Data Mining: Mushroom Classification

```
# Data Mining Project for IST 707: Mushroom Classification
# Team Members: Lauren Foltz & Baskar Dakshin
# Date: December 12, 2018
# Note: Last run on R Version: 3.6.1 Eggshell Igloo

# Required Data:
  # mushrooms.csv (in project folder)
  # mushrooms how to grow them.txt (in folder called "corpus1")
  # the mushroom cultivator.txt (in folder called "corpus2")

# Table of Contents:
  #   52 Text Mining (Lauren)
  #  289 Read in Data and Explore (Lauren)
  #  341 Update Variable Name (Lauren)
  #  443 Visual of Bar Charts (Lauren)
  #  563 Create Test and Train (Baskar)
  #  605 Decision Tree (Baskar)
  #  693 Random Forest (Baskar)
  #  759 K-Nearest Neighbors (Baskar)
  #  798 Support Vector Machines (Baskar)
  #  851 Clustering (Baskar)
  # 1040 Naïve Bayes (Lauren)
  # 1106 Association Rules Mining (Lauren)

################################################################################
# Optional Prep Work
################################################################################

# Clean up Packages:
# Check which packages are currently loaded
#search()

# Detach previously loaded packages
#lapply(paste('package:',names(sessionInfo()$otherPkgs),sep=""),detach,character.only=TRUE,unload=TRUE)
```

```r
# Set Working Directory:
# Check current working directory
#getwd()

# Set new working directory using "setwd" with the relevant path, then comment out
#setwd()

#==============================================================================
# Text Mining (Lauren Foltz)
#==============================================================================

# Load Text Mining Libraries
#install.packages("tm")
library(tm)  # Text mining, for reading corpus
#install.packages("stringr")
library(stringr)
#install.packages("wordcloud")  # For creating a word cloud
library(wordcloud)
#install.packages("slam")
library(slam)
#install.packages("SnowballC")
library(SnowballC)
#install.packages("stringi")
library(stringi)
#install.packages("Matrix")
library(Matrix)
#install.packages("tidytext")
library(tidytext)  # To convert Document Term Matrix into a Data Frame
#install.packages("textmineR")
library(textmineR)

#########
# Book 1: Mushrooms and How to Grow Them, 1892
#########

# Read in text document; Corpus should appear in the Environment under Data with length of 1
Corpus <- Corpus(DirSource("corpus1"))
# View Corpus; it should come up in console as "Simple Corpus"
paste("View of Corpus:")
```

```
## [1] "View of Corpus:"
```

```
(Corpus)
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 1
```

```
# View summary; make sure title is as expected
paste("Summary of Corpus:")
```

```
## [1] "Summary of Corpus:"
```

```
summary(Corpus)
```

```
##                                 Length Class             Mode
## mushrooms how to grow them.txt 2      PlainTextDocument list
```

```
# Store length as ndocs; it should appear under Values
paste("Number of Documents:")
```

```
## [1] "Number of Documents:"
```

```
(ndocs<-length(Corpus))
```

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

```
# Clean the Corpus and create Document Term Matrix; "dtm" should appear under Data
dtm <- DocumentTermMatrix(Corpus,
                          control = list(
                            stopwords = TRUE,
                            wordLengths=c(3, 15),
                            removePunctuation = T,
                            removeNumbers = T,
                            tolower=T,
                            #stemming = T,
                            remove_separators = T
                          ))

# Perform Checks, either by viewing or with inspect

#paste("View of dtm:")
#(dtm)
```

```r
paste("Inspect dtm:")
```

```
## [1] "Inspect dtm:"
```

```r
inspect(dtm)  # Shows number of documents and terms
```

```
## <<DocumentTermMatrix (documents: 1, terms: 5036)>>
## Non-/sparse entries: 5036/0
## Sparsity           : 0%
## Maximal term length: 15
## Weighting          : term frequency (tf)
## Sample             :
##                                 Terms
## Docs                             bed beds can good manure may mushroom
##   mushrooms how to grow them.txt 254  443 160  153    408 168      334
##                                 Terms
## Docs                             mushrooms one spawn
##   mushrooms how to grow them.txt       564 216   332
```

```r
# We expect sparcity to be 0 because there is only one document

# Convert what we just built into a matrix; "dtm_M" should appear under Data
dtm_M <- as.matrix(dtm)
# Check it
paste("Check dtm_M:")
```

```
## [1] "Check dtm_M:"
```

```r
(dtm_M[1,1:10])
```

```
##   abandoned       abide        able        ably      abound     abounds
##           3           1          11           1           3           2
## aboveground       abram    abruptly     absence
##           3           2           1           1
```

```r
# Look at word frequencies
WordFreq <- colSums(dtm_M)  # Get column sums and store as WordFreq (should appear in Values)
paste("Length:")
```

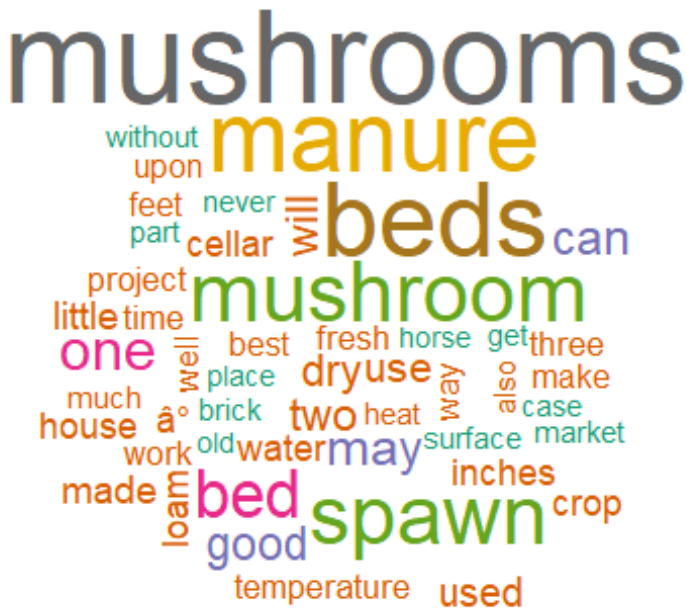
```
## [1] "Length:"
```

```r
(length(WordFreq))  # Get length in console
```

```
## [1] 5036
```

```r
ord <- order(WordFreq)  # Put them in order, and store as ord (should appear under values)
paste("6 Least Frequent:")

## [1] "6 Least Frequent:"

(WordFreq[head(ord)])  # Least frequent will appear in console

##    abide     ably abruptly  absence absorbed  absorbs
##        1        1        1        1        1        1

paste("6 Most Frequent:")

## [1] "6 Most Frequent:"

(WordFreq[tail(ord)])  # Most frequent will appear in console

##       bed     spawn  mushroom    manure      beds mushrooms
##       254       332       334       408       443       564

# Row Sums gives a number for each document, good for normalizing by hand.
# Not needed for this document, but added as a reference.
# (Row_Sum_Per_doc <- rowSums(dtm_M))

# Convert un-normalized Matrix to a DataFrame; "mush_DF" should appear under Data
mush_DF <- as.data.frame(dtm_M)
# View portion of Structure
paste("Portion of mush_DF Structure:")

## [1] "Portion of mush_DF Structure:"

str(mush_DF [1,1:10])

## 'data.frame':    1 obs. of  10 variables:
##  $ abandoned  : num 3
##  $ abide      : num 1
##  $ able       : num 11
##  $ ably       : num 1
##  $ abound     : num 3
##  $ abounds    : num 2
##  $ aboveground: num 3
##  $ abram      : num 2
##  $ abruptly   : num 1
##  $ absence    : num 1
```

```r
# Check how many rows are in the DF; it should be the same as the number of text documents
paste("Number of Rows in mush_DF:")

## [1] "Number of Rows in mush_DF:"

(nrow(mush_DF))

## [1] 1

# Visual: Create a word cloud using a matrix
wordcloud(colnames(dtm_M), dtm_M[1, ], max.words = 50,colors=brewer.pal(8,"Dark2"))
```




```r
# If desired, select a word to look up. Will get the freqency in each document.
paste("Mushrooms and How to Grow Them, 1892 ")

## [1] "Mushrooms and How to Grow Them, 1892 "

cat("Frequency of beds:",(mush_DF$beds),"\n")

## Frequency of beds: 443
```

```r
cat("Frequency of manure:",(mush_DF$manure),"\n")

## Frequency of manure: 408

cat("Frequency of spawn:",(mush_DF$spawn),"\n")

## Frequency of spawn: 332

cat("Frequency of cellar:",(mush_DF$cellar),"\n")

## Frequency of cellar: 103

cat("Frequency of loam:",(mush_DF$loam),"\n")

## Frequency of loam: 97

cat("Frequency of temperature:",(mush_DF$temperature),"\n")

## Frequency of temperature: 91

cat("Frequency of compost:",(mush_DF$compost),"\n")

## Frequency of compost: 1

cat("Frequency of straw:",(mush_DF$straw),"\n")

## Frequency of straw: 36

cat("Frequency of agar:",(mush_DF$agar),"\n")

## Frequency of agar:

cat("Frequency of species:",(mush_DF$species),"\n")

## Frequency of species: 7

#########
# Book 2: The Mushroom Cultivator, 1983
#########

# This is the same code as above, but using "corpus2:

# Read in text document; Corpus should appear in the Environment under Data with length of 1
Corpus <- Corpus(DirSource("corpus2"))
```

```r
# View Corpus; it should come up in console as "Simple Corpus"
paste("View of Corpus:")

## [1] "View of Corpus:"

(Corpus)

## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 1

# View summary; make sure title is as expected
paste("Summary of Corpus:")

## [1] "Summary of Corpus:"

summary(Corpus)

##                                                                     Length
## The-Mushroom-Cultivator-A-Practical-Guide-to-Growing-Mushrooms-at-Home.txt 2
##                                                                     Class
## The-Mushroom-Cultivator-A-Practical-Guide-to-Growing-Mushrooms-at-Home.txt PlainTextDocument
##                                                                     Mode
## The-Mushroom-Cultivator-A-Practical-Guide-to-Growing-Mushrooms-at-Home.txt list

# Store length as ndocs; it should appear under Values
paste("Number of Documents:")

## [1] "Number of Documents:"

(ndocs<-length(Corpus))

## [1] 1

# Clean the Corpus and create Document Term Matrix; "dtm" should appear under Data
dtm <- DocumentTermMatrix(Corpus,
                          control = list(
                            stopwords = TRUE,
                            wordLengths=c(3, 15),
                            removePunctuation = T,
                            removeNumbers = T,
                            tolower=T,
                            #stemming = T,
                            remove_separators = T
```

```r
                          ))

# Perform Checks, either by viewing or with inspect
#paste("View of dtm:")
#(dtm)

paste("Inspect dtm:")

## [1] "Inspect dtm:"

inspect(dtm)  # Shows number of documents and terms

## <<DocumentTermMatrix (documents: 1, terms: 3221)>>
## Non-/sparse entries: 3221/0
## Sparsity           : 0%
## Maximal term length: 15
## Weighting          : term frequency (tf)
## Sample             :
##                                                                      Terms
## Docs                                                                  can
##   The-Mushroom-Cultivator-A-Practical-Guide-to-Growing-Mushrooms-at-Home.txt 120
##                                                                      Terms
## Docs                                                                  casing
##   The-Mushroom-Cultivator-A-Practical-Guide-to-Growing-Mushrooms-at-Home.txt     73
##                                                                      Terms
## Docs                                                                  compost
##   The-Mushroom-Cultivator-A-Practical-Guide-to-Growing-Mushrooms-at-Home.txt      76
##                                                                      Terms
## Docs                                                                  culture
##   The-Mushroom-Cultivator-A-Practical-Guide-to-Growing-Mushrooms-at-Home.txt      74
##                                                                      Terms
## Docs                                                                  figure
##   The-Mushroom-Cultivator-A-Practical-Guide-to-Growing-Mushrooms-at-Home.txt    157
##                                                                      Terms
## Docs                                                                  mushroom
##   The-Mushroom-Cultivator-A-Practical-Guide-to-Growing-Mushrooms-at-Home.txt      198
##                                                                      Terms
## Docs                                                                  mycelium
##   The-Mushroom-Cultivator-A-Practical-Guide-to-Growing-Mushrooms-at-Home.txt       86
##                                                                      Terms
## Docs                                                                  spawn
##   The-Mushroom-Cultivator-A-Practical-Guide-to-Growing-Mushrooms-at-Home.txt    82
```

```
##                                                                                  Terms
## Docs                                                                              species
##   The-Mushroom-Cultivator-A-Practical-Guide-to-Growing-Mushrooms-at-Home.txt      95
##                                                                                  Terms
## Docs                                                                              spores
##   The-Mushroom-Cultivator-A-Practical-Guide-to-Growing-Mushrooms-at-Home.txt     76
```

```r
# We expect sparcity to be 0 because there is only one document

# Convert what we just built into a matrix; "dtm_M" should appear under Data
dtm_M <- as.matrix(dtm)
# Check it
paste("Check dtm_M:")
```

```
## [1] "Check dtm_M:"
```

```r
(dtm_M[1,1:10])
```

```
##     ability abnormality       abort     aborted     absence      absent
##           3           2           1           1           1           3
##  absolutely      absorb   absorbing  absorption
##           5           1           1           1
```

```r
# Look at word frequencies; WordFreq
WordFreq <- colSums(dtm_M)  # Get column sums and store as WordFreq (should appear in Values)
paste("Length:")
```

```
## [1] "Length:"
```

```r
(length(WordFreq))  # Get length in console
```

```
## [1] 3221
```

```r
ord <- order(WordFreq)  # Put them in order, and store as ord (should appear under values)
paste("6 Least Frequent:")
```

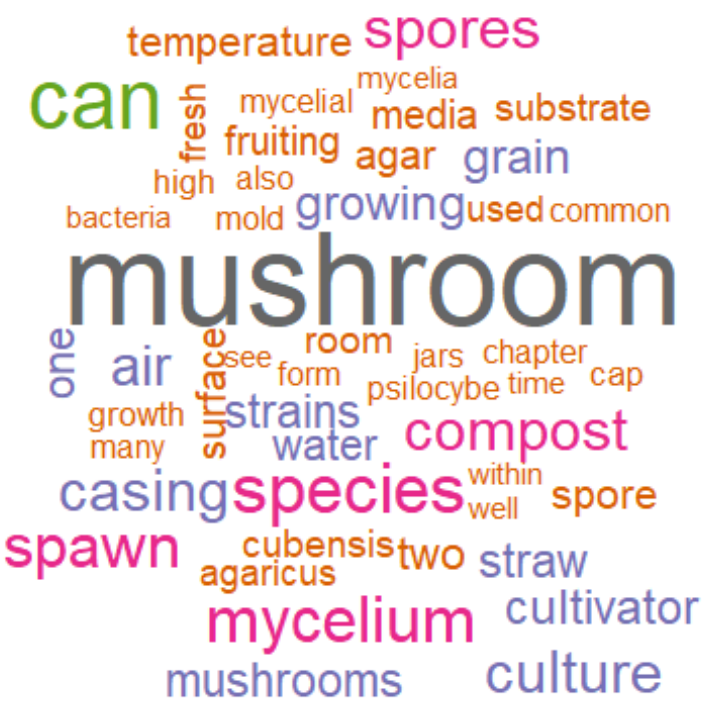
```
## [1] "6 Least Frequent:"
```

```r
(WordFreq[head(ord)])  # Least frequent will appear in console
```

```
##      abort    aborted    absence     absorb  absorbing absorption
##          1          1          1          1          1          1
```

```r
paste("6 Most Frequent:")
```

```
## [1] "6 Most Frequent:"

(WordFreq[tail(ord)])  # Most frequent will appear in console

##    spawn mycelium  species      can   figure mushroom
##       82       86       95      120      157      198

# Row Sums gives a number for each document, good for normalizing by hand.
# Not needed for this document, but added as a reference.
#(Row_Sum_Per_doc <- rowSums(dtm_M))

# Convert un-normalized Matrix to a DataFrame; "mush_DF" should appear under Data
mush_DF <- as.data.frame(dtm_M)
# View portion of Structure
paste("Portion of mush_DF Structure:")

## [1] "Portion of mush_DF Structure:"

str(mush_DF [1,1:10])

## 'data.frame':    1 obs. of  10 variables:
##  $ ability    : num 3
##  $ abnormality: num 2
##  $ abort      : num 1
##  $ aborted    : num 1
##  $ absence    : num 1
##  $ absent     : num 3
##  $ absolutely : num 5
##  $ absorb     : num 1
##  $ absorbing  : num 1
##  $ absorption : num 1

# Check how many rows are in the DF; it should be the same as the number of text documents
paste("Number of Rows in mush_DF:")

## [1] "Number of Rows in mush_DF:"

(nrow(mush_DF))

## [1] 1

# Visual: Create a word cloud using a matrix
wordcloud(colnames(dtm_M), dtm_M[1, ], max.words = 50,colors=brewer.pal(8,"Dark2"))
```




```
# If desired, select a word to look up. Will get the freqency in each document
paste("The Mushroom Cultivator, 1983")

## [1] "The Mushroom Cultivator, 1983"

cat("Frequency of beds:",(mush_DF$beds),"\n")

## Frequency of beds: 8

cat("Frequency of manure:",(mush_DF$manure),"\n")

## Frequency of manure: 5

cat("Frequency of spawn:",(mush_DF$spawn),"\n")

## Frequency of spawn: 82

cat("Frequency of cellar:",(mush_DF$cellar),"\n")

## Frequency of cellar:
```

```r
cat("Frequency of loam:",(mush_DF$loam),"\n")

## Frequency of loam:

cat("Frequency of temperature:",(mush_DF$temperature),"\n")

## Frequency of temperature: 46

cat("Frequency of compost:",(mush_DF$compost),"\n")

## Frequency of compost: 76

cat("Frequency of straw:",(mush_DF$straw),"\n")

## Frequency of straw: 53

cat("Frequency of agar:",(mush_DF$agar),"\n")

## Frequency of agar: 44

cat("Frequency of species:",(mush_DF$species),"\n")

## Frequency of species: 95

##############################################################################
# Prep Work: Load Libraries
##############################################################################
#install.packages("ggplot2")
library(ggplot2)
#install.packages("plyr")
library(plyr)
#install.packages("tidyverse")
library(tidyverse)
#install.packages("e1071")
library(e1071)
#install.packages("mlr")
library(mlr)
#install.packages("caret",
#                  repos = "http://cran.r-project.org",
#                  dependencies = TRUE)
library(caret)
#install.packages("naivebayes")
```

```r
library(naivebayes)
#install.packages("mclust")
library(mclust)
#install.packages("cluster")
library(cluster)
#install.packages("rpart")
library(rpart)
#install.packages('rattle')
library(rattle)
#install.packages('rpart.plot')
library(rpart.plot)
#install.packages("Cairo")
library(Cairo)
#install.packages("corrplot")
library(corrplot)

##############################################################################
# Read in Data and Explore (Lauren Foltz)
##############################################################################

# Place "mushrooms.csv" in the R project folder
# Read the file into R using the read.csv function
# Use header=TRUE to let R know that headers are present
# Use "na.strings = "NA" so R will replace spaces/blanks with "NA"
# Note: R will convert dashes to dots (cap-shape will be cap.shape)

filename="mushrooms.csv"
m<- read.csv(filename, header = TRUE, na.strings = "NA",stringsAsFactors = TRUE)
# m now appears in the Environment under Data as 8124 obs. of 23 variables

# Look at the data, then comment out
#View(m)

# Check for missing values
Total <-sum(is.na(m))
cat("The number of missing values in Mushroom data is ", Total )

## The number of missing values in Mushroom data is  0
```

```r
# The number of missing values in Mushroom data is  0

# Look at a table
paste("Habitat Table:")
```

```
## [1] "Habitat Table:"
```

```r
(table(m$class,m$habitat))
```

```
##
##        d    g    l    m    p    u    w
##   e 1880 1408  240  256  136   96  192
##   p 1268  740  592   36 1008  272    0
```

```r
# Use a loop to create all of the tables at once, then review the data (appears in console)
paste("Loop of all Tables:")
```

```
## [1] "Loop of all Tables:"
```

```r
for(i in 1:ncol(m)){
  print(table(m[i]))
}
```

```
##
##    e    p
## 4208 3916
##
##    b    c    f    k    s    x
##  452    4 3152  828   32 3656
##
##    f    g    s    y
## 2320    4 2556 3244
##
##    b    c    e    g    n    p    r    u    w    y
##  168   44 1500 1840 2284  144   16   16 1040 1072
##
##    f    t
## 4748 3376
##
##    a    c    f    l    m    n    p    s    y
##  400  192 2160  400   36 3528  256  576  576
##
##    a    f
```

```
##  210 7914
##
##    c    w
## 6812 1312
##
##    b    n
## 5612 2512
##
##    b    e    g    h    k    n    o    p    r    u    w    y
## 1728   96  752  732  408 1048   64 1492   24  492 1202   86
##
##    e    t
## 3516 4608
##
##    ?    b    c    e    r
## 2480 3776  556 1120  192
##
##    f    k    s    y
##  552 2372 5176   24
##
##    f    k    s    y
##  600 2304 4936  284
##
##    b    c    e    g    n    o    p    w    y
##  432   36   96  576  448  192 1872 4464    8
##
##    b    c    e    g    n    o    p    w    y
##  432   36   96  576  512  192 1872 4384   24
##
##    p
## 8124
##
##    n    o    w    y
##   96   96 7924    8
##
##    n    o    t
##   36 7488  600
##
##    e    f    l    n    p
## 2776   48 1296   36 3968
##
```

```
##    b    h    k    n    o    r    u    w    y
##   48 1632 1872 1968   48   72   48 2388   48
##
##    a    c    n    s    v    y
##  384  340  400 1248 4040 1712
##
##    d    g    l    m    p    u    w
## 3148 2148  832  292 1144  368  192
```

```r
# The tables show that there are 2480 instances of "?" in column L "stalk-root"

# Below are optional checks that can be run.
#(colnames(m2))  # To get column names
#(head(m2))  # To see the first 6 rows
#(m2)  # To see all the data

# Notes on Which variables contain important information.
  # veil.type is not valuable. There is only one level.
  # All other variables may have value.
  # Class is important; it is our label.

# Use "str" to check the data types.
# Important: The labels must be factors for models to work properly!
paste("Structure of Dataframe M:")
```

```
## [1] "Structure of Dataframe M:"
```

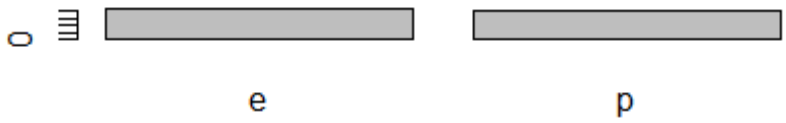
```r
str(m)
```

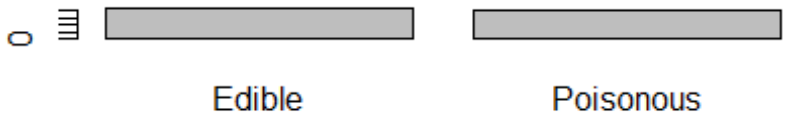
```
## 'data.frame':    8124 obs. of  23 variables:
##  $ class                   : Factor w/ 2 levels "e","p": 2 1 1 2 1 1 1 1 2 1 ...
##  $ cap.shape               : Factor w/ 6 levels "b","c","f","k",..: 6 6 1 6 6 6 1 1 6 1 ...
##  $ cap.surface             : Factor w/ 4 levels "f","g","s","y": 3 3 3 4 3 4 3 4 4 3 ...
##  $ cap.color               : Factor w/ 10 levels "b","c","e","g",..: 5 10 9 9 4 10 9 9 9 10 ...
##  $ bruises                 : Factor w/ 2 levels "f","t": 2 2 2 2 1 2 2 2 2 2 ...
##  $ odor                    : Factor w/ 9 levels "a","c","f","l",..: 7 1 4 7 6 1 1 4 7 1 ...
##  $ gill.attachment         : Factor w/ 2 levels "a","f": 2 2 2 2 2 2 2 2 2 2 ...
##  $ gill.spacing            : Factor w/ 2 levels "c","w": 1 1 1 1 2 1 1 1 1 1 ...
##  $ gill.size               : Factor w/ 2 levels "b","n": 2 1 1 2 1 1 1 1 2 1 ...
##  $ gill.color              : Factor w/ 12 levels "b","e","g","h",..: 5 5 6 6 5 6 3 6 8 3 ...
##  $ stalk.shape             : Factor w/ 2 levels "e","t": 1 1 1 1 2 1 1 1 1 1 ...
##  $ stalk.root              : Factor w/ 5 levels "?","b","c","e",..: 4 3 3 4 4 3 3 3 4 3 ...
```

```
##  $ stalk.surface.above.ring: Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.surface.below.ring: Factor w/ 4 levels "f","k","s","y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.color.above.ring  : Factor w/ 9 levels "b","c","e","g",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ stalk.color.below.ring  : Factor w/ 9 levels "b","c","e","g",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ veil.type               : Factor w/ 1 level "p": 1 1 1 1 1 1 1 1 1 1 ...
##  $ veil.color              : Factor w/ 4 levels "n","o","w","y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ ring.number             : Factor w/ 3 levels "n","o","t": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ring.type               : Factor w/ 5 levels "e","f","l","n",..: 5 5 5 5 1 5 5 5 5 5 ...
##  $ spore.print.color       : Factor w/ 9 levels "b","h","k","n",..: 3 4 4 3 4 3 3 4 3 3 ...
##  $ population              : Factor w/ 6 levels "a","c","n","s",..: 4 3 3 4 1 3 3 4 5 4 ...
##  $ habitat                 : Factor w/ 7 levels "d","g","l","m",..: 6 2 4 6 2 2 4 4 2 4 ...
```

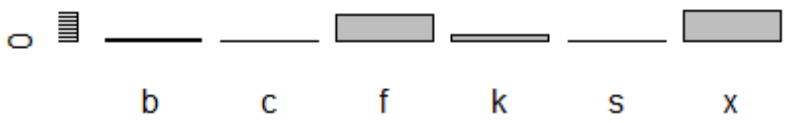
```r
# All are factors.

##############################################################################
# Update Variable Names (Lauren Foltz)
##############################################################################

# Look at barplot, update variable, check barplot again to make sure correct
par(mfrow=c(2,1))  # Set plot parameters for easy comparison
barplot(table(m$class))
```

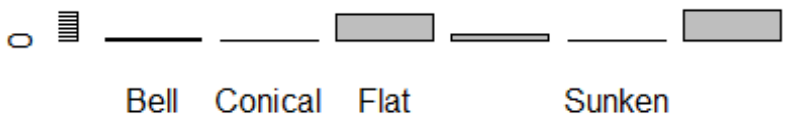
```r
m$class<-recode(m$class, e = "Edible", p = "Poisonous")
barplot(table(m$class))
```



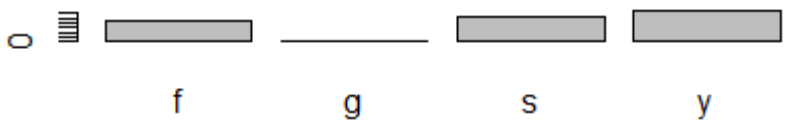


```r
barplot(table(m$cap.shape))
m$cap.shape<-recode(m$cap.shape, b = "Bell", c = "Conical" , x = "Convex" , f = "Flat" , k = "Knobbed" , s = "Sunken")
barplot(table(m$cap.shape))
```

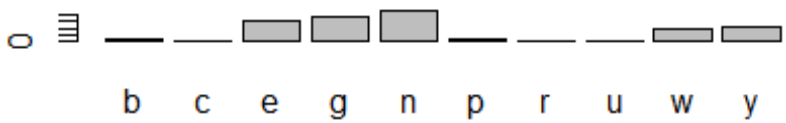


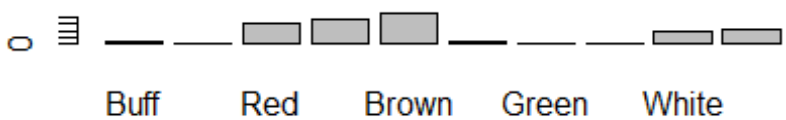


```r
barplot(table(m$cap.surface))
m$cap.surface<-recode(m$cap.surface, f = "Fibrous", g = "Grooves" , y = "Scaly" , s = "Smooth" )
barplot(table(m$cap.surface))
```

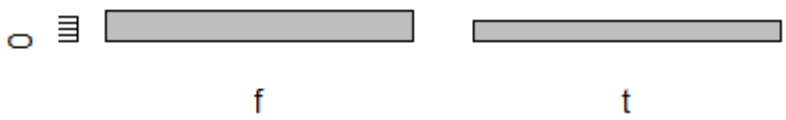


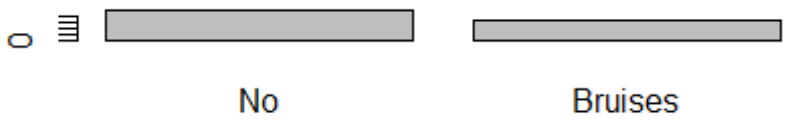
```r
barplot(table(m$cap.color))
m$cap.color<-recode(m$cap.color, n = "Brown", b= "Buff" , c = "Cinnamon" , g = "Gray", r = "Green", p = "Pink" , u =
"Purple" , e = "Red" , w = "White" ,y = "Yellow")
barplot(table(m$cap.color))
```



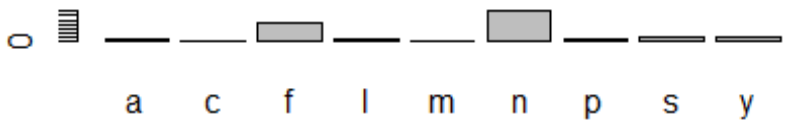


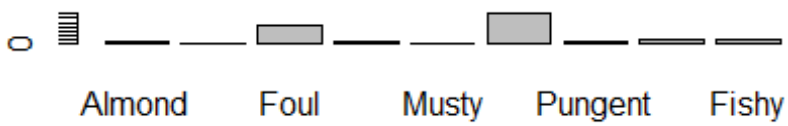
```r
barplot(table(m$bruises))
m$bruises<-recode(m$bruises, t = "Bruises", f= "No")
barplot(table(m$bruises))
```





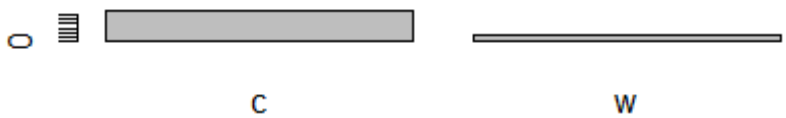
```r
barplot(table(m$odor))
m$odor <-recode(m$odor, a = "Almond", l= "Anise" , c = "Creosote" , y = "Fishy", f = "Foul", m = "Musty" , n = "None" ,
p = "Pungent" , s = "Spicy")
barplot(table(m$odor))
```

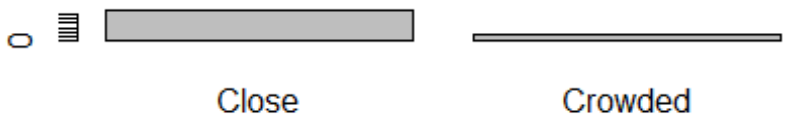




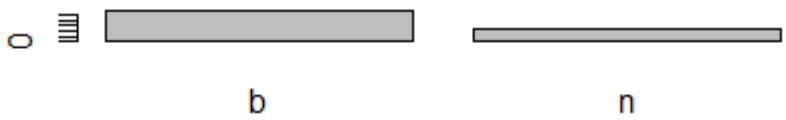
```r
barplot(table(m$gill.attachment))
m$gill.attachment<-recode(m$gill.attachment, a = "Attached", f = "Free")
barplot(table(m$gill.attachment))
```

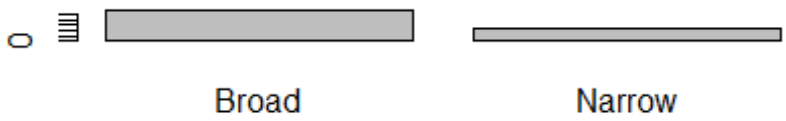
```r
barplot(table(m$gill.spacing))
m$gill.spacing<-recode(m$gill.spacing, c = "Close", w = "Crowded")
barplot(table(m$gill.spacing))
```



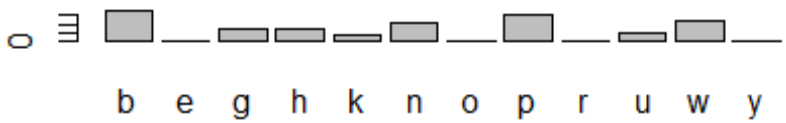


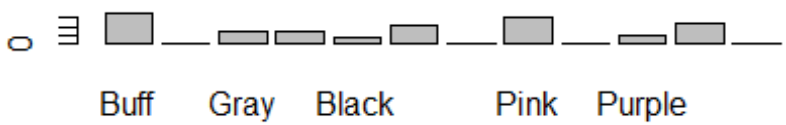
```r
barplot(table(m$gill.size))
m$gill.size<-recode(m$gill.size, b = "Broad", n = "Narrow")
barplot(table(m$gill.size))
```



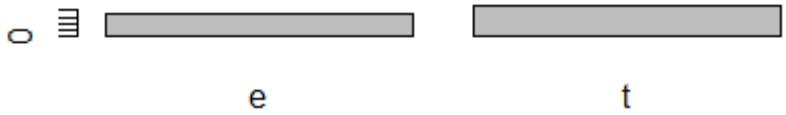


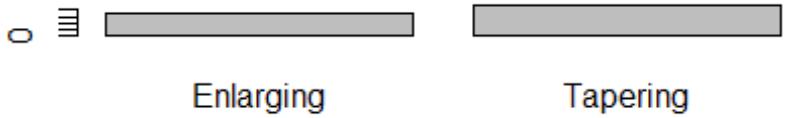
```r
barplot(table(m$gill.color))
m$gill.color<-recode(m$gill.color, k = "Black", n = "Brown", b= "Buff" , h = "Chocolate" , g = "Gray", r = "Green", o =
"Orange", p = "Pink" , u = "Purple" , e = "Red" , w = "White" ,y = "Yellow")
barplot(table(m$gill.color))
```



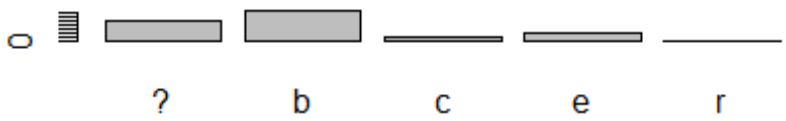


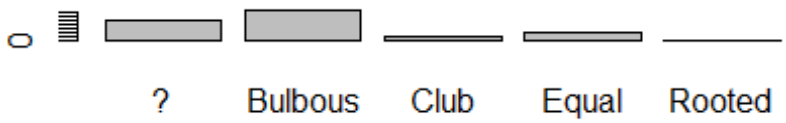
```r
barplot(table(m$stalk.shape))
m$stalk.shape<-recode(m$stalk.shape, e = "Enlarging", t = "Tapering")
barplot(table(m$stalk.shape))
```



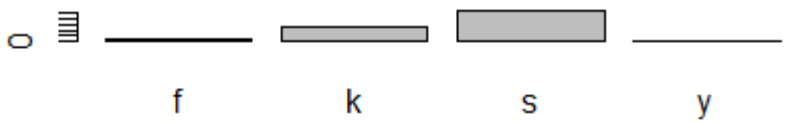


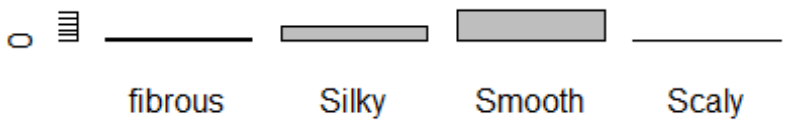
```r
barplot(table(m$stalk.root))
m$stalk.root<-recode(m$stalk.root, b = "Bulbous", c = "Club", e = "Equal" ,r = "Rooted")
barplot(table(m$stalk.root))
```



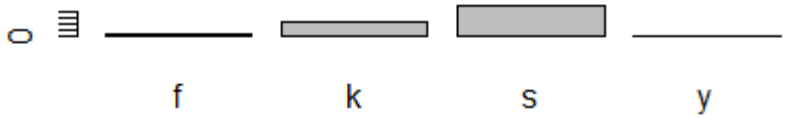


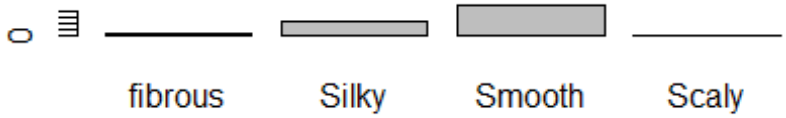
```r
barplot(table(m$stalk.surface.above.ring))
m$stalk.surface.above.ring<-recode(m$stalk.surface.above.ring, f= "fibrous", y = "Scaly", k= "Silky" , s = "Smooth")
barplot(table(m$stalk.surface.above.ring))
```



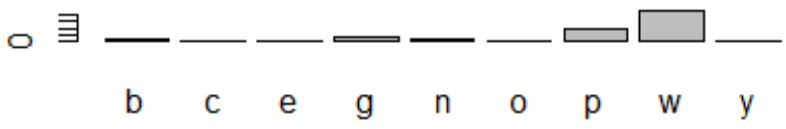


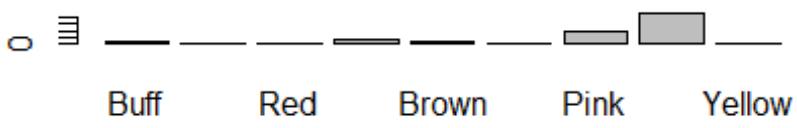
```r
barplot(table(m$stalk.surface.below.ring))
m$stalk.surface.below.ring<-recode(m$stalk.surface.below.ring, f= "fibrous", y = "Scaly", k= "Silky" , s = "Smooth")
barplot(table(m$stalk.surface.below.ring))
```



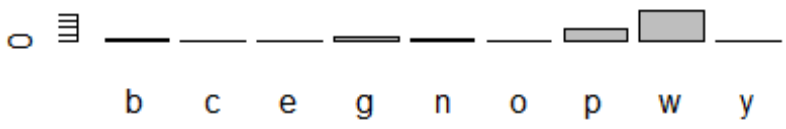


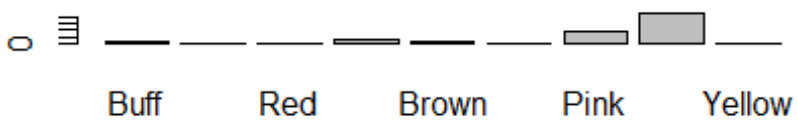
```r
barplot(table(m$stalk.color.above.ring))
m$stalk.color.above.ring<-recode(m$stalk.color.above.ring, n = "Brown", b= "Buff" , c = "Cinnamon" , g = "Gray", o =
"Orange", p = "Pink" , e = "Red" , w = "White" ,y = "Yellow")
barplot(table(m$stalk.color.above.ring))
```



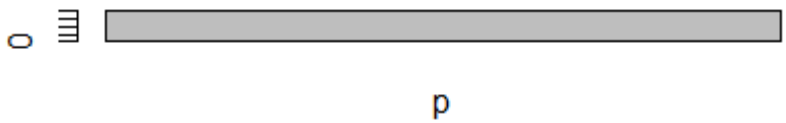


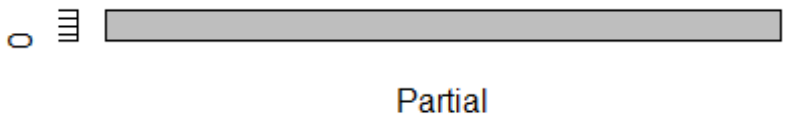
```r
barplot(table(m$stalk.color.below.ring))
m$stalk.color.below.ring<-recode(m$stalk.color.below.ring , n = "Brown", b= "Buff" , c = "Cinnamon" , g = "Gray", o =
"Orange", p = "Pink" , e = "Red" , w = "White" ,y = "Yellow")
barplot(table(m$stalk.color.below.ring))
```



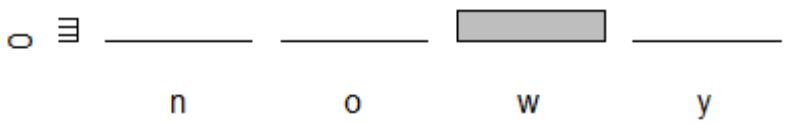


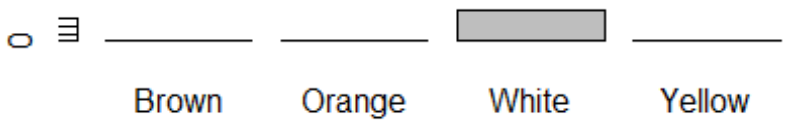
```r
barplot(table(m$veil.type))
m$veil.type<-recode(m$veil.type , p = "Partial")
barplot(table(m$veil.type))
```



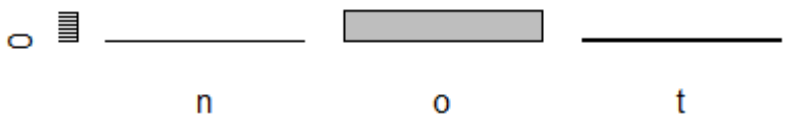


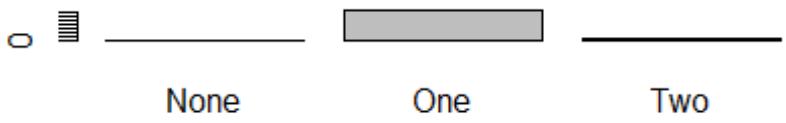
```r
barplot(table(m$veil.color))
m$veil.color<-recode(m$veil.color , n = "Brown", o = "Orange", w = "White" ,y = "Yellow")
barplot(table(m$veil.color))
```



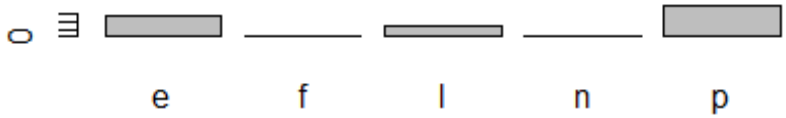


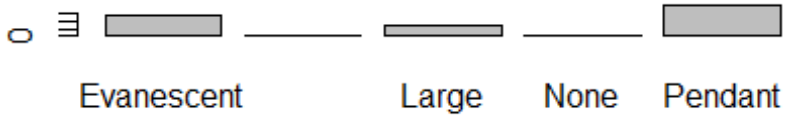
```r
barplot(table(m$ring.number))
m$ring.number<-recode(m$ring.number , n = "None", o = "One", t = "Two")
barplot(table(m$ring.number))
```



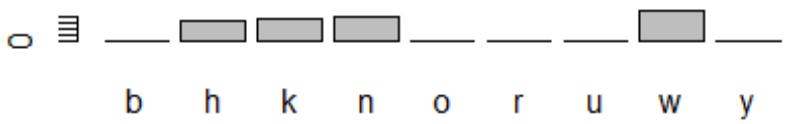


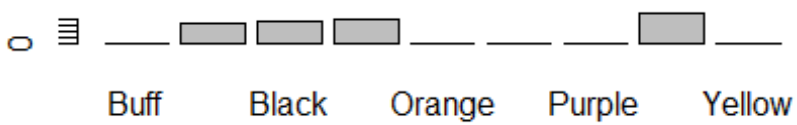
```r
barplot(table(m$ring.type))
m$ring.type<-recode(m$ring.type , e= "Evanescent" , f = "Flaring" , l = "Large", n = "None", p = "Pendant" )
barplot(table(m$ring.type))
```



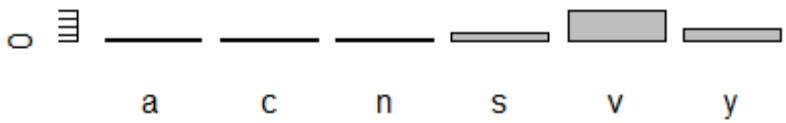


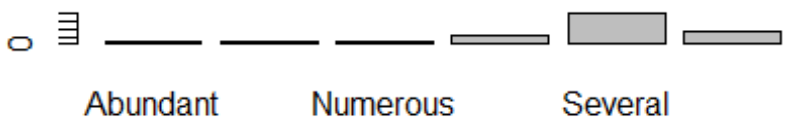
```r
barplot(table(m$spore.print.color))
m$spore.print.color<-recode(m$spore.print.color , k = "Black", n = "Brown", b= "Buff" ,h = "Chocolate" , r = "Green", o
= "Orange" , u = "Purple" , w = "White" ,y = "Yellow")
barplot(table(m$spore.print.color))
```



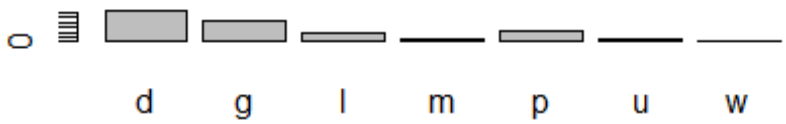


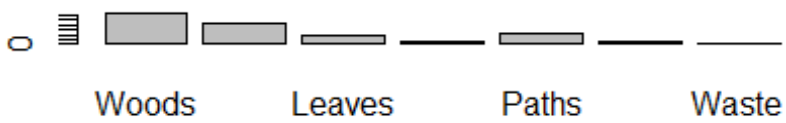
```r
barplot(table(m$population))
m$population<-recode(m$population , a = "Abundant", c= "Clustered" , n = "Numerous" , s = "Scattered", v= "Several", y
= "Solitary")
barplot(table(m$population))
```





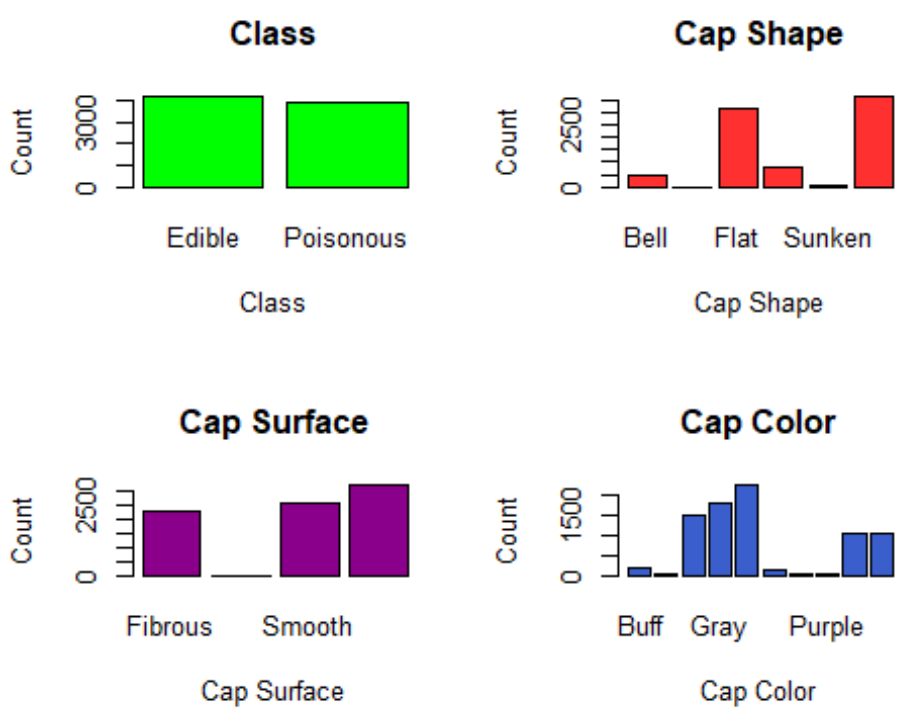
```r
barplot(table(m$habitat))
m$habitat<-recode(m$habitat , g = "Grasses", l= "Leaves" , m = "Meadows" , p = "Paths", u= "Urban", w = "Waste", d =
"Woods")
barplot(table(m$habitat))
```



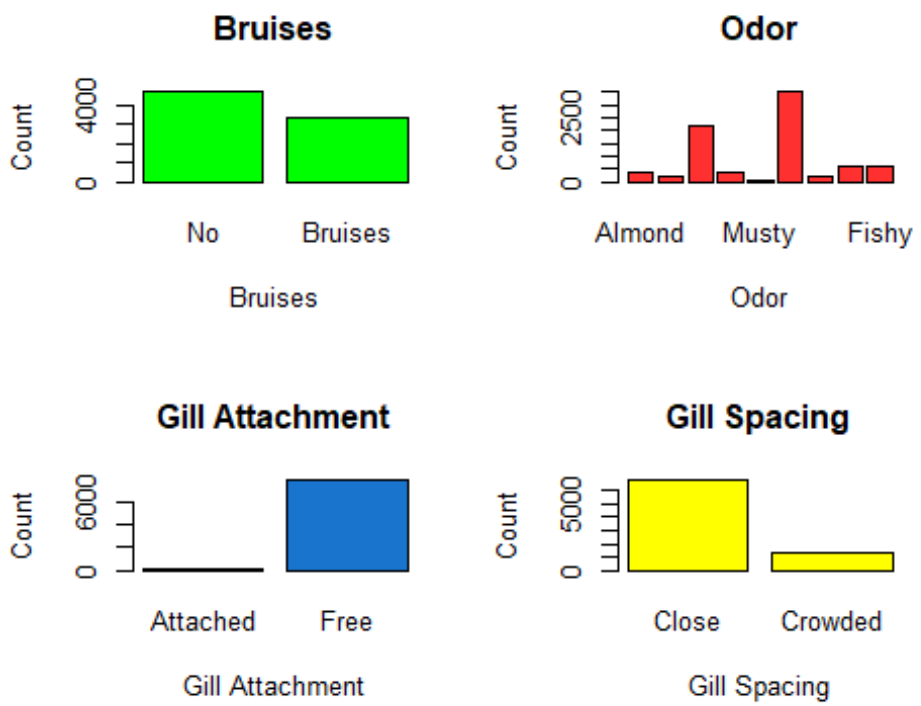



```r
##############################################################################
# Visual of Barcharts (Lauren Foltz)
##############################################################################
```

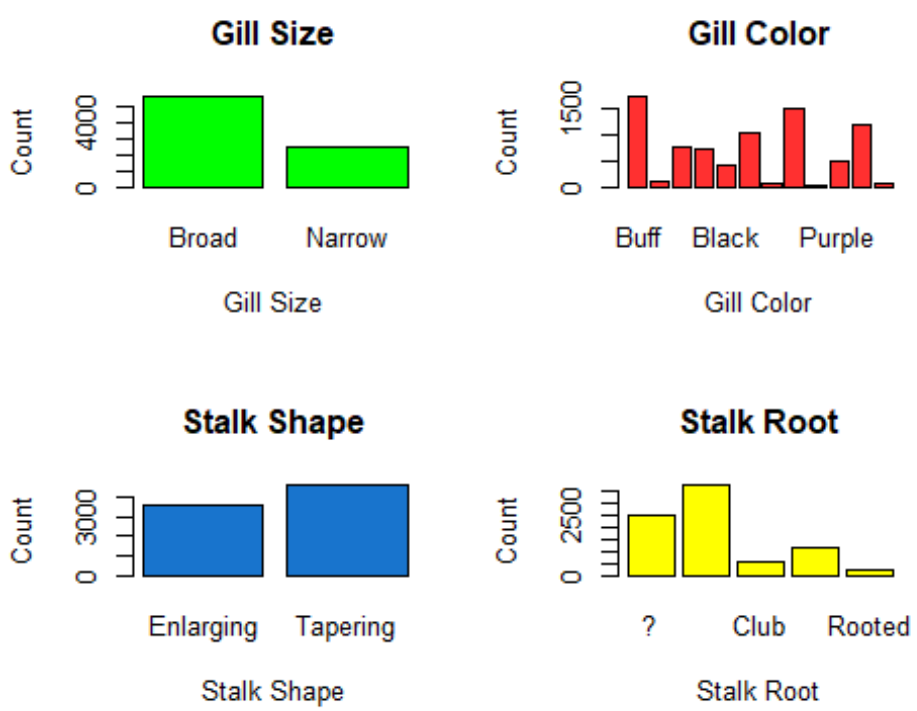
```r
#Set A
par(mfrow=c(2,2))
barplot(table(m$class),
        col='green',
        xlab='Class',ylab='Count',
        main='Class')
barplot(table(m$cap.shape),
        col='firebrick1',
        xlab='Cap Shape',ylab='Count',
        main='Cap Shape')
barplot(table(m$cap.surface),
        col='darkmagenta',
        xlab='Cap Surface',ylab='Count',
        main='Cap Surface')
barplot(table(m$cap.color),
        col='royalblue3',
        xlab='Cap Color',ylab='Count',
        main='Cap Color')
```



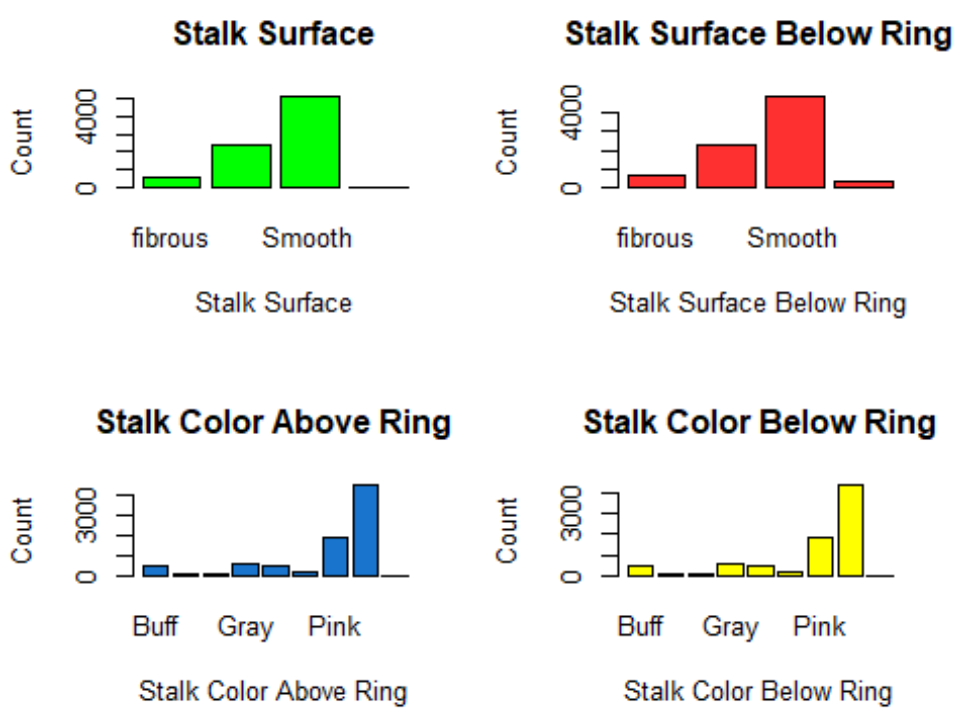
```r
#Set B
par(mfrow=c(2,2))
barplot(table(m$bruises),
        col='green',
        xlab='Bruises',ylab='Count',
        main='Bruises')
barplot(table(m$odor),
        col='firebrick1',
        xlab='Odor',ylab='Count',
        main='Odor')
barplot(table(m$gill.attachment),
        col='dodgerblue3',
        xlab='Gill Attachment',ylab='Count',
        main='Gill Attachment')
barplot(table(m$gill.spacing),
        col='yellow',
        xlab='Gill Spacing',ylab='Count',
        main='Gill Spacing')
```



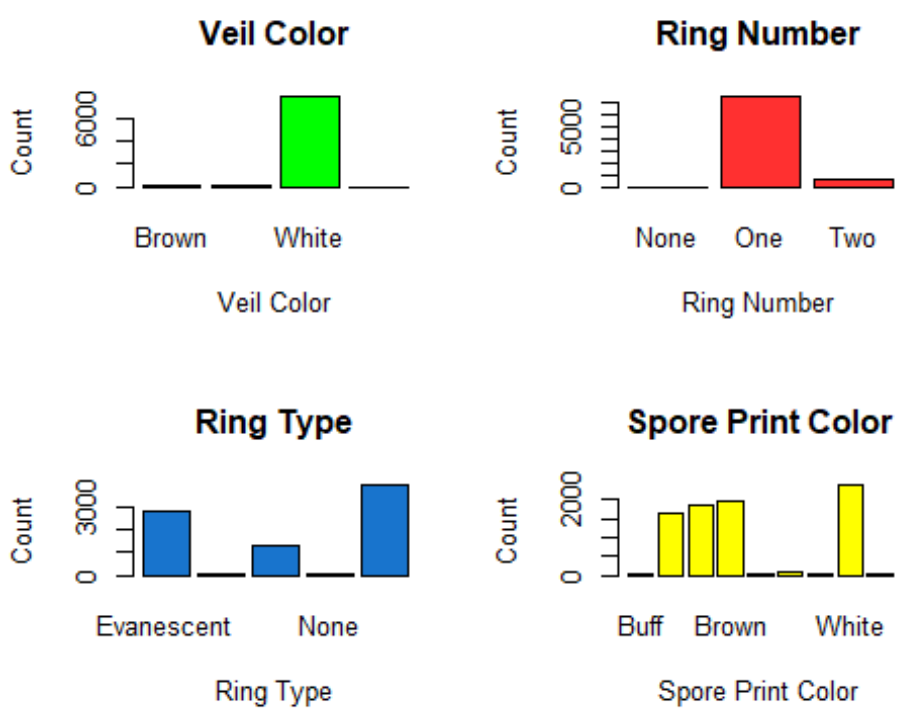
```r
#Set C
par(mfrow=c(2,2))
barplot(table(m$gill.size),
        col='green',
        xlab='Gill Size',ylab='Count',
        main='Gill Size')
barplot(table(m$gill.color),
        col='firebrick1',
        xlab='Gill Color',ylab='Count',
        main='Gill Color')
barplot(table(m$stalk.shape),
        col='dodgerblue3',
        xlab='Stalk Shape',ylab='Count',
        main='Stalk Shape')
barplot(table(m$stalk.root),
        col='yellow',
        xlab='Stalk Root',ylab='Count',
        main='Stalk Root')
```



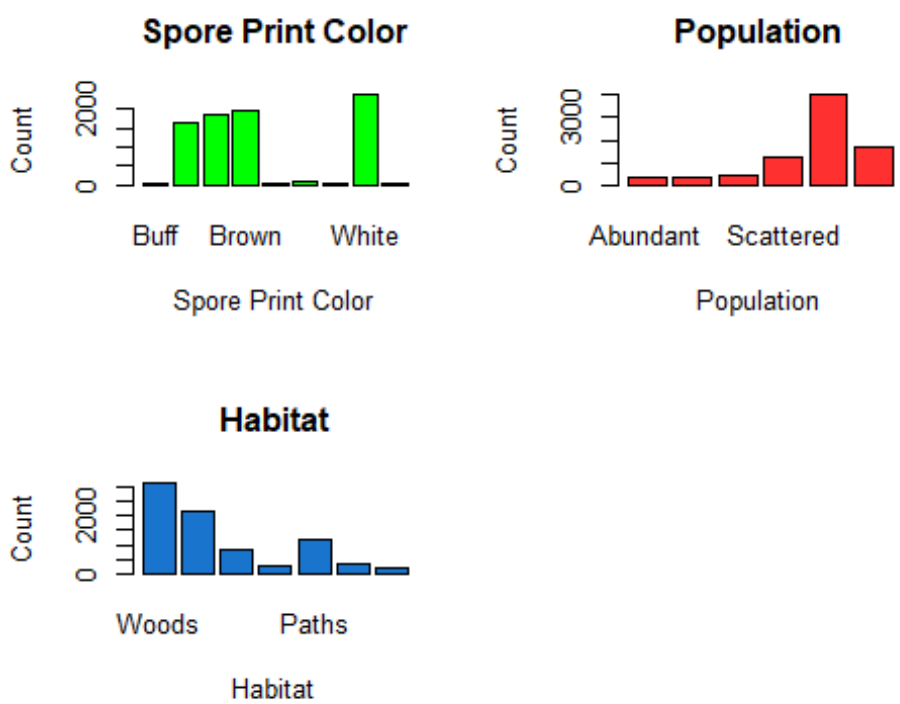
```r
#Set D
par(mfrow=c(2,2))
barplot(table(m$stalk.surface.above.ring),
        col='green',
        xlab='Stalk Surface',ylab='Count',
        main='Stalk Surface')
barplot(table(m$stalk.surface.below.ring),
        col='firebrick1',
        xlab='Stalk Surface Below Ring',ylab='Count',
        main='Stalk Surface Below Ring')
barplot(table(m$stalk.color.above.ring),
        col='dodgerblue3',
        xlab='Stalk Color Above Ring',ylab='Count',
        main='Stalk Color Above Ring')
barplot(table(m$stalk.color.below.ring),
        col='yellow',
        xlab='Stalk Color Below Ring',ylab='Count',
        main='Stalk Color Below Ring')
```



```r
#Set E
par(mfrow=c(2,2))
barplot(table(m$veil.color),
        col='green',
        xlab='Veil Color',ylab='Count',
        main='Veil Color')
barplot(table(m$ring.number),
        col='firebrick1',
        xlab='Ring Number',ylab='Count',
        main='Ring Number')
barplot(table(m$ring.type),
        col='dodgerblue3',
        xlab='Ring Type',ylab='Count',
        main='Ring Type')
barplot(table(m$spore.print.color),
        col='yellow',
        xlab='Spore Print Color',ylab='Count',
        main='Spore Print Color')
```



```r
#Set F
par(mfrow=c(2,2))
barplot(table(m$spore.print.color),
        col='green',
        xlab='Spore Print Color',ylab='Count',
        main='Spore Print Color')
barplot(table(m$population),
        col='firebrick1',
        xlab='Population',ylab='Count',
        main='Population')
barplot(table(m$habitat),
        col='dodgerblue3',
        xlab='Habitat',ylab='Count',
        main='Habitat')

# Reset parameters before moving forward
par(mfrow=c(1,1))
```



```r
#==============================================================================
# Create Train and Test set for Decision Tree and Random Forest (Baskar Dakshin)
#==============================================================================

# Make a copy of the data frame and remove veil.type, as it has only one value
m2<-m[,-c(17)]
str(m2)
```

```
## 'data.frame':    8124 obs. of  22 variables:
##  $ class                   : Factor w/ 2 levels "Edible","Poisonous": 2 1 1 2 1 1 1 1 2 1 ...
##  $ cap.shape               : Factor w/ 6 levels "Bell","Conical",..: 6 6 1 6 6 6 1 1 6 1 ...
##  $ cap.surface             : Factor w/ 4 levels "Fibrous","Grooves",..: 3 3 3 4 3 4 3 4 4 3 ...
##  $ cap.color               : Factor w/ 10 levels "Buff","Cinnamon",..: 5 10 9 9 4 10 9 9 9 10 ...
##  $ bruises                 : Factor w/ 2 levels "No","Bruises": 2 2 2 2 1 2 2 2 2 2 ...
##  $ odor                    : Factor w/ 9 levels "Almond","Creosote",..: 7 1 4 7 6 1 1 4 7 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 "Buff","Red","Gray",..: 5 5 6 6 5 6 3 6 8 3 ...
##  $ 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",..: 4 3 3 4 4 3 3 3 4 3 ...
##  $ stalk.surface.above.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.surface.below.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.color.above.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ stalk.color.below.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ 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 "Buff","Chocolate",..: 3 4 4 3 4 3 3 4 3 3 ...
##  $ population              : Factor w/ 6 levels "Abundant","Clustered",..: 4 3 3 4 1 3 3 4 5 4 ...
##  $ habitat                 : Factor w/ 7 levels "Woods","Grasses",..: 6 2 4 6 2 2 4 4 2 4 ...
```

```r
# Create Training and Test Dataset using Sample method.
# Took 70% for Training and 30% for Testing data
n = nrow(m2)
n
```

```
## [1] 8124
```

```r
trainIndex = sample(1:n, size = round(0.7*n), replace=FALSE)
```

```r
# Create Training Set
mush_train = m2[trainIndex ,]
str(mush_train)
```

```
## 'data.frame':    5687 obs. of  22 variables:
##  $ class                   : Factor w/ 2 levels "Edible","Poisonous": 2 1 1 1 2 1 1 1 1 1 ...
##  $ cap.shape               : Factor w/ 6 levels "Bell","Conical",..: 3 3 6 6 3 4 3 6 6 6 ...
##  $ cap.surface             : Factor w/ 4 levels "Fibrous","Grooves",..: 4 1 3 4 3 4 4 1 3 1 ...
##  $ cap.color               : Factor w/ 10 levels "Buff","Cinnamon",..: 5 5 5 3 5 1 3 10 5 9 ...
##  $ bruises                 : Factor w/ 2 levels "No","Bruises": 1 2 1 2 1 2 2 2 1 1 ...
##  $ odor                    : Factor w/ 9 levels "Almond","Creosote",..: 8 6 6 6 3 6 6 4 6 6 ...
##  $ gill.attachment         : Factor w/ 2 levels "Attached","Free": 2 2 2 2 2 2 2 2 1 2 ...
##  $ gill.spacing            : Factor w/ 2 levels "Close","Crowded": 1 1 2 1 1 1 1 2 1 2 ...
##  $ gill.size               : Factor w/ 2 levels "Broad","Narrow": 2 1 1 1 2 1 1 2 1 1 ...
##  $ gill.color              : Factor w/ 12 levels "Buff","Red","Gray",..: 1 6 8 11 1 11 10 11 7 8 ...
##  $ stalk.shape             : Factor w/ 2 levels "Enlarging","Tapering": 2 2 2 2 2 1 2 2 1 2 ...
##  $ stalk.root              : Factor w/ 5 levels "?","Bulbous",..: 1 2 4 2 1 1 2 2 1 4 ...
##  $ stalk.surface.above.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 3 3 3 3 3 3 3 1 ...
##  $ stalk.surface.below.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 1 3 2 3 3 3 3 1 ...
##  $ stalk.color.above.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 8 4 8 4 7 8 8 8 6 8 ...
##  $ stalk.color.below.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 7 8 8 8 8 8 4 8 6 8 ...
##  $ veil.color              : Factor w/ 4 levels "Brown","Orange",..: 3 3 3 3 3 3 3 3 1 3 ...
##  $ ring.number             : Factor w/ 3 levels "None","One","Two": 2 2 2 2 2 3 2 2 2 2 ...
##  $ ring.type               : Factor w/ 5 levels "Evanescent","Flaring",..: 1 5 1 5 1 1 5 5 5 1 ...
##  $ spore.print.color       : Factor w/ 9 levels "Buff","Chocolate",..: 8 4 3 4 8 8 4 7 1 4 ...
##  $ population              : Factor w/ 6 levels "Abundant","Clustered",..: 5 5 1 5 5 2 6 5 2 1 ...
##  $ habitat                 : Factor w/ 7 levels "Woods","Grasses",..: 3 1 2 1 3 7 1 1 3 2 ...
```

```r
dim(mush_train)
```

```
## [1] 5687   22
```

```r
# Create Testing set
mush_test = m2[-trainIndex ,]
dim(mush_test)
```

```
## [1] 2437   22
```

```r
# Remove the Decision Class from the Testing set
mush_test_nolabels<-mush_test[-c(1)]
dim(mush_test_nolabels)
```

```
## [1] 2437   21
```

```r
# Remove the Decision Class from the Training set
mush_train_nolabels<-mush_train[-c(1)]
dim(mush_train_nolabels)
```

```
## [1] 5687   21
```

```r
str(mush_test_nolabels)
```

```
## 'data.frame':    2437 obs. of  21 variables:
##  $ cap.shape               : Factor w/ 6 levels "Bell","Conical",..: 1 6 1 6 6 5 6 1 6 6 ...
##  $ cap.surface             : Factor w/ 4 levels "Fibrous","Grooves",..: 3 4 4 4 1 1 3 3 4 4 ...
##  $ cap.color               : Factor w/ 10 levels "Buff","Cinnamon",..: 9 9 9 10 5 4 5 10 9 10 ...
##  $ bruises                 : Factor w/ 2 levels "No","Bruises": 2 2 2 2 1 1 2 2 2 2 ...
##  $ odor                    : Factor w/ 9 levels "Almond","Creosote",..: 4 7 4 4 6 6 7 4 7 4 ...
##  $ 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": 1 2 1 1 1 2 2 1 2 1 ...
##  $ gill.color              : Factor w/ 12 levels "Buff","Red","Gray",..: 6 6 6 3 6 5 6 3 5 6 ...
##  $ 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",..: 3 4 3 3 4 4 4 3 4 3 ...
##  $ stalk.surface.above.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.surface.below.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 3 3 1 3 3 3 3 3 ...
##  $ stalk.color.above.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ stalk.color.below.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ 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 "Buff","Chocolate",..: 4 3 4 4 3 4 3 4 4 4 ...
##  $ population              : Factor w/ 6 levels "Abundant","Clustered",..: 3 4 4 3 1 6 4 3 4 3 ...
##  $ habitat                 : Factor w/ 7 levels "Woods","Grasses",..: 4 6 4 2 2 6 2 4 6 4 ...
```

```r
# Create a Label Data Frame for Class variable
TestClassLabels<-mush_test$class
length(TestClassLabels)
```

```
## [1] 2437
```

```r
TrainClassLabels<-mush_train$class
length(TrainClassLabels)
```

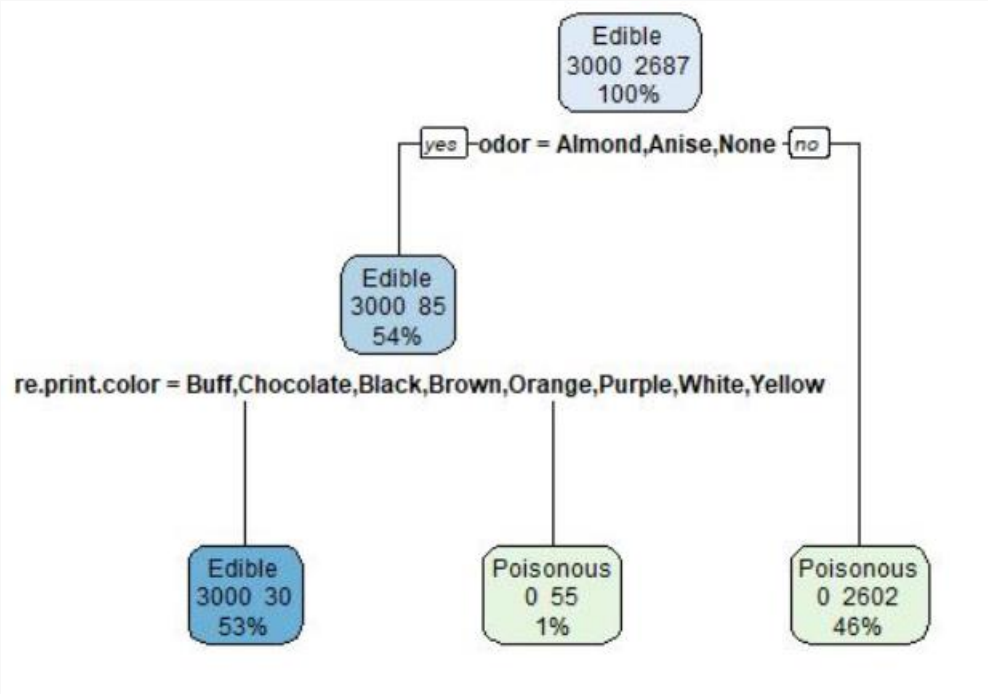
```
## [1] 5687
```

```r
#==============================================================================
# Decision Tree Classification Modelling (Baskar Dakshin)
#==============================================================================

# Running the Model to find out what variables are important
set.seed(123)
# Create a model with split as gini
model <- rpart( class ~ ., data=mush_train,method="class",parms = list(split = "gini"))
summary(model)
```

```
## Call:
## rpart(formula = class ~ ., data = mush_train, method = "class",
##     parms = list(split = "gini"))
##   n= 5687
##
##           CP nsplit  rel error     xerror        xstd
## 1 0.96836621      0 1.00000000 1.00000000 0.014011518
## 2 0.02046892      1 0.03163379 0.03163379 0.003405428
## 3 0.01000000      2 0.01116487 0.01116487 0.002033033
##
## Variable importance
##                     odor        spore.print.color               gill.color
##                       26                       19                       16
## stalk.surface.above.ring stalk.surface.below.ring                ring.type
##                       14                       13                       13
##
## Node number 1: 5687 observations,    complexity param=0.9683662
##   predicted class=Edible     expected loss=0.4724811  P(node) =1
##     class counts:  3000  2687
##    probabilities: 0.528 0.472
##   left son=2 (3085 obs) right son=3 (2602 obs)
##   Primary splits:
##       odor                     splits as  LRRLRLRRR,    improve=2669.5710, (0 missing)
##       spore.print.color        splits as  LRLLLRLRL,    improve=1513.6540, (0 missing)
##       gill.color               splits as  RLRRLLLLRLLL, improve=1105.0000, (0 missing)
##       stalk.surface.above.ring splits as  LRLL,         improve= 977.3323, (0 missing)
##       stalk.surface.below.ring splits as  LRLL,         improve= 914.1120, (0 missing)
##   Surrogate splits:
##       spore.print.color        splits as  LRLLLLLRL,    agree=0.860, adj=0.693, (0 split)
##       gill.color               splits as  RLRRLLLLLLLL, agree=0.815, adj=0.595, (0 split)
##       stalk.surface.above.ring splits as  LRLL,         agree=0.784, adj=0.528, (0 split)
##       stalk.surface.below.ring splits as  LRLL,         agree=0.781, adj=0.521, (0 split)
```

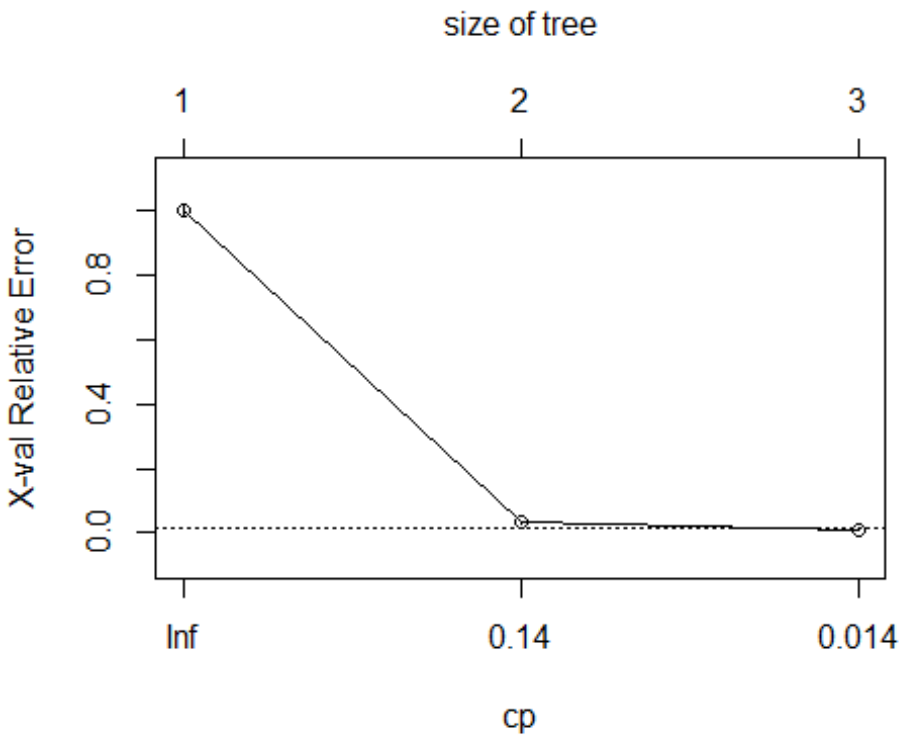
```
##       ring.type                splits as  RLRRL,        agree=0.777, adj=0.512, (0 split)
##
## Node number 2: 3085 observations,    complexity param=0.02046892
##   predicted class=Edible     expected loss=0.02755267  P(node) =0.5424653
##     class counts:  3000    85
##    probabilities: 0.972 0.028
##   left son=4 (3030 obs) right son=5 (55 obs)
##   Primary splits:
##       spore.print.color      splits as  LLLLLRLLL,    improve=105.910100, (0 missing)
##       gill.color             splits as  -LLLLLLLRLLL, improve= 39.989670, (0 missing)
##       stalk.color.below.ring splits as  --LLLLLLR,    improve= 32.330390, (0 missing)
##       cap.color              splits as  RLLLLRLLLL,   improve= 21.700640, (0 missing)
##       ring.number            splits as  -LR,          improve=  9.684872, (0 missing)
##   Surrogate splits:
##       gill.color splits as  -LLLLLLLRLLL, agree=0.989, adj=0.382, (0 split)
##
## Node number 3: 2602 observations
##   predicted class=Poisonous  expected loss=0  P(node) =0.4575347
##     class counts:     0  2602
##    probabilities: 0.000 1.000
##
## Node number 4: 3030 observations
##   predicted class=Edible     expected loss=0.00990099  P(node) =0.5327941
##     class counts:  3000    30
##    probabilities: 0.990 0.010
##
## Node number 5: 55 observations
##   predicted class=Poisonous  expected loss=0  P(node) =0.00967118
##     class counts:     0    55
##    probabilities: 0.000 1.000
```

```r
# Create the Decision Tree and save as Jpeg
jpeg("DecisionTree_Mushroom_Sample1.jpg")
fancyRpartPlot(model)
rpart.plot(model,extr=101)
dev.off()
```




```
## png
##   2
```

```r
plotcp(model)
```




```r
printcp(model)

##
## Classification tree:
## rpart(formula = class ~ ., data = mush_train, method = "class",
##     parms = list(split = "gini"))
##
## Variables actually used in tree construction:
## [1] odor              spore.print.color
##
## Root node error: 2687/5687 = 0.47248
##
## n= 5687
```

```
##
##         CP nsplit rel error   xerror      xstd
## 1 0.968366      0  1.000000 1.000000 0.0140115
## 2 0.020469      1  0.031634 0.031634 0.0034054
## 3 0.010000      2  0.011165 0.011165 0.0020330
```

```r
# Do the prediction
#mush_test_nolabels
predicted=predict(model,mush_test_nolabels,type="class")
length(predicted)
```

```
## [1] 2437
```

```r
Results<-data.frame(predicted=predicted,Actual=TestClassLabels)
(table(Results))
```

```
##            Actual
## predicted   Edible Poisonous
##   Edible      1208        18
##   Poisonous      0      1211
```

```r
# Calculate Accuracy using the ConfusionMatrix- Accuracy comes in at 99.6%
confusionMatrix(predicted,TestClassLabels)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Edible Poisonous
##   Edible      1208        18
##   Poisonous      0      1211
##
##                Accuracy : 0.9926
##                  95% CI : (0.9884, 0.9956)
##     No Information Rate : 0.5043
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9852
##
##  Mcnemar's Test P-Value : 6.151e-05
##
##             Sensitivity : 1.0000
##             Specificity : 0.9854
##          Pos Pred Value : 0.9853
```

```
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4957
##          Detection Rate : 0.4957
##    Detection Prevalence : 0.5031
##       Balanced Accuracy : 0.9927
##
##        'Positive' Class : Edible
##
```

```r
# I tried different combination of Variables against Decision Class
model1 <- rpart( class ~ odor+spore.print.color+gill.color+stalk.surface.above.ring+stalk.color.below.ring+ring.type,
                 data=mush_train,method="class",control = rpart.control(cp = 0, maxdepth = 8,minsplit = 10))
summary(model1)
```

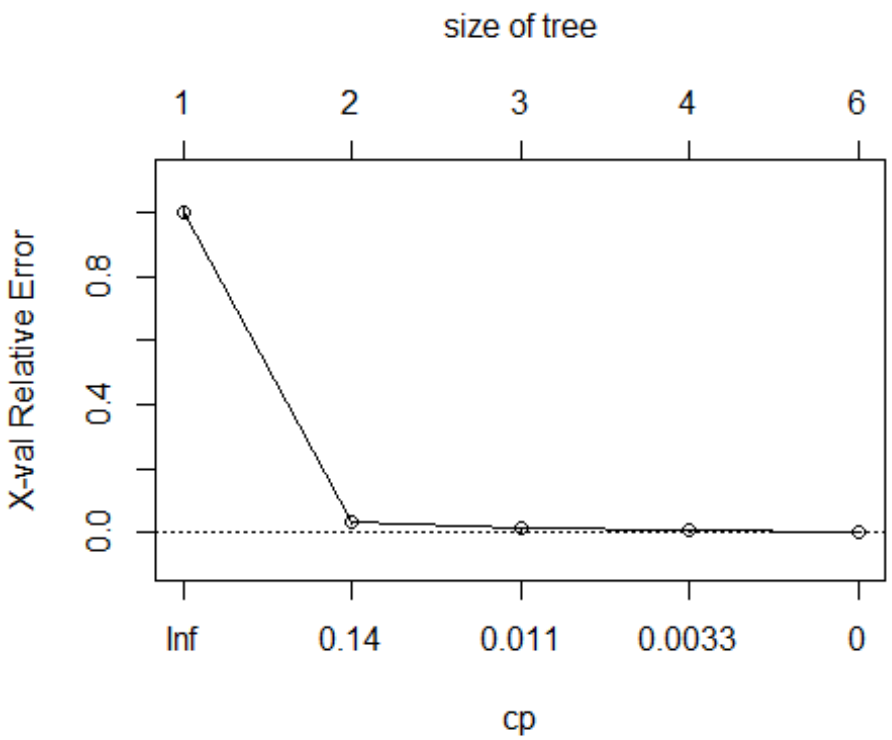
```
## Call:
## rpart(formula = class ~ odor + spore.print.color + gill.color +
##     stalk.surface.above.ring + stalk.color.below.ring + ring.type,
##     data = mush_train, method = "class", control = rpart.control(cp = 0,
##         maxdepth = 8, minsplit = 10))
##   n= 5687
##
##            CP nsplit   rel error      xerror         xstd
## 1 0.968366208      0 1.000000000 1.000000000 0.0140115177
## 2 0.020468924      1 0.031633792 0.031633792 0.0034054283
## 3 0.006326758      2 0.011164868 0.011164868 0.0020330330
## 4 0.001674730      3 0.004838109 0.004838109 0.0013403156
## 5 0.000000000      5 0.001488649 0.001488649 0.0007440627
##
## Variable importance
##                     odor        spore.print.color               gill.color
##                       26                       19                       16
## stalk.surface.above.ring                ring.type   stalk.color.below.ring
##                       14                       13                       11
##
## Node number 1: 5687 observations,    complexity param=0.9683662
##   predicted class=Edible     expected loss=0.4724811  P(node) =1
##     class counts:  3000  2687
##    probabilities: 0.528 0.472
##   left son=2 (3085 obs) right son=3 (2602 obs)
##   Primary splits:
##       odor                     splits as  LRRLRLRRR,    improve=2669.5710, (0 missing)
##       spore.print.color        splits as  LRLLLRLRL,    improve=1513.6540, (0 missing)
```

```
##       gill.color               splits as  RLRRLLLLRLLL, improve=1105.0000, (0 missing)
##       stalk.surface.above.ring splits as  LRLL,         improve= 977.3323, (0 missing)
##       ring.type                splits as  RLRRL,        improve= 833.0020, (0 missing)
##   Surrogate splits:
##       spore.print.color        splits as  LRLLLLLRL,    agree=0.860, adj=0.693, (0 split)
##       gill.color               splits as  RLRRLLLLLLLL, agree=0.815, adj=0.595, (0 split)
##       stalk.surface.above.ring splits as  LRLL,         agree=0.784, adj=0.528, (0 split)
##       ring.type                splits as  RLRRL,        agree=0.777, adj=0.512, (0 split)
##       stalk.color.below.ring   splits as  RRLLRLRLL,    agree=0.722, adj=0.393, (0 split)
##
## Node number 2: 3085 observations,    complexity param=0.02046892
##   predicted class=Edible     expected loss=0.02755267  P(node) =0.5424653
##     class counts:  3000    85
##    probabilities: 0.972 0.028
##   left son=4 (3030 obs) right son=5 (55 obs)
##   Primary splits:
##       spore.print.color        splits as  LLLLLRLLL,    improve=105.910100, (0 missing)
##       gill.color               splits as  -LLLLLLLRLLL, improve= 39.989670, (0 missing)
##       stalk.color.below.ring   splits as  --LLLLLLR,    improve= 32.330390, (0 missing)
##       stalk.surface.above.ring splits as  LRLR,         improve=  7.475549, (0 missing)
##       odor                     splits as  L--L-R---,    improve=  1.050172, (0 missing)
##   Surrogate splits:
##       gill.color splits as  -LLLLLLLRLLL, agree=0.989, adj=0.382, (0 split)
##
## Node number 3: 2602 observations
##   predicted class=Poisonous  expected loss=0  P(node) =0.4575347
##     class counts:     0  2602
##    probabilities: 0.000 1.000
##
## Node number 4: 3030 observations,    complexity param=0.006326758
##   predicted class=Edible     expected loss=0.00990099  P(node) =0.5327941
##     class counts:  3000    30
##    probabilities: 0.990 0.010
##   left son=8 (3013 obs) right son=9 (17 obs)
##   Primary splits:
##       stalk.color.below.ring   splits as  --LLLLLLR,    improve=33.518120, (0 missing)
##       stalk.surface.above.ring splits as  LRLR,         improve= 9.214107, (0 missing)
##       spore.print.color        splits as  LLLLL-LRL,    improve= 3.405941, (0 missing)
##       gill.color               splits as  -LLLLLLL-LRR, improve= 1.871694, (0 missing)
##       ring.type                splits as  RL--L,        improve= 1.217860, (0 missing)
##
```

```
## Node number 5: 55 observations
##   predicted class=Poisonous  expected loss=0  P(node) =0.00967118
##     class counts:     0    55
##    probabilities: 0.000 1.000
##
## Node number 8: 3013 observations,    complexity param=0.00167473
##   predicted class=Edible     expected loss=0.004314637  P(node) =0.5298048
##     class counts:  3000    13
##    probabilities: 0.996 0.004
##   left son=16 (2960 obs) right son=17 (53 obs)
##   Primary splits:
##       stalk.color.below.ring   splits as  --LLRLLL-,    improve=2.9552340, (0 missing)
##       stalk.surface.above.ring splits as  LRLL,         improve=1.3715700, (0 missing)
##       spore.print.color        splits as  LLLLL-LRL,    improve=0.6684199, (0 missing)
##       gill.color               splits as  -LLLLLLL-LRL, improve=0.3870809, (0 missing)
##       ring.type                splits as  RL--L,        improve=0.1222750, (0 missing)
##   Surrogate splits:
##       stalk.surface.above.ring splits as  LLLR, agree=0.986, adj=0.208, (0 split)
##
## Node number 9: 17 observations
##   predicted class=Poisonous  expected loss=0  P(node) =0.002989274
##     class counts:     0    17
##    probabilities: 0.000 1.000
##
## Node number 16: 2960 observations
##   predicted class=Edible     expected loss=0.001351351  P(node) =0.5204853
##     class counts:  2956     4
##    probabilities: 0.999 0.001
##
## Node number 17: 53 observations,    complexity param=0.00167473
##   predicted class=Edible     expected loss=0.1698113  P(node) =0.009319501
##     class counts:    44     9
##    probabilities: 0.830 0.170
##   left son=34 (44 obs) right son=35 (9 obs)
##   Primary splits:
##       stalk.surface.above.ring splits as  LRLL,  improve=14.9434000, (0 missing)
##       ring.type                splits as  R---L, improve= 0.8005391, (0 missing)
##
## Node number 34: 44 observations
##   predicted class=Edible     expected loss=0  P(node) =0.007736944
##     class counts:    44     0
```

```
##    probabilities: 1.000 0.000
##
## Node number 35: 9 observations
##   predicted class=Poisonous  expected loss=0  P(node) =0.001582557
##     class counts:     0     9
##    probabilities: 0.000 1.000
```

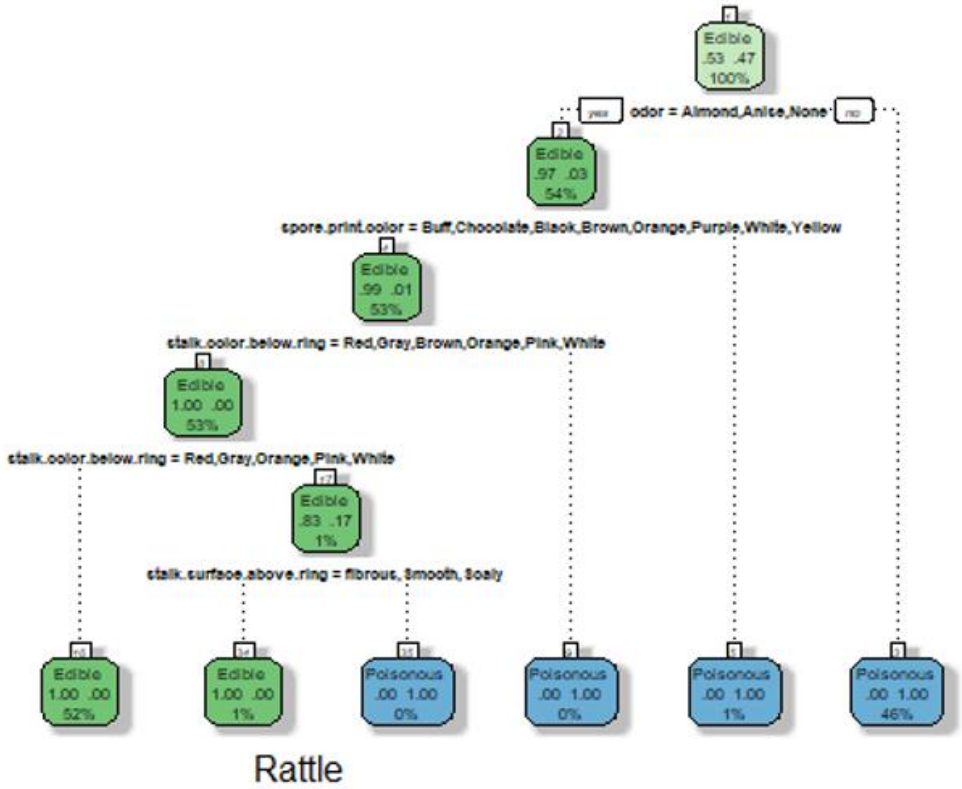
```
plotcp(model1)
```




```
printcp(model1)

##
## Classification tree:
## rpart(formula = class ~ odor + spore.print.color + gill.color +
##     stalk.surface.above.ring + stalk.color.below.ring + ring.type,
##     data = mush_train, method = "class", control = rpart.control(cp = 0,
##         maxdepth = 8, minsplit = 10))
```

```
##
## Variables actually used in tree construction:
## [1] odor                      spore.print.color
## [3] stalk.color.below.ring    stalk.surface.above.ring
##
## Root node error: 2687/5687 = 0.47248
##
## n= 5687
##
##          CP nsplit rel error    xerror       xstd
## 1 0.9683662      0 1.0000000 1.0000000 0.01401152
## 2 0.0204689      1 0.0316338 0.0316338 0.00340543
## 3 0.0063268      2 0.0111649 0.0111649 0.00203303
## 4 0.0016747      3 0.0048381 0.0048381 0.00134032
## 5 0.0000000      5 0.0014886 0.0014886 0.00074406
```

```r
fancyRpartPlot(model1)  # Visual of tree
```



Rattle

```r
predicted=predict(model1,mush_test_nolabels,type="class")
Results<-data.frame(predicted=predicted,Actual=TestClassLabels)
(table(Results))  # Shows Actual vs Predicted in Console
```

```
##            Actual
## predicted   Edible Poisonous
##   Edible      1208         4
##   Poisonous      0      1225
```

```r
confusionMatrix(predicted,TestClassLabels)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Edible Poisonous
##   Edible      1208         4
##   Poisonous      0      1225
##
##                Accuracy : 0.9984
##                  95% CI : (0.9958, 0.9996)
##     No Information Rate : 0.5043
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9967
##
##  Mcnemar's Test P-Value : 0.1336
##
##             Sensitivity : 1.0000
##             Specificity : 0.9967
##          Pos Pred Value : 0.9967
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4957
##          Detection Rate : 0.4957
##    Detection Prevalence : 0.4973
##       Balanced Accuracy : 0.9984
##
##        'Positive' Class : Edible
##
```

```r
#class ~ cap.color+cap.shape+habitat
# Accuracy Comes in at 78%.
# We can do this for all other combinations but looks like Odor and Spore.print.color are variables of Importance.
```

```r
model2 <- rpart( class ~ cap.color+cap.shape+habitat, data=mush_train,method="class",minsplit = 2, minbucket = 1)
summary(model2)
```

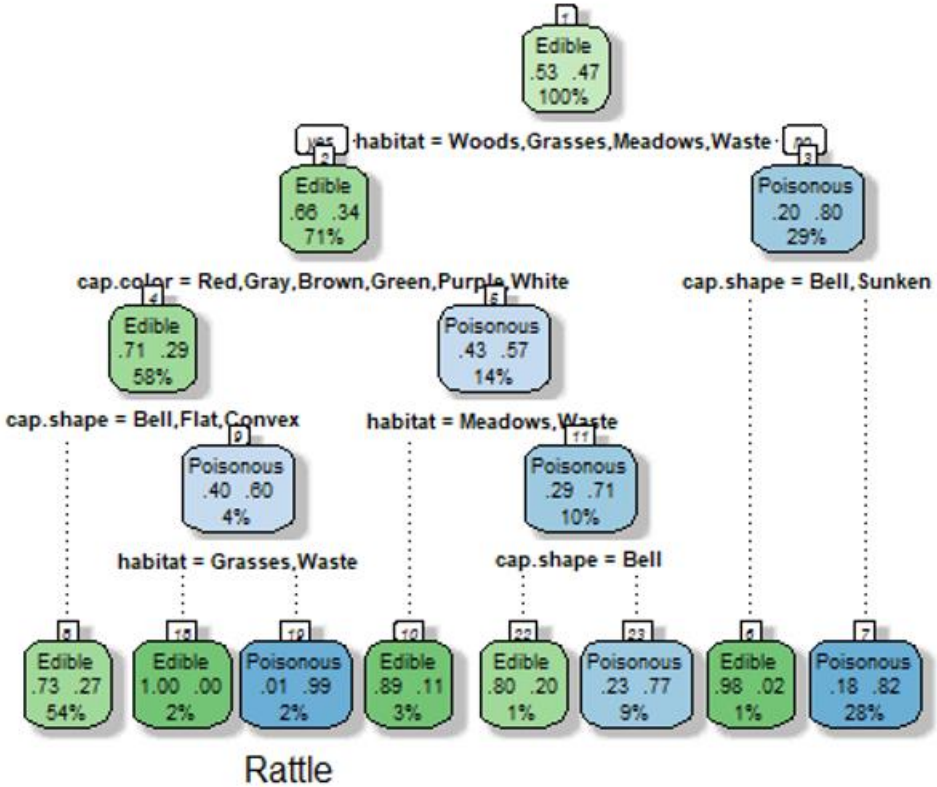
```
## Call:
## rpart(formula = class ~ cap.color + cap.shape + habitat, data = mush_train,
##     method = "class", minsplit = 2, minbucket = 1)
##   n= 5687
##
##           CP nsplit rel error    xerror       xstd
## 1 0.35653145      0 1.0000000 1.0000000 0.01401152
## 2 0.04558988      1 0.6434686 0.6434686 0.01291000
## 3 0.02512095      3 0.5522888 0.5522888 0.01232501
## 4 0.01898028      5 0.5020469 0.5020469 0.01193827
## 5 0.01414217      6 0.4830666 0.4901377 0.01183925
## 6 0.01000000      7 0.4689245 0.4759955 0.01171782
##
## Variable importance
##   habitat cap.shape cap.color
##        65        18        17
##
## Node number 1: 5687 observations,    complexity param=0.3565314
##   predicted class=Edible     expected loss=0.4724811  P(node) =1
##     class counts:  3000  2687
##    probabilities: 0.528 0.472
##   left son=2 (4065 obs) right son=3 (1622 obs)
##   Primary splits:
##       habitat   splits as  LLRLRRL,    improve=472.9990, (0 missing)
##       cap.shape splits as  LRRRLR,     improve=103.3253, (0 missing)
##       cap.color splits as  RLRLLRLLLR, improve=101.4764, (0 missing)
##   Surrogate splits:
##       cap.shape splits as  LRLRRL,     agree=0.733, adj=0.062, (0 split)
##       cap.color splits as  LRLLLLLLLL, agree=0.717, adj=0.009, (0 split)
##
## Node number 2: 4065 observations,    complexity param=0.04558988
##   predicted class=Edible     expected loss=0.3436654  P(node) =0.7147881
##     class counts:  2668  1397
##    probabilities: 0.656 0.344
##   left son=4 (3296 obs) right son=5 (769 obs)
##   Primary splits:
##       cap.color splits as  RRLLLRLLLR, improve=96.80171, (0 missing)
##       habitat   splits as  RR-L--L,    improve=49.02731, (0 missing)
##       cap.shape splits as  L-RR-R,     improve=33.38058, (0 missing)
```

```
##   Surrogate splits:
##       habitat splits as  LL-R--R, agree=0.815, adj=0.023, (0 split)
##
## Node number 3: 1622 observations,    complexity param=0.01898028
##   predicted class=Poisonous  expected loss=0.2046856  P(node) =0.2852119
##     class counts:   332  1290
##    probabilities: 0.205 0.795
##   left son=6 (53 obs) right son=7 (1569 obs)
##   Primary splits:
##       cap.shape splits as  LRRRLR,     improve=66.06278, (0 missing)
##       cap.color splits as  RLRRLL--RR, improve=65.12134, (0 missing)
##       habitat   splits as  --L-RL-,    improve=18.53685, (0 missing)
##
## Node number 4: 3296 observations,    complexity param=0.02512095
##   predicted class=Edible     expected loss=0.2909587  P(node) =0.5795674
##     class counts:  2337   959
##    probabilities: 0.709 0.291
##   left son=8 (3069 obs) right son=9 (227 obs)
##   Primary splits:
##       cap.shape splits as  L-LR-L,     improve=46.30195, (0 missing)
##       habitat   splits as  RR-L--L,    improve=18.52762, (0 missing)
##       cap.color splits as  --RRL-LLL-, improve=14.05372, (0 missing)
##
## Node number 5: 769 observations,    complexity param=0.04558988
##   predicted class=Poisonous  expected loss=0.4304291  P(node) =0.1352207
##     class counts:   331   438
##    probabilities: 0.430 0.570
##   left son=10 (178 obs) right son=11 (591 obs)
##   Primary splits:
##       habitat   splits as  RR-L--L,    improve=96.832870, (0 missing)
##       cap.shape splits as  L-RL-R,     improve=58.729050, (0 missing)
##       cap.color splits as  RR---R---L, improve= 3.165144, (0 missing)
##   Surrogate splits:
##       cap.shape splits as  R-RL-R, agree=0.793, adj=0.107, (0 split)
##
## Node number 6: 53 observations
##   predicted class=Edible     expected loss=0.01886792  P(node) =0.009319501
##     class counts:    52     1
##    probabilities: 0.981 0.019
##
## Node number 7: 1569 observations
```

```
##   predicted class=Poisonous  expected loss=0.1784576  P(node) =0.2758924
##     class counts:   280  1289
##    probabilities: 0.178 0.822
##
## Node number 8: 3069 observations
##   predicted class=Edible     expected loss=0.2681655  P(node) =0.5396518
##     class counts:  2246   823
##    probabilities: 0.732 0.268
##
## Node number 9: 227 observations,    complexity param=0.02512095
##   predicted class=Poisonous  expected loss=0.4008811  P(node) =0.0399156
##     class counts:    91   136
##    probabilities: 0.401 0.599
##   left son=18 (90 obs) right son=19 (137 obs)
##   Primary splits:
##       habitat   splits as  RL----L,    improve=107.05420, (0 missing)
##       cap.color splits as  --RLR---L-, improve= 69.69374, (0 missing)
##   Surrogate splits:
##       cap.color splits as  --RLR---L-, agree=0.903, adj=0.756, (0 split)
##
## Node number 10: 178 observations
##   predicted class=Edible     expected loss=0.1123596  P(node) =0.03129945
##     class counts:   158    20
##    probabilities: 0.888 0.112
##
## Node number 11: 591 observations,    complexity param=0.01414217
##   predicted class=Poisonous  expected loss=0.2927242  P(node) =0.1039212
##     class counts:   173   418
##    probabilities: 0.293 0.707
##   left son=22 (64 obs) right son=23 (527 obs)
##   Primary splits:
##       cap.shape splits as  L-RR-R,     improve=36.48445, (0 missing)
##       cap.color splits as  RR---R---L, improve=23.42159, (0 missing)
##       habitat   splits as  RL-----,    improve=16.80623, (0 missing)
##
## Node number 18: 90 observations
##   predicted class=Edible     expected loss=0  P(node) =0.01582557
##     class counts:    90     0
##    probabilities: 1.000 0.000
##
## Node number 19: 137 observations
```

```
##   predicted class=Poisonous  expected loss=0.00729927  P(node) =0.02409003
##     class counts:     1   136
##    probabilities: 0.007 0.993
##
## Node number 22: 64 observations
##   predicted class=Edible     expected loss=0.203125  P(node) =0.01125374
##     class counts:    51    13
##    probabilities: 0.797 0.203
##
## Node number 23: 527 observations
##   predicted class=Poisonous  expected loss=0.2314991  P(node) =0.09266749
##     class counts:   122   405
##    probabilities: 0.231 0.769
```

```r
fancyRpartPlot(model2) # Visual of tree
```



```r
predicted=predict(model2,mush_test_nolabels,type="class")
Results<-data.frame(predicted=predicted,Actual=TestClassLabels)
(table(Results))
```

```
##            Actual
## predicted   Edible Poisonous
##   Edible      1059       407
##   Poisonous    149       822
```

```r
confusionMatrix(predicted,TestClassLabels)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Edible Poisonous
##   Edible      1059       407
##   Poisonous    149       822
##
##                Accuracy : 0.7719
##                  95% CI : (0.7547, 0.7884)
##     No Information Rate : 0.5043
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5445
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8767
##             Specificity : 0.6688
##          Pos Pred Value : 0.7224
##          Neg Pred Value : 0.8465
##              Prevalence : 0.4957
##          Detection Rate : 0.4346
##    Detection Prevalence : 0.6016
##       Balanced Accuracy : 0.7727
##
##        'Positive' Class : Edible
##
```

```r
#model <- rpart( class ~ cap.shape, data=mush_train,method="class",minsplit = 2, minbucket = 1)
#model <- rpart( class ~ habitat, data=mush_train,method="class",minsplit = 2, minbucket = 1)
```

```r
# Another Sample method to validate our results from previous Model
every7_indexes<-seq(1,nrow(m2),7)
mush_Df_sampletest=m2[every7_indexes, ]
mush_Df_sampletrain=m2[-every7_indexes, ]
str(mush_Df_sampletrain)
```

```
## 'data.frame':    6963 obs. of  22 variables:
##  $ class                   : Factor w/ 2 levels "Edible","Poisonous": 1 1 2 1 1 1 2 1 1 1 ...
##  $ cap.shape               : Factor w/ 6 levels "Bell","Conical",..: 6 1 6 6 6 1 6 1 6 6 ...
##  $ cap.surface             : Factor w/ 4 levels "Fibrous","Grooves",..: 3 3 4 3 4 3 4 3 4 4 ...
##  $ cap.color               : Factor w/ 10 levels "Buff","Cinnamon",..: 10 9 9 4 10 9 9 10 10 10 ...
##  $ bruises                 : Factor w/ 2 levels "No","Bruises": 2 2 2 1 2 2 2 2 2 2 ...
##  $ odor                    : Factor w/ 9 levels "Almond","Creosote",..: 1 4 7 6 1 1 7 1 4 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 2 1 1 1 1 1 1 ...
##  $ gill.size               : Factor w/ 2 levels "Broad","Narrow": 1 1 2 1 1 1 2 1 1 1 ...
##  $ gill.color              : Factor w/ 12 levels "Buff","Red","Gray",..: 5 6 6 5 6 3 8 3 3 6 ...
##  $ stalk.shape             : Factor w/ 2 levels "Enlarging","Tapering": 1 1 1 2 1 1 1 1 1 1 ...
##  $ stalk.root              : Factor w/ 5 levels "?","Bulbous",..: 3 3 4 4 3 3 4 3 3 3 ...
##  $ stalk.surface.above.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.surface.below.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.color.above.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ stalk.color.below.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ 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 1 5 5 5 5 5 5 ...
##  $ spore.print.color       : Factor w/ 9 levels "Buff","Chocolate",..: 4 4 3 4 3 3 3 3 4 3 ...
##  $ population              : Factor w/ 6 levels "Abundant","Clustered",..: 3 3 4 1 3 3 5 4 3 4 ...
##  $ habitat                 : Factor w/ 7 levels "Woods","Grasses",..: 2 4 6 2 2 4 2 4 2 4 ...
```

```r
dim(mush_Df_sampletrain)
```

```
## [1] 6963   22
```

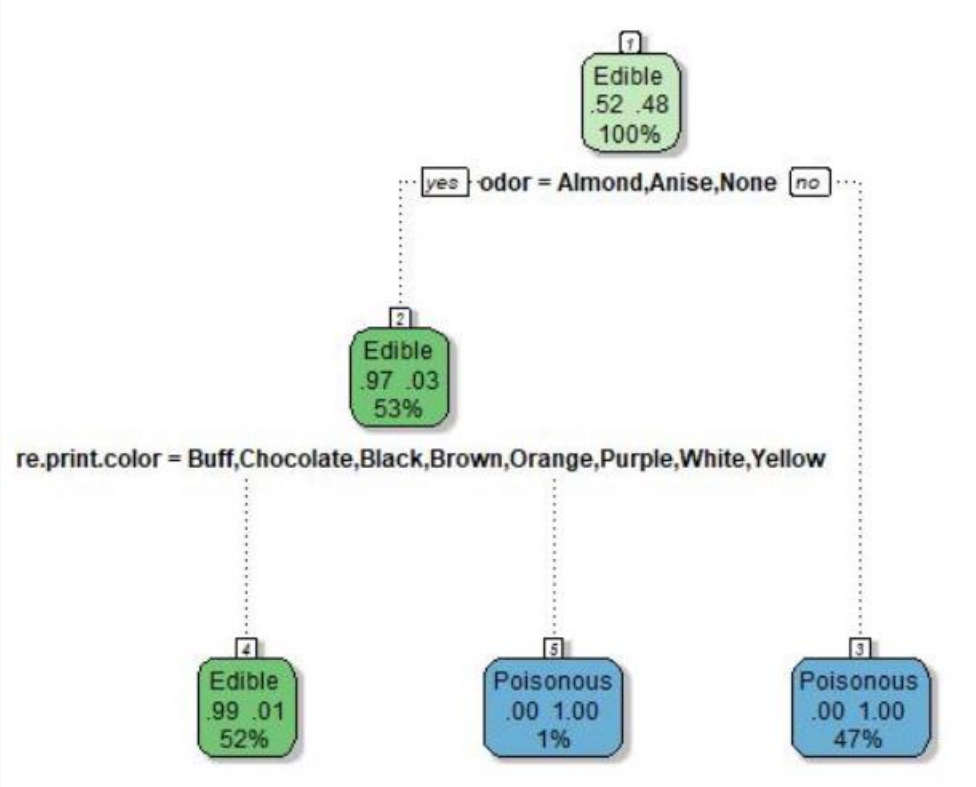
```r
mush_test_labels<-mush_Df_sampletest$class
mush_Df_sampletest1<-mush_Df_sampletest[-c(1)]

head(mush_test_labels)
```

```
## [1] Poisonous Edible    Edible    Poisonous Edible    Edible
## Levels: Edible Poisonous
```

```r
TestClassLabels1<-mush_test_labels
dim(mush_Df_sampletest1)

## [1] 1161   21

model_new <- rpart( class ~ ., data=mush_Df_sampletrain,method="class")
#model_new <- rpart( class ~ habitat, data=mush_Df_sampletrain,method="class",minsplit = 2, minbucket = 1)
summary(model)

## Call:
## rpart(formula = class ~ ., data = mush_train, method = "class",
##     parms = list(split = "gini"))
##   n= 5687
##
##           CP nsplit  rel error     xerror        xstd
## 1 0.96836621      0 1.00000000 1.00000000 0.014011518
## 2 0.02046892      1 0.03163379 0.03163379 0.003405428
## 3 0.01000000      2 0.01116487 0.01116487 0.002033033
##
## Variable importance
##                     odor        spore.print.color               gill.color
##                       26                       19                       16
## stalk.surface.above.ring stalk.surface.below.ring                ring.type
##                       14                       13                       13
##
## Node number 1: 5687 observations,    complexity param=0.9683662
##   predicted class=Edible     expected loss=0.4724811  P(node) =1
##     class counts:  3000  2687
##    probabilities: 0.528 0.472
##   left son=2 (3085 obs) right son=3 (2602 obs)
##   Primary splits:
##       odor                     splits as  LRRLRLRRR,    improve=2669.5710, (0 missing)
##       spore.print.color        splits as  LRLLLRLRL,    improve=1513.6540, (0 missing)
##       gill.color               splits as  RLRRLLLLRLLL, improve=1105.0000, (0 missing)
##       stalk.surface.above.ring splits as  LRLL,         improve= 977.3323, (0 missing)
##       stalk.surface.below.ring splits as  LRLL,         improve= 914.1120, (0 missing)
##   Surrogate splits:
##       spore.print.color        splits as  LRLLLLLRL,    agree=0.860, adj=0.693, (0 split)
##       gill.color               splits as  RLRRLLLLLLLL, agree=0.815, adj=0.595, (0 split)
##       stalk.surface.above.ring splits as  LRLL,         agree=0.784, adj=0.528, (0 split)
##       stalk.surface.below.ring splits as  LRLL,         agree=0.781, adj=0.521, (0 split)
##       ring.type                splits as  RLRRL,        agree=0.777, adj=0.512, (0 split)
```

```
##
## Node number 2: 3085 observations,    complexity param=0.02046892
##   predicted class=Edible     expected loss=0.02755267  P(node) =0.5424653
##     class counts:  3000    85
##    probabilities: 0.972 0.028
##   left son=4 (3030 obs) right son=5 (55 obs)
##   Primary splits:
##       spore.print.color      splits as  LLLLLRLLL,    improve=105.910100, (0 missing)
##       gill.color             splits as  -LLLLLLLRLLL, improve= