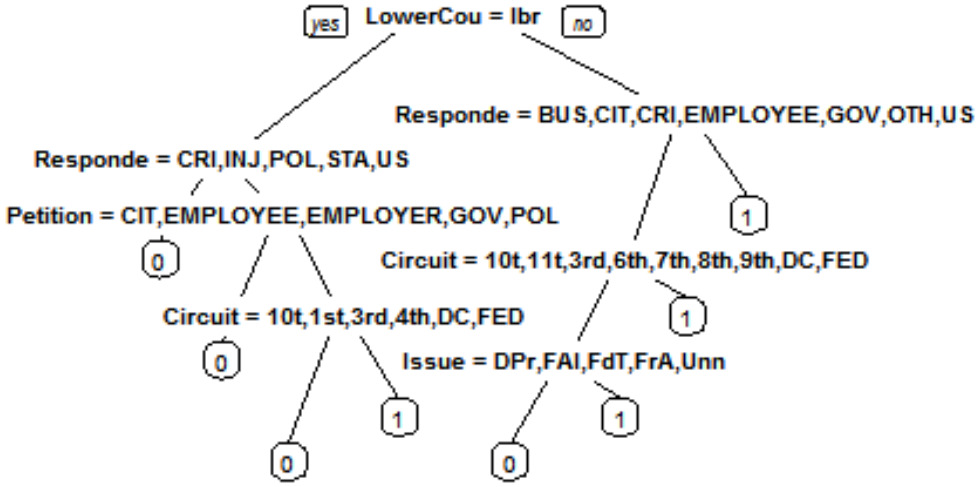
Building the Model

- Install and load the “rpart” package and the “rpart.plot” plotting package.
- Now we can create our CART model using the rpart function.
- We'll call our model StevensTree.
- The first two arguments are the same as if we were building a linear or logistic regression model (formula,data)
- Now we'll give two additional arguments here.
  - method = "class". This tells rpart to build a classification tree, instead of a regression tree.
  - minbucket = 25. This limits the tree so that it doesn't overfit to our training set.

```
> StevensTree<-rpart( Reverse ~ Circuit + Issue + Petitioner + Respondent + LowerCourt + Unconst, data=Train,method ="class",minbucket=25)
```

Plotting the TREE

- We will plot our tree using the prp function.
- The only argument is the name of our model, StevensTree.
  - prp(StevensTree)
- You should see the tree pop up in the graphics window.



Understanding the TREE

- The first split of our tree is whether or not the lower court decision is liberal.
- If it is, then we move to the left in the tree.
- Then we check the respondent. If the respondent is a criminal defendant, injured person, politician, state, or the United States, we predict 0, or affirm.
- If the respondent is not one of these types, we move on to the next split, and we check the petitioner.
- If the petitioner is a city, employee, employer, government official, or politician, then we predict 0, or affirm.

Understanding the TREE

- If not, then we check the circuit court of origin.
- If it's the 10th, 1st, 3rd, 4th, DC or Federal Court, then we predict 0.
- Otherwise, we predict 1, or reverse.
- We can repeat this same process on the other side of the tree if the lower court decision is not liberal.
- The prp function abbreviates the values of the independent variables.
- Comparing this to a logistic regression model, we can see that it's very interpretable.

Making Predictions

- We'll call our predictions PredictCART, and we'll use the predict function.
- The first argument is the name of our model, StevensTree.
- The second argument is the new data we want to make predictions for, Test.
- And we'll add a third argument here, which is type = "class".
- We need to give this argument when making predictions for our CART model if we want the majority class predictions.
- This is like using a threshold of 0.5.

```
>PredictCart<-predict(StevensTree,newdata = Test,type="class")
```

Accuracy of the model

- Compute the accuracy of our model by building a confusion matrix.
  - table(Test$Reverse,PredictCart)

```
   PredictCart
     0  1
0 41 36
1 22 71
```

- Accuracy = (41+71)/(41+71+22+36) = 0.659
- A logistic regression model, would get an accuracy of 0.665.
- A baseline model that always predicts Reverse, the most common outcome, has an accuracy of 0.547.
- So our CART model significantly beats the baseline and is competitive with logistic regression.

