Decision trees

Nik Bear Brown

In the this lesson we study the theory and practice of Decision Trees. We show how to use them in R with labeled data.

Additional packages needed

To run the code you may need additional packages.

- If necessary install the followings packages.

```
install.packages("ggplot2");
install.packages("C50");
install.packages("gmodels");
install.packages("rpart");
install.packages("rattle");
install.packages("RColorBrewer");
install.packages("tree");
install.packages("party");
```

```
require("ggplot2");
```

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

```
require("C50");
```

```
## Loading required package: C50
```

```
require("gmodels");
```

```
## Loading required package: gmodels
```

```
require("rpart");
```

```
## Loading required package: rpart
```

```
require("RColorBrewer");
```

```
## Loading required package: RColorBrewer
```

```
require("tree");
```

```
## Loading required package: tree
```

```
require("party");
```

```
## Loading required package: party
```

```
## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

## Loading required package: strucchange

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich
```

Data

We will be using the UCI Machine Learning Repository: Mushroom Data Set. This data is mushrooms described in terms of physical characteristics; along with the classification (label): poisonous or edible drawn from The Audubon Society Field Guide to North American Mushrooms (1981). G. H. Lincoff (Pres.), New York: Alfred A. Knopf

Feel free to tweet questions to [@NikBearBrown](https://twitter.com/NikBearBrown)

```
# Load our data
data_url <-
'http://nikbearbrown.com/YouTube/MachineLearning/M07/mushrooms.csv'
mushrooms <- read.csv(url(data_url))
head(mushrooms)

##        type cap_shape cap_surface cap_color bruises    odor
## 1 poisonous    convex      smooth     brown     yes pungent
## 2    edible    convex      smooth    yellow     yes  almond
## 3    edible      bell      smooth     white     yes   anise
## 4 poisonous    convex       scaly     white     yes pungent
## 5    edible    convex      smooth      gray      no    none
## 6    edible    convex       scaly    yellow     yes  almond
##   gill_attachment gill_spacing gill_size gill_color stalk_shape
stalk_root
## 1            free        close    narrow      black   enlarging
equal
## 2            free        close     broad      black   enlarging
club
```

```
## 3            free        close     broad      brown   enlarging
club
## 4            free        close    narrow      brown   enlarging
equal
## 5            free      crowded     broad      black    tapering
equal
## 6            free        close     broad      brown   enlarging
club
##   stalk_surface_above_ring stalk_surface_below_ring
stalk_color_above_ring
## 1                   smooth                   smooth
white
## 2                   smooth                   smooth
white
## 3                   smooth                   smooth
white
## 4                   smooth                   smooth
white
## 5                   smooth                   smooth
white
## 6                   smooth                   smooth
white
##   stalk_color_below_ring veil_type veil_color ring_number  ring_type
## 1                  white   partial      white         one    pendant
## 2                  white   partial      white         one    pendant
## 3                  white   partial      white         one    pendant
## 4                  white   partial      white         one    pendant
## 5                  white   partial      white         one evanescent
## 6                  white   partial      white         one    pendant
##   spore_print_color population habitat
## 1             black  scattered   urban
## 2             brown   numerous grasses
## 3             brown   numerous meadows
## 4             black  scattered   urban
## 5             brown   abundant grasses
## 6             black   numerous grasses
```

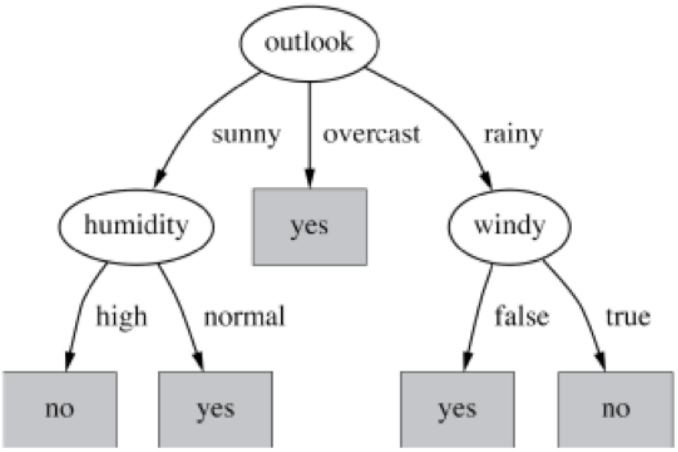
Decision trees

A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their outcomes. The decision tree can be linearized into decision rules, where the outcome is the contents of the leaf node, and the conditions along the path form a conjunction in the if clause. In general, the rules have the form:

$$if \quad condition1 \quad and \quad condition2 \quad and \quad condition3 \quad then \quad outcome$$

Each node in the tree is a decisions/tests. Each path from the tree root to a leaf corresponds to a conjunction of attribute decisions/tests. The tree itself corresponds to a disjunction of these conjunctions.

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No



If outlook = sunny, temp = hot, humidity = high, wind= strong then play is which class? (No, Yes)

Decision Tree

Growing a Decision Tree

Top-down: Which attribute shoud ne the root?

We construct a tree from the top down starting with the question: which attribute should be tested at the root of the tree? That is, which attribute best splits/sperates the labled training data.

Then build subtrees recursively, asking the same question on the remaining attributes.

Information gain

Heuristic: choose the attribute that produces the “purest” nodes. That is, the most homogeneous splits. A popular impurity criterion is information gain. Information

gain increases with the average purity of the subsets. The idea is to choose the attribute that gives greatest information gain as the root of the tree.

Entropy

The notion of using entropy as a measure of change in system state and dynamics comes both from statistical physics and from information theory. In statistical physics, entropy is a measure of disorder and uncertainty in a random variable; the higher the entropy, the greater the disorder. [(Gray,1990), (Behara et al., 1973), (Yeung,2002)] In the statistical physics context, the term usually refers to Gibbs entropy, which measures the macroscopic state of the system as defined by a distribution of atoms and molecules in a thermodynamic system. Gibbs entropy is a measure of the disorder in the arrangements of its particles. As the position of a particle becomes less predictable, the entropy increases. For a classical system (i.e., a collection of classical particles) with a discrete set of microstates, if E_i is the energy of microstate i, and p_i is the probability that it occurs during the system's fluctuations, then the entropy of the system is

$$S = -k_{\mathrm{B}} \sum_i p_i \ln p_i$$

The quantity k_{B} is a physical constant known as Boltzmann's constant, which, like the entropy, has units of heat capacity. The logarithm is dimensionless.

In information theory, entropy is also a measure of the uncertainty in a random variable. [(Cover & Thomas, 1991),(Emmert-Streib & Dehmer, 2009)] In this context, however, the term usually refers to the Shannon entropy, which quantifies the expected value of the information contained in a message (or the expected value of the information of the probability distribution). The concept was introduced by Claude E. Shannon in his 1948 paper "A Mathematical Theory of Communication." [(Shannon, 1948)] Shannon entropy establishes the limits to possible data compression and channel capacity. That is, the entropy gives a lower bound for the efficiency of an encoding scheme (in other words, a lower bound on the possible compression of a data stream). Typically this is expressed in the number of 'bits' or 'nats' that are required to encode a given message. Given the probability of each of n events, the information required to predict an event is the distribution's entropy. Low entropy means the system is very ordered, that is, very predictable. High entropy means the system is mixed, that is, very un predictable; a lot of information is needed for prediction.

