Forecasting Supreme Court Decisions

Tool : Classification and regression trees (CART), Random Forests

The Analytics Edge :

Political scientists and legal academics constantly seek to understand what motivates the justice decision of courts. In 2002 a group of academic predicted the decisions of the US Supreme Court using information on past Supreme Court decisions and compared it to the expert judgements of legal academics & professionals. Using an interpretable analytics technique known as Classification & Regression Trees, the model got 75 % of the Court's affirm / reverse results correct while experts collectively got 59.1 % right. At the individual justice level, the model got 66.7 % decisions correct while experts got 67.9 % correct. Using analytics can provide an edge in a traditionally qualitative application here. At the individual justice level, the experts got better predictions on the more "predictable" justice votes while the model did a better job on the more "unpredic[illegible] justices. Combining the two approaches - the consistent and unemotional nature of a model with the intuition & knowl[illegible] of experts might be a reasonable approach in this case.

Background of the federal judiciary of US

Highest federal court

Supreme Court

Chief Justice +
8 associate justices

Each justice has one vote and while many cases are decided unanimously, many high profile cases expose ideological beliefs.
For example justices might be categorized as conservative or liberal based on their votes in previous cases.
The Supreme Court decisions often impact a variety of social, economic, structural questions.

For example in the 2002 Term Court some of the important cases were:

1) Grutter vs Bollinger
This case dealt with the constitutionality of affirmative action (Policy of favoring members of a disadvantaged group who suffers from discrimination)
This case was upheld by the Supreme Court ruling that the affirmative action policy of the University of Michigan law School was justified. Case was affirmed 5 – 4.

2) Lawrence vs Texas

This case dealt with the right to engage in consensual homosexual sodomy.

In a <u>6-3 ruling</u> the Court <u>reversed</u> and struck down the sodomy law in Texas making same sex sexual activity legal in the US states.

Many of the court decisions are hard to predict. Martin and his colleagues predicted the 2002 Term court decisions using simple prediction variables. The model is indifferent to many of the specific legal and factual aspects of the case that legal experts might use in making predictions.

Estimate   Predict

$\underbrace{1994 \text{ ——————— } 2001}$ $\underbrace{2002}$

Same nine justices were involved in this period

(One of the longest periods of time with the same justices - greater availability of data)

Question : Is it possible to predict whether the Supreme Court Judges will affirm or reverse the lower court decision (at the individual judge level, at the overall case level) using simple analytics ?

Circuit of origin
(1 - 11 , DC , FED)

Issue area of case
(Criminal procedure ,
Civil rights , First amendment
... , Economic activity , Federal
taxation)

Petitioner type
(Business , city , employee ,
American Indian , Politician , ...)

Respondent type
(Business , city , employee ,
American Indian , Politician , ...)

Ideological direction of
lower court ruling
(Liberal or Conservative)

Petitioner argued that law
or practice was constitutional
(Yes or No)

Reverse the lower court decision

1 = reverse

0 = affirm

Classification & regression trees

Suppose you want to predict a value or response Y from predictors $X_1, \ldots, X_p$.

Models such as linear regression or logistic regression are global models where the single predictive formula is supposed to hold over the entire data region with linear dependence of the dependent variable on the independent variables. One might be able to use nonlinear regression techniques if the data interacts in complex, nonlinear ways. However these models are often less interpretable.

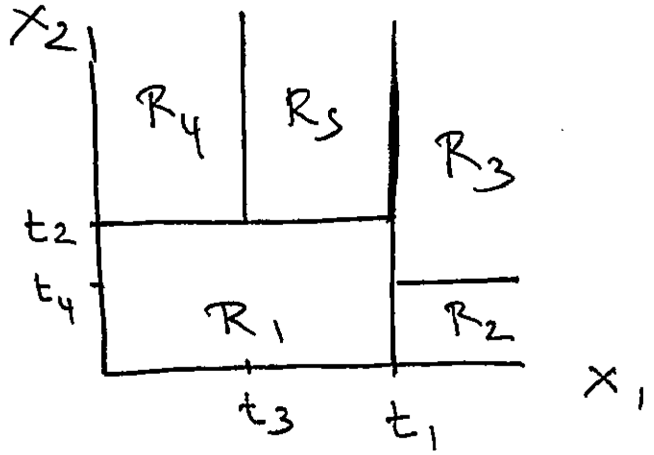
Alternative approach

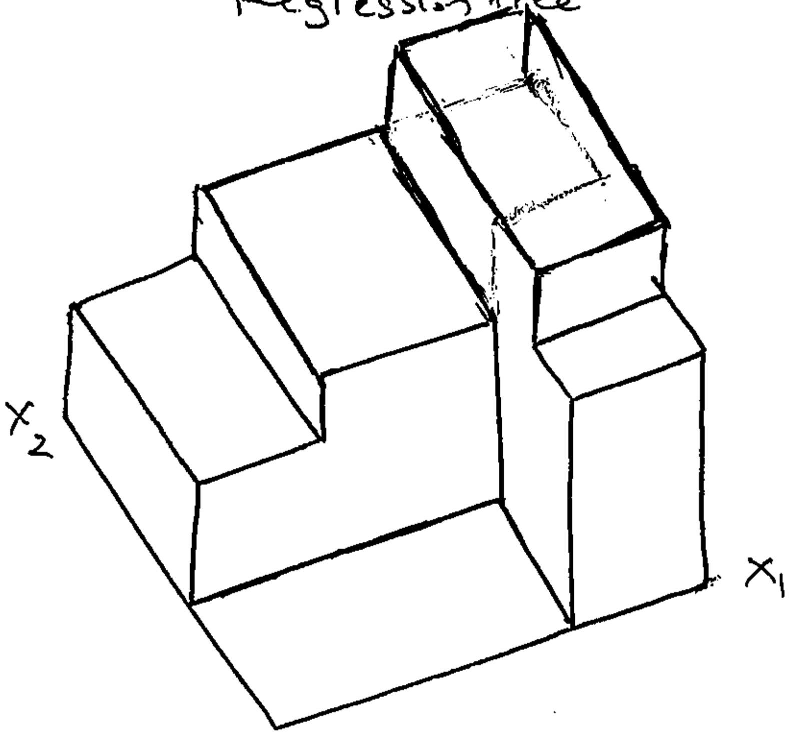
Divide (partition) the predictor space into a number of simpler regions and use simple rules such as using the mean of the observations (or mode) in the space partitioned to make predictions.

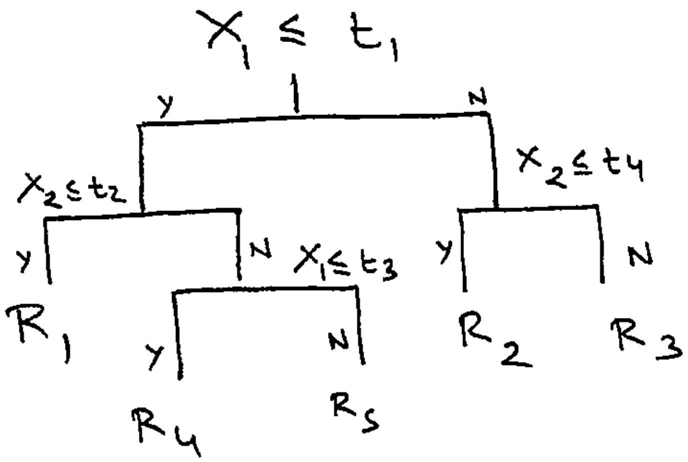
By partitioning the sub-regions again (recursive partitioning), we can develop regions where the models within regions are very simple.