ROC Curve for CART

- To evaluate our model, let's generate an ROC curve.
- We need to generate our predictions again, this time without the type = "class" argument.
- We'll call them PredictROC.
- Look at what this looks like by just typing PredictROC.
- For each observation in the test set, it gives two numbers which can be thought of as the probability of outcome 0 and the probability of outcome 1.
- These numbers give the percentage of training set data in that subset with outcome 0 and the percentage of data in the training set in that subset with outcome 1.

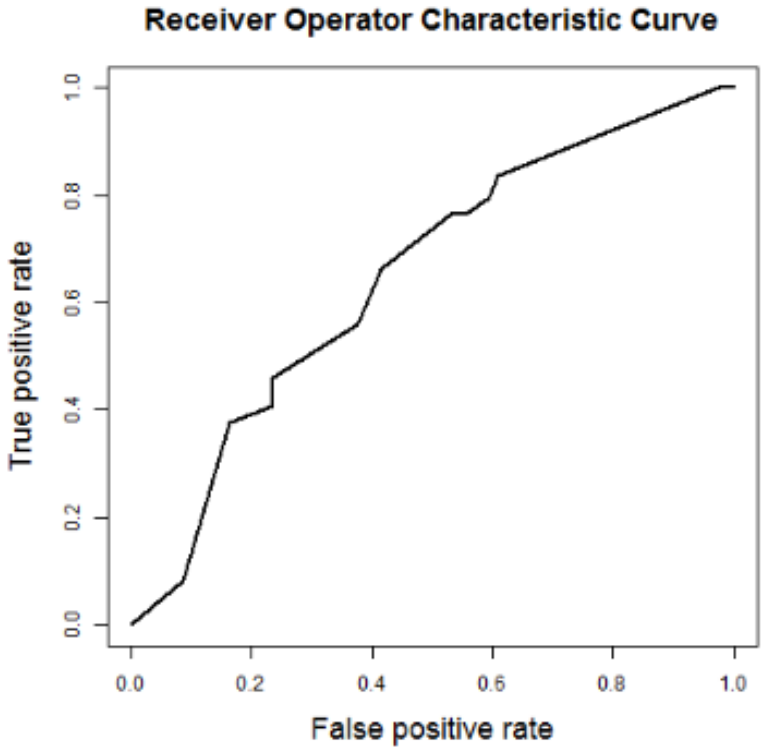
ROC Curve for CART

- We'll use the second column as our probabilities to generate an ROC curve.
- We'll start by using the prediction function. We'll call the output ROCRpred.
- Then we will use the performance function, we'll call it ROCRperf.
- Now we can just plot our ROC curve by typing plot(ROCRperf).

```
>ROCRpred<-prediction(PredictROC[,2],Test$Reverse)
>ROCRperf<-performance(ROCRpred,'tpr','fpr')
>plot(ROCRperf,colorize=TRUE)
```

ROC Curve for CART

- Vary the threshold to obtain an ROC curve



Quick Question

Compute the AUC of the CART model made earlier?

Build a CART model that is similar to the one we built, except change the minbucket parameter to 5. Plot the tree.How many splits does the tree have?

Now build a CART model that is similar to the one we built, except change the minbucket parameter to 100. Plot the tree. How many splits does the tree have?

