Session 1: Decision Trees in R

JR Ferrer-Paris

02/08/2021

Abstract

This document will walk you through examples to fit decision trees for classification using the *rpart* package and two different datasets. This is part of the UNSW codeRs workshop: *Introduction to Classification Trees and Random Forests in R* at https://github.com/UNSW-codeRs/workshop-random-forests

UNSW codeRs is a student and staff run community dedicated for 'R' users for anyone who wants to further develop their coding skills. It is our goal to create a safe and open space for members to share and gain new experiences relating to R, coding and statistics.

https://unsw-coders.netlify.app/

Overview

Decision trees are recursive partitioning methods that divide the predictor spaces into simpler regions and can be visualized in a tree-like structure. They attempt to classify data by dividing it into subsets according to a Y output variable and based on some predictors.

What data do we need?

- Y: The output or response variable is a categorical variable with two or more classes (in R: factor with two or more levels)
- X: A set of predictors or features, might be a mix of continuous and categorical variables, they should not have any missing values

Load data

Here we will work with two examples.

First, we will use the *iris* dataset from base R. This dataset has 150 observations with four measurements (continuous variables) for three species (categorical variable with three categories):

```
data(iris)
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

As a second example we will use the Breast Cancer dataset from the *mlbench* package. This dataset has 699 observations with 9 nominal or ordinal variables describing cell properties and the output or target variable is the class of tumor with two possible values: bening or malignant:

```
require(mlbench)
```

```
## Loading required package: mlbench
```

```
data(BreastCancer)
str(BreastCancer)
```

```
## 'data.frame':    699 obs. of  11 variables:
##  $ Id             : chr  "1000025" "1002945" "1015425" "1016277" ...
##  $ Cl.thickness   : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 5 5 3 6 4 8 1 2 2 4 ...
##  $ Cell.size      : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 4 1 8 1 10 1 1 1 2 ...
##  $ Cell.shape     : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 4 1 8 1 10 1 2 1 1 ...
##  $ Marg.adhesion  : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 5 1 1 3 8 1 1 1 1 ...
##  $ Epith.c.size   : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 2 7 2 3 2 7 2 2 2 2 ...
##  $ Bare.nuclei    : Factor w/ 10 levels "1","2","3","4",..: 1 10 2 4 1 10 10 1 1 1 ...
##  $ Bl.cromatin    : Factor w/ 10 levels "1","2","3","4",..: 3 3 3 3 3 9 3 3 1 2 ...
##  $ Normal.nucleoli: Factor w/ 10 levels "1","2","3","4",..: 1 2 1 7 1 7 1 1 1 1 ...
##  $ Mitoses        : Factor w/ 9 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 5 1 ...
##  $ Class          : Factor w/ 2 levels "benign","malignant": 1 1 1 1 1 2 1 1 1 1 ...
```

What package to use

Classification trees are implemented in packages:

- *tree*: Classification and Regression Trees
- *rpart*: Recursive Partitioning and Regression Trees

Load packages

Here we will work with package *rpart*, and we will also load additional packages for creating the plots

```
library(rpart)
# auxiliary packages for plotting:
library(rpart.plot)
```

Fit a model

iris dataset

Let's start with a familiar dataset:

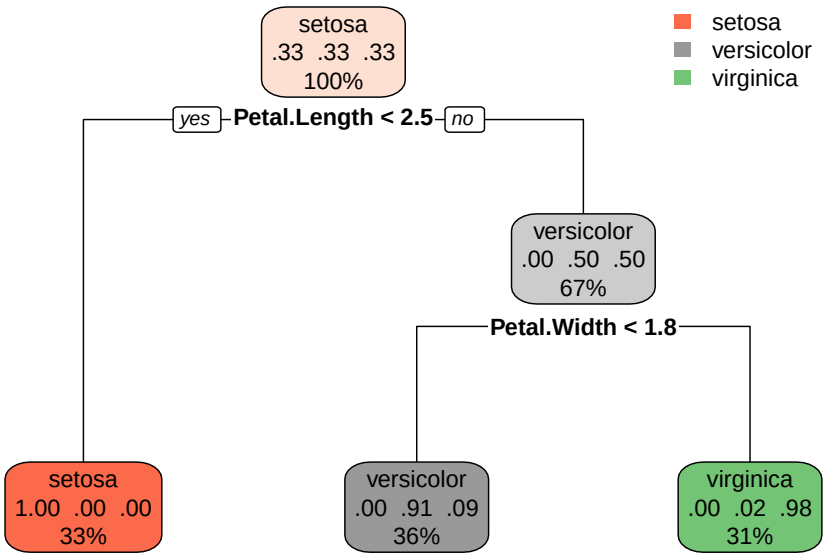
```
set.seed(3)