Basics

Regression tree







Model

$$f(x) = \sum_{m=1}^{M} \underbrace{w_m}_{\text{mean response in region}} \mathbb{I}(x \in \underbrace{R_m}_{\text{region}})$$

$X_1 \le t_1$ If Yes, check if $x_2 \le t_2$

If yes, we are in region R_1

If no, check if $x_1 \le t_3$

If Yes, region R_4

If No, region R_5

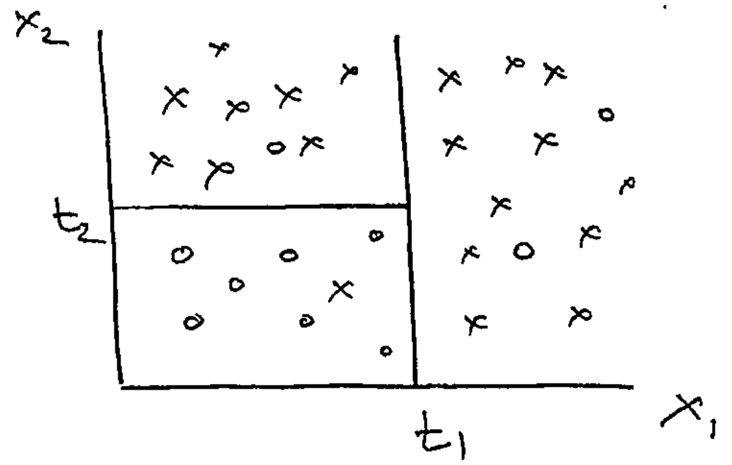
If No, check if $X_2 \le t_4$

If Yes, region R_2

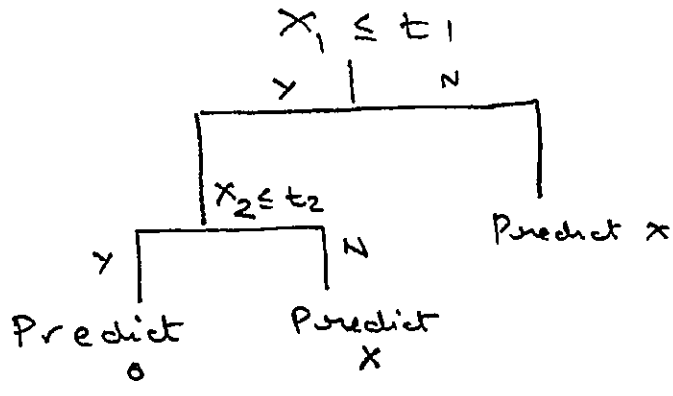
If No, region R_3

Axis parallel splits (splits entire region into regions R_1 to R_5)

Basics



Classification tree



Here the CART model predicts x or o based on the splits and the majority number in each split.

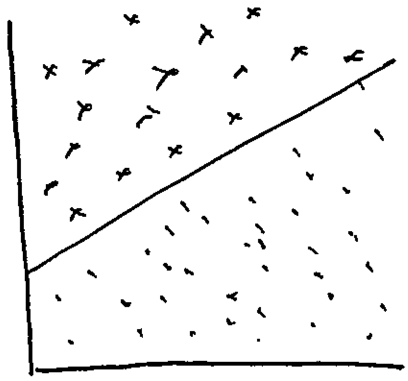
To figure out how many splits are needed, one approach is to select the minimum number of points needed in each subset. If this is too small, fits training data very well but the performance on test set might not be good. This is captured by minibucket parameter in R. in the rpart package.

Note that for prediction with binary outcome we can use threshold to choose which outcome to predict (The majority example previously corresponds to threshold of 0.5)

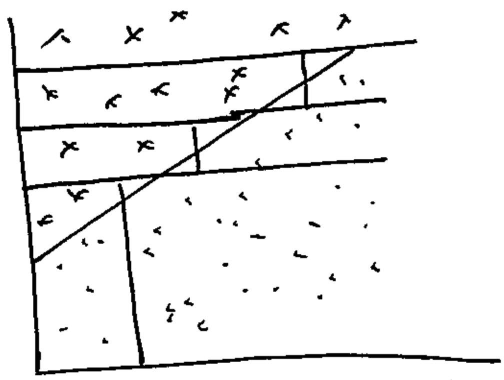
Advantages of CART

1) Model is easily interpretable even to non-experts

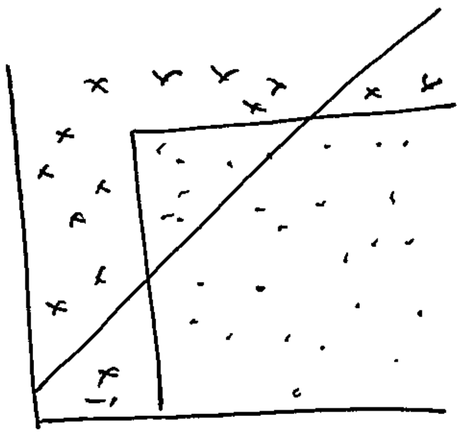
2) Model does not assume a linear relationship between input and output variables and hence captures nonlinearities in data.



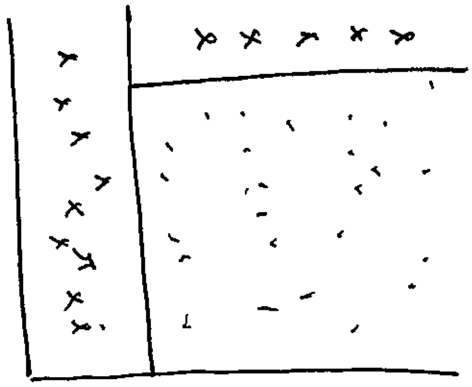
Linear model works well



CART does not work well



Linear model does not work well



CART works well.

Technique to find the tree in CART

Divide the set of possible values of the predictor variables $(X_1, \dots, X_p)$ into M distinct and non-overlapping regions $R_1, \dots, R_m$.

In CART, these regions are defined by rectangles (boxes, high dimension rectangles) where the cuts are parallel to the axes.

Residual sum of squares

$$\min_{R_1,\dots,R_m} \sum_{m=1}^{m} \sum_{i \in R_m} (\underbrace{Y_i}_{\substack{\text{observation}\\ \text{in } R_m}} - \underbrace{\hat{Y}_{R_m}}_{\substack{\text{mean value of}\\ \text{observations}\\ \text{in } R_m}})^2$$

$R_1, \dots, R_m$ are boxes that partition space

$R_i \cap R_j = \phi \ \forall i \neq j$ (except at boundary)

This is a difficult optimization problem to solve typically.

A classic book on classification and regression trees was written by Leo Breiman, a distinguished statistician at UC Berkley.

Technique is based on a greedy approach

Start with all observations in a single region.
Select the predictor variable X_j and cutpoint s such that splitting into region $\{X \mid X_j \geq s\}$ and $\{X \mid X_j < s\}$ results in smallest RSS.

$$\min_{j=1,\dots,p} \; \min_{s} \; \sum_{i:\, x_{ij} < s} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_{ij} \geq s} (y_i - \hat{y}_{R_2})^2$$

Here $R_1 = \{X \mid X_j < s\}$ & $R_2 = \{X \mid X_j \geq s\}$.

