Classification Tree: Carseats dataset

Nathacia Nathacia

2022-07-09

```
library(ISLR)
library(tree)
library(caret)
```

Load packages and dataset

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
data(package="ISLR")
dtree <- Carseats
class(dtree)
dtree$Sales
View(dtree)
```

Investigate the data

```
names(dtree)
```

```
##  [1] "Sales"       "CompPrice"   "Income"      "Advertising" "Population"
##  [6] "Price"       "ShelveLoc"   "Age"         "Education"   "Urban"
## [11] "US"
```

```
head(dtree)
```

```
##   Sales CompPrice Income Advertising Population Price ShelveLoc Age Education
## 1  9.50       138     73          11        276   120       Bad  42        17
## 2 11.22       111     48          16        260    83      Good  65        10
## 3 10.06       113     35          10        269    80    Medium  59        12
## 4  7.40       117    100           4        466    97    Medium  55        14
## 5  4.15       141     64           3        340   128       Bad  38        13
## 6 10.81       124    113          13        501    72       Bad  78        16
##   Urban  US
## 1   Yes Yes
## 2   Yes Yes
## 3   Yes Yes
## 4   Yes Yes
## 5   Yes  No
## 6    No Yes
```

```
tail(dtree)
```

```
##     Sales CompPrice Income Advertising Population Price ShelveLoc Age Education
## 395  5.35       130     58          19        366   139       Bad  33        16
```

```
## 396 12.57       138    108          17        203   128      Good  33        14
## 397  6.14       139     23           3         37   120    Medium  55        11
## 398  7.41       162     26          12        368   159    Medium  40        18
## 399  5.94       100     79           7        284    95       Bad  50        12
## 400  9.71       134     37           0         27   120      Good  49        16
##     Urban  US
## 395   Yes Yes
## 396   Yes Yes
## 397    No Yes
## 398   Yes Yes
## 399   Yes Yes
## 400   Yes Yes
```

```
summary(dtree)
```

```
##      Sales          CompPrice       Income        Advertising
##  Min.   : 0.000   Min.   : 77   Min.   : 21.00   Min.   : 0.000
##  1st Qu.: 5.390   1st Qu.:115   1st Qu.: 42.75   1st Qu.: 0.000
##  Median : 7.490   Median :125   Median : 69.00   Median : 5.000
##  Mean   : 7.496   Mean   :125   Mean   : 68.66   Mean   : 6.635
##  3rd Qu.: 9.320   3rd Qu.:135   3rd Qu.: 91.00   3rd Qu.:12.000
##  Max.   :16.270   Max.   :175   Max.   :120.00   Max.   :29.000
##    Population        Price        ShelveLoc        Age          Education
##  Min.   : 10.0   Min.   : 24.0   Bad   : 96   Min.   :25.00   Min.   :10.0
##  1st Qu.:139.0   1st Qu.:100.0   Good  : 85   1st Qu.:39.75   1st Qu.:12.0
##  Median :272.0   Median :117.0   Medium:219   Median :54.50   Median :14.0
##  Mean   :264.8   Mean   :115.8                Mean   :53.32   Mean   :13.9
##  3rd Qu.:398.5   3rd Qu.:131.0                3rd Qu.:66.00   3rd Qu.:16.0
##  Max.   :509.0   Max.   :191.0                Max.   :80.00   Max.   :18.0
##  Urban       US
##  No :118   No :142
##  Yes:282   Yes:258
##
##
##
##
```

Preprocessing - discretization

```
High <- ifelse(dtree$Sales<=8, "NO", "YES")
class(High)
```

```
## [1] "character"
```

```
High <- as.factor(High)
class(High)
```

```
## [1] "factor"
```

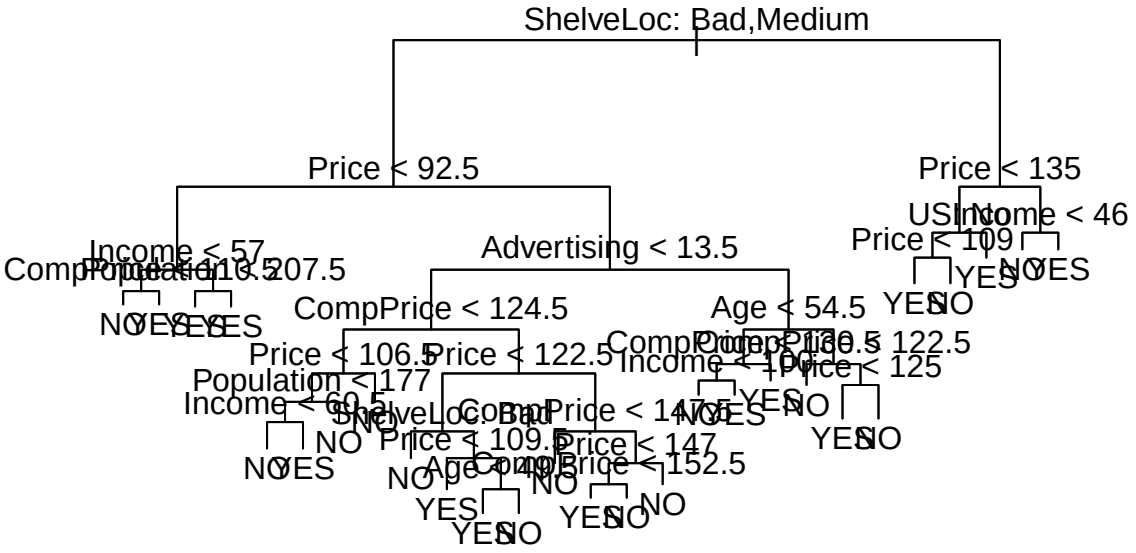
```
dtree <- data.frame(dtree,High)
dtree <- subset(dtree, select=-Sales)
```

Training

```
dtree.carseats <- tree(High~., dtree)
summary(dtree.carseats)
```

```
##
## Classification tree:
## tree(formula = High ~ ., data = dtree)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Income"      "CompPrice"   "Population"
## [6] "Advertising" "Age"         "US"
## Number of terminal nodes:  27
## Residual mean deviance:  0.4575 = 170.7 / 373
## Misclassification error rate: 0.09 = 36 / 400
```

```
plot(dtree.carseats)
text(dtree.carseats, pretty = 0)
```




```
print(dtree.carseats)
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##   1) root 400 541.500 NO ( 0.59000 0.41000 )
##     2) ShelveLoc: Bad,Medium 315 390.600 NO ( 0.68889 0.31111 )
##       4) Price < 92.5 46  56.530 YES ( 0.30435 0.69565 )
##         8) Income < 57 10  12.220 NO ( 0.70000 0.30000 )
##          16) CompPrice < 110.5 5   0.000 NO ( 1.00000 0.00000 ) *
##          17) CompPrice > 110.5 5   6.730 YES ( 0.40000 0.60000 ) *
##         9) Income > 57 36  35.470 YES ( 0.19444 0.80556 )
##          18) Population < 207.5 16  21.170 YES ( 0.37500 0.62500 ) *
##          19) Population > 207.5 20   7.941 YES ( 0.05000 0.95000 ) *
##       5) Price > 92.5 269 299.800 NO ( 0.75465 0.24535 )
##        10) Advertising < 13.5 224 213.200 NO ( 0.81696 0.18304 )
##          20) CompPrice < 124.5 96  44.890 NO ( 0.93750 0.06250 )
##            40) Price < 106.5 38  33.150 NO ( 0.84211 0.15789 )
##              80) Population < 177 12  16.300 NO ( 0.58333 0.41667 )
##               160) Income < 60.5 6   0.000 NO ( 1.00000 0.00000 ) *
##               161) Income > 60.5 6   5.407 YES ( 0.16667 0.83333 ) *
##              81) Population > 177 26   8.477 NO ( 0.96154 0.03846 ) *
##            41) Price > 106.5 58   0.000 NO ( 1.00000 0.00000 ) *
```

```
##          21) CompPrice > 124.5 128 150.200 NO ( 0.72656 0.27344 )
##            42) Price < 122.5 51  70.680 YES ( 0.49020 0.50980 )
##              84) ShelveLoc: Bad 11   6.702 NO ( 0.90909 0.09091 ) *
##              85) ShelveLoc: Medium 40  52.930 YES ( 0.37500 0.62500 )
##               170) Price < 109.5 16   7.481 YES ( 0.06250 0.93750 ) *
##               171) Price > 109.5 24  32.600 NO ( 0.58333 0.41667 )
##                 342) Age < 49.5 13  16.050 YES ( 0.30769 0.69231 ) *
##                 343) Age > 49.5 11   6.702 NO ( 0.90909 0.09091 ) *
##            43) Price > 122.5 77  55.540 NO ( 0.88312 0.11688 )
##              86) CompPrice < 147.5 58  17.400 NO ( 0.96552 0.03448 ) *
##              87) CompPrice > 147.5 19  25.010 NO ( 0.63158 0.36842 )
##               174) Price < 147 12  16.300 YES ( 0.41667 0.58333 )
##                 348) CompPrice < 152.5 7   5.742 YES ( 0.14286 0.85714 ) *
##                 349) CompPrice > 152.5 5   5.004 NO ( 0.80000 0.20000 ) *
##               175) Price > 147 7   0.000 NO ( 1.00000 0.00000 ) *
##        11) Advertising > 13.5 45  61.830 YES ( 0.44444 0.55556 )
##          22) Age < 54.5 25  25.020 YES ( 0.20000 0.80000 )
##            44) CompPrice < 130.5 14  18.250 YES ( 0.35714 0.64286 )
##              88) Income < 100 9  12.370 NO ( 0.55556 0.44444 ) *
##              89) Income > 100 5   0.000 YES ( 0.00000 1.00000 ) *
##            45) CompPrice > 130.5 11   0.000 YES ( 0.00000 1.00000 ) *
##          23) Age > 54.5 20  22.490 NO ( 0.75000 0.25000 )
##            46) CompPrice < 122.5 10   0.000 NO ( 1.00000 0.00000 ) *
##            47) CompPrice > 122.5 10  13.860 NO ( 0.50000 0.50000 )
##              94) Price < 125 5   0.000 YES ( 0.00000 1.00000 ) *
##              95) Price > 125 5   0.000 NO ( 1.00000 0.00000 ) *
##     3) ShelveLoc: Good 85  90.330 YES ( 0.22353 0.77647 )
##       6) Price < 135 68  49.260 YES ( 0.11765 0.88235 )
##        12) US: No 17  22.070 YES ( 0.35294 0.64706 )
##          24) Price < 109 8   0.000 YES ( 0.00000 1.00000 ) *
##          25) Price > 109 9  11.460 NO ( 0.66667 0.33333 ) *
##        13) US: Yes 51  16.880 YES ( 0.03922 0.96078 ) *
##       7) Price > 135 17  22.070 NO ( 0.64706 0.35294 )
##        14) Income < 46 6   0.000 NO ( 1.00000 0.00000 ) *
##        15) Income > 46 11  15.160 YES ( 0.45455 0.54545 ) *
```

Splitting the data

Splitting the original data set to create training and testing data sets

```
set.seed(123)
train.index <- sample(1:nrow(dtree), 200)
nrow(dtree)
```

```
## [1] 400
```

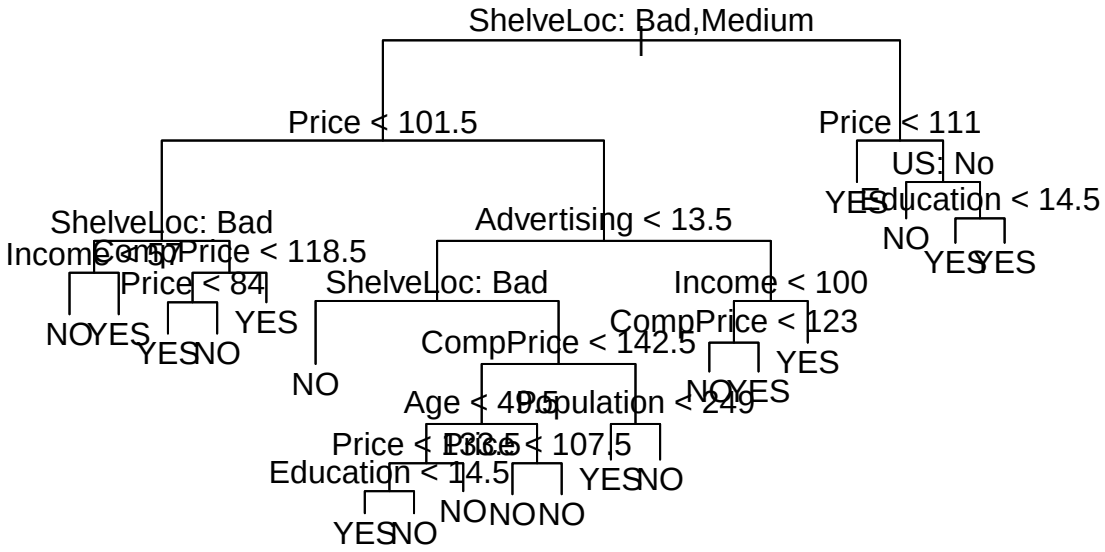
Splitting

```
train.set <- dtree[train.index,]
class(train.set)
```

```
## [1] "data.frame"
```

Training

```
dtree.tree <- tree(High~., train.set)
plot(dtree.tree)
text(dtree.tree, pretty = 0)
```




```
summary(dtree.tree)
```

```
##
## Classification tree:
## tree(formula = High ~ ., data = train.set)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Income"      "CompPrice"   "Advertising"
## [6] "Age"         "Education"   "Population"  "US"
## Number of terminal nodes:  20
## Residual mean deviance:  0.5056 = 91.02 / 180
## Misclassification error rate: 0.115 = 23 / 200
```

Testing

```
test.set <- dtree[-train.index,]
High.test <- High[-train.index]

tree.pred <- predict(dtree.tree, test.set, type="class")
```

Accuracy

```
table(tree.pred, High.test)
```

```
##          High.test
## tree.pred NO YES
##       NO  87  13
##       YES 35  65
```

```
accuracy = (87+65)/200
accuracy
```

```
## [1] 0.76
```

Testing continued

Predict the class of the test set using the trained decision tree

```
tree.pred <- predict(dtree.tree, test.set, type="class")
#If type = "class": for a classification tree, a factor of the predicted
#classes (that with highest posterior probability, with ties split randomly).
```

Confusion matrix

Compare the predicted classes with the actual classes to evaluate the performance of the decision tree

```
table(tree.pred, High.test)
```

```
##          High.test
## tree.pred NO YES
##       NO  87  13
##       YES 35  65
```

```
#calculate the accuracy of the decision tree on the test set
accuracy = (87+65)/200
accuracy
```

```
## [1] 0.76
```

```
#calculating the misclassification rate (1-accuracy)
misclassification_rate <- 1-accuracy
misclassification_rate
```

```
## [1] 0.24
```

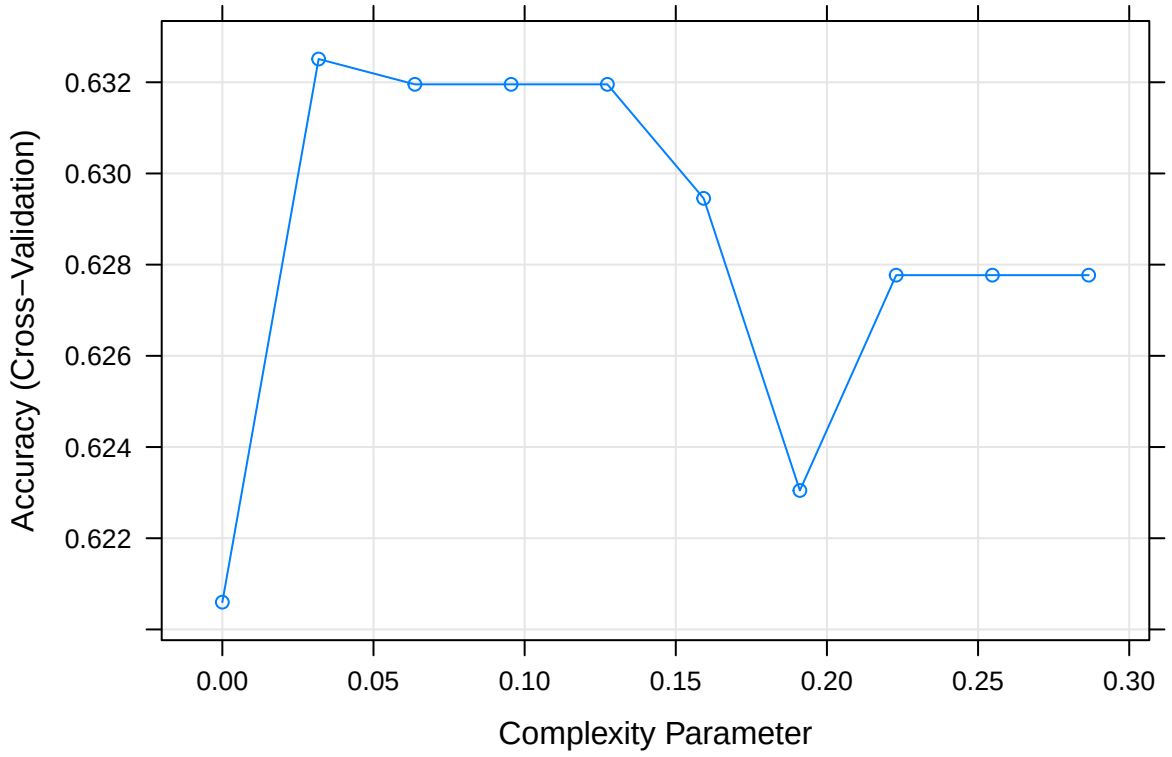
K-fold cross-validation

```
set.seed(123)
folds <- createFolds(dtree$High, k=10, list=TRUE, returnTrain=FALSE)
ctrl <- trainControl(method="cv", index=folds, savePredictions="final", classProbs=TRUE)
set.seed(123)
dtree.tree.cv <- train(High~., data=dtree, method="rpart", trControl=ctrl, tuneLength=10)
dtree.tree.cv
```

```
## CART
##
## 400 samples
##  10 predictor
##   2 classes: 'NO', 'YES'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 40, 40, 40, 39, 40, 41, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.00000000  0.6205982  0.2172401
##   0.03184282  0.6325095  0.2316952
##   0.06368564  0.6319540  0.2143365
```

```
##   0.09552846  0.6319540  0.2143365
##   0.12737127  0.6319540  0.2143365
##   0.15921409  0.6294540  0.2041994
##   0.19105691  0.6230473  0.1845560
##   0.22289973  0.6277695  0.1835560
##   0.25474255  0.6277695  0.1835560
##   0.28658537  0.6277695  0.1835560
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03184282.
```

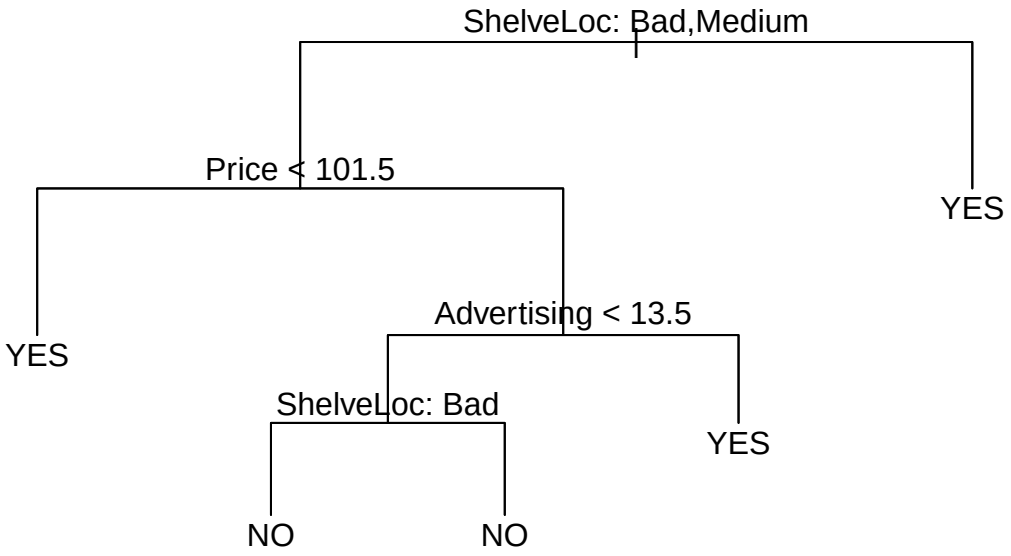
```
plot(dtree.tree.cv)
```



Fine-tuning the decision tree

Pruning the decision tree to avoid overfitting

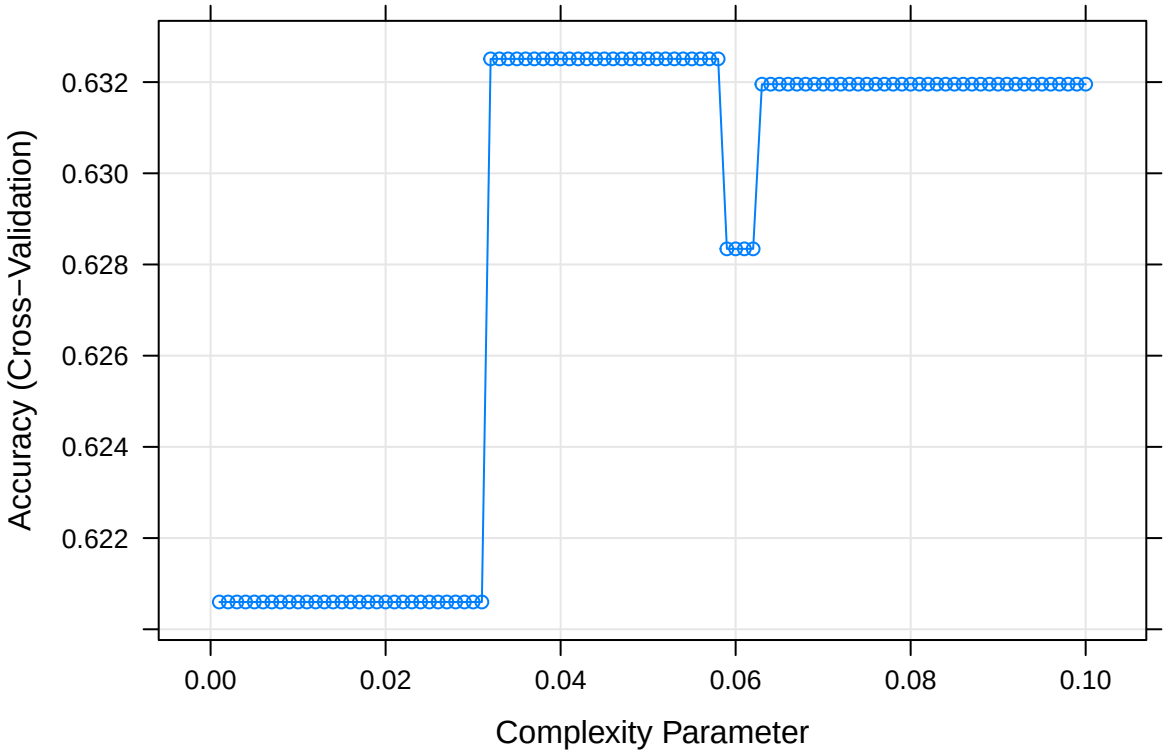
```
dtree.tree.prune <- prune.tree(dtree.tree, best=4)
plot(dtree.tree.prune)
text(dtree.tree.prune, pretty = 0)
```



Cross-validation

Using cross-validation to determine the optimal complexity parameter for pruning

```
dtree.tree.cv.prune <- train(High~., data=dtree, method="rpart", trControl=ctrl, tuneLength=10, tuneGri
plot(dtree.tree.cv.prune)
```



Predicting class of test set

Using the pruned decision tree to predict the class of the test set

```
tree.pred.prune <- predict(dtree.tree.prune, test.set, type="class")
table(tree.pred.prune, High.test)
```

```
##                High.test
## tree.pred.prune NO YES
##              NO  77  12
##              YES 45  66
```

Accuracy and misclassification rate

```
accuracy.prune <- sum(tree.pred.prune == High.test)/length(High.test)
misclassification_rate.prune <- 1-accuracy.prune
accuracy.prune
```

```
## [1] 0.715
```

```
misclassification_rate.prune
```

```
## [1] 0.285
```