Random Forests

Random Forests

- Designed to improve prediction accuracy of CART
- Works by building a large number of CART trees
  - Makes model less interpretable
- To make a prediction for a new observation, each tree "votes" on the outcome, and we pick the outcome that receives the majority of the votes

Building Many Trees

- Each tree can split on only a random subset of the variables
- Each tree is built from a "bagged"/"bootstrapped" sample of the data
  - Select observations randomly with replacement
  - Example – original data: 1 2 3 4 5
  - New "data":

    2 4 5 2 1 ⟶ 1st tree

    3 5 1 5 2 ⟶ 2nd tree

    ⋮

Random Forest Parameters

- Minimum number of observations in a subset
  - In R, this is controlled by the nodesize parameter
  - Smaller nodesize may take longer in R
- Number of trees
  - In R, this is the ntree parameter
  - Should not be too small, because bagging procedure may miss observations
  - More trees take longer to build

Implementing in R

randomForest

- Install and load the package "randomForest".
- We'll call our model "StevensForest" and use the randomForest function.
- Syntax is similar to lm() or glm().
- We need to give 2 additional arguments
  - nodesize=25 (minbucket for CART)
  - Ntree=200 (no. of trees, couple of hundred trees are sufficient)

```
>StevensForest<- randomForest(Reverse~ Circuit + Issue + Petitioner + Respondent + LowerCourt + Unconst, data=Train, nodesize=25, ntree=200)
```

- You should see an interesting warning message here.

randomForest

- There is no argument to choose between classification and regression tree like method="class" for CART function.
- So we will convert our dependent variable for Train and Test data to a factor to get a factor outcome.

```
>Train$Reverse<-as.factor(Train$Reverse)
>Test$Reverse<- as.factor(Test$Reverse)
```

Making Predictions

- Let's compute predictions on our test set.
- We'll call our predictions PredictForest.
- We'll use predict function on our model StevensForest and dataset Test

```
> PredictForest<-predict(StevensForest,newdata = Test)
```

- Look at the confusion matrix to compute our accuracy.

```
>table(Test$Reverse,PredictForest)
```

```
  PredictForest
    0  1
0 39 38
1 19 74
```

- Accuracy = (39+74)/(39+38+19+74)

Comparing the Models

- The accuracy of our Random Forest model is about 67%.
- Our logistic regression model had an accuracy of 66.5%.
- Our CART model had an accuracy of 65.9%.
- So our random forest model improved our accuracy a little bit over CART.
- Sometimes you'll see a smaller improvement in accuracy and sometimes you'll see that random forests can significantly improve in accuracy over CART.
- Random Forests has a random component. You may have gotten a different confusion matrix than me.

Quick Question

Let's see what happens if we set the seed to two different values and create two different random forest models.

- First, set the seed to 100, and the re-build the random forest model. Then make predictions on the test set. What is the accuracy of the model on the test set?
- Now, set the seed to 200, and then re-build the random forest model. Then make predictions on the test set. What is the accuracy of this model on the test set?

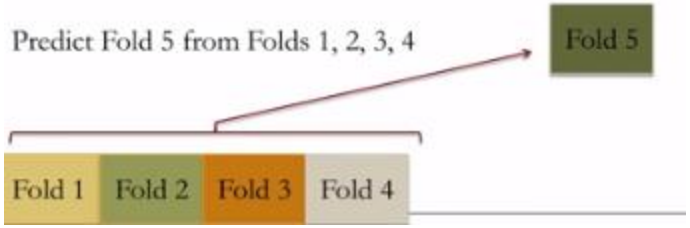
Cross Validation

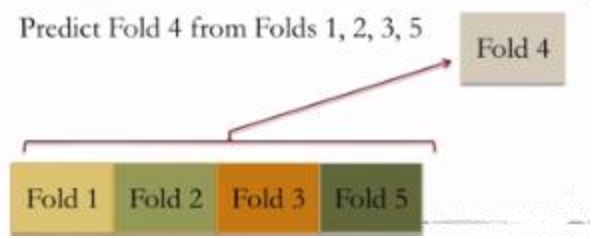
Parameter Selection

- In CART, the value of "minbucket" can affect the model's out-of-sample accuracy
- How should we set this parameter?
- We could select the value that gives the best testing set accuracy
  - This is not right!

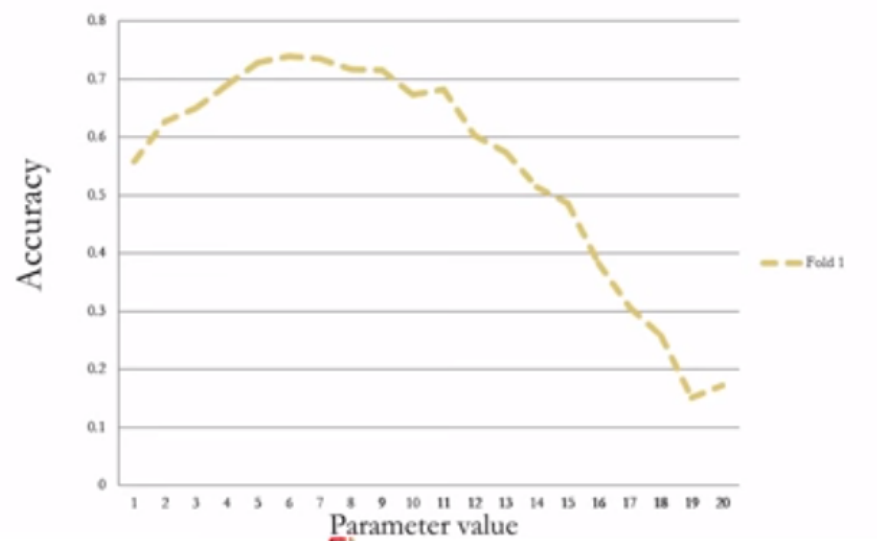
K-Fold Cross-Validation

- Given training set, split into k pieces (here k = 5)
- Use k-1 folds to estimate a model, and test model on remaining one fold ("validation set") for each candidate parameter value
- Repeat for each of the k folds



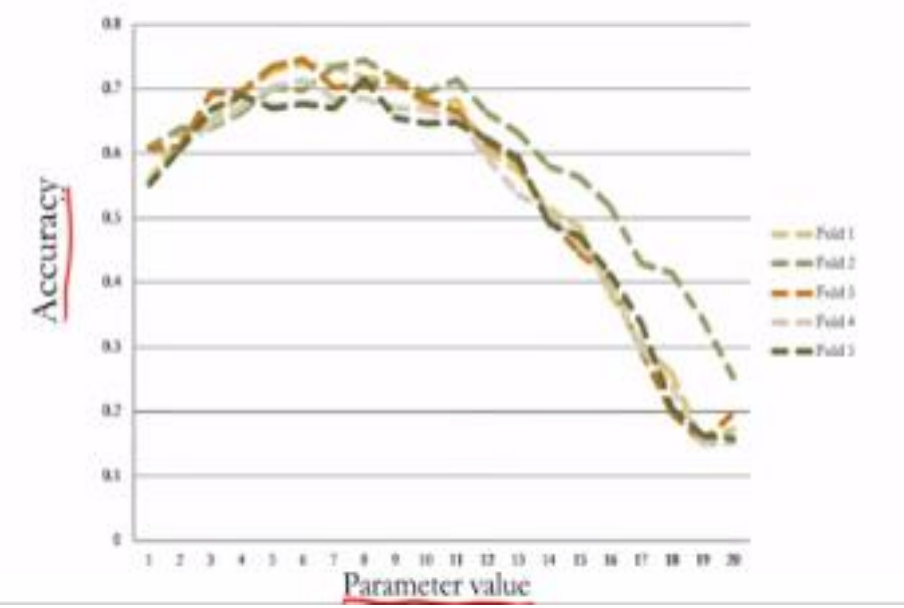


Output of k-fold Cross Validation



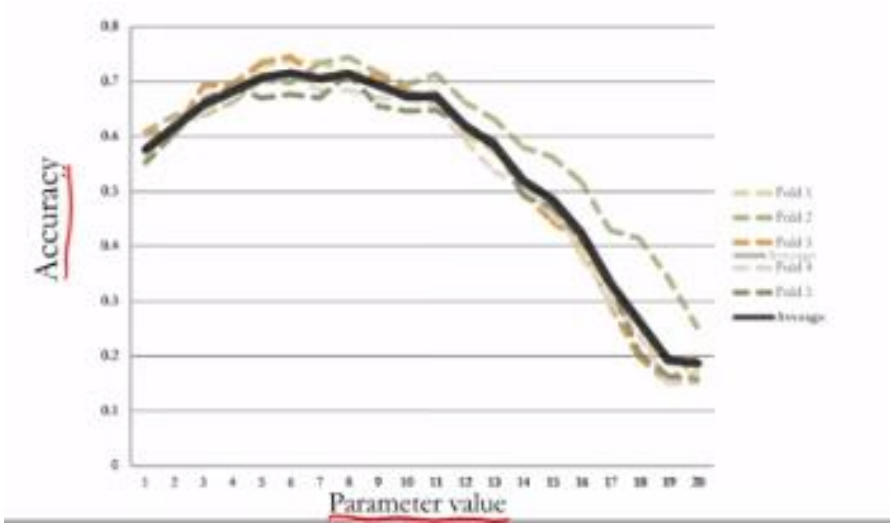
- This plot shows the possible parameter values on the x-axis, and the accuracy of the model on the y-axis.
- This line shows the accuracy of our model on fold 1.

Output of k-fold Cross Validation



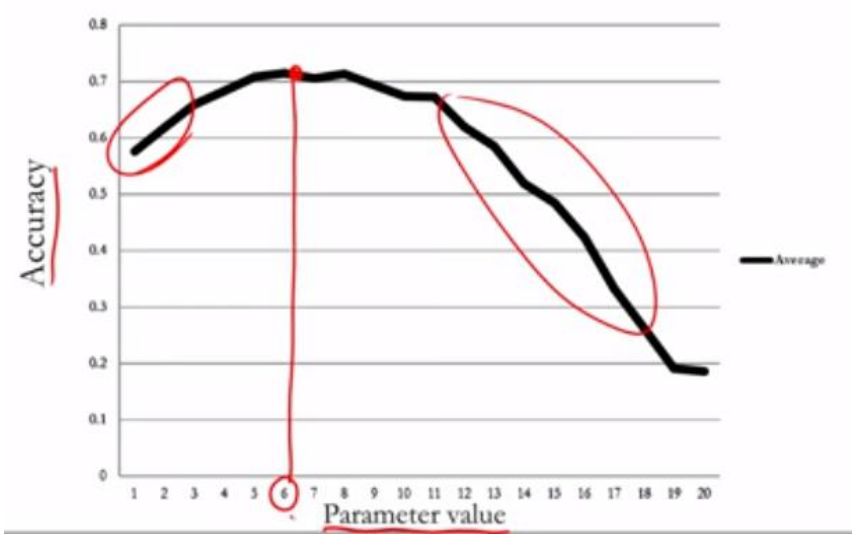
- We can also compute the accuracy of the model using each of the other folds as the validation sets.

Output of k-fold Cross Validation



- We then average the accuracy over the k-folds to determine the final parameter value that we want to use.

Output of k-fold Cross Validation



- Typically, the behavior looks like this.
- If the parameter value is too small, then the accuracy is lower, because the model is probably over-fit to the training set.
- But if the parameter value is too large, then the accuracy is also lower, because the model is too simple.
- In this case, we would pick a parameter value around six, because it leads to the maximum average accuracy over all parameter values.

Cross-Validation in R

- Before, we limited our tree using minbucket
- When we use cross-validation in R, we'll use a parameter called cp instead
  - Complexity Parameter
- Like Adjusted R^2 and AIC
  - Measures trade-off between model complexity and accuracy on the training set
- Smaller cp leads to a bigger tree (might overfit)

Cross-Validation in R

- We need to install and load two new packages "caret" and "e1071".
- Also install "ggplot2" and "class" packages, in case the R console is giving error.
- Now, we'll define our cross validation experiment.
- First, we need to define how many folds we want.
- We can do this using the trainControl function.

```
> numFolds<-trainControl(method="cv", number = 10)
```

Quick Question

Plot the tree that we created using cross-validation. How many splits does it have?

Ans:- 1

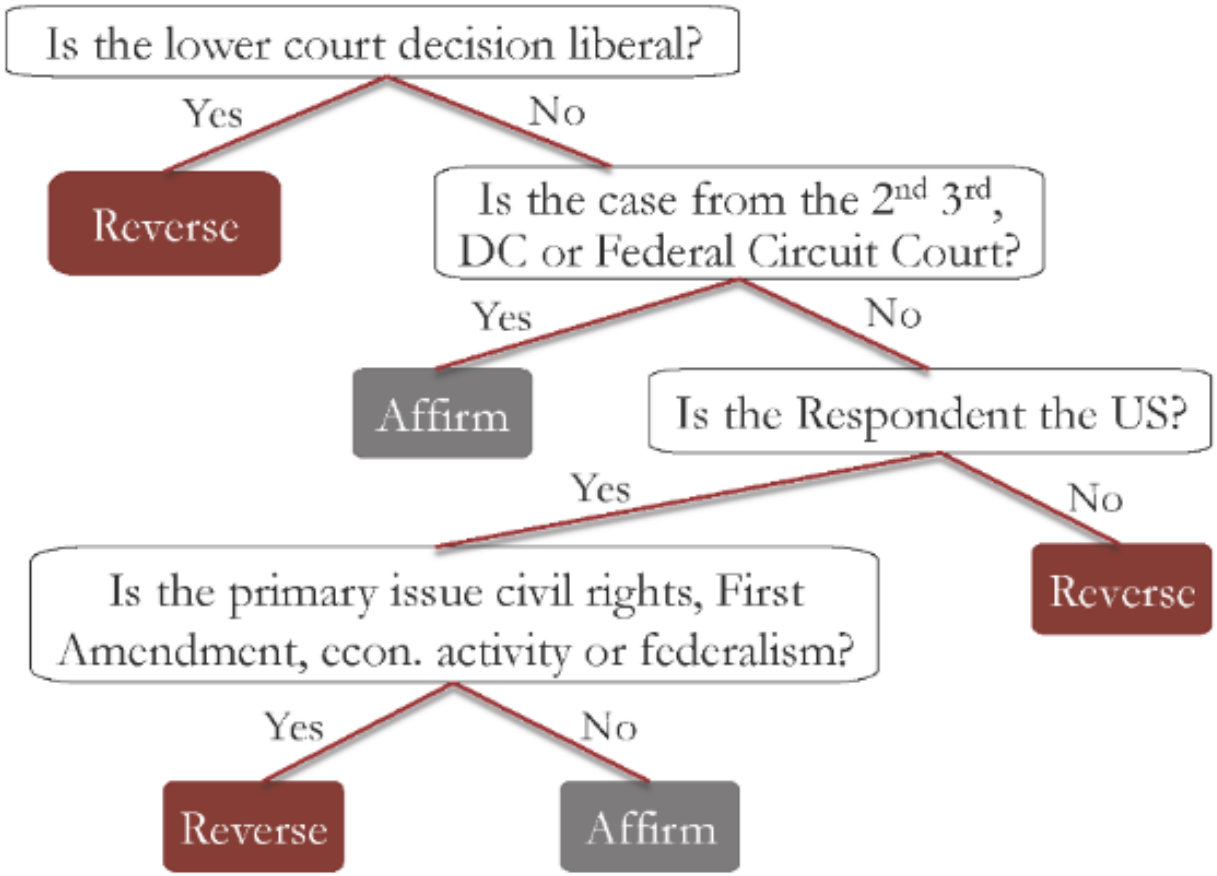
The tree with the best accuracy only has one split! When we were picking different minbucket parameters before, it seemed like this tree was probably not doing a good job of fitting the data. However, this tree with one split gives us the best out-of-sample accuracy. This reminds us that sometimes the simplest models are the best!

The Model v. The Experts

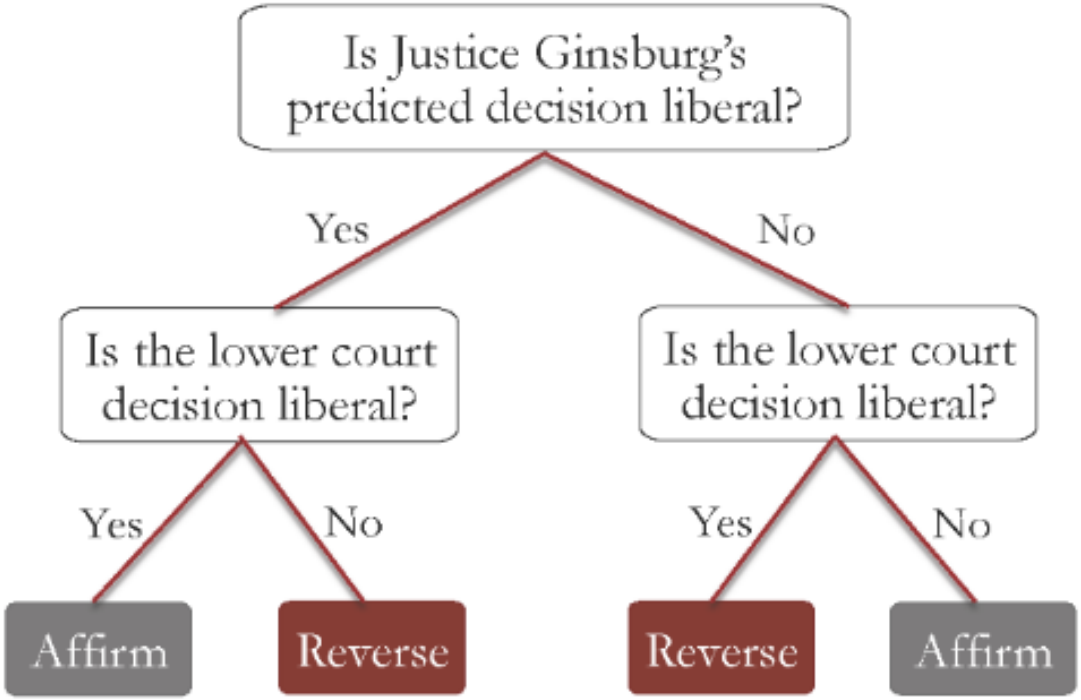
Martin's Model

- Used 628 previous SCOTUS cases between 1994 and 2001
- Made predictions for the 68 cases that would be decided in October 2002, before the term started
- Two stage approach based on CART:
  - First stage: one tree to predict a unanimous liberal decision, other tree to predict unanimous conservative decision
    - If conflicting predictions or predict no, move to next stage
  - Second stage consists of predicting decision of each individual justice, and using majority decision as prediction

Tree for Justice O'Connor



Tree for Justice Souter



"Make a liberal decision"

"Make a conservative decision"

The Experts

- Martin and his colleagues recruited 83 legal experts
  - 71 academics and 12 attorneys
  - 38 previously clerked for a Supreme Court justice, 33 were chaired professors and 5 were current or former law school deans
- Experts only asked to predict within their area of expertise; more than one expert to each case
- Allowed to consider any source of information, but not allowed to communicate with each other regarding predictions

The Results

- For the 68 cases in October 2002:
- Overall case predictions:
  - Model accuracy: 75%
  - Experts accuracy: 59%
- Individual justice predictions:
  - Model accuracy: 67%
  - Experts accuracy: 68%

The Analytics Edge

- Predicting Supreme Court decisions is very valuable to firms, politicians and non-governmental organizations
- A model that predicts these decisions is both more accurate and faster than experts
  - CART model based on very high-level details of case beats experts who can process much more detailed and complex information