Once this problem is solved find j^*, s^*.
Use this as the first split.

Now repeat this step for the sub-regions individually. The best split now gives a total of three regions since we split one of the two regions
Continue until a stopping criterion is reached such as not having too few observations in a region
Note at top, all observations belong to a single region beyond which greedy splits are made at each step without looking necessarily at a best split that might lead to a better tree in future steps. (greedy).

Pruning a tree

Remember while we build our tree, our ultimate goal often is to get a good prediction on the test set.

Small tree with fewer splits $\Rightarrow$ More interpretable (lower variance) Poorer fit in training set (higher bias)

Large tree with many splits $\Rightarrow$ Less interpretable (Higher variance) Better fit in training set (lower bias)

To prune the tree, the typical approach is to grow a large tree and then prune back to get a subtree with the objective of getting low test error rates.

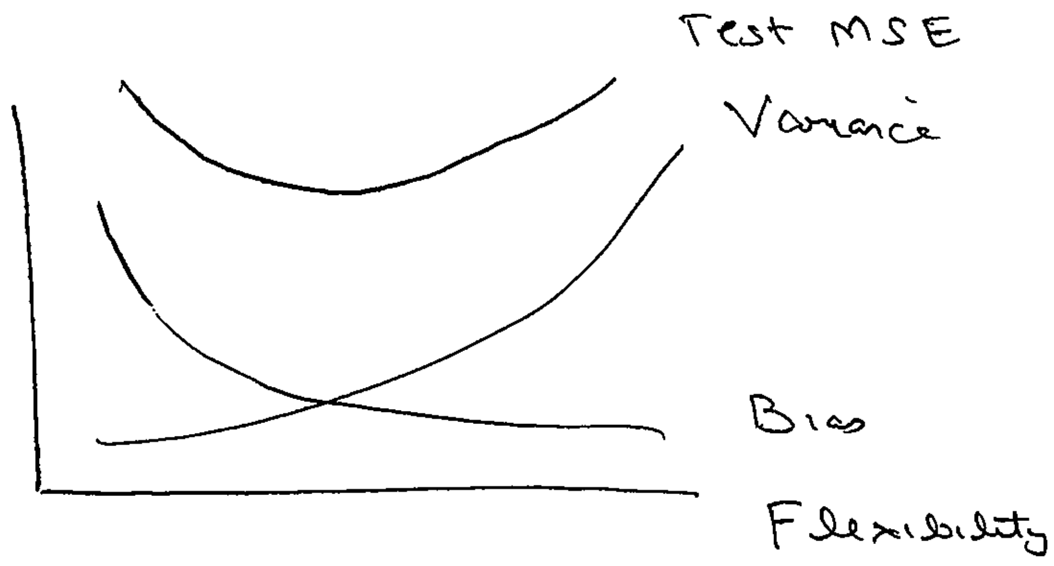
Variance here refers to how the estimator would change if we use a different training set

Bias is introduced by approximating a function f with a simpler function $\hat{f}$.

Say nonlinear function with a linear function

More complex the model $\hat{f}$ $\Longrightarrow$ Lower the bias, Higher the variance since it will capture many features of the model

Simpler the model $\hat{f}$ $\Longrightarrow$ Higher the bias, Lower the variance



Cost complexity parameter (cp): tradeoff between the fit to the training data and the complexity of the model

For any value of α, find the sub tree $T \subseteq T_0$ of the original tree (T_0) such that

$$\min_{T \subseteq T_0} \sum_{m=1}^{|T|} \sum_{i \in R_m} \left(y_i - \hat{y}_{R_m}\right)^2 + \alpha \underbrace{|T|}_{\text{No. of terminal nodes of tree } T}$$

$\alpha = 0 \implies T = T_0$ (original tree)

As $\alpha \uparrow$, there is a price to be paid for having too many leaf nodes and so we look for smaller subtrees (Similar to LASSO)

To find the best α, one can use cross-validation methods.

In CART, splits are made at each step to make the buckets as homogeneous or pure as possible. Note it may not be possible to do so always since we could have observations with the same independent variable values but different dependent variable values. It might also not necessarily be a good idea to get all buckets to be pure due to overfitting.

Classification Trees

To measure the purity of a bucket, a couple of measures are popularly used to measure impurity:

1) $$\text{Gini index} = \sum_{k=1}^{K} P_{mk}(1 - P_{mk})$$

P_{mk} → proportion of training observations in mth region from the kth class

2) $$\text{Entropy} = -\sum_{k=1}^{K} P_{mk} \log(P_{mk})$$

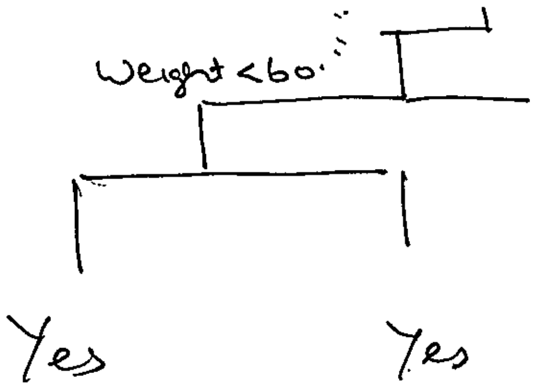
These two indices take a value near 0 as long as the P_{mk} values are close to 0 or 1. (all in one class).
These measures help indicate how pure a bucket is (no. of observations in the same class) and are also

Another related mesure is the

3) Classification error rate $= 1 - \underbrace{\max_{k} \hat{p}_{mk}}_{\text{assuming the class with maximum proportion is to be predicted.}}$

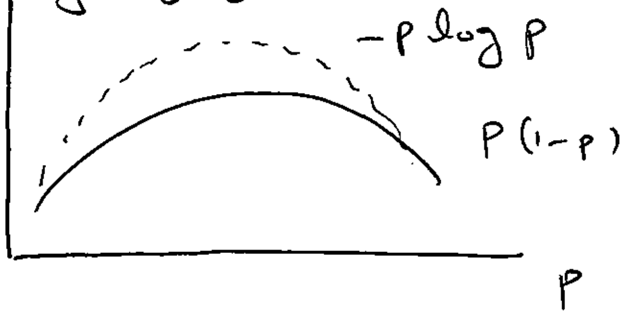
Note that in CART models, splitting on qualitative predictor variables is straightforward too.

In some splits for CART models, you might end up with a situations as follows:



The bottom split seems superficial. However note that such a split might give improved node purity where though the same response is predicted (YES), it might be that in one leaf, the observations are all in the same class (so you are more sure) while in the other there might be some impurity (so you are less sure)

To build the classification tree with rpart(·), it chooses a split with maximal impurity reduction in going from top node to bottom two nodes.



By default, rpart( ) uses the Gini rule for splitting.

In doing pruning of trees for classification problems, rpart( ) uses proportion misclassified and the number of terminal nodes to do cross validation.

By default, when you run rpart(·) it also does pruning using k-fold cross validation.