tree = rpart::rpart(Species ~ ., data = iris,
             method = "class")
print(tree)
```

```
## n= 150
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)
##   2) Petal.Length< 2.45 50   0 setosa (1.00000000 0.00000000 0.00000000) *
##   3) Petal.Length>=2.45 100  50 versicolor (0.00000000 0.50000000 0.50000000)
##     6) Petal.Width< 1.75 54   5 versicolor (0.00000000 0.90740741 0.09259259) *
##     7) Petal.Width>=1.75 46   1 virginica (0.00000000 0.02173913 0.97826087) *
```

This is a very simple tree and we can walk through the output recognising three levels of nodes: (1) is the root node, (2) and (3) are the branches based on Petal Length. Branch (2) has 50 samples all belong to the first class (setosa), branch (3) has 100 samples of two different classes. Branch (3) splits into two further branches (6) and (7) based on petal width, these end-nodes (or leaf-nodes) have 54 and 46 samples respectively.

We can visualise the same information in a fancy *rpart.plot*:

```
rpart.plot::rpart.plot(tree)
```



This function use different colors for each category, splits are labelled with the variable and threshold used. Root nodes are on top, and end-nodes are at the bottom, each node is labelled with the modal category and has information on the proportion of observation in each category and the percentage of the total sample size.

When an end node only contains samples from a single class it is considered to be "pure". So the end-node for *I. setosa* at the bottom left is pure, the other end-nodes have 2 and 9% impurity.

Breast cancer dataset

Now let's look at a more challenging dataset.

```
set.seed(3)

BC.data <- BreastCancer[,-1]

tree = rpart::rpart(Class ~ ., data = BC.data,
             method = "class")
print(tree)
```

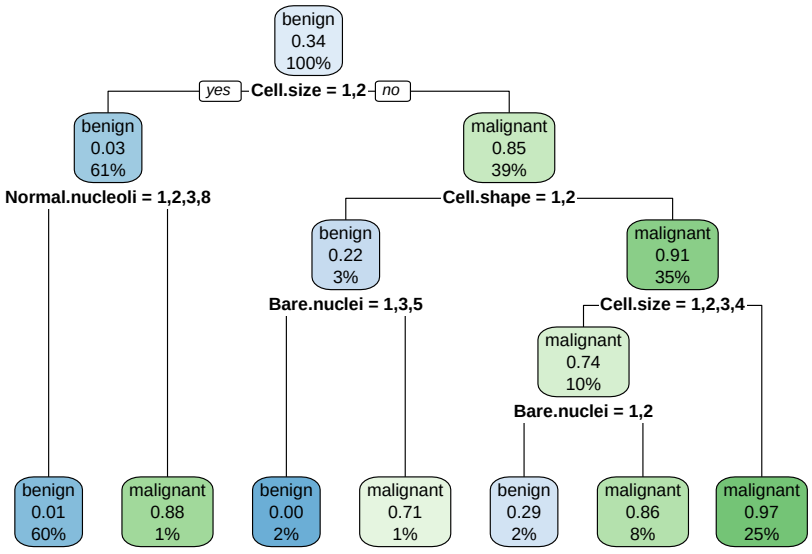
```
## n= 699
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
```

```
##  1) root 699 241 benign (0.65522175 0.34477825)
##    2) Cell.size=1,2 429  12 benign (0.97202797 0.02797203)
##      4) Normal.nucleoli=1,2,3,8 421   5 benign (0.98812352 0.01187648) *
##      5) Normal.nucleoli=4,6,7,10 8   1 malignant (0.12500000 0.87500000) *
##    3) Cell.size=3,4,5,6,7,8,9,10 270  41 malignant (0.15185185 0.84814815)
##      6) Cell.shape=1,2 23   5 benign (0.78260870 0.21739130)
##       12) Bare.nuclei=1,3,5 16   0 benign (1.00000000 0.00000000) *
##       13) Bare.nuclei=2,4,10 7   2 malignant (0.28571429 0.71428571) *
##      7) Cell.shape=3,4,5,6,7,8,9,10 247  23 malignant (0.09311741 0.90688259)
##       14) Cell.size=1,2,3,4 70  18 malignant (0.25714286 0.74285714)
##         28) Bare.nuclei=1,2 14   4 benign (0.71428571 0.28571429) *
##         29) Bare.nuclei=3,4,5,7,8,9,10 56   8 malignant (0.14285714 0.85714286) *
##       15) Cell.size=5,6,7,8,9,10 177   5 malignant (0.02824859 0.97175141) *
```

This is a more complex tree with up to five levels of branching, can you see them?

The plot is a great visual aid, but what do all these values mean?

```
rpart.plot::rpart.plot(tree)
```



Here the output or response variable has two categories, so the rules are slightly simplified, but is actually all pretty similar as the previos example. Each box is labelled with the modal category on top, the proportion of observations in the second class within each group (in this case 'malignant'), and the percentage of total observation within the group. Compare figure and text to try make sense of this.

Variable importance We can also look inside of `tree` object to see its components, for example "variable.importance":

```
names(tree)
```

```
##  [1] "frame"               "where"               "call"
##  [4] "terms"               "cptable"             "method"
##  [7] "parms"               "control"             "functions"
## [10] "numresp"             "splits"              "csplit"
## [13] "variable.importance" "y"                   "ordered"
```

```
data.frame(tree$variable.importance)
```

```
##                 tree.variable.importance
## Cell.size                       228.196290
## Cell.shape                      195.580593
## Normal.nucleoli                 167.558093
## Epith.c.size                    164.615713
## Bl.cromatin                     160.203228
## Bare.nuclei                     154.590550
## Mitoses                           5.763756
## Cl.thickness                      4.301576
## Marg.adhesion                     2.655170
```

Complexity parameter In decision trees the main hyperparameter (configuration setting) is the **complexity parameter** (CP), but the name is a little counterintuitive; a high CP results in a simple decision tree with few splits, whereas a low CP results in a larger decision tree with many splits.

`rpart` uses cross-validation internally to estimate the accuracy at various CP settings. We can review those to see what setting seems best.

Print the results for various CP settings - we want the one with the lowest "xerror".

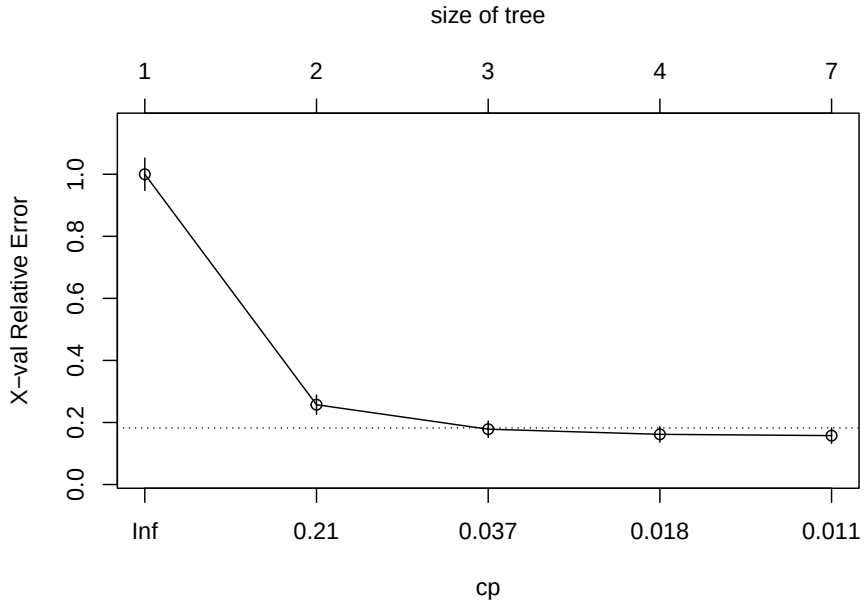
```
printcp(tree)
```

```
##
## Classification tree:
## rpart::rpart(formula = Class ~ ., data = BC.data, method = "class")
##
## Variables actually used in tree construction:
## [1] Bare.nuclei     Cell.shape      Cell.size       Normal.nucleoli
##
## Root node error: 241/699 = 0.34478
##
## n= 699
##
##         CP nsplit rel error  xerror     xstd
```

```
## 1 0.780083      0   1.00000 1.00000 0.052142
## 2 0.053942      1   0.21992 0.25726 0.031190
## 3 0.024896      2   0.16598 0.17842 0.026359
## 4 0.012448      3   0.14108 0.16183 0.025180
## 5 0.010000      6   0.10373 0.15768 0.024873
```

We can visualise this using this function:

```
plotcp(tree)
```

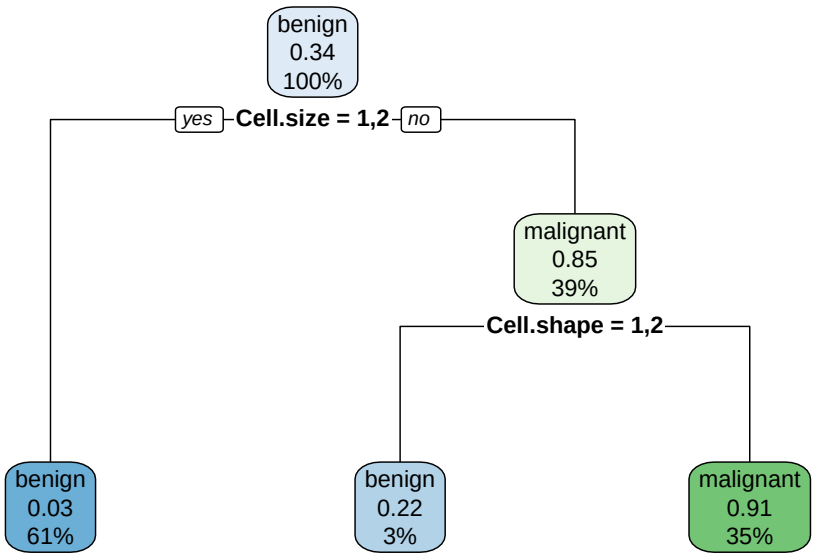


There is an obvious drop between 1 and 2, but afterwards the differences in xerror are pretty small. Considering that a tree with fewer splits might be easier to interpret we can adjust the `cp` value:

```
tree_pruned2 = prune(tree, cp = 0.037)
```

How does it look?

```
rpart.plot(tree_pruned2)
```



So this tree looks much simpler, but is it good enough? Notice that the end-nodes on the left and right have relative low impurity and together include 96% of the original sample. So with only two variables we can get a good discrimination of most of the samples.

If we want to look at the detailed results, variable importance, and summary of splits we can use:

```
summary(tree_pruned2)
```

```
## Call:
## rpart::rpart(formula = Class ~ ., data = BC.data, method = "class")
##   n= 699
##
##           CP nsplit rel error    xerror       xstd
## 1 0.78008299      0 1.0000000 1.0000000 0.05214175
## 2 0.05394191      1 0.2199170 0.2572614 0.03118961
## 3 0.03700000      2 0.1659751 0.1784232 0.02635910
##
## Variable importance
##       Cell.size      Cell.shape    Epith.c.size     Bl.cromatin Normal.nucleoli
##              22              19              16              15              15
##     Bare.nuclei
##              14
##
## Node number 1: 699 observations,    complexity param=0.780083
```

```
##   predicted class=benign     expected loss=0.3447783  P(node) =1
##     class counts:   458   241
##    probabilities: 0.655 0.345
##   left son=2 (429 obs) right son=3 (270 obs)
##   Primary splits:
##       Cell.size    splits as  LLRRRRRRRR, improve=222.9401, (0 missing)
##       Cell.shape   splits as  LLLRRRRRRR, improve=216.3834, (0 missing)
##       Bare.nuclei  splits as  LLRRRRRRRR, improve=203.7284, (16 missing)
##       Bl.cromatin  splits as  LLLRRRRRRR, improve=197.9057, (0 missing)
##       Epith.c.size splits as  LLRRRRRRRR, improve=190.5300, (0 missing)
##   Surrogate splits:
##       Cell.shape      splits as  LLLRRRRRRR, agree=0.916, adj=0.781, (0 split)
##       Epith.c.size    splits as  LLRRRRRRRR, agree=0.897, adj=0.733, (0 split)
##       Normal.nucleoli splits as  LLRRRRRRRR, agree=0.880, adj=0.689, (0 split)
##       Bl.cromatin     splits as  LLLRRRRRRR, agree=0.877, adj=0.681, (0 split)
##       Bare.nuclei     splits as  LLRRRRRRRR, agree=0.860, adj=0.637, (0 split)
##
## Node number 2: 429 observations
##   predicted class=benign     expected loss=0.02797203  P(node) =0.6137339
##     class counts:   417    12
##    probabilities: 0.972 0.028
##
## Node number 3: 270 observations,    complexity param=0.05394191
##   predicted class=malignant  expected loss=0.1518519  P(node) =0.3862661
##     class counts:    41   229
##    probabilities: 0.152 0.848
##   left son=6 (23 obs) right son=7 (247 obs)
##   Primary splits:
##       Cell.shape    splits as  LLRRRRRRRR, improve=20.00546, (0 missing)
##       Cell.size     splits as  LLLRRRRRRR, improve=17.38477, (0 missing)
##       Bare.nuclei   splits as  LRRRRRRRRR, improve=16.81493, (5 missing)
##       Bl.cromatin   splits as  LLRRRRRRRR, improve=14.86975, (0 missing)
##       Marg.adhesion splits as  LLRRRRRRRR, improve=10.74334, (0 missing)
##   Surrogate splits:
##       Bl.cromatin splits as  LRRRRRRRRR, agree=0.933, adj=0.217, (0 split)
##
## Node number 6: 23 observations
##   predicted class=benign     expected loss=0.2173913  P(node) =0.03290415
##     class counts:    18     5
##    probabilities: 0.783 0.217
##
## Node number 7: 247 observations
##   predicted class=malignant  expected loss=0.09311741  P(node) =0.3533619
##     class counts:    23   224
##    probabilities: 0.093 0.907
```

And compare this output with the previous `tree` object.

That's it for now! Let's move to the next document.

Post-scriptum

Additional resources

- Davis David **Random Forest Classifier Tutorial: How to Use Tree-Based Algorithms for Machine Learning**
- Evan Muzzall and Chris Kennedy **Introduction to Machine Learning in R**
- Dave Tang **Building a classification tree in R**
- Zach @ Statology **How to Fit Classification and Regression Trees in R**
- Ben Gorman **Decision Trees in R using rpart**

```
sessionInfo()
```

Session information:

```
## R version 4.1.0 (2021-05-18)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_AU.UTF-8/en_AU.UTF-8/en_AU.UTF-8/C/en_AU.UTF-8/en_AU.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] rpart.plot_3.1.0 rpart_4.1-15     mlbench_2.1-3
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.1.0    magrittr_2.0.1    tools_4.1.0       htmltools_0.5.1.1
##  [5] yaml_2.2.1        stringi_1.7.3     rmarkdown_2.9     highr_0.9
##  [9] knitr_1.33        stringr_1.4.0     xfun_0.24         digest_0.6.27
## [13] rlang_0.4.11      evaluate_0.14
```