The Shannon entropy can explicitly be written as

$$E(X) = \sum_i \mathrm{P}(x_i)\,\mathrm{I}(x_i) = -\sum_i \mathrm{P}(x_i)\log_b \mathrm{P}(x_i),$$

where b is the base of the logarithm used. Common values of b are 2, Euler's number e, and 10, and the unit of entropy is shannon for b = 2, nat for b = e, and hartley for b = 10.When b = 2, the units of entropy are also commonly referred to as bits.

The Shannon entropy is by far the most common information-theoretic measure there are others. Other information-theoretic measures include: plog,Rényi entropy, Hartley entropy, collision entropy, min-entropy, Kullback-Leibler divergence and the information dimension.

Plog

Plog (which we pronounce 'plog, ' for positive log) (Equation 3) (Gray, 1990) is simply the negative log of the frequency. As the value of plog increases, the frequency decreases.

$$E(X) = -\sum \ln p_i$$

freq	(base 2)
0.5	1
0.25	2
1/16	5

Big plog means low frequency.

Rényi entropies

The Rényi entropies generalize the Shannon entropy, the Hartley entropy, the min-entropy, and the collision entropy. As such, these entropies as an ensemble are often called the Rényi entropies (or the Rényi entropy, even though this usually refers to a class of entropies). The difference between these entropies is in the respective value for each of an order parameter called alpha: the values of alpha are greater than or equal to zero but cannot equal one. The Renyi entropy ordering is related to the underlying probability distributions and allows more probable events to be weighted more heavily. As alpha approaches zero, the Rényi entropy increasingly weighs all possible events more equally, regardless of their probabilities. A higher alpha (a) weighs more probable events more heavily. The base used to calculate entropies is usually base 2 or Euler's number base e. If the base of the logarithm is 2, then the uncertainty is measured in bits. If it is the natural logarithm, then the unit is nats.

Rényi entropies

The Rényi entropy of order α, where $\alpha \geq 0$ and $\alpha \neq 1$, is defined as

$$H_\alpha(X) = \frac{1}{1-\alpha} \log\left(\sum_{i=1}^{n} p_i^\alpha\right)$$

Here, X is a discrete random variable with possible outcomes 1,2,...,n and corresponding probabilities $p_i \doteq \Pr(X = i) for i = 1, \ldots, n$, and the logarithm is base 2.

Hartley entropy

The Hartley entropy (Gray, 1990) is the Rényi entropy with an alpha of zero.

the probabilities are nonzero, H_0 is the logarithm of the cardinality of X, sometimes called the Hartley entropy of X:

$$H_0(X) = \log n = \log|X|$$

Shannon entropy

The Shannon entropy (Gray, 1990) is the Rényi entropy with an alpha of one. The Shannon entropy is a simple estimate of the expected value of the information contained in a message. It assumes independence and identically distributed random variables, which is a simplification when applied to word counts. In this sense it is analogous to naïve Bayes, in that it is very commonly used and thought to work well in spite of violating some assumptions upon which it is based.

The limiting value of $H_\alpha as \alpha \to 1$ is the Shannon entropy:

$$H_1(X) = -\sum_{i=1}^{n} p_i \log p_i.$$

collision entropy

The collision entropy (Gray, 1990) is the Rényi entropy with an alpha of two and is sometimes just called "Rényi entropy," refers to the case $\alpha = 2$,

$$H_2(X) = -\log \sum_{i=1}^{n} p_i^2 = -\log P(X = Y)$$

where X and Y are independent and identically distributed.

min-entropy

The min-entropy (Gray, 1990) is the Rényi entropy as the limit of alpha approaches infinity. The name min-entropy stems from the fact that it is the smallest entropy measure in the Rényi family of entropies. In the limit as $\alpha \to \infty$, the Rényi entropy $H_\alpha converges to the min-entropy H_\infty$:

$$H_\infty(X) \doteq \min_i(-\log p_i) = -(\max_i \log p_i) = -\log \max_i p_i \,.$$

Equivalently, the min-entropy $H_\infty(X)$ is the largest real number b such that all events occur with probability at most 2^{-b}.

Kullback-Leibler divergence

Kullback-Leibler divergence (Gray, 1990) is a non-symmetric measure of the difference between two probability distributions. The Kullback-Leibler measure

goes by several names: relative entropy, discrimination information, Kullback-Leibler (KL) number, directed divergence, informational divergence, and cross entropy. Kullback-Leibler divergence is a measure of the difference between the observed entropy and its excepted entropy. We calculate the KL divergence by weighting one distribution (like an observed frequency distribution) by the log of probabilities of some other distribution D2. For discrete probability distributions P and Q, the Kullback–Leibler divergence of Q from P is defined to be

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{i} P(i) \ln \frac{P(i)}{Q(i)}$$

In words, it is the expectation of the logarithmic difference between the probabilities P and Q, where the expectation is taken using the probabilities P.

Mutual Information

Mutual information (Gray, 1990) quantifies the mutual dependence of the two random variables. It is a measure of the "stickiness" between two items. It measures how much knowing one of these variables reduces uncertainty about the other. We can use mutual information to quantify the association between two tags. Mutual information (Equation 10) is given by:

the mutual information of two discrete random variables X and Y can be defined as:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p(x)\, p(y)} \right),$$

where $p(x, y)$ is the joint probability distribution function of X and Y, and $p(x)$ and $p(y)$ are the marginal probability distribution functions of X and Y respectively. In the case of continuous random variables, the summation is replaced by a definite double integral:

$$I(X;Y) = \int_{Y} \int_{X} p(x,y) \log \left(\frac{p(x,y)}{p(x)\, p(y)} \right) \, dx\, dy,$$

where $p(x, y)$ is now the joint probability density function of X and Y, and $p(x)$ and $p(y)$ are the marginal probability density functions of X and Y respectively.