Random forests

Random forests are a combination of tree predictors (ensemble method) that operate by constructing a multitude of trees during the training phase and asking each tree to output the mode of the classes (most popular class in classification) or mean prediction (in regression). Pick the majority or average prediction across trees.

Random forest correct for the habit of decision trees to overfit the data and reduce the variance of the estimator.

$$f(x) = \sum_{t=1}^{T'} \frac{f_t(x)}{T'}$$

Here each $f_t(x)$ is a CART that trains on a subset of data that is chosen randomly with replacement

Bagging (Bootstrap aggregating)

(This is known as bootstrapping)

One of the challenges with this approach is that if we use the same algorithm then the predictors are highly correlated. If only a few features are strong predictors. This can be improved by learning trees on randomly chosen subset of input variables ~~[illegible]~~

Note that random forests provide less interpretability but often better predictive power.

More the trees - longer to build

Smaller node sizes - longer to build

In a typical implementation, a random sample of $m \approx \sqrt{p}$ predictors are considered at each split of the tree and the split is made based only ~~one~~ one of this candidate predictor variables.

This helps to decorrelate the trees chosen by the random forest method & reduces the variance.

Suppose you have one strong & many moderately strong predictors, then the trees will have the same split on the top & look similar.

(Rough) Algorithm for random forest

For a selected number of trees T^*

1) Sample observations with replacement to create a subset of data

2) Build a tree

   At each node

   1) For a chosen m, select only m random predictor variables from the entire set

   2) Find the best among these m predictors

   3) At the next node choose another m random predictors & repeat.

Note that by repeating this multiple times;

1) Bagging helps in ensuring that the average of many trees is not sensitive to noise in training set as just a single tree might be.
It helps decorrelate trees by showing different data.

2) Using subset of predictors randomly, it helps reduce the strong correlation that might arise if a few predictor variables are very strong, then these kept getting chosen.

Typically $m \approx \sqrt{p}$ (classification), $m \approx p/3$ (regression)

Analytics on Supreme Court dataset

Supreme <- read.csv ("supreme.csv")

head (supreme) 623 observation of

summary (supreme) 20 variables

str(supreme)

Variables are described below:

docket (case number)

term (year case was discussed)

party_1
party_2 (Two parties involved in case)

rehndir, stevdir, ocondir, scaldir, kendir,
soutdir, thomdir, gindir, brydir

(These variables provide the direction of the judgement of the 9 Judges -
Rehnquist, Stevens, O'Connor, Scalia, Kennedy, Souter,
Thomas, Ginsburg, Breyer)

Here 0 = liberal vote, 1 = conservative vote,
9 = not available for the case.

result (1 = conservative
0 = liberal)

petit (Petitioner type)

respon (Respondent type)

circuit (Circuit of origin)

unconst (Binary number indicating if petitioner argued practice was unconstitutional)

lctdir (lower court direction of result - liberal or conservative)

issue (issue area)

Let's focus on a particular judge (say Stevens)

```
stevens <- subset(supreme[, c("docket", "term",
                              "stevdir", "petit", "respon",
                              "circuit", "unconst",
                              "lctdir", "issue")],
                  supreme$stevdir != 9)
```

We focus only on those cases where Judge Stevens was present.

Processing the output result to affirm or reverse for judge Stevens.

```
stevens$rev <- as.integer((stevens$lctdir == "conser" &
                           stevens$stevdir == 0) |
                          (steven$lctdir = "liberal" &
                           stevens$stevdir == 1))
```

This creates a new variable $rev in the data frome stevens that takes a value of 1 if Steven's decision reverses the lower court decision and 0 if it affirms the lower case decision.

Note similar analysis can be done for other judges and overall case result

Split the data into training & test set

```
library (caTools)
set.seed (1)
spl <- sample.split (stevens$rev, SplitRatio = 0.7)
train <- subset (stevens, spl == TRUE)
test <- subset (stevens, spl == FALSE)
```

(Used to create training & test sets balanced on response)

Logistic regression

```
m1 <- glm (rev ~ petit + respon + circuit + unconst +
           lctdir + issue,
           data = train, family = binomial)

p1 <- predict (m1, newdata = test, type = "response")
```

This gives an error & this is because issue has only one realisation of the IR value (Interstate relations) which is not estimated in training set.

We modify the test set as follows:

```
test <- subset (test, test$issue != "IR")
p1 <- predict (m1, newdata = test, type = "response")
table (p1 >= 0.5, test$rev)
```

Confusion matrix

	0	1
0	53	32
1	29	70

Accuracy = 0.6684

We now fit a CART tree to the model

```
install.packages("rpart")
install.packages("rpart.plot")
library(rpart)
library(rpart.plot)
```

The Recursive Partitioning (rpart) builds CART models in R, while rpart.plot extends some of the functions in rpart

```
cart1 <- rpart(as.factor(rev) ~ petit + respon +
               circuit + unconst + lctdir + issue,
               data = train)
```

This fits a classification tree to the training set data. Note the as.factor( ) command is used since we want to fit a classification tree rather than a regression tree in this example

Alternatively

```
cart1 <- rpart(rev ~ petit + respon + circuit +
               unconst + lctdir + issue,
               data = train, method = "class")
```

$\underbrace{\text{class}}_{\text{classification}}$

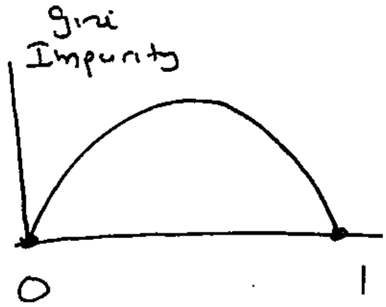
```
cart1
```

Provides the classification tree with nodes, branches and the corresponding probabilities of each class.

prp (cart 1) This plots the tree and adds labels to the tree

Summary (cart 1) Provides details of the CART model

By default, when the CART model for classification is run with rpart(), the Gini rule is used for splitting



$p(1-p)$

Here when we split, we choose it such that there is maximal reduction in impurity.

In the output tree, the children of node x are numbered to $2x$ and $2x+1$.

Note that a variable might also appear in the tree many times

Cart 1

$n = 434$ No. of observations

node), split, n, loss, yval, (yprob)

* denotes terminal node

Node number

Variable split on

No. of observations at node

Loss (error at node)

Predicted class

Predicted probability

38) Issue = CP, FA 16 4 0 (0.75 0.25)*

Node no

Variable split on

CP (criminal procedure)

FA (first amendment)

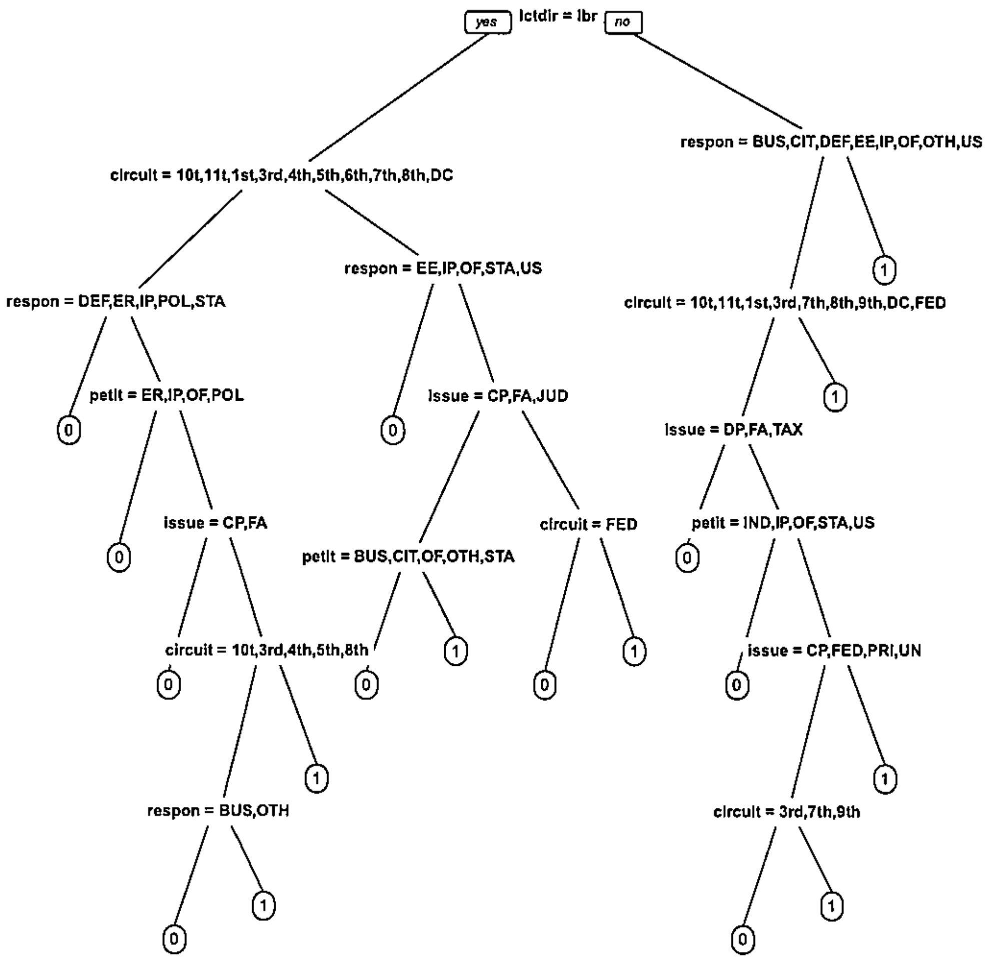
16 obs

4 misclassified

predicted value

prob

Terminal node


yes
lctdir = lbr
no
circuit = 10t,11t,1st,3rd,4th,5th,6th,7th,8th,DC
respon = BUS,CIT,DEF,EE,IP,OF,OTH,US
respon = DEF,ER,IP,POL,STA
respon = EE,IP,OF,STA,US
circuit = 10t,11t,1st,3rd,7th,8th,9th,DC,FED
petit = ER,IP,OF,POL
issue = CP,FA,JUD
issue = DP,FA,TAX
issue = CP,FA
petit = BUS,CIT,OF,OTH,STA
circuit = FED
petit = IND,IP,OF,STA,US
circuit = 10t,3rd,4th,5th,8th
issue = CP,FED,PRI,UN
respon = BUS,OTH
circuit = 3rd,7th,9th
0
1

prp (cart1, type = 1)

Labels all nodes
not just leaves

prp (cart1, type = 4)

Draws separate labels
for left & right directions
for all nodes and labels nodes

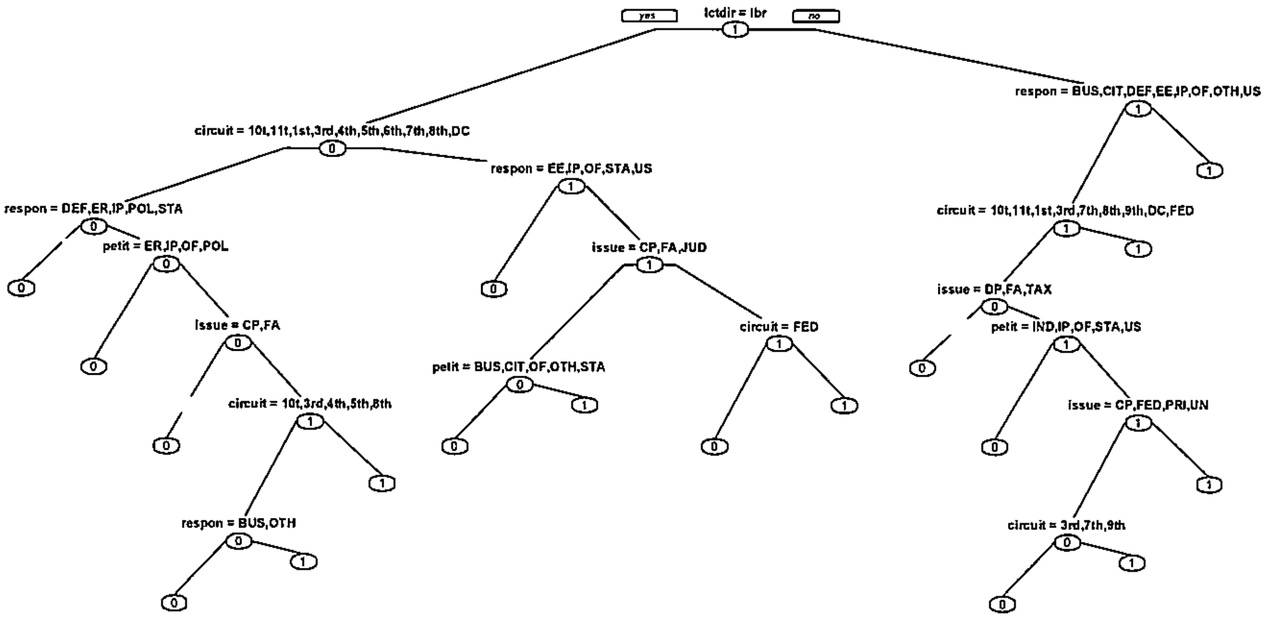
prp (cart1, extra = 4, type = 4)

In addition also
plots probability per
class of observations

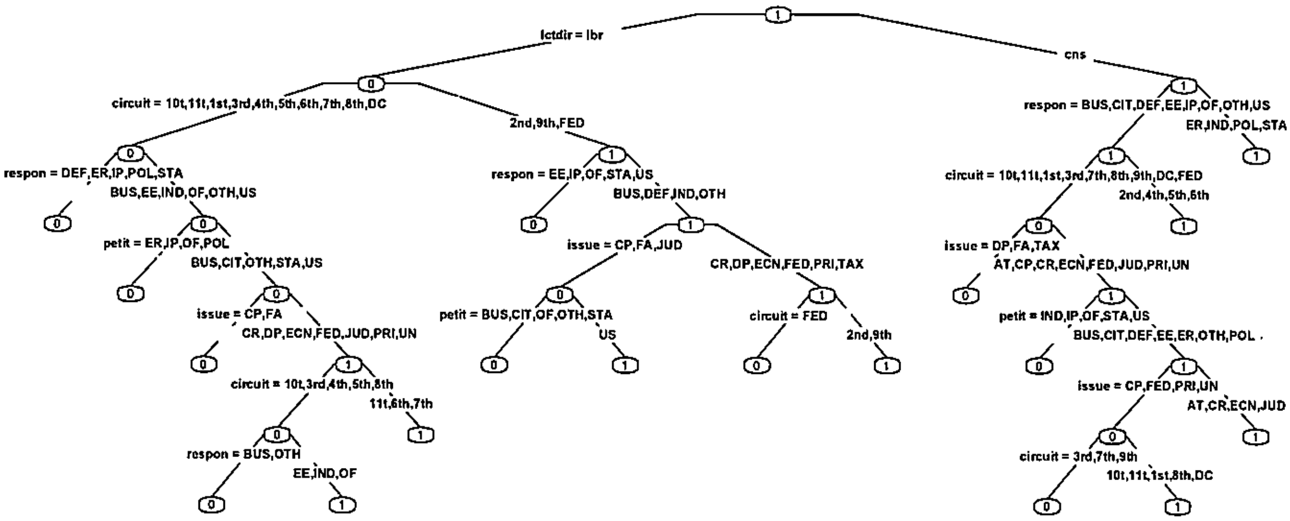
prp (cart1, extra = 9, type = 4)

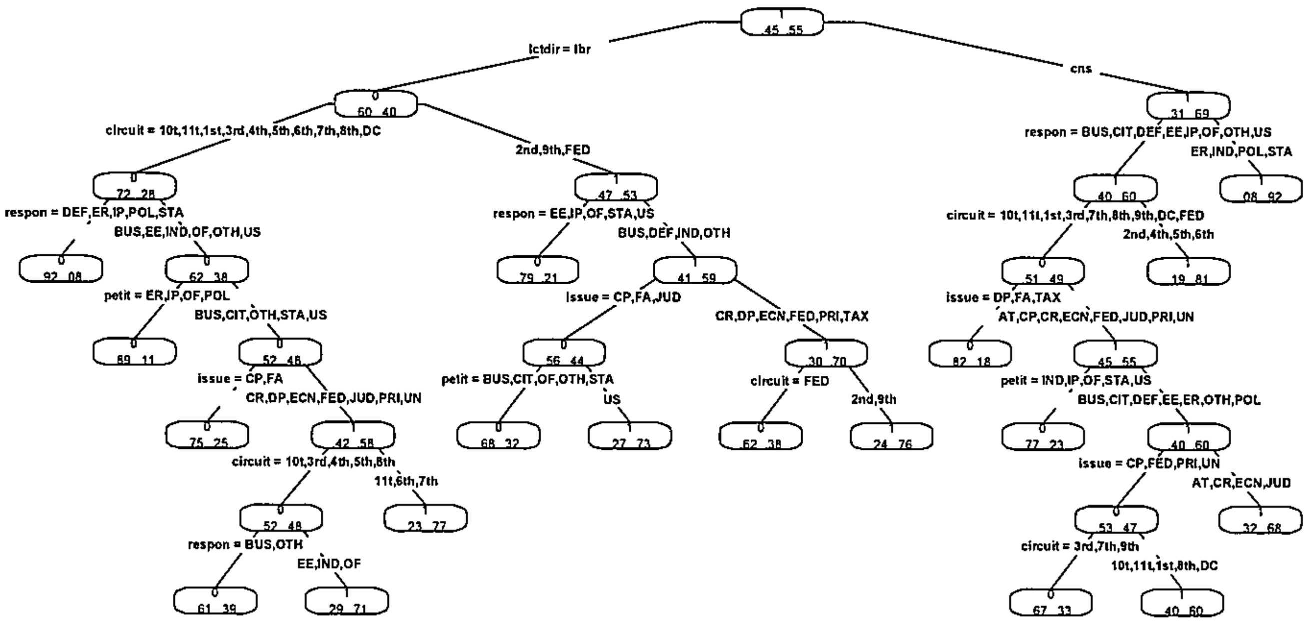
Probabilities times
fraction of observations
at the node
(Sum across all leaves is 1)

There are many plotting options in prp(·) that
can be explored further.

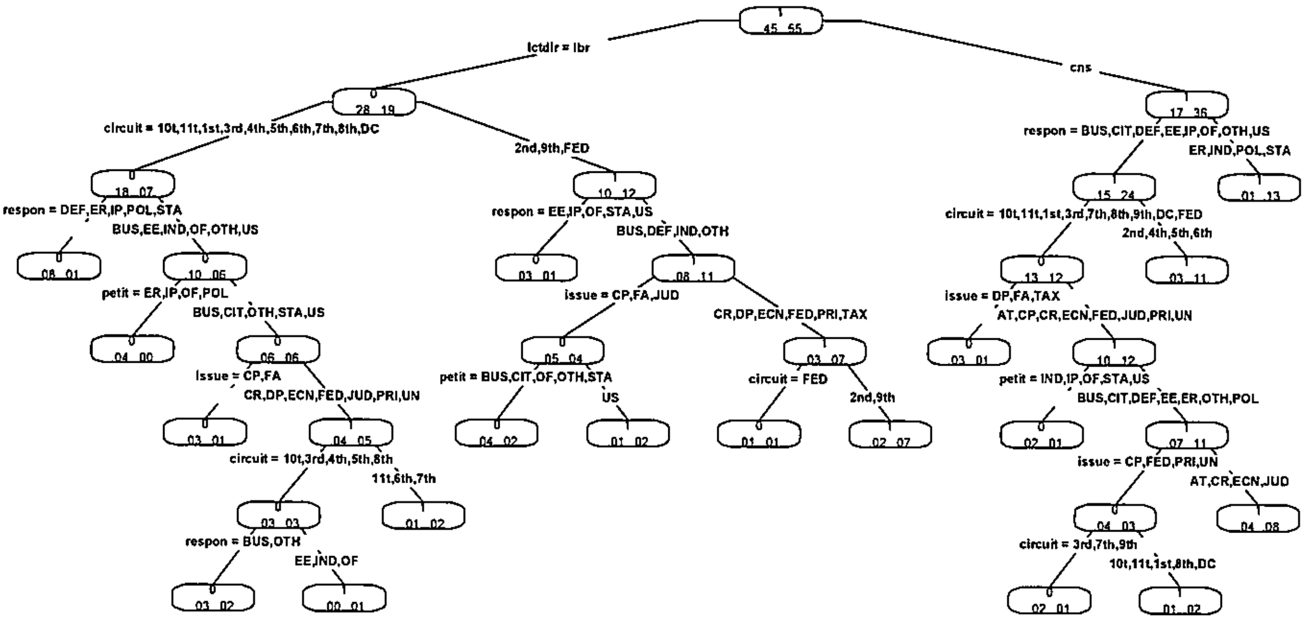


yes
lctdir = lbr
no
circuit = 10t,11t,1st,3rd,4th,5th,6th,7th,8th,DC
respon = BUS,CIT,DEF,EE,IP,OF,OTH,US
respon = DEF,ER,IP,POL,STA
petit = ER,IP,OF,POL
issue = CP,FA
circuit = 10t,3rd,4th,5th,8th
respon = BUS,OTH
respon = EE,IP,OF,STA,US
issue = CP,FA,JUD
petit = BUS,CIT,OF,OTH,STA
circuit = FED
circuit = 10t,11t,1st,3rd,7th,8th,9th,DC,FED
issue = DP,FA,TAX
petit = IND,IP,OF,STA,US
issue = CP,FED,PRI,UN
circuit = 3rd,7th,9th


lctdir = lbr
cns
circuit = 10t,11t,1st,3rd,4th,5th,6th,7th,8th,DC
2nd,9th,FED
respon = DEF,ER,IP,POL,STA
BUS,EE,IND,OF,OTH,US
petit = ER,IP,OF,POL
BUS,CIT,OTH,STA,US
issue = CP,FA
CR,DP,ECN,FED,JUD,PRI,UN
circuit = 10t,3rd,4th,5th,8th
11t,6th,7th
respon = BUS,OTH
EE,IND,OF
respon = EE,IP,OF,STA,US
BUS,DEF,IND,OTH
issue = CP,FA,JUD
CR,DP,ECN,FED,PRI,TAX
petit = BUS,CIT,OF,OTH,STA
US
circuit = FED
2nd,9th
respon = BUS,CIT,DEF,EE,IP,OF,OTH,US
ER,IND,POL,STA
circuit = 10t,11t,1st,3rd,7th,8th,9th,DC,FED
2nd,4th,5th,6th
issue = DP,FA,TAX
AT,CP,CR,ECN,FED,JUD,PRI,UN
petit = IND,IP,OF,STA,US
BUS,CIT,DEF,EE,ER,OTH,POL
issue = CP,FED,PRI,UN
AT,CR,ECN,JUD
circuit = 3rd,7th,9th
10t,11t,1st,8th,DC



1
45 55
lctdir = lbr
cns
0
60 40
circuit = 10t,11t,1st,3rd,4th,5th,6th,7th,8th,DC
2nd,9th,FED
0
72 28
respon = DEF,ER,IP,POL,STA
BUS,EE,IND,OF,OTH,US
0
92 08
0
62 38
petit = ER,IP,OF,POL
BUS,CIT,OTH,STA,US
0
89 11
0
52 48
issue = CP,FA
CR,DP,ECN,FED,JUD,PRI,UN
0
75 25
1
42 58
circuit = 10t,3rd,4th,5th,8th
11t,6th,7th
0
52 48
respon = BUS,OTH
EE,IND,OF
0
61 39
1
29 71
1
23 77
1
47 53
respon = EE,IP,OF,STA,US
BUS,DEF,IND,OTH
0
79 21
1
41 59
issue = CP,FA,JUD
CR,DP,ECN,FED,PRI,TAX
0
56 44
petit = BUS,CIT,OF,OTH,STA
US
0
68 32
1
27 73
1
30 70
circuit = FED
2nd,9th
0
62 38
1
24 76
1
31 69
respon = BUS,CIT,DEF,EE,IP,OF,OTH,US
ER,IND,POL,STA
1
40 60
circuit = 10t,11t,1st,3rd,7th,8th,9th,DC,FED
2nd,4th,5th,6th
1
08 92
0
51 49
issue = DP,FA,TAX
AT,CP,CR,ECN,FED,JUD,PRI,UN
1
19 81
0
82 18
1
45 55
petit = IND,IP,OF,STA,US
BUS,CIT,DEF,EE,ER,OTH,POL
0
77 23
1
40 60
issue = CP,FED,PRI,UN
AT,CR,ECN,JUD
0
53 47
circuit = 3rd,7th,9th
10t,11t,1st,8th,DC
1
32 68
0
67 33
1
40 60


1
45 55
lctdir = lbr
cns
0
28 19
circuit = 10t,11t,1st,3rd,4th,5th,6th,7th,8th,DC
2nd,9th,FED
0
18 07
respon = DEF,ER,IP,POL,STA
BUS,EE,IND,OF,OTH,US
0
08 01
0
10 06
petit = ER,IP,OF,POL
BUS,CIT,OTH,STA,US
0
04 00
0
06 06
issue = CP,FA
CR,DP,ECN,FED,JUD,PRI,UN
0
03 01
0
04 05
circuit = 10t,3rd,4th,5th,8th
11t,6th,7th
0
03 03
respon = BUS,OTH
EE,IND,OF
0
03 02
00 01
01 02
10 12
respon = EE,IP,OF,STA,US
BUS,DEF,IND,OTH
0
03 01
08 11
issue = CP,FA,JUD
CR,DP,ECN,FED,PRI,TAX
0
05 04
petit = BUS,CIT,OF,OTH,STA
US
0
04 02
01 02
03 07
circuit = FED
2nd,9th
0
01 01
02 07
17 36
respon = BUS,CIT,DEF,EE,IP,OF,OTH,US
ER,IND,POL,STA
15 24
01 13
circuit = 10t,11t,1st,3rd,7th,8th,9th,DC,FED
2nd,4th,5th,6th
0
13 12
03 11
issue = DP,FA,TAX
AT,CP,CR,ECN,FED,JUD,PRI,UN
0
03 01
10 12
petit = IND,IP,OF,STA,US
BUS,CIT,DEF,EE,ER,OTH,POL
0
02 01
07 11
issue = CP,FED,PRI,UN
AT,CR,ECN,JUD
0
04 03
04 08
circuit = 3rd,7th,9th
10t,11t,1st,8th,DC
0
02 01
01 02

Prediction

```
predictcart1 <- predict(cart1, newdata = test,
                        type = "class")
```

This provides the class prediction for the test set using the CART model applied to it

It chooses the class with the higher probability

```
table(predictcart1, test$rev)
```

Confusion matrix

Accuracy = 0.6684

	0	1
0	54	33
1	28	69

This CART model is very close to the logistic regression

Baseline model

```
table(train$rev)
```

0	1
195	234

```
table(test$rev)
```

	0	1
1	82	102

Baseline model predicts 1 for all observations based on traning set with accuracy = 0.5543

We also compare AUC using ROCR package

predictcart1prob <- predict (cart1, newdata = test)

This provides predicted probabilities of rev = 0 and rev = 1 for each test observation

library (ROCR)

predictroclog <- prediction (p1, test$rev)

perfroclog <- performance (predictroclog, measure = "tpr", x.measure = "fpr")

plot (perfroclog)

perfroclogAUC <- performance (predictroclog, measure = "AUC")

AUC for logistic regression <u>0.7404</u>

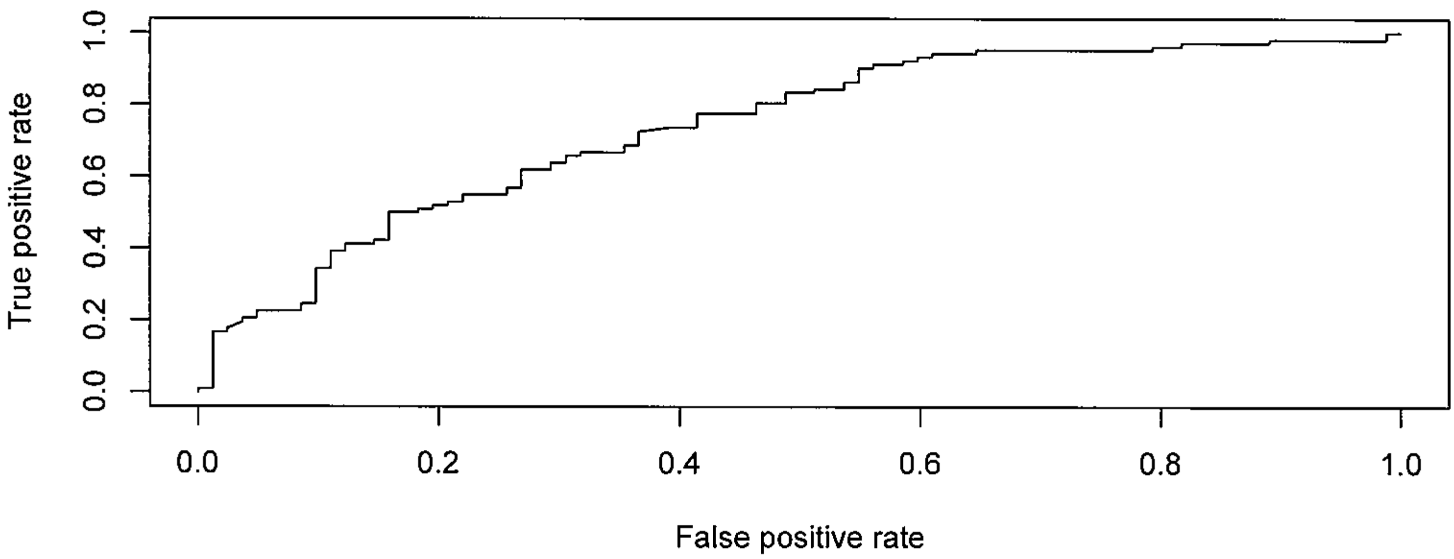
predictroccart <- prediction (predictcart1prob[,2], test$rev)

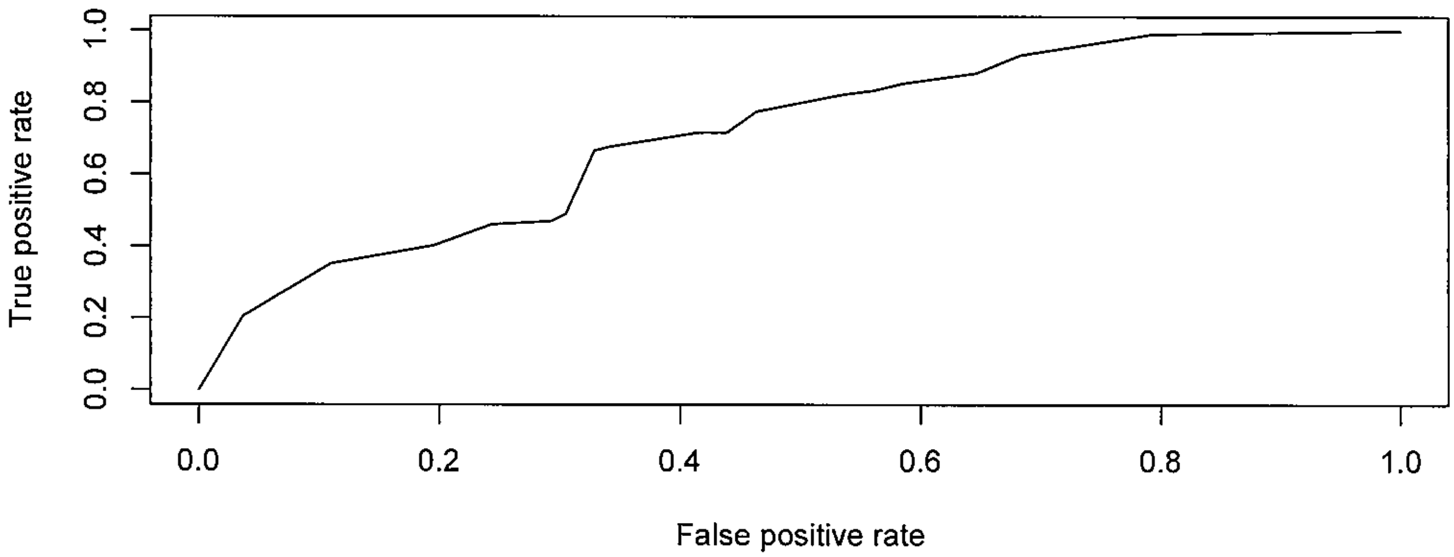
perfroccart <- performance (predictroccart, measure = "tpr", x.measure = "fpr")

plot (perfroccart)

perfroccartAUC <- performance (predictroccart, measure = "auc")

AUC for CART 0.7119


True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate

printcp (cart1) Displays cp table for model

Variables used in tree construction

Circuit issue letdin petit respon

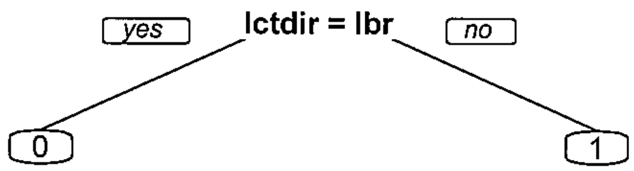
Root node error 195/434 = 0.44931

n = 434

	CP	nsplit	relerror	xerror	xstd
1	0.210	0	1	1	0.053
2	0.035	1	0.78	0.78	0.051
3	0.023	3	0.71	0.87	0.052
7	0.01	17	0.49	0.88	0.052

Smallest tree (no splits) to largest one (17 splits)
The errors are scaled so that the first node has error = 1. The cp parameter is also scaled.
Default stopping value = 0.01 in rpart.
nsplit = No. of terminal nodes − 1

Roughly, relative error can be converted to absolute error by multiplying by root node error.
Xerror & xstd refers to output from cross-validation
~~Obs~~ Roughly you would like to pick the tree with lowest xerror + xstd (1 std deviation)
Other rules are also possible such as xerror min


lctdir = lbr
yes
no
0
1

More sophisticated analytics models with random forests

install.packages("randomForest") — Package to fit random forests for classification & regression

library(randomForest)

set.seed(100)

forest <- randomForest(as.factor(rev) ~ petit + respon + circuit + unconst + lctdir + issue, data = train, nodesize = 5, ntree = 200)

This creates a random forest fit to the training data set where nodesize = 5 denotes the minimum number of observations in a terminal node (this is the default value for CART) and ntree = 200 (the number of trees to grow)

predictforest <- predict(forest, newdata = Test, type = c("class"))

This predicts a class based on the majority vote of each of the trees

table(predictforest, test$rev)

Confusion matrix

	0	1
0	46	16
1	36	86

$$\text{Accuracy} = \frac{46+86}{46+86+36+16} = 0.71$$

Comparison of legal experts and analytics tools

628 cases from 1994 to 2001 → 68 cases for 2002

1) 1st stage of tree building

A tree to build & predict unanimous liberal decisions

A tree to predict unanimous conservative decisions

If trees did not conflict then the case was predicted using the tree

2) For the more complex cases, trees were built for each judge and then the majority was used to predict decisions

This two stage approach was used by the authors in their work

"The Supreme Court Forecasting Project : Legal & Political Science Approaches to Predicting Supreme Court Decision Making"

In their paper, the authors also used the results predicted for certain justices to make predictions for new justices.

Legal experts: 83 experts (71 academics + 12 attorneys)

Experts asked to predict only cases in their expertise area. They could use any information to make their judgements on the decisions.

Main findings

1) Model predicted 75% of affirm/reverse results correctly while experts got 59%.

2) On the justice level, model got 66.7% correct and experts got 67.9% correct

3) Experts were most accurate at predicting the votes of the most ideologically extreme justices and least successful at forecasting votes of centrist judges (Kennedy and O'Connor)