Regression Trees and Classification Trees

Yifei Sun

Contents

```
library(ISLR)
library(mlbench)
library(caret)
library(rpart)
library(rpart.plot)
library(party)
library(partykit)
library(plotmo)
library(pROC)
```

Regression Trees

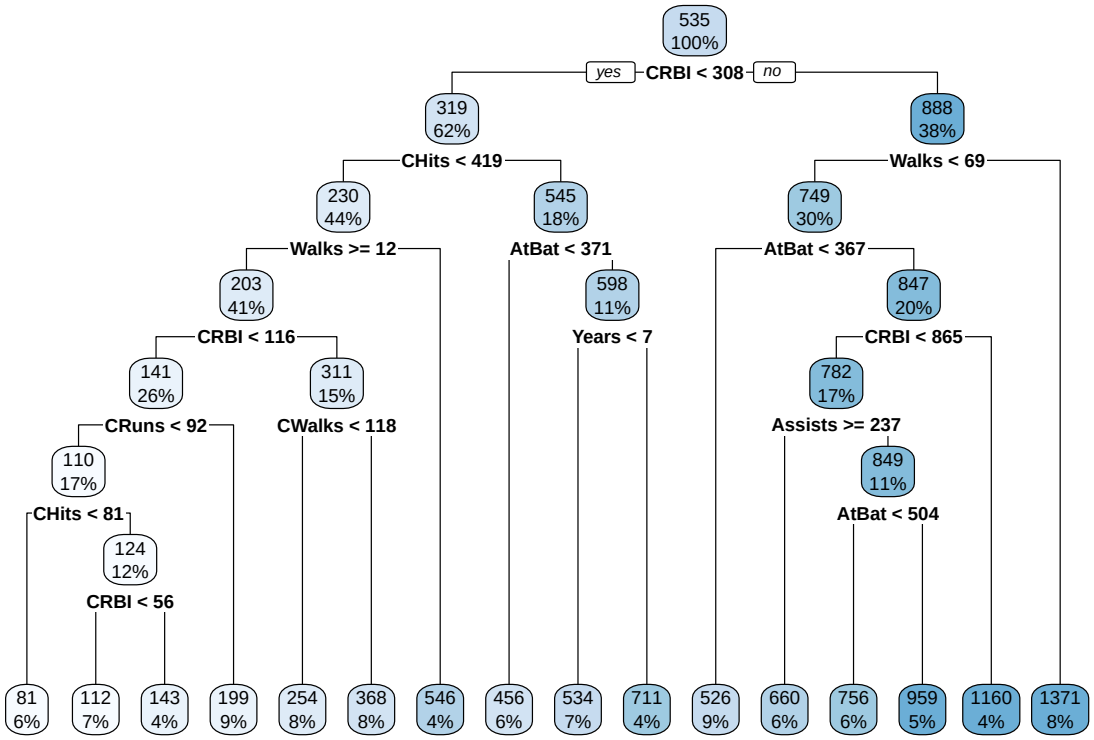
Predict a baseball player's salary on the basis of various statistics associated with performance in the previous year. Use `?Hitters` for more details.

```
data(Hitters)
Hitters <- na.omit(Hitters)

set.seed(2021)
trRows <- createDataPartition(Hitters$Salary,
                              p = .75,
                              list = F)
```

The CART approach

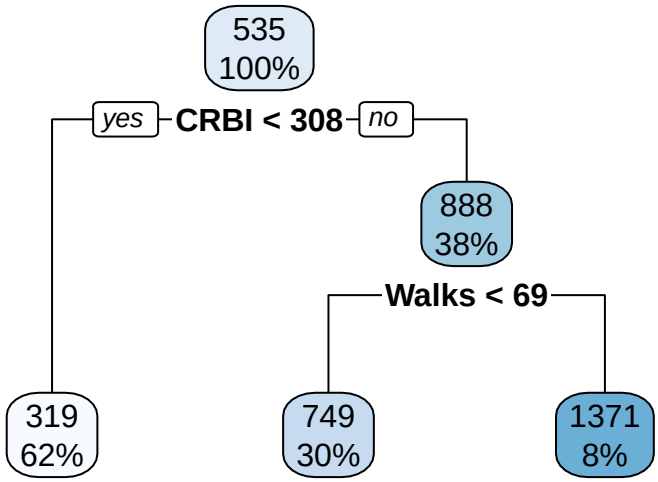
We first apply the regression tree method to the Hitters data. `cp` is the complexity parameter. The default value for `cp` is 0.01. Sometimes the default value may over prune the tree.

```
set.seed(1)
tree1 <- rpart(formula = Salary ~ . ,
               data = Hitters, subset = trRows,
               control = rpart.control(cp = 0))

## plot.rpart
# plot(tree1)
# text(tree1)

rpart.plot(tree1)
```



We get a smaller tree by increasing the complexity parameter.

```
set.seed(1)
tree2 <- rpart(Salary ~ . ,
               data = Hitters, subset = trRows,
               control = rpart.control(cp = 0.1))
rpart.plot(tree2)
```



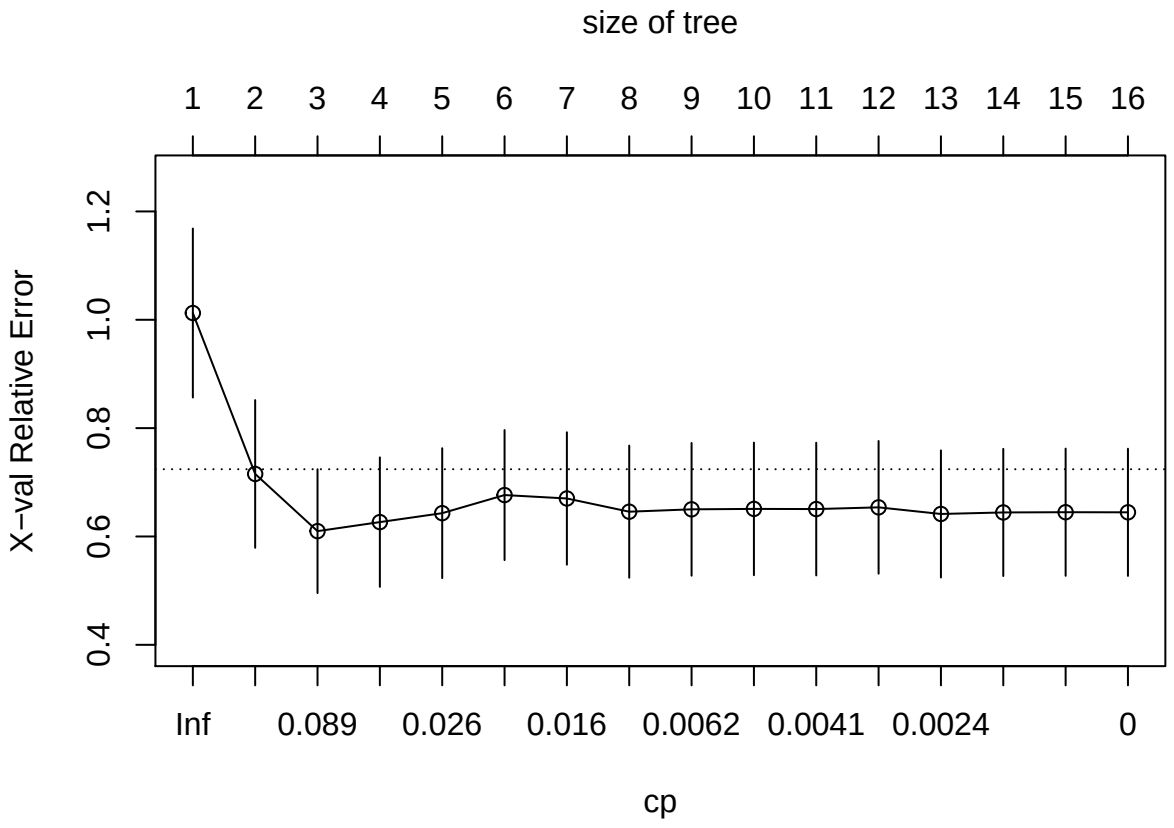
We next apply cost complexity pruning to obtain a tree with the right size. The functions `printcp()` and `plotcp()` give the set of possible cost-complexity prunings of a tree from a nested set. For the geometric means of the intervals of values of `cp` for which a pruning is optimal, a cross-validation has been done in the initial construction by `rpart()`.

The `cptable` in the fit contains the mean and standard deviation of the errors in the cross-validated prediction against each of the geometric means, and these are plotted by `plotcp()`. `Rel error` (relative error) is `\(1 - R^2\)`. The x-error is the cross-validation error generated by built-in cross validation. A good choice of `cp` for pruning is often the leftmost value for which the mean lies below the horizontal line.

```
printcp(tree1)
```

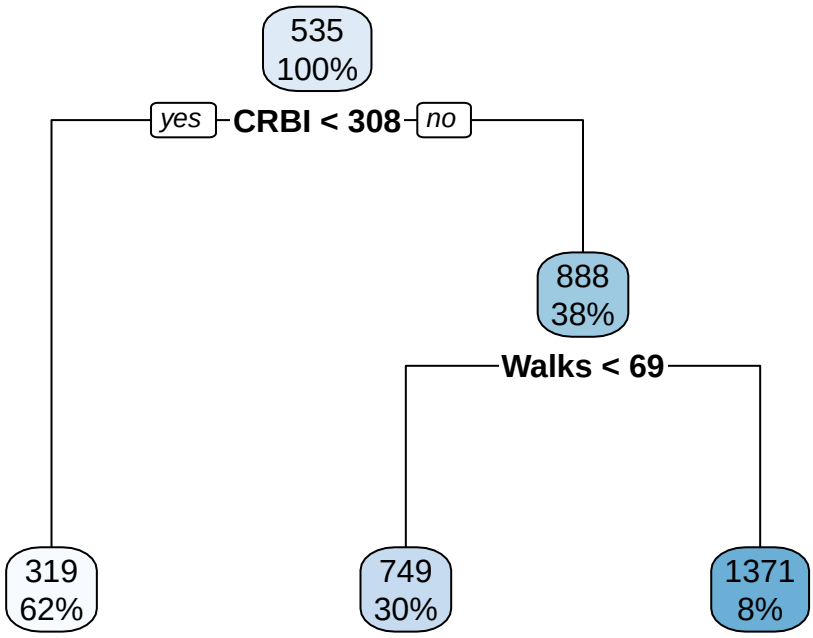
```
##
## Regression tree:
## rpart(formula = Salary ~ ., data = Hitters, subset = trRows,
##     control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
## [1] Assists AtBat   CHits   CRBI    CRuns   CWalks  Walks   Years
##
## Root node error: 39912455/200 = 199562
##
## n= 200
##
##            CP nsplit rel error  xerror    xstd
## 1  0.38192590      0   1.00000 1.01246 0.15599
## 2  0.12805512      1   0.61807 0.71544 0.13636
## 3  0.06250035      2   0.49002 0.60987 0.11431
## 4  0.03215783      3   0.42752 0.62652 0.11953
## 5  0.02075481      4   0.39536 0.64304 0.11992
## 6  0.01897401      5   0.37461 0.67643 0.12001
## 7  0.01378337      6   0.35563 0.67016 0.12223
## 8  0.00690975      7   0.34185 0.64575 0.12195
## 9  0.00561556      8   0.33494 0.65010 0.12252
## 10 0.00414395      9   0.32932 0.65090 0.12242
## 11 0.00401868     10   0.32518 0.65060 0.12248
## 12 0.00245210     11   0.32116 0.65378 0.12251
## 13 0.00233886     12   0.31871 0.64156 0.11732
## 14 0.00034311     13   0.31637 0.64435 0.11740
## 15 0.00013069     14   0.31603 0.64479 0.11740
## 16 0.00000000     15   0.31590 0.64461 0.11740
```

```
cpTable <- tree1$cptable
plotcp(tree1)
```

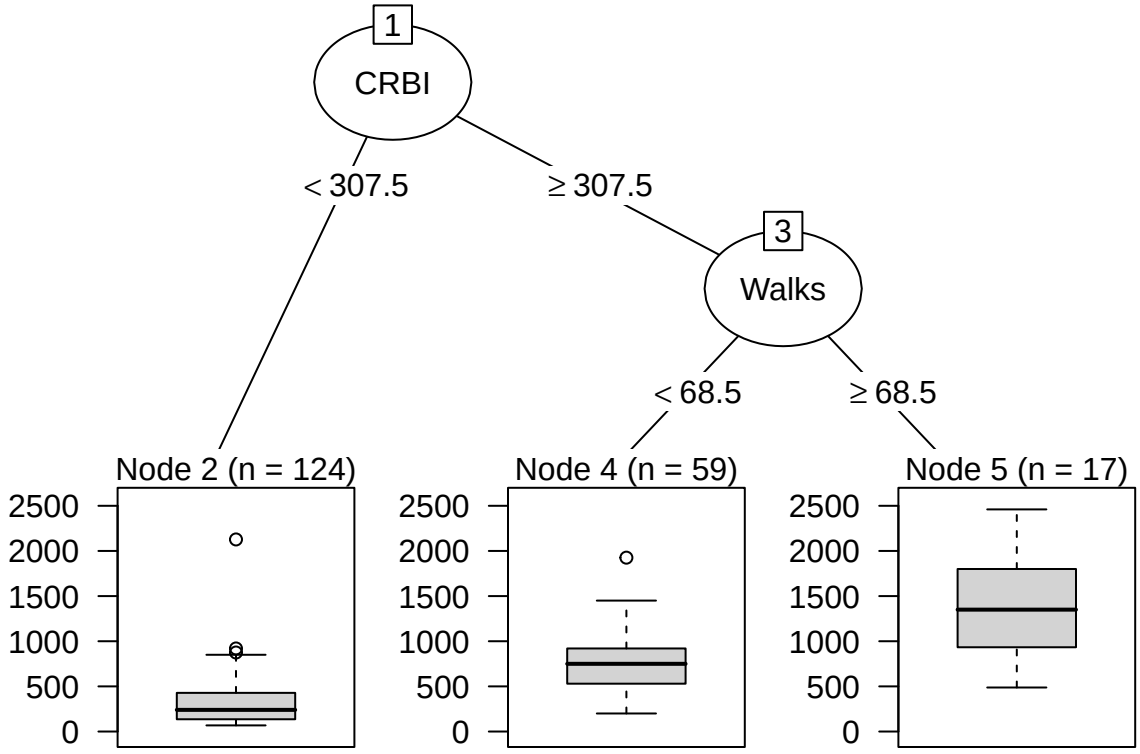


Prune the tree based on the `cp` table.

```
# minimum cross-validation error
minErr <- which.min(cpTable[,4])
tree3 <- prune(tree1, cp = cpTable[minErr,1])
rpart.plot(tree3)
```




```
plot(as.party(tree3))
```




```
summary(tree3)
```

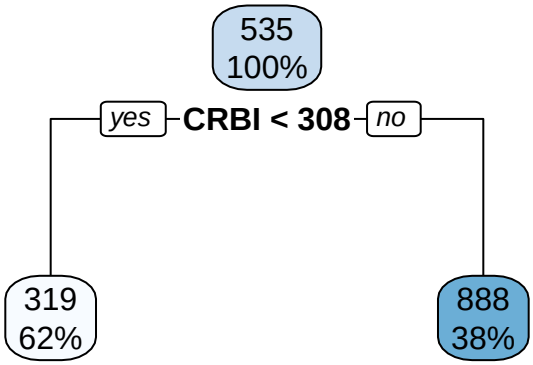
```
## Call:
## rpart(formula = Salary ~ ., data = Hitters, subset = trRows,
##     control = rpart.control(cp = 0))
##   n= 200
##
##           CP nsplit rel error    xerror      xstd
## 1 0.38192590      0 1.0000000 1.0124567 0.1559889
## 2 0.12805512      1 0.6180741 0.7154425 0.1363587
## 3 0.06250035      2 0.4900190 0.6098656 0.1143107
##
## Variable importance
##    CRBI  CAtBat   CHits   CRuns  CHmRun  CWalks   Walks    Runs PutOuts    Hits
##      18      16      16      14      14      13       6       2       1       1
##
## Node number 1: 200 observations,    complexity param=0.3819259
##   mean=535.4117, MSE=199562.3
##   left son=2 (124 obs) right son=3 (76 obs)
##   Primary splits:
##       CRBI   < 307.5  to the left,  improve=0.3819259, (0 missing)
##       CHits  < 450    to the left,  improve=0.3792192, (0 missing)
##       CRuns  < 218.5  to the left,  improve=0.3735094, (0 missing)
##       CAtBat < 1929.5 to the left,  improve=0.3726771, (0 missing)
##       CWalks < 216    to the left,  improve=0.3330730, (0 missing)
```

```
##   Surrogate splits:
##       CAtBat < 2316.5 to the left,  agree=0.95, adj=0.868, (0 split)
##       CHits  < 669    to the left,  agree=0.95, adj=0.868, (0 split)
##       CRuns  < 301    to the left,  agree=0.92, adj=0.789, (0 split)
##       CHmRun < 54.5   to the left,  agree=0.90, adj=0.737, (0 split)
##       CWalks < 216    to the left,  agree=0.90, adj=0.737, (0 split)
##
## Node number 2: 124 observations
##   mean=319.2769, MSE=72904.09
##
## Node number 3: 76 observations,    complexity param=0.1280551
##   mean=888.0527, MSE=205641.4
##   left son=6 (59 obs) right son=7 (17 obs)
##   Primary splits:
##       Walks   < 68.5   to the left,  improve=0.3270252, (0 missing)
##       AtBat   < 424    to the left,  improve=0.2172819, (0 missing)
##       Hits    < 123.5  to the left,  improve=0.2106275, (0 missing)
##       Runs    < 55     to the left,  improve=0.2010831, (0 missing)
##       PutOuts < 809    to the left,  improve=0.1887649, (0 missing)
##   Surrogate splits:
##       Runs    < 84.5   to the left,  agree=0.842, adj=0.294, (0 split)
##       PutOuts < 1171   to the left,  agree=0.816, adj=0.176, (0 split)
##       Hits    < 184.5  to the left,  agree=0.803, adj=0.118, (0 split)
##       AtBat   < 603.5  to the left,  agree=0.789, adj=0.059, (0 split)
##       CHmRun  < 273    to the left,  agree=0.789, adj=0.059, (0 split)
##
## Node number 6: 59 observations
##   mean=748.8511, MSE=98543.27
##
## Node number 7: 17 observations
##   mean=1371.164, MSE=276688.3
```

```
with(Hitters[trRows,], table(cut(CRBI, c(-Inf, 307.5, Inf)),
                             cut(CAtBat, c(-Inf, 2316.5, Inf))))
```

```
##
##              (-Inf,2.32e+03] (2.32e+03, Inf]
##   (-Inf,308]             114              10
##   (308, Inf]               0              76
```

```
# 1SE rule
tree4 <- prune(tree1, cp = cpTable[cpTable[,4]<cpTable[minErr,4]+cpTable[minErr,5],1][1])
rpart.plot(tree4)
```



Finally, the function `predict()` can be used for prediction from a fitted `rpart` object.

```
head(predict(tree3, newdata = Hitters[-trRows,]))
```

```
## -Andres Galarraga       -Buddy Bell       -Bob Brenly       -Bob Melvin
##          319.2769         1371.1643          319.2769          319.2769
## -BillyJo Robidoux      -Chris Bando
##          319.2769          319.2769
```

Missing data

```
Hitters2 <- Hitters
Hitters2$CRBI[sample(1:nrow(Hitters2), 50)] <- NA

set.seed(1)
tree_m <- rpart(Salary ~ . ,
                data = Hitters2,
                subset = trRows,
                control = rpart.control(cp = 0))

cpTable_m <- tree_m$cptable
tree2_m <- prune(tree_m, cp = cpTable_m[which.min(cpTable_m[,4]),1])

summary(tree_m, cp = cpTable_m[which.min(cpTable_m[,4]),1])
```

```
## Call:
## rpart(formula = Salary ~ ., data = Hitters2, subset = trRows,
##     control = rpart.control(cp = 0))
##   n= 200
##
##              CP nsplit rel error    xerror      xstd
## 1  0.3792191672      0 1.0000000 1.0124567 0.1559889
## 2  0.1346129886      1 0.6207808 0.7051947 0.1375911
## 3  0.0466533613      2 0.4861678 0.6162435 0.1162865
## 4  0.0406985313      3 0.4395145 0.7006591 0.1228920
## 5  0.0230367844      4 0.3988160 0.6979655 0.1221345
## 6  0.0184611980      5 0.3757792 0.7157471 0.1214443
## 7  0.0176141868      6 0.3573180 0.7245008 0.1213859
## 8  0.0151385132      7 0.3397038 0.7174090 0.1235906
## 9  0.0058500417      8 0.3245653 0.6932186 0.1236323
## 10 0.0045489952      9 0.3187152 0.6969256 0.1237107
## 11 0.0038721016     10 0.3141662 0.6936291 0.1235962
## 12 0.0018360746     11 0.3102941 0.6849839 0.1236191
## 13 0.0006536496     12 0.3084581 0.6812802 0.1192290
## 14 0.0003431087     13 0.3078044 0.6809116 0.1192366
## 15 0.0001055580     14 0.3074613 0.6808523 0.1192365
## 16 0.0000000000     15 0.3073557 0.6807566 0.1192392
##
## Variable importance
##  CAtBat   CHits   CRuns  CWalks   Years  CHmRun   Walks    Runs    Hits   AtBat
##      16      16      15      14      11      10       6       3       3       3
## PutOuts     RBI
##       2       1
##
## Node number 1: 200 observations,    complexity param=0.3792192
##   mean=535.4117, MSE=199562.3
##   left son=2 (91 obs) right son=3 (109 obs)
##   Primary splits:
##       CHits  < 450    to the left,  improve=0.3792192, (0 missing)
##       CRuns  < 218.5  to the left,  improve=0.3735094, (0 missing)
##       CAtBat < 1929.5 to the left,  improve=0.3726771, (0 missing)
##       CWalks < 216    to the left,  improve=0.3330730, (0 missing)
##       CRBI   < 310    to the left,  improve=0.3230103, (35 missing)
##   Surrogate splits:
##       CAtBat < 1537   to the left,  agree=0.975, adj=0.945, (0 split)
##       CRuns  < 210.5  to the left,  agree=0.965, adj=0.923, (0 split)
##       CWalks < 131    to the left,  agree=0.910, adj=0.802, (0 split)
##       Years  < 5.5    to the left,  agree=0.860, adj=0.692, (0 split)
##       CHmRun < 31.5   to the left,  agree=0.830, adj=0.626, (0 split)
##
## Node number 2: 91 observations
##   mean=234.3352, MSE=58560.08
##
## Node number 3: 109 observations,    complexity param=0.134613
##   mean=786.7692, MSE=178421.3
##   left son=6 (86 obs) right son=7 (23 obs)
##   Primary splits:
##       Walks  < 67     to the left,  improve=0.2762627, (0 missing)
##       AtBat  < 426.5  to the left,  improve=0.2124849, (0 missing)
```

```
##       RBI    < 59.5   to the left,  improve=0.1941918, (0 missing)
##       Hits   < 124.5  to the left,  improve=0.1932516, (0 missing)
##       CHmRun < 102.5  to the left,  improve=0.1766360, (0 missing)
##   Surrogate splits:
##       Runs    < 83.5   to the left,  agree=0.826, adj=0.174, (0 split)
##       CWalks  < 687.5  to the left,  agree=0.817, adj=0.130, (0 split)
##       PutOuts < 1171   to the left,  agree=0.817, adj=0.130, (0 split)
##       CAtBat  < 1635   to the right, agree=0.807, adj=0.087, (0 split)
##       CHmRun  < 273    to the left,  agree=0.798, adj=0.043, (0 split)
##
## Node number 6: 86 observations
##   mean=671.954, MSE=86244.51
##
## Node number 7: 23 observations,    complexity param=0.04665336
##   mean=1216.078, MSE=289485.1
```

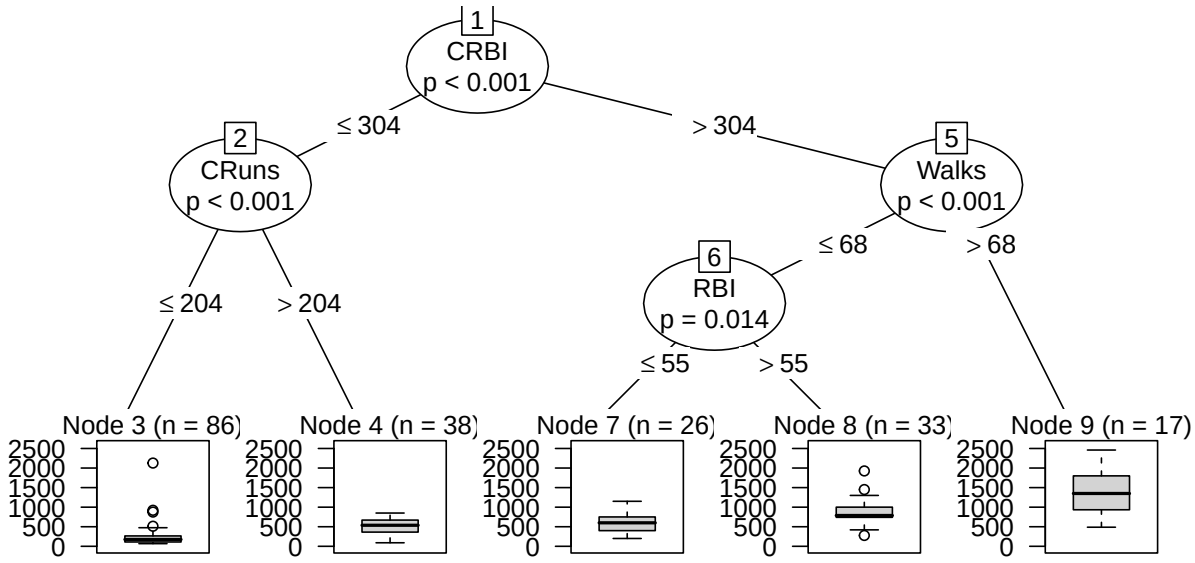
```
head(predict(tree2_m, newdata = Hitters2[-trRows,]))
```

```
## -Andres Galarraga       -Buddy Bell       -Bob Brenly       -Bob Melvin
##          234.3352         1216.0780         1216.0780          234.3352
## -BillyJo Robidoux      -Chris Bando
##          234.3352          234.3352
```

Conditional inference trees

The implementation utilizes a unified framework for conditional inference, or permutation tests. Unlike CART, the stopping criterion is based on p-values. A split is implemented when (1 - p-value) exceeds the value given by `mincriterion` as specified in `ctree_control()`. This approach ensures that the right-sized tree is grown without additional pruning or cross-validation, but can stop early. At each step, the splitting variable is selected as the input variable with strongest association to the response (measured by a p-value corresponding to a test for the partial null hypothesis of a single input variable and the response). Such a splitting procedure can avoid a variable selection bias towards predictors with many possible cutpoints.

```
tree5 <- ctree(Salary ~ . , Hitters,
               subset = trRows)
plot(tree5)
```



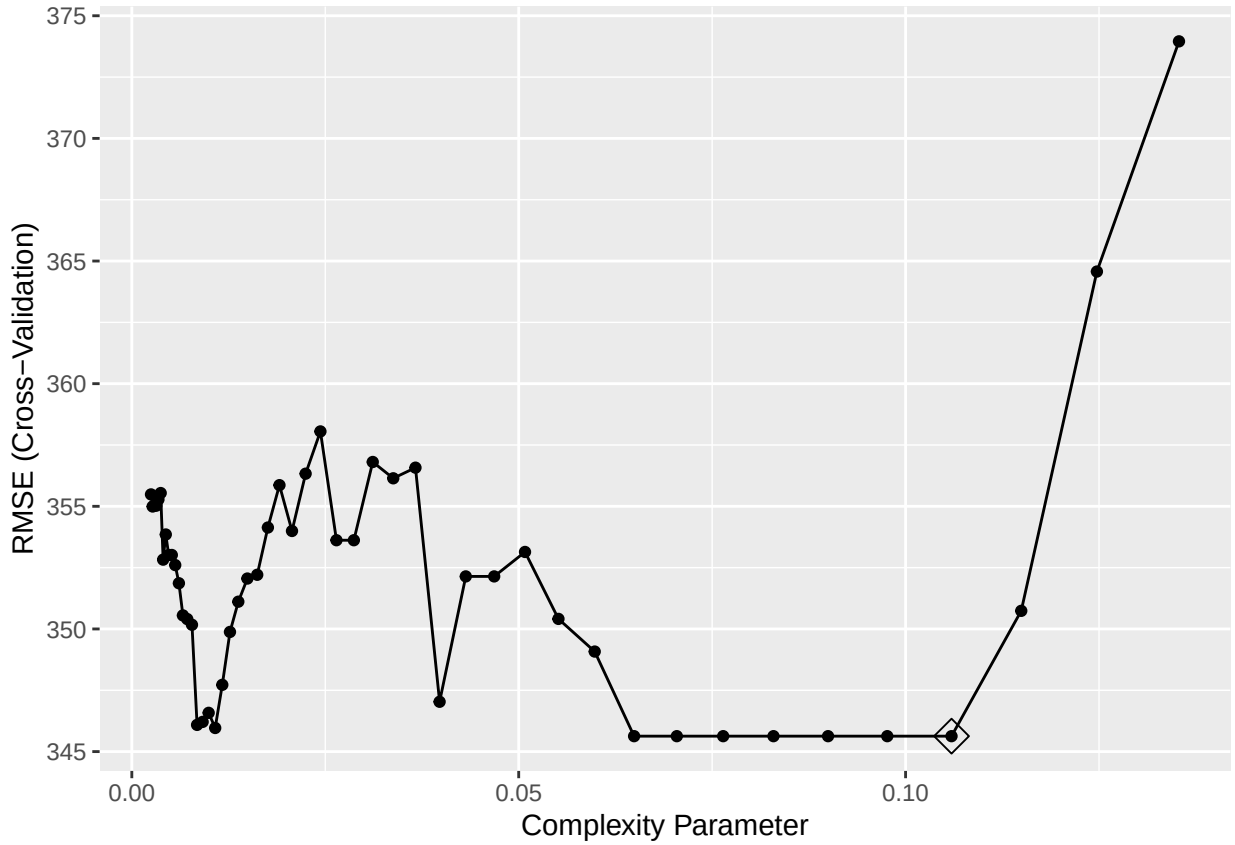
Note that `tree5` is a `party` object. The function `predict()` can be used for prediction from a fitted `party` object.

```
head(predict(tree5, newdata = Hitters[-trRows,]))
```

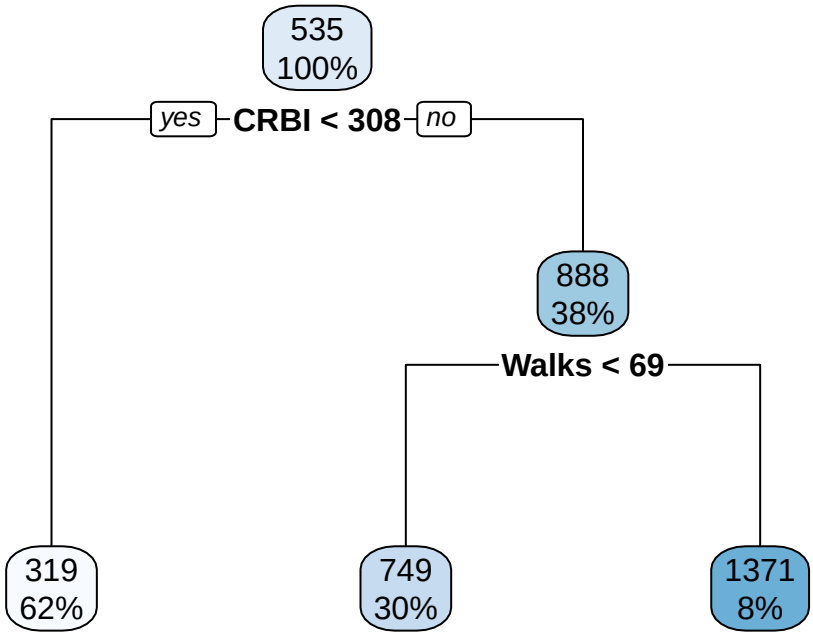
```
## -Andres Galarraga       -Buddy Bell       -Bob Brenly       -Bob Melvin
##          229.2093         1371.1643          523.1141          229.2093
## -BillyJo Robidoux      -Chris Bando
##          229.2093          229.2093
```

caret

```
ctrl <- trainControl(method = "cv")

set.seed(1)
rpart.fit <- train(Salary ~ . ,
                   Hitters[trRows,],
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-6,-2, length = 50))),
                   trControl = ctrl)
ggplot(rpart.fit, highlight = TRUE)
```

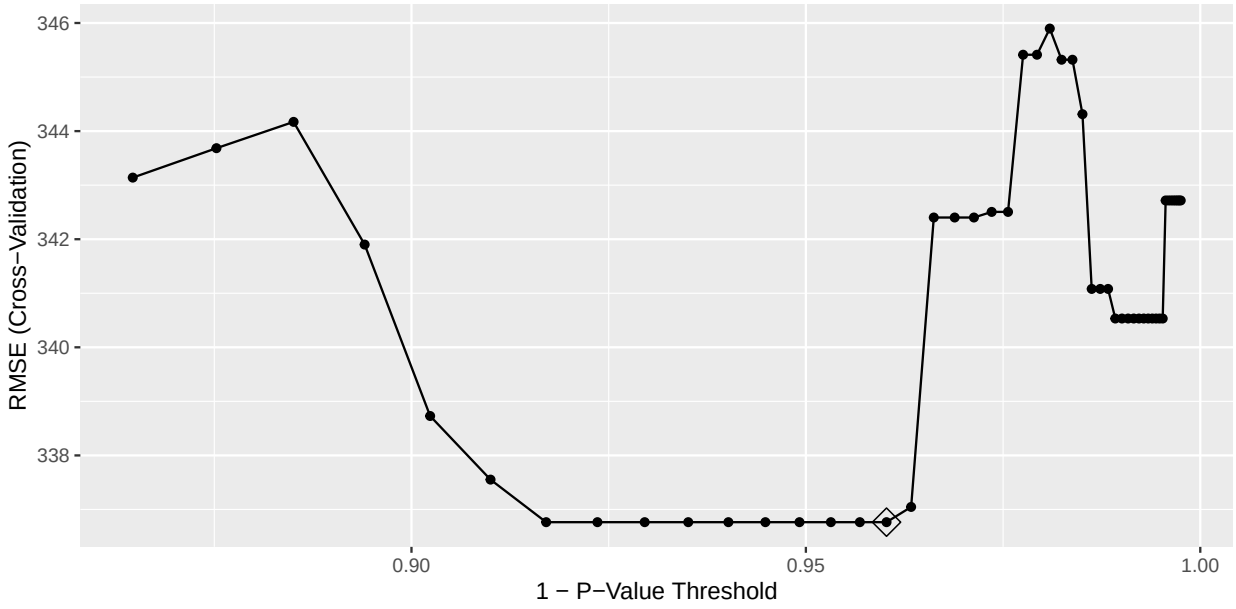



```
rpart.plot(rpart.fit$finalModel)
```

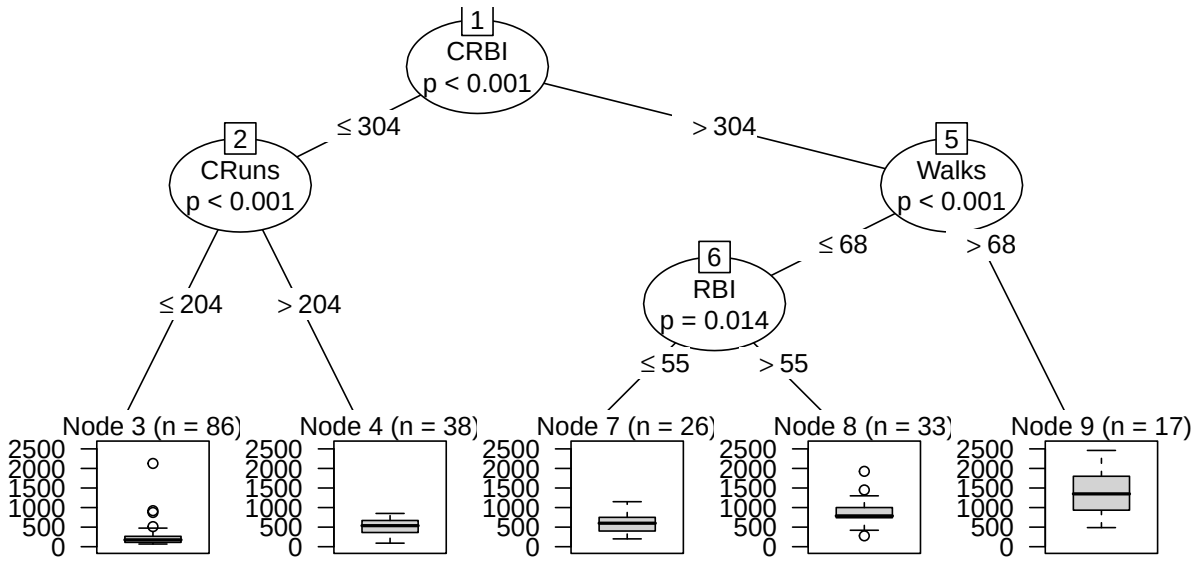


We can also fit a conditional inference tree model. The tuning parameter is `mincriterion`.

```
set.seed(1)
ctree.fit <- train(Salary ~ . ,
                   Hitters[trRows,],
                   method = "ctree",
                   tuneGrid = data.frame(mincriterion = 1-exp(seq(-6, -2, length = 50))),
                   trControl = ctrl)
ggplot(ctree.fit, highlight = TRUE)
```




```
plot(ctree.fit$finalModel)
```




```
summary(resamples(list(rpart.fit, ctree.fit)))
```

```
##
## Call:
## summary.resamples(object = resamples(list(rpart.fit, ctree.fit)))
##
## Models: Model1, Model2
## Number of resamples: 10
##
## MAE
```

```
##            Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## Model1 197.0586 213.6735 253.6719 249.9659 279.2418 319.9003    0
## Model2 132.8015 211.5980 221.4816 229.0189 260.9666 298.8747    0
##
## RMSE
##            Min.  1st Qu.   Median     Mean 3rd Qu.     Max. NA's
## Model1 235.4672 284.1624 331.0820 345.6295 357.909 509.5493    0
## Model2 171.9460 274.9534 335.9821 336.7650 373.203 531.4344    0
##
## Rsquared
##                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Model1 2.353772e-05 0.3227309 0.4309238 0.4308392 0.6016268 0.7701476    0
## Model2 8.244983e-02 0.3366112 0.5148029 0.4795865 0.6603520 0.8140052    0
```

```
RMSE(predict(rpart.fit, newdata = Hitters[-trRows,]), Hitters$Salary[-trRows])
```

```
## [1] 389.4528
```

```
RMSE(predict(ctree.fit, newdata = Hitters[-trRows,]), Hitters$Salary[-trRows])
```

```
## [1] 346.8838
```

Classification trees

We use the Pima Indians Diabetes Database for illustration. The data contain 768 observations and 9 variables. The outcome is a binary variable `diabetes`.

```
data(PimaIndiansDiabetes)
dat <- PimaIndiansDiabetes
dat$diabetes <- factor(dat$diabetes, c("pos", "neg"))

set.seed(1)
rowTrain <- createDataPartition(y = dat$diabetes,
                                p = 2/3,
                                list = FALSE)
```

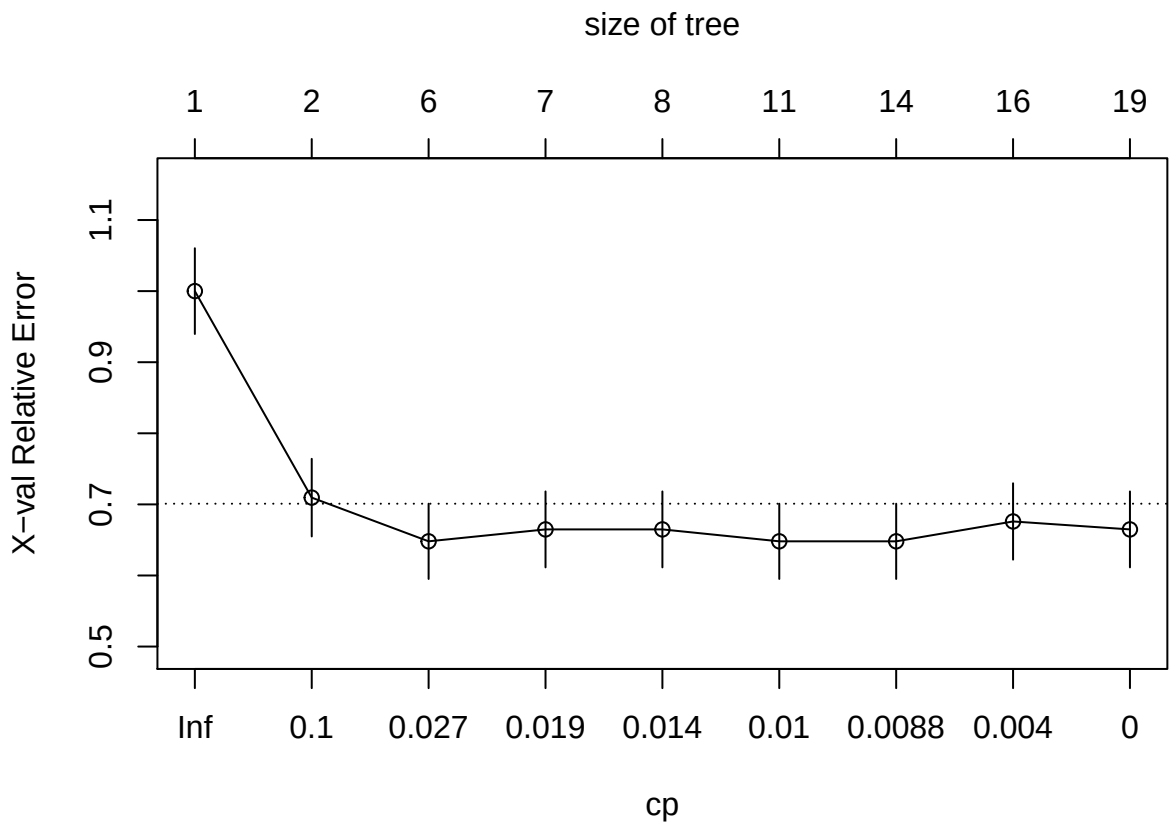
rpart

```
set.seed(1)
tree1 <- rpart(formula = diabetes ~ . ,
               data = dat,
               subset = rowTrain,
               control = rpart.control(cp = 0))

cpTable <- printcp(tree1)
```

```
##
## Classification tree:
```

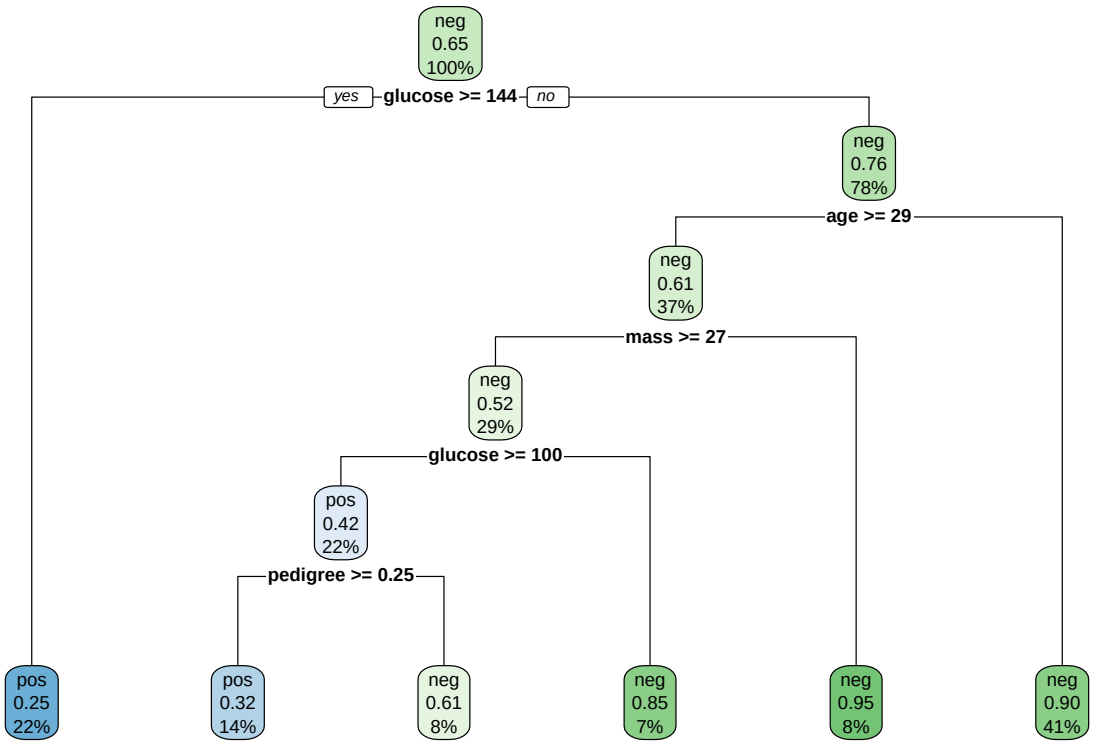
```
## rpart(formula = diabetes ~ ., data = dat, subset = rowTrain,
##     control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
## [1] age      glucose  insulin  mass     pedigree pressure triceps
##
## Root node error: 179/513 = 0.34893
##
## n= 513
##
##          CP nsplit rel error  xerror     xstd
## 1 0.3184358      0   1.00000 1.00000 0.060310
## 2 0.0335196      1   0.68156 0.70950 0.054611
## 3 0.0223464      5   0.53073 0.64804 0.052931
## 4 0.0167598      6   0.50838 0.66480 0.053408
## 5 0.0111732      7   0.49162 0.66480 0.053408
## 6 0.0093110     10   0.45810 0.64804 0.052931
## 7 0.0083799     13   0.43017 0.64804 0.052931
## 8 0.0018622     15   0.41341 0.67598 0.053719
## 9 0.0000000     18   0.40782 0.66480 0.053408
```

```
plotcp(tree1)
```




```
# minimum cross-validation error; may also use the 1SE rule
minErr <- which.min(cpTable[,4])
```

```
tree2 <- prune(tree1, cp = cpTable[minErr,1])
rpart.plot(tree2)
```




```
summary(tree2)
```

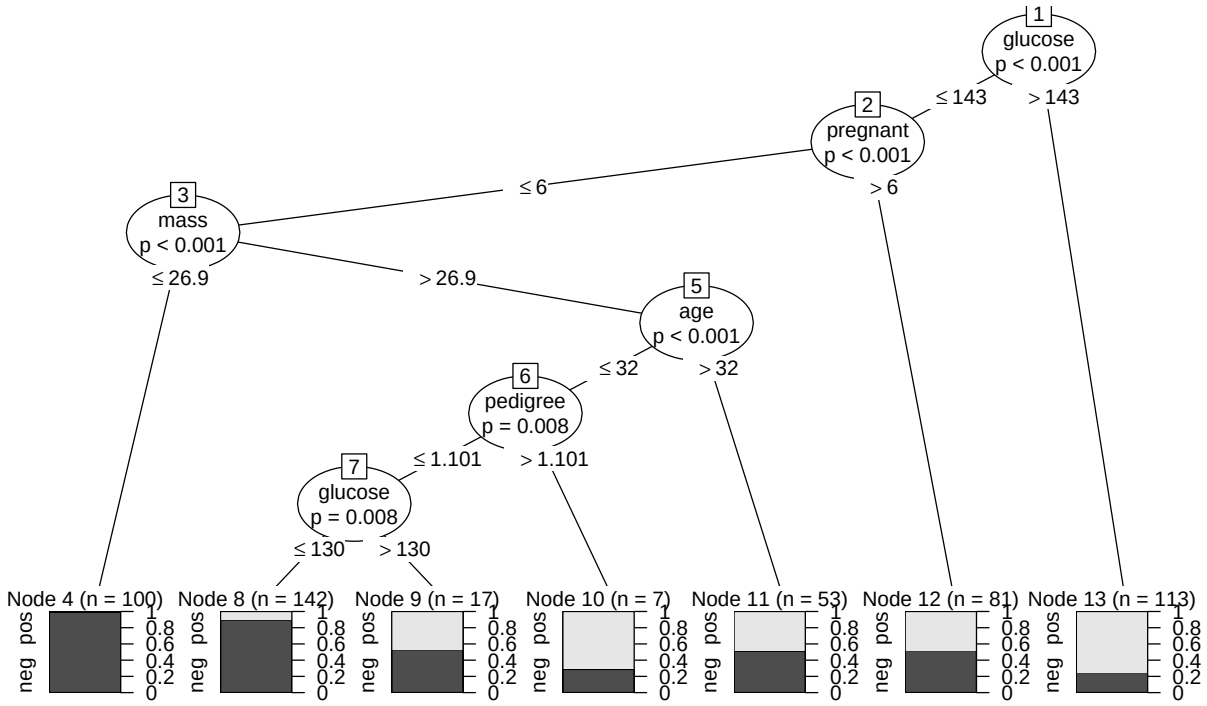
```
## Call:
## rpart(formula = diabetes ~ ., data = dat, subset = rowTrain,
##     control = rpart.control(cp = 0))
##   n= 513
##
##           CP nsplit rel error    xerror       xstd
## 1 0.31843575      0 1.0000000 1.0000000 0.06030982
## 2 0.03351955      1 0.6815642 0.7094972 0.05461146
## 3 0.02234637      5 0.5307263 0.6480447 0.05293130
##
## Variable importance
##  glucose      age     mass pregnant  insulin pedigree pressure  triceps
##       49       15       10        8        5        5        5        3
##
## Node number 1: 513 observations,    complexity param=0.3184358
##   predicted class=neg  expected loss=0.3489279  P(node) =1
##     class counts:   179   334
##    probabilities: 0.349 0.651
##   left son=2 (113 obs) right son=3 (400 obs)
##   Primary splits:
```

```
##       glucose  < 143.5  to the right, improve=47.13993, (0 missing)
##       age      < 28.5   to the right, improve=25.25772, (0 missing)
##       mass     < 26.45  to the right, improve=24.97639, (0 missing)
##       pregnant < 6.5    to the right, improve=18.37466, (0 missing)
##       pedigree < 0.7305 to the right, improve=10.54990, (0 missing)
##   Surrogate splits:
##       insulin  < 281    to the right, agree=0.793, adj=0.062, (0 split)
##       pedigree < 1.756  to the right, agree=0.788, adj=0.035, (0 split)
##       pressure < 109    to the right, agree=0.784, adj=0.018, (0 split)
##       triceps  < 53     to the right, agree=0.784, adj=0.018, (0 split)
##       mass     < 47.55  to the right, agree=0.782, adj=0.009, (0 split)
##
## Node number 2: 113 observations
##   predicted class=pos  expected loss=0.2477876  P(node) =0.2202729
##     class counts:    85    28
##    probabilities: 0.752 0.248
##
## Node number 3: 400 observations,    complexity param=0.03351955
##   predicted class=neg  expected loss=0.235  P(node) =0.7797271
##     class counts:    94   306
##    probabilities: 0.235 0.765
##   left son=6 (188 obs) right son=7 (212 obs)
##   Primary splits:
##       age      < 28.5   to the right, improve=16.671870, (0 missing)
##       mass     < 26.95  to the right, improve=14.292070, (0 missing)
##       pregnant < 6.5    to the right, improve=12.341070, (0 missing)
##       glucose  < 101.5  to the right, improve=11.532000, (0 missing)
##       pedigree < 0.731  to the right, improve= 7.224033, (0 missing)
##   Surrogate splits:
##       pregnant < 3.5    to the right, agree=0.802, adj=0.580, (0 split)
##       pressure < 71     to the right, agree=0.662, adj=0.282, (0 split)
##       insulin  < 8      to the left,  agree=0.625, adj=0.202, (0 split)
##       triceps  < 7.5    to the left,  agree=0.623, adj=0.197, (0 split)
##       glucose  < 113.5  to the right, agree=0.615, adj=0.181, (0 split)
##
## Node number 6: 188 observations,    complexity param=0.03351955
##   predicted class=neg  expected loss=0.3882979  P(node) =0.3664717
##     class counts:    73   115
##    probabilities: 0.388 0.612
##   left son=12 (148 obs) right son=13 (40 obs)
##   Primary splits:
##       mass     < 26.95  to the right, improve=11.630130, (0 missing)
##       glucose  < 99.5   to the right, improve= 9.960993, (0 missing)
##       insulin  < 128    to the right, improve= 4.907038, (0 missing)
##       age      < 56.5   to the left,  improve= 4.407857, (0 missing)
##       pedigree < 1.105  to the right, improve= 4.053126, (0 missing)
##   Surrogate splits:
##       age     < 66.5   to the left,  agree=0.809, adj=0.100, (0 split)
##       glucose < 59     to the right, agree=0.793, adj=0.025, (0 split)
##
## Node number 7: 212 observations
##   predicted class=neg  expected loss=0.0990566  P(node) =0.4132554
##     class counts:    21   191
##    probabilities: 0.099 0.901
```

```
##
## Node number 12: 148 observations,    complexity param=0.03351955
##   predicted class=neg  expected loss=0.4797297  P(node) =0.288499
##     class counts:    71    77
##    probabilities: 0.480 0.520
##   left son=24 (114 obs) right son=25 (34 obs)
##   Primary splits:
##       glucose  < 99.5   to the right, improve=9.770019, (0 missing)
##       pedigree < 0.528  to the right, improve=5.757891, (0 missing)
##       insulin  < 128    to the right, improve=2.993970, (0 missing)
##       age      < 56.5   to the left,  improve=2.604198, (0 missing)
##       pregnant < 6.5    to the right, improve=1.545045, (0 missing)
##
## Node number 13: 40 observations
##   predicted class=neg  expected loss=0.05  P(node) =0.07797271
##     class counts:     2    38
##    probabilities: 0.050 0.950
##
## Node number 24: 114 observations,    complexity param=0.03351955
##   predicted class=pos  expected loss=0.4210526  P(node) =0.2222222
##     class counts:    66    48
##    probabilities: 0.579 0.421
##   left son=48 (73 obs) right son=49 (41 obs)
##   Primary splits:
##       pedigree < 0.253  to the right, improve=4.559903, (0 missing)
##       age      < 56.5   to the left,  improve=2.836624, (0 missing)
##       pressure < 67     to the left,  improve=2.390223, (0 missing)
##       glucose  < 107.5  to the right, improve=1.601170, (0 missing)
##       insulin  < 123.5  to the right, improve=1.478613, (0 missing)
##   Surrogate splits:
##       glucose  < 135.5  to the left,  agree=0.702, adj=0.171, (0 split)
##       pressure < 99     to the left,  agree=0.667, adj=0.073, (0 split)
##       age      < 58     to the left,  agree=0.658, adj=0.049, (0 split)
##
## Node number 25: 34 observations
##   predicted class=neg  expected loss=0.1470588  P(node) =0.0662768
##     class counts:     5    29
##    probabilities: 0.147 0.853
##
## Node number 48: 73 observations
##   predicted class=pos  expected loss=0.3150685  P(node) =0.1423002
##     class counts:    50    23
##    probabilities: 0.685 0.315
##
## Node number 49: 41 observations
##   predicted class=neg  expected loss=0.3902439  P(node) =0.07992203
##     class counts:    16    25
##    probabilities: 0.390 0.610
```

ctree

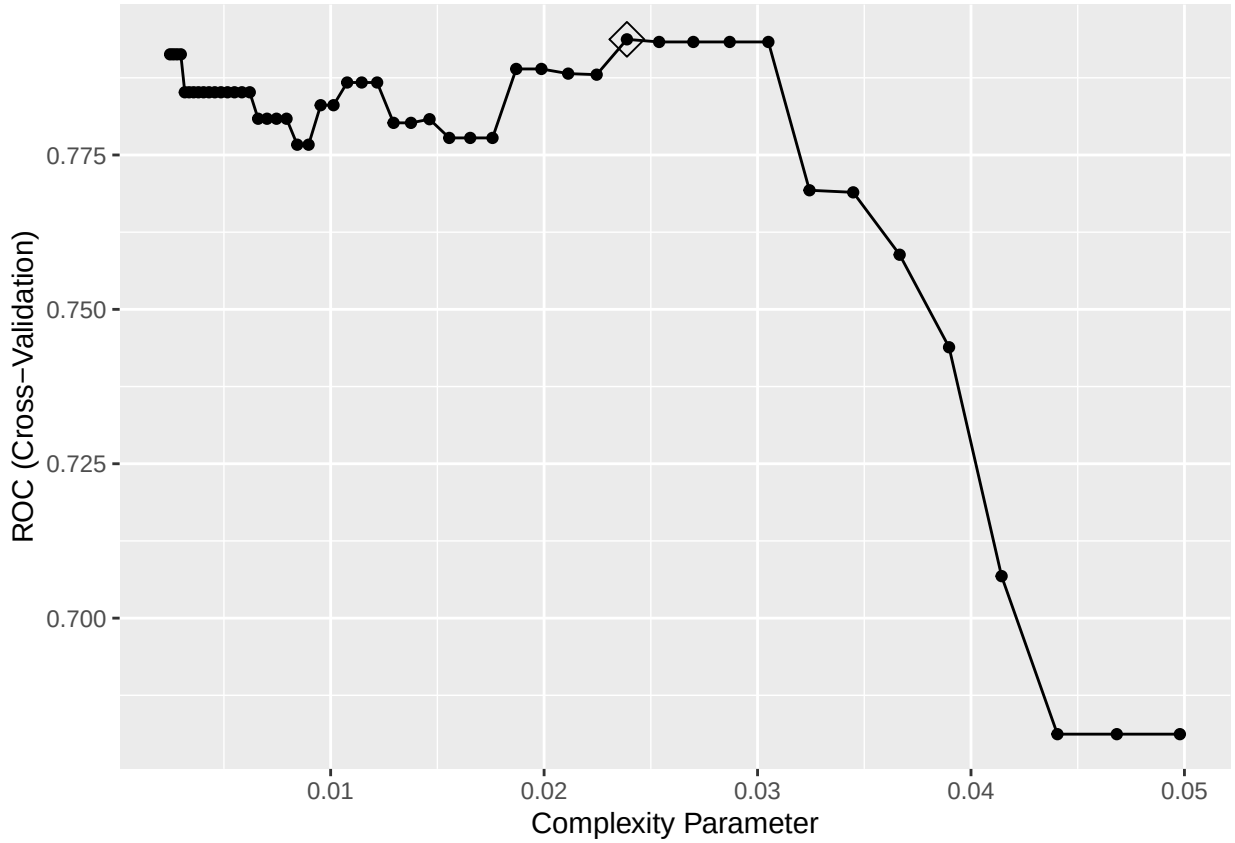
```
tree2 <- ctree(formula = diabetes ~ . ,
               data = dat,
               subset = rowTrain)
plot(tree2)
```



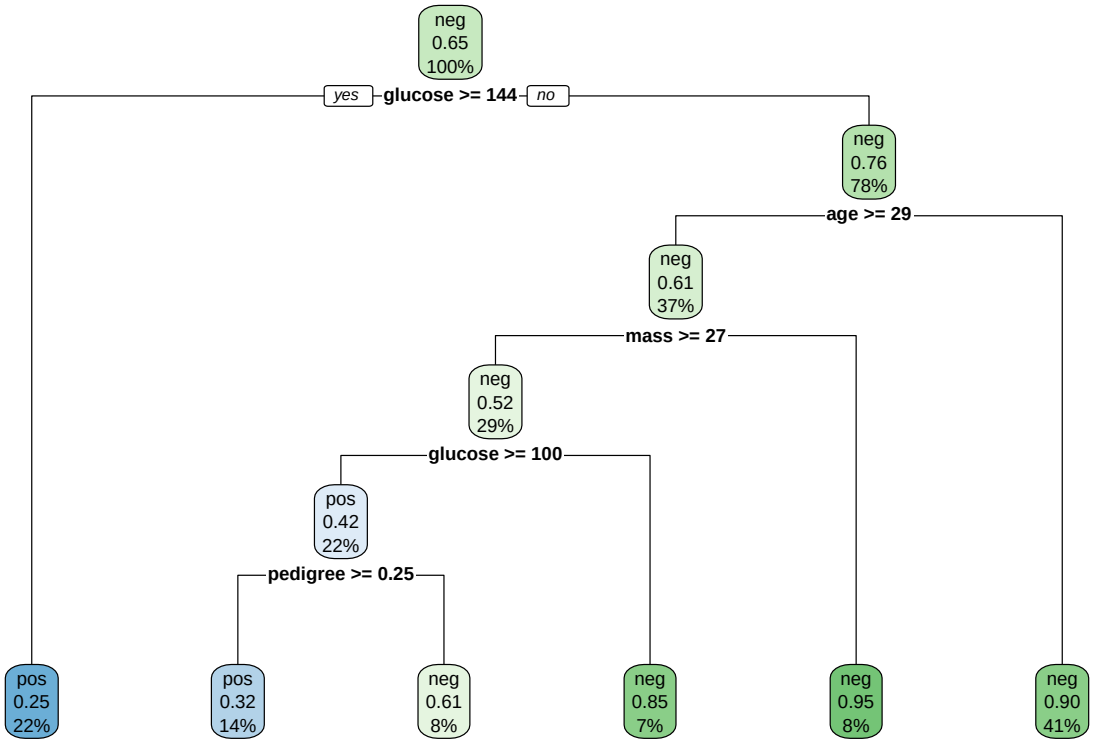
caret

CART

```
ctrl <- trainControl(method = "cv",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)

set.seed(1)
rpart.fit <- train(diabetes ~ . ,
                   dat,
                   subset = rowTrain,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-6,-3, len = 50))),
                   trControl = ctrl,
                   metric = "ROC")
ggplot(rpart.fit, highlight = TRUE)
```

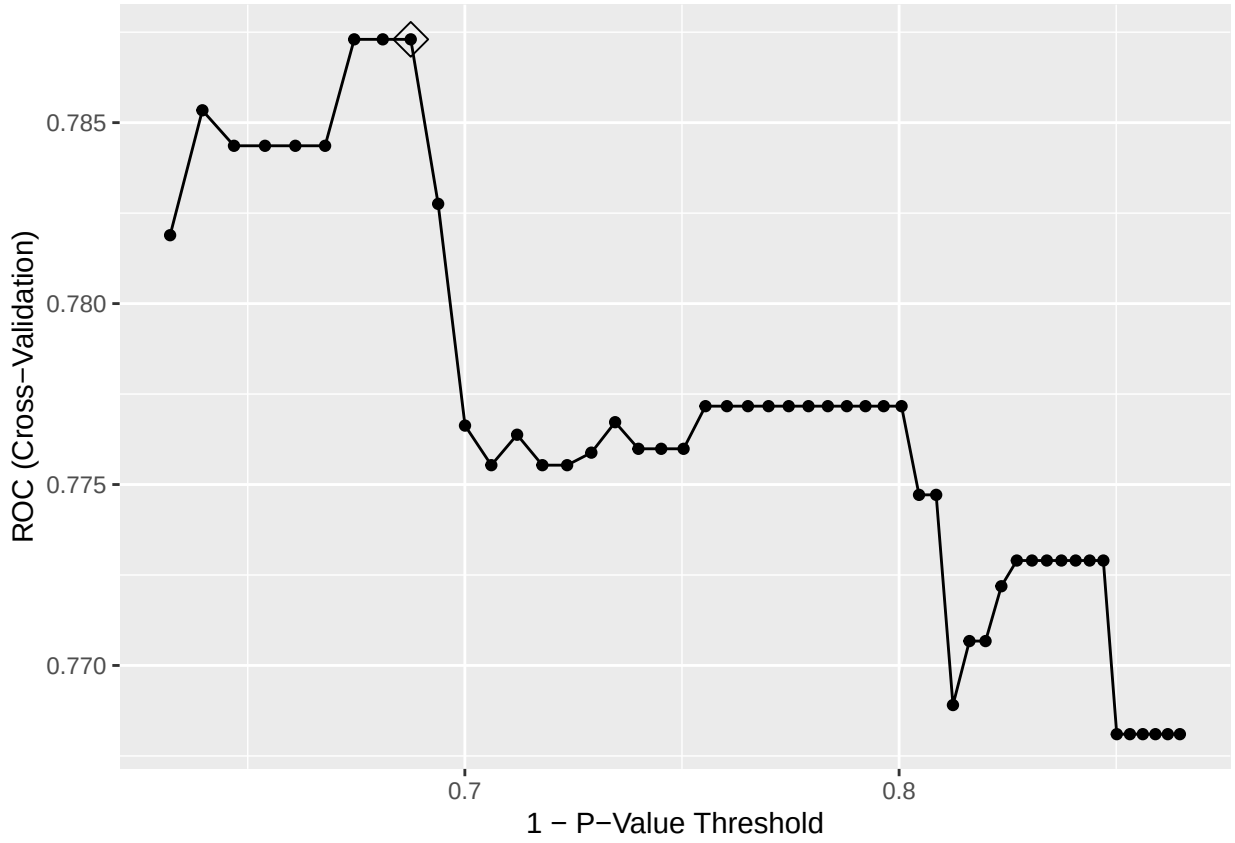



```
rpart.plot(rpart.fit$finalModel)
```

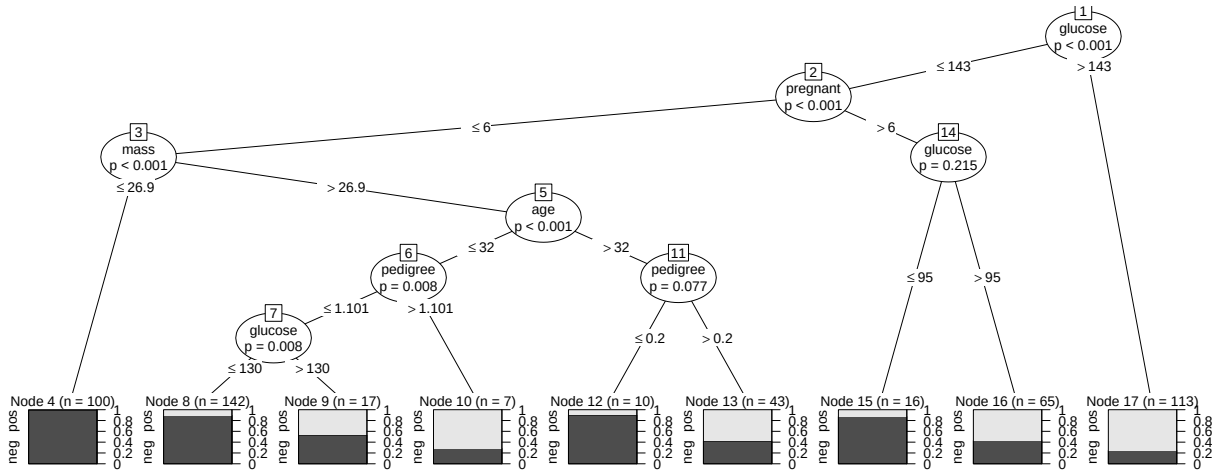


CIT

```
set.seed(1)
ctree.fit <- train(diabetes ~ . , dat,
                   subset = rowTrain,
                   method = "ctree",
                   tuneGrid = data.frame(mincriterion = 1-exp(seq(-2, -1, length = 50))),
                   metric = "ROC",
                   trControl = ctrl)
ggplot(ctree.fit, highlight = TRUE)
```




```
plot(ctree.fit$finalModel)
```

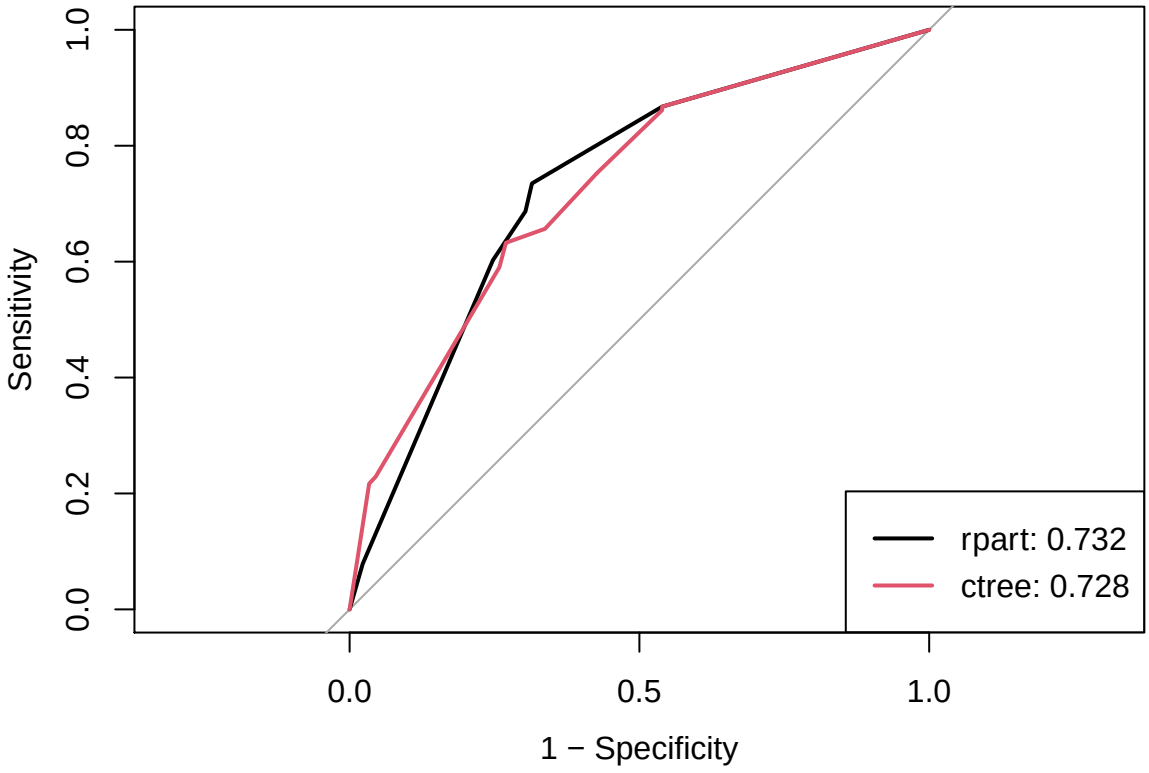



```
summary(resamples(list(rpart.fit, ctree.fit)))
```

```
##
## Call:
## summary.resamples(object = resamples(list(rpart.fit, ctree.fit)))
##
```

```
## Models: Model1, Model2
## Number of resamples: 10
##
## ROC
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Model1 0.6784512 0.7396886 0.7852421 0.7937263 0.8472037 0.9011438    0
## Model2 0.7045455 0.7417929 0.8063478 0.7873018 0.8164983 0.8685121    0
##
## Sens
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Model1 0.6111111 0.6805556 0.7222222 0.7490196 0.8120915 0.9444444    0
## Model2 0.5000000 0.5833333 0.6944444 0.6767974 0.7638889 0.8235294    0
##
## Spec
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Model1 0.6363636 0.7112299 0.8333333 0.7869875 0.8518271 0.8823529    0
## Model2 0.6666667 0.7352941 0.7575758 0.7604278 0.7941176 0.8484848    0
```

```
rpart.pred <- predict(tree1, newdata = dat[-rowTrain,])[,1]

rpart.pred2 <- predict(rpart.fit, newdata = dat[-rowTrain,],
                       type = "prob")[,1]

ctree.pred <- predict(ctree.fit, newdata = dat[-rowTrain,],
                       type = "prob")[,1]

roc.rpart <- roc(dat$diabetes[-rowTrain], rpart.pred2)
roc.ctree <- roc(dat$diabetes[-rowTrain], ctree.pred)

auc <- c(roc.rpart$auc[1], roc.ctree$auc[1])

plot(roc.rpart, legacy.axes = TRUE)
plot(roc.ctree, col = 2, add = TRUE)


modelNames <- c("rpart","ctree")
legend("bottomright", legend = paste0(modelNames, ": ", round(auc,3)),
       col = 1:2, lwd = 2)
```


Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.5
1.0
1 − Specificity
rpart: 0.732
ctree: 0.728