Computing Information Gain

To calculate information gain, we can calculate the information difference, $-p_1 \log p_1 - p_2 \log p_2$. Generalizing this to n events, we get:

$$entropy(p_1, p_2, \ldots p_n) = -p_1 \log p_1 - p_2 \log p_2 \ldots - p_n \log p_n$$

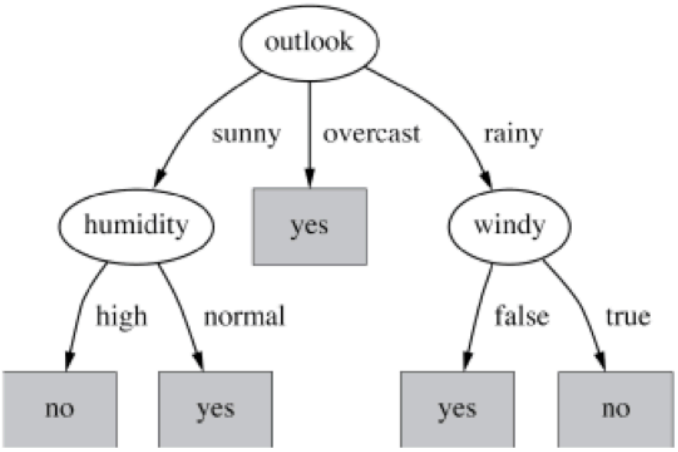
which is just the Shannon entropy

$$H_1(X) = -\sum_{i=1}^{n} p_i \log p_i.$$

For example, if entropy = $-1.0\log(1.0) - 0.0\log(0.0) = 0$ then this provides no information. If entropy = $-0.5\log(0.5) - 0.5\log(0.5) = 1.0$ then this provides one "bit" of information.

For example, let's calculate the information gain for the Outlook variable in the table below.

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No



If outlook = sunny, temp = hot, humidity = high, wind= strong then play is which class? (No, Yes)

decision tree play or n

For Outlook = Sunny are there are 5 Sunny rows of which 2 are "yes" and 3 are "no" and then InfoGain[2,3]= $-2/5\log(2/5) - 3/5\log(3/5) = 0.97 \quad bits$.

For Outlook = Overcast are there are 4 Overcast rows of which 0 are "yes" and 4 are "no" and then InfoGain[0,4]= $-0\log(0) - 1\log(1) = 0.0 \quad bits$.

For Outlook = Rainy are there are 5 Rainy rows of which 3 are "yes" and 2 are "no" and then InfoGain[3,2]= $-3/5\log(3/5) - 2/5\log(2/5) = 0.97 \quad bits$.

So the expected information gain for the for attribute Outlook would be,

$$InfoGain([2,3],[0,4][3,2]) = \frac{5}{14}\times 0.97 + \frac{4}{14}\times 0.0 + \frac{5}{14}\times 0.97 = 0.69 \quad bits$$

ID3 algorithm

This idea of iteratively finding the attribute with the most information gain to find a root in decision tree learning is called the ID3 (Iterative Dichotomiser 3) algorithm. The invented by Ross Quinlan. It is a simple algorithm once one understands the concept of entropy and information gain.

1. Calculate the entropy of every attribute using the data set S, using the Shannon entropy.
2. Split the set S into subsets using the attribute for which entropy is minimum (or, equivalently, information gain is maximum)
3. Make the decision tree (or sub-tree) root node that attribute.
4. Recur on subsets using remaining attributes.

C4.5 algorithm

C4.5 is an extension of Quinlan's earlier ID3 algorithm. The splitting criterion is based on statistical confidence estimates. This technique has the advantage that it allows all of the available labeled data to be used for training. To generate this confidence one calculates the error rate over n labled training instances. The observed error rate e is analaogous to the observed fraction of heads in n tosses of a biased coin (i.e. the probability of heads may not be 0.5). One wishes to estimate the true error rate, p from the observed error rate e.

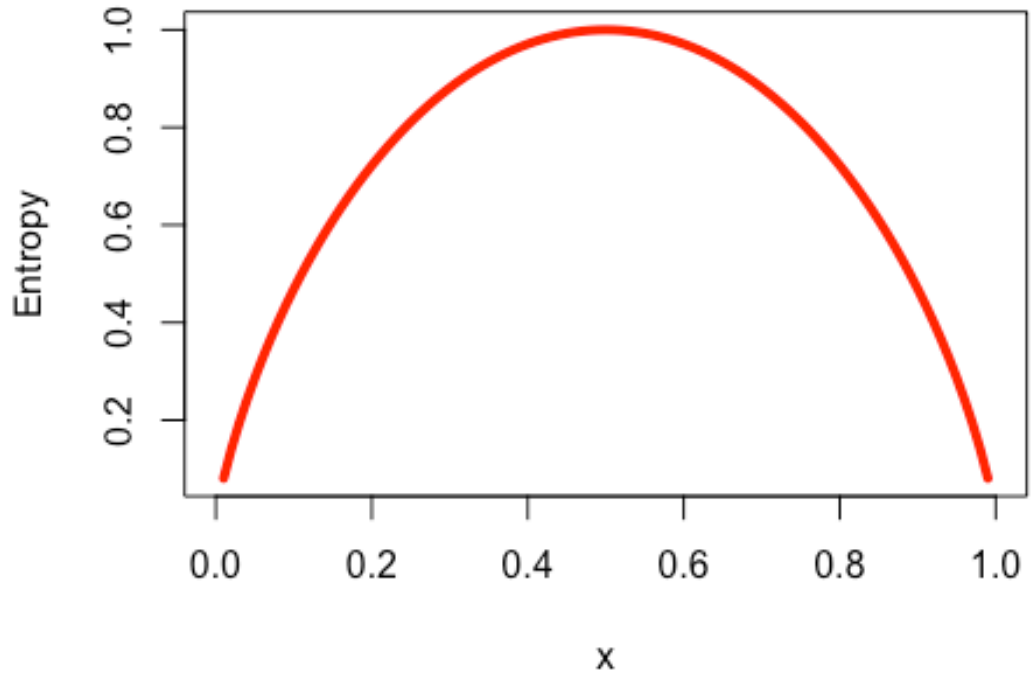
The confidence interval, is calculated as follows, if one chooses a level of confidence z then

$$p = e + z\times\sqrt{e\times\frac{1-e}{n}}$$

Paired values for z and confidence levels (z,confidence) are in the following lists: (0.67 z, 50% confidence), (1.0 z, 68% confidence) , (1.64 z, 90% confidence) and (1.96 z, 95% confidence).

Decision Trees in R

```
#####  Decision Trees  ------------------

#### Step 1: Decision Trees -------------------

## Understanding Decision Trees ----
# calculate entropy of a two-class segment


curve(-x * log2(x) - (1 - x) * log2(1 - x),
      col="red", xlab = "x", ylab = "Entropy", lwd=4)
```




```
## Example: Identifying Mushroom Type: Either 'poisonous' or 'edible' -
---

##Step 2: Exploring and preparing the data ----


str(mushrooms)

## 'data.frame':    8124 obs. of  23 variables:
##  $ type                    : Factor w/ 2 levels
```

```
"edible","poisonous": 2 1 1 2 1 1 1 1 2 1 ...
##  $ cap_shape               : Factor w/ 6 levels "bell","conical",..:
3 3 1 3 3 3 1 1 3 1 ...
##  $ cap_surface             : Factor w/ 4 levels
"fibrous","grooves",..: 4 4 4 3 4 3 4 3 3 4 ...
##  $ cap_color               : Factor w/ 10 levels "brown","buff",..:
1 10 9 9 4 10 9 9 9 10 ...
##  $ bruises                 : Factor w/ 2 levels "no","yes": 2 2 2 2
1 2 2 2 2 2 ...
##  $ odor                    : Factor w/ 9 levels "almond","anise",..:
8 1 2 8 7 1 1 2 8 1 ...
##  $ gill_attachment         : Factor w/ 2 levels "attached","free": 2
2 2 2 2 2 2 2 2 2 ...
##  $ gill_spacing            : Factor w/ 2 levels "close","crowded": 1
1 1 1 2 1 1 1 1 1 ...
##  $ gill_size               : Factor w/ 2 levels "broad","narrow": 2
1 1 2 1 1 1 1 2 1 ...
##  $ gill_color              : Factor w/ 12 levels "black","brown",..:
1 1 2 2 1 2 5 2 8 5 ...
##  $ stalk_shape             : Factor w/ 2 levels
"enlarging","tapering": 1 1 1 1 2 1 1 1 1 1 ...
##  $ stalk_root              : Factor w/ 5 levels "bulbous","club",..:
3 2 2 3 3 2 2 2 3 2 ...
##  $ stalk_surface_above_ring: Factor w/ 4 levels
"fibrous","scaly",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ stalk_surface_below_ring: Factor w/ 4 levels
"fibrous","scaly",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ stalk_color_above_ring  : Factor w/ 9 levels "brown","buff",..: 8
8 8 8 8 8 8 8 8 8 ...
##  $ stalk_color_below_ring  : Factor w/ 9 levels "brown","buff",..: 8
8 8 8 8 8 8 8 8 8 ...
##  $ veil_type               : Factor w/ 1 level "partial": 1 1 1 1 1
1 1 1 1 1 ...
##  $ veil_color              : Factor w/ 4 levels "brown","orange",..:
3 3 3 3 3 3 3 3 3 3 ...
##  $ ring_number             : Factor w/ 3 levels "none","one","two":
2 2 2 2 2 2 2 2 2 2 ...
##  $ ring_type               : Factor w/ 5 levels
"evanescent","flaring",..: 5 5 5 5 1 5 5 5 5 5 ...
##  $ spore_print_color       : Factor w/ 9 levels "black","brown",..:
1 2 2 1 2 1 1 2 1 1 ...
##  $ population              : Factor w/ 6 levels
"abundant","clustered",..: 4 3 3 4 1 3 3 4 5 4 ...
##  $ habitat                 : Factor w/ 7 levels
"grasses","leaves",..: 5 1 3 5 1 1 3 3 1 3 ...

# look at the class variable
table(mushrooms$type)
```

```
##
##    edible poisonous
##      4208      3916
```

```
# create a random sample for training and test data
```

```
set.seed(12345)
mush_rand <- mushrooms[order(runif(8124)), ]
```

```
# compare the Mushrooms(In original order) and mush_rand( random order)
data frames
```

```
summary(mushrooms$habitat)
```

```
## grasses  leaves meadows   paths   urban   waste   woods
##    2148     832     292    1144     368     192    3148
```

```
summary(mush_rand$habitat)
```

```
## grasses  leaves meadows   paths   urban   waste   woods
##    2148     832     292    1144     368     192    3148
```

```
head(mushrooms$habitat)
```

```
## [1] urban   grasses meadows urban   grasses grasses
## Levels: grasses leaves meadows paths urban waste woods
```

```
head(mush_rand$habitat)
```

```
## [1] paths   woods   grasses paths   paths   woods
## Levels: grasses leaves meadows paths urban waste woods
```

```
# split the data frames
```

```
mushrooms_train <- mush_rand[1:8000,-17 ]
mushrooms_test  <- mush_rand[8000:8124, ]
```

```
# check the proportion of class variable
```

```
prop.table(table(mushrooms_train$type))
```

```
##
##    edible poisonous
##  0.517625  0.482375
```

```
prop.table(table(mushrooms_test$type))
```

```
##
##    edible poisonous
##     0.544     0.456
```

```
## Step 3: Training a model on the data ----


model <- C5.0(mushrooms_train[-1], mushrooms_train$type)

# display simple facts about the tree
model

##
## Call:
## C5.0.default(x = mushrooms_train[-1], y = mushrooms_train$type)
##
## Classification Tree
## Number of samples: 8000
## Number of predictors: 21
##
## Tree size: 7
##
## Non-standard options: attempt to group attributes

# display detailed information about the tree
summary(model)

##
## Call:
## C5.0.default(x = mushrooms_train[-1], y = mushrooms_train$type)
##
##
## C5.0 [Release 2.07 GPL Edition]      Thu May  4 14:16:58 2017
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 8000 cases (22 attributes) from undefined.data
##
## Decision tree:
##
## odor in {creosote,fishy,foul,musty,pungent,spicy}: poisonous (3741)
## odor in {almond,anise,none}:
## :...spore_print_color = green: poisonous (71)
##     spore_print_color in
{black,brown,buff,chocolate,orange,purple,white,
##     :                       yellow}:
##     :...cap_surface = grooves: poisonous (4)
##         cap_surface in {fibrous,scaly,smooth}:
##         :...stalk_color_below_ring in
{buff,cinnamon,gray,orange,pink,red,
##             :                           white}: edible (4083/4)
##             stalk_color_below_ring = yellow: poisonous (23)
##             stalk_color_below_ring = brown:
```

```
##             :...stalk_root in {bulbous,club,equal,rooted}: edible
(62)
##                 stalk_root = missing: poisonous (16)
##
##
## Evaluation on training data (8000 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##       7    4( 0.1%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##    4141          (a): class edible
##       4  3855    (b): class poisonous
##
##
##  Attribute usage:
##
##  100.00% odor
##   53.24% spore_print_color
##   52.35% cap_surface
##   52.30% stalk_color_below_ring
##    0.98% stalk_root
##
##
## Time: 0.0 secs
```

```
## Step 4: Evaluating model performance ----
# create a factor vector of predictions(model) on test data

Mushroom_type_pred <- predict(model, mushrooms_test)

# cross tabulation of predicted versus actual classes


CrossTable(mushrooms_test$type, Mushroom_type_pred,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual type', 'predicted type'))
```

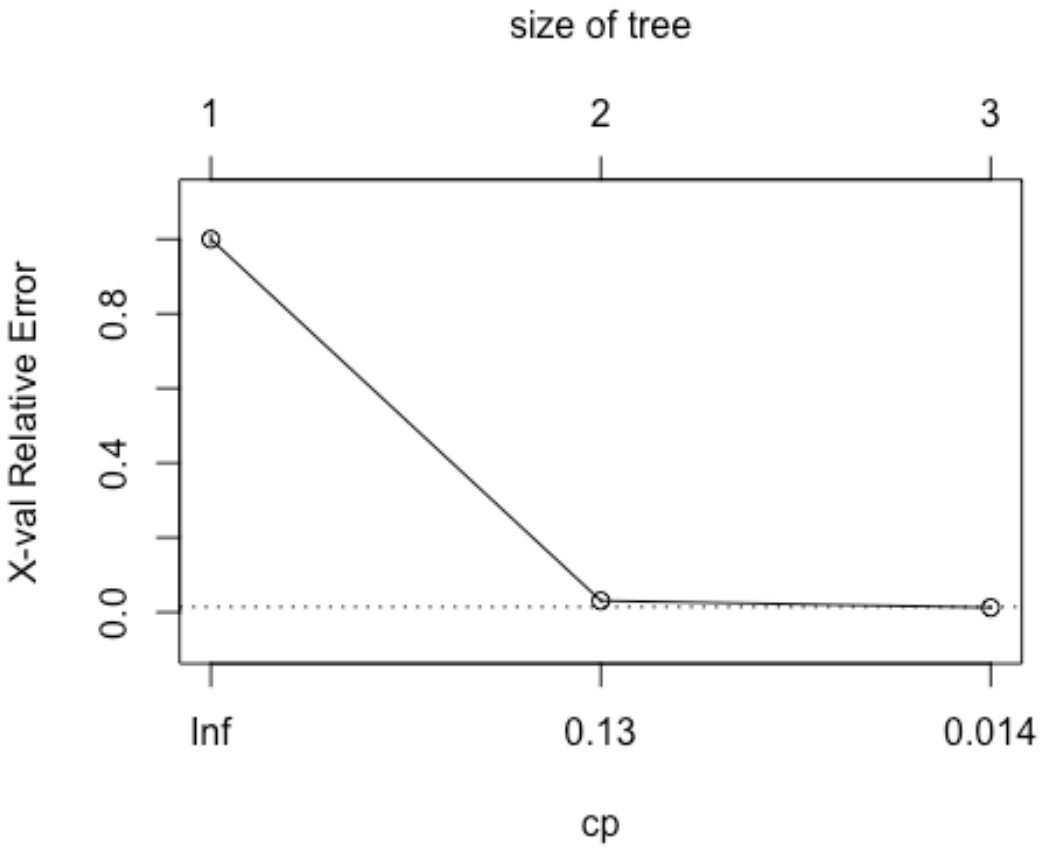
```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
```

```
##
##
## Total Observations in Table:  125
##
##
##              | predicted type
##  actual type |    edible | poisonous | Row Total |
## -------------|-----------|-----------|-----------|
##       edible |        68 |         0 |        68 |
##              |     0.544 |     0.000 |           |
## -------------|-----------|-----------|-----------|
##    poisonous |         0 |        57 |        57 |
##              |     0.000 |     0.456 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        68 |        57 |       125 |
## -------------|-----------|-----------|-----------|
##
##
```

```
formula<-type  ~ cap_shape + cap_surface +  cap_color+  bruises +
          odor + gill_attachment    + gill_spacing  + gill_size +
gill_color+   stalk_shape+    stalk_root +
          stalk_surface_above_ring +stalk_surface_below_ring +
stalk_color_above_ring   +
          stalk_color_below_ring +  veil_color+ring_number+ring_type+
spore_print_color+population+
           habitat

fit = rpart(formula, method="class", data=mushrooms_train)

printcp(fit) # display the results
```

```
##
## Classification tree:
## rpart(formula = formula, data = mushrooms_train, method = "class")
##
## Variables actually used in tree construction:
## [1] odor              spore_print_color
##
## Root node error: 3859/8000 = 0.48237
##
## n= 8000
##
##         CP nsplit rel error   xerror      xstd
## 1 0.969422      0  1.000000 1.000000 0.0115816
## 2 0.018399      1  0.030578 0.030578 0.0027941
## 3 0.010000      2  0.012179 0.012179 0.0017713
```

```
plotcp(fit) # visualize cross-validation results
```




```
summary(fit) # detailed summary of splits

## Call:
## rpart(formula = formula, data = mushrooms_train, method = "class")
##   n= 8000
##
##           CP nsplit  rel error     xerror        xstd
## 1 0.96942213      0 1.00000000 1.00000000 0.011581645
## 2 0.01839855      1 0.03057787 0.03057787 0.002794084
## 3 0.01000000      2 0.01217932 0.01217932 0.001771310
##
## Variable importance
##                      odor        spore_print_color
gill_color
##                        25                       19
15
## stalk_surface_above_ring stalk_surface_below_ring
ring_type
##                        14                       13
13
##
## Node number 1: 8000 observations,    complexity param=0.9694221
##   predicted class=edible     expected loss=0.482375  P(node) =1
```

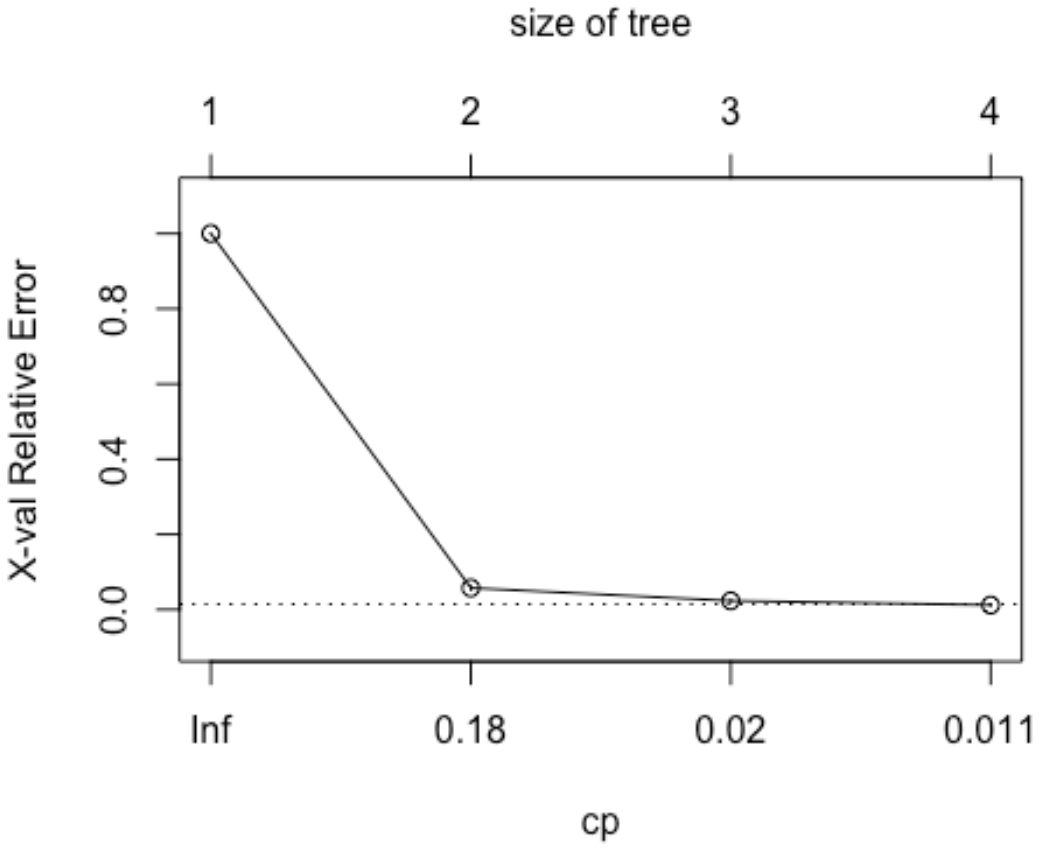
```
##     class counts:  4141  3859
##    probabilities: 0.518 0.482
##   left son=2 (4259 obs) right son=3 (3741 obs)
##   Primary splits:
##       odor                     splits as  LLRRRRLRR,
improve=3765.568, (0 missing)
##       spore_print_color        splits as  LLLRRLLRL,
improve=2164.168, (0 missing)
##       gill_color               splits as  LLRRRRLLLLLL,
improve=1509.883, (0 missing)
##       stalk_surface_above_ring splits as  LLRL,
improve=1378.779, (0 missing)
##       stalk_surface_below_ring splits as  LLRL,
improve=1313.004, (0 missing)
##   Surrogate splits:
##       spore_print_color        splits as  LLLRLLLRL,    agree=0.862,
adj=0.704, (0 split)
##       gill_color               splits as  LLRRRLLLLLL, agree=0.810,
adj=0.594, (0 split)
##       stalk_surface_above_ring splits as  LLRL,         agree=0.781,
adj=0.532, (0 split)
##       stalk_surface_below_ring splits as  LLRL,         agree=0.781,
adj=0.531, (0 split)
##       ring_type                splits as  RLRRL,        agree=0.780,
adj=0.530, (0 split)
##
## Node number 2: 4259 observations,    complexity param=0.01839855
##   predicted class=edible     expected loss=0.02770603  P(node)
=0.532375
##     class counts:  4141   118
##    probabilities: 0.972 0.028
##   left son=4 (4188 obs) right son=5 (71 obs)
##   Primary splits:
##       spore_print_color      splits as  LLLLRLLLL,
improve=136.51630, (0 missing)
##       gill_color             splits as  LL-LLRLLLLL, improve=
43.72247, (0 missing)
##       stalk_color_below_ring splits as  L--LLLLLR,    improve=
43.72247, (0 missing)
##       cap_color              splits as  LRLLLRLLLL,   improve=
26.36352, (0 missing)
##       stalk_color_above_ring splits as  L--LLLLLR,    improve=
15.15415, (0 missing)
##   Surrogate splits:
##       gill_color splits as  LL-LLRLLLLL, agree=0.989, adj=0.324, (0
split)
##
## Node number 3: 3741 observations
##   predicted class=poisonous  expected loss=0  P(node) =0.467625
##     class counts:     0  3741
```

```
##    probabilities: 0.000 1.000
##
## Node number 4: 4188 observations
##   predicted class=edible     expected loss=0.01122254  P(node)
=0.5235
##     class counts:  4141    47
##    probabilities: 0.989 0.011
##
## Node number 5: 71 observations
##   predicted class=poisonous  expected loss=0  P(node) =0.008875
##     class counts:     0    71
##    probabilities: 0.000 1.000
```

```
###- Regression Tree Example

# grow tree
fit <- rpart(formula, method="anova", data=mushrooms_train)

printcp(fit) # display the results
```

```
##
## Regression tree:
## rpart(formula = formula, data = mushrooms_train, method = "anova")
##
## Variables actually used in tree construction:
## [1] odor                   spore_print_color
stalk_color_below_ring
##
## Root node error: 1997.5/8000 = 0.24969
##
## n= 8000
##
##         CP nsplit rel error   xerror       xstd
## 1 0.942563      0  1.000000 1.000309 0.00079406
## 2 0.034172      1  0.057437 0.057461 0.00510281
## 3 0.011319      2  0.023265 0.023271 0.00334617
## 4 0.010000      3  0.011946 0.011950 0.00242156
```

```
plotcp(fit) # visualize cross-validation results
```



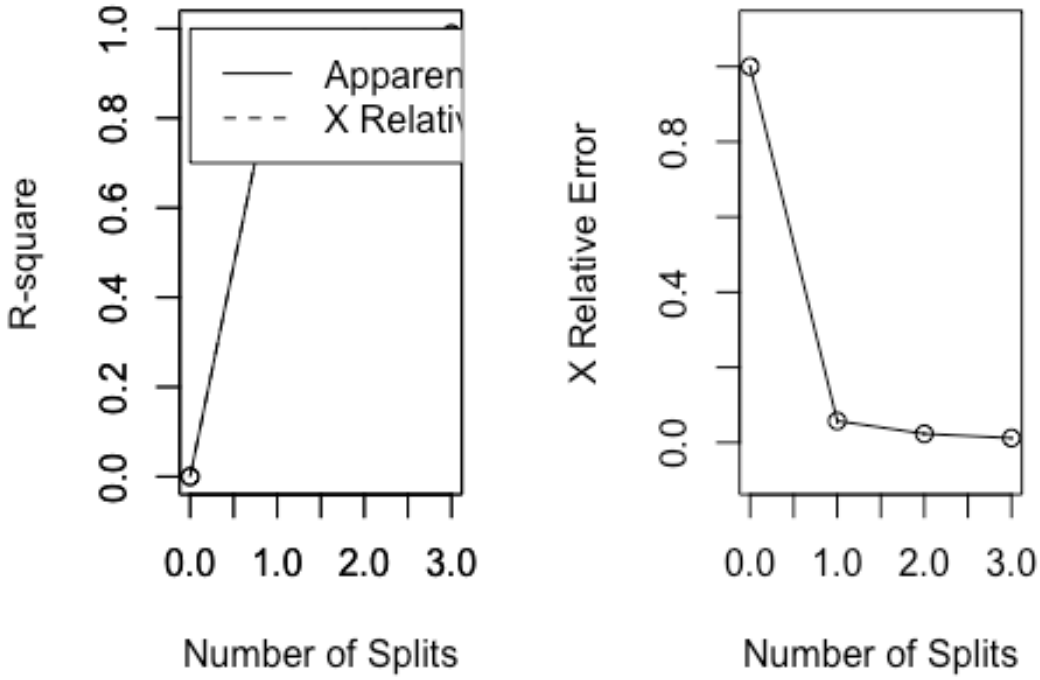

```
summary(fit) # detailed summary

## Call:
## rpart(formula = formula, data = mushrooms_train, method = "anova")
##   n= 8000
##
##           CP nsplit  rel error     xerror         xstd
## 1 0.94256329      0 1.00000000 1.00030868 0.0007940618
## 2 0.03417153      1 0.05743671 0.05746054 0.0051028134
## 3 0.01131948      2 0.02326518 0.02327095 0.0033461670
## 4 0.01000000      3 0.01194570 0.01195029 0.0024215560
##
## Variable importance
##                      odor        spore_print_color
gill_color
##                        25                       19
15
## stalk_surface_above_ring stalk_surface_below_ring
ring_type
##                        13                       13
13
##
## Node number 1: 8000 observations,    complexity param=0.9425633
```

```
##   mean=1.482375, MSE=0.2496894
##   left son=2 (4259 obs) right son=3 (3741 obs)
##   Primary splits:
##       odor                     splits as  LLRRRRLRR,
improve=0.9425633, (0 missing)
##       spore_print_color        splits as  LLLRRLLRL,
improve=0.5417151, (0 missing)
##       gill_color               splits as  LLRRRRLLLLLL,
improve=0.3779405, (0 missing)
##       stalk_surface_above_ring splits as  LLRL,
improve=0.3451235, (0 missing)
##       stalk_surface_below_ring splits as  LLRL,
improve=0.3286594, (0 missing)
##   Surrogate splits:
##       spore_print_color        splits as  LLLRLLLRL,    agree=0.862,
adj=0.704, (0 split)
##       gill_color               splits as  LLRRRLLLLLL, agree=0.810,
adj=0.594, (0 split)
##       stalk_surface_above_ring splits as  LLRL,         agree=0.781,
adj=0.532, (0 split)
##       stalk_surface_below_ring splits as  LLRL,         agree=0.781,
adj=0.531, (0 split)
##       ring_type                splits as  RLRRL,        agree=0.780,
adj=0.530, (0 split)
##
## Node number 2: 4259 observations,    complexity param=0.03417153
##   mean=1.027706, MSE=0.02693841
##   left son=4 (4188 obs) right son=5 (71 obs)
##   Primary splits:
##       spore_print_color      splits as  LLLLRLLLL,
improve=0.59494240, (0 missing)
##       stalk_color_below_ring splits as  L--LLLLLR,
improve=0.19054390, (0 missing)
##       gill_color             splits as  LL-LLRLLLLL,
improve=0.19054390, (0 missing)
##       cap_color              splits as  LRLLLRLLLL,
improve=0.11489310, (0 missing)
##       veil_color             splits as  LLLR,
improve=0.06604229, (0 missing)
##   Surrogate splits:
##       gill_color splits as  LL-LLRLLLLL, agree=0.989, adj=0.324, (0
split)
##
## Node number 3: 3741 observations
##   mean=2, MSE=0
##
## Node number 4: 4188 observations,    complexity param=0.01131948
##   mean=1.011223, MSE=0.0110966
##   left son=8 (4165 obs) right son=9 (23 obs)
##   Primary splits:
```

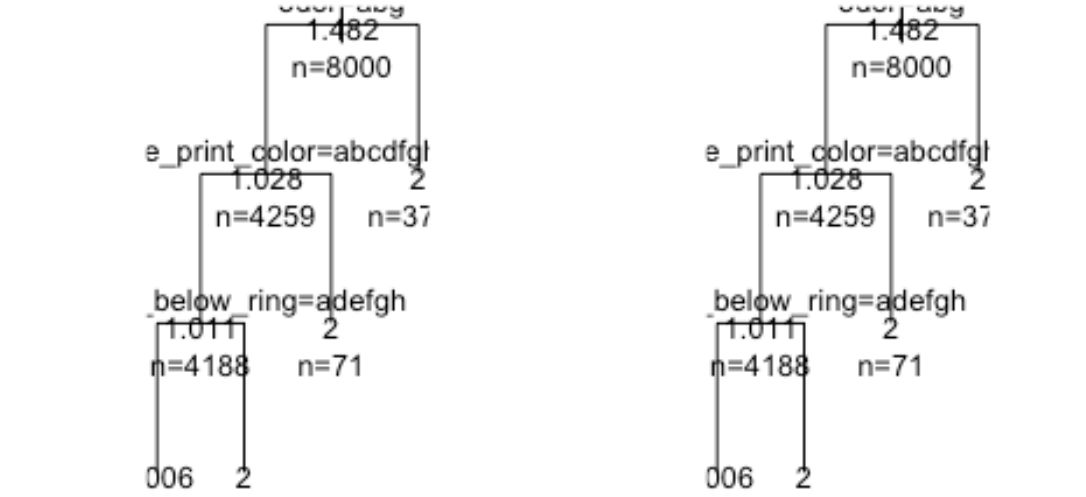
```
##       stalk_color_below_ring   splits as  L--LLLLLR,
improve=0.4865419, (0 missing)
##       stalk_color_above_ring   splits as  L--LLLLLR,
improve=0.1686247, (0 missing)
##       veil_color               splits as  LLLR,
improve=0.1686247, (0 missing)
##       stalk_surface_above_ring splits as  LRRL,
improve=0.1559798, (0 missing)
##       gill_size                splits as  LR,
improve=0.1318231, (0 missing)
##   Surrogate splits:
##       stalk_color_above_ring splits as  L--LLLLLR, agree=0.996,
adj=0.348, (0 split)
##       veil_color             splits as  LLLR,      agree=0.996,
adj=0.348, (0 split)
##
## Node number 5: 71 observations
##   mean=2, MSE=0
##
## Node number 8: 4165 observations
##   mean=1.005762, MSE=0.005729101
##
## Node number 9: 23 observations
##   mean=2, MSE=0
```

```
# create additional plots
par(mfrow=c(1,2)) # two plots on one page
rsq.rpart(fit) # visualize cross-validation results
```

```
##
## Regression tree:
## rpart(formula = formula, data = mushrooms_train, method = "anova")
##
## Variables actually used in tree construction:
## [1] odor                  spore_print_color
stalk_color_below_ring
##
## Root node error: 1997.5/8000 = 0.24969
##
## n= 8000
##
##         CP nsplit rel error   xerror       xstd
## 1 0.942563      0  1.000000 1.000309 0.00079406
## 2 0.034172      1  0.057437 0.057461 0.00510281
## 3 0.011319      2  0.023265 0.023271 0.00334617
## 4 0.010000      3  0.011946 0.011950 0.00242156
```

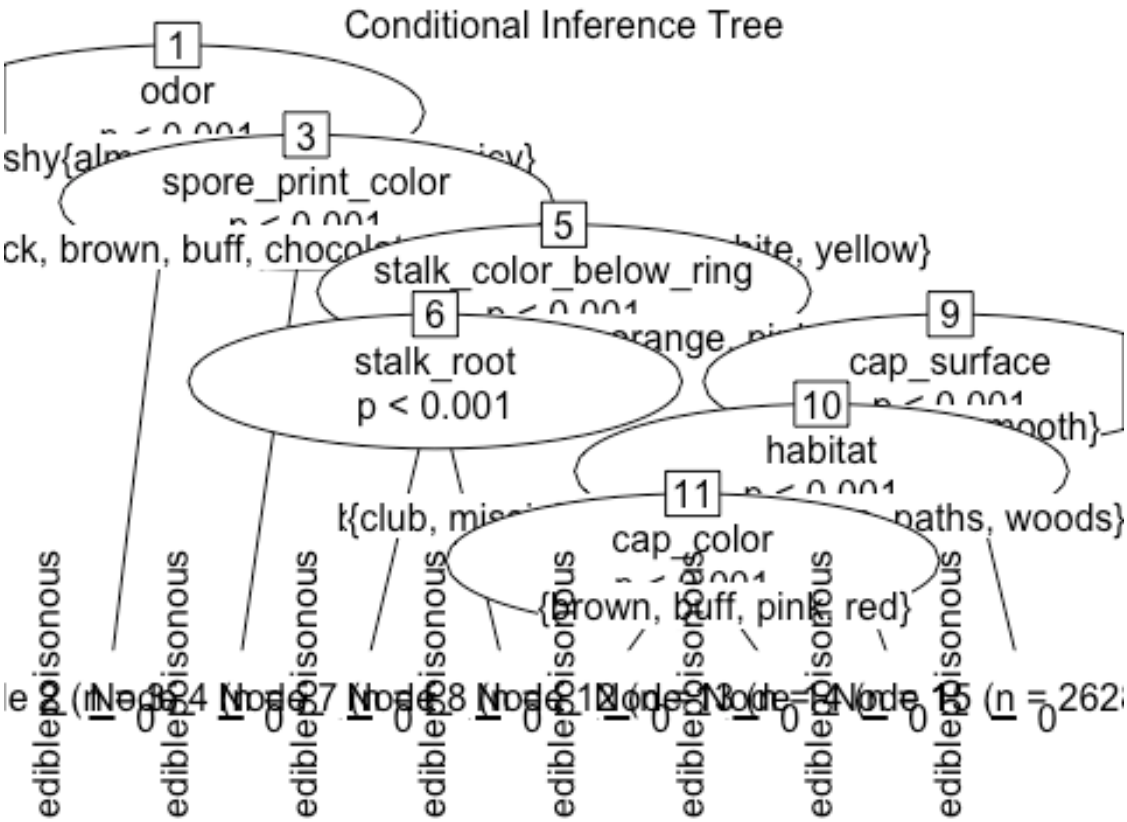



```r
# plot tree
plot(fit, uniform=TRUE,
     main="Regression Tree for 'type' ")
text(fit, use.n=TRUE, all=TRUE, cex=.8)




### ---------------  plot tree

plot(fit, uniform=T, main="Classification Tree for Mushrooms Types")
text(fit, use.n=TRUE, all=TRUE, cex=.8)
```

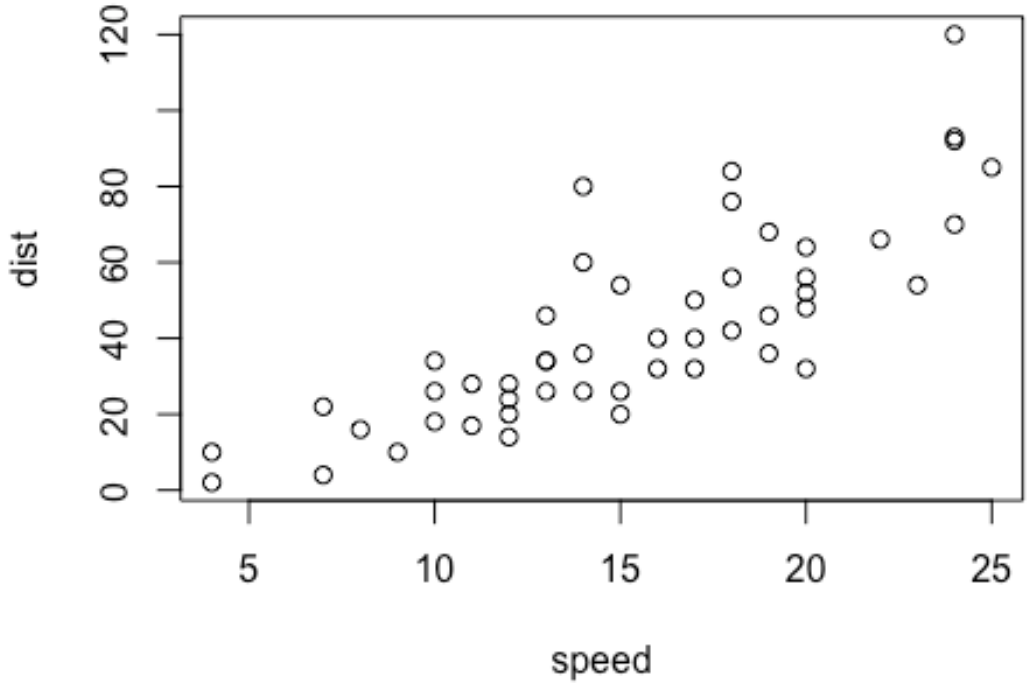



```
##----------------- TREE package


tr = tree(formula, data=mushrooms_train)
summary(tr)

##
## Classification tree:
## tree(formula = formula, data = mushrooms_train)
## Variables actually used in tree construction:
## [1] "odor"                   "spore_print_color"
## [3] "stalk_color_below_ring" "stalk_root"
## Number of terminal nodes:  5
## Residual mean deviance:  0.01448 = 115.8 / 7995
## Misclassification error rate: 0.001 = 8 / 8000

plot(tr); text(tr)



##-------------------Party package
```

```
ct = ctree(formula, data = mushrooms_train)
plot(ct, main="Conditional Inference Tree")


# Estimated class probabilities
tr.pred = predict(ct, newdata=mushrooms_train, type="prob")

#Table of prediction errors
table(predict(ct), mushrooms_train$type)

##
##              edible poisonous
##   edible       4141         0
##   poisonous       0      3859
```



You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

References

- Behara, E. b. M., Krickeberg, K., & Wolfowitz, J. (1973). "Probability and information theory II" Springer-Verlag.
- Yeung, R. W. 2002. "A first course in information theory Kluwer Academic/Plenum Publishers.
- Cover, T. M., & Thomas, J. A. (1991). "Elements of Information Theory"" Wiley.
- Deng,H.; Runger, G.; Tuv, E. (2011). "Bias of importance measures for multi-valued attributes and solutions."" Proceedings of the 21st International Conference on Artificial Neural Networks (ICANN).
- Emmert-Streib, F., & Dehmer, M. (2009). "Information Theory and Statistical Learning."" Springer-Verlag.
- Gray, R. M. (1990). Entropy and Information Theory: Springer-Verlag.

- Quinlan, J. R. (1987). "Simplifying decision trees". International Journal of Man-Machine Studies 27 (3): 221. doi:10.1016/S0020-7373(87)80053-6.
- Shannon, C. E. (1948). "A Mathematical Theory of Communication."" Bell System Technical Journal, 27(3), 379-423.
- Theil. (1972). "Statistical Decomposition Analysis."" Studies in Mathematical and Managerial Economics, 14.

Resources

- Decision Trees
- Quick-R: Tree-Based Models
- A Brief Tour of the Trees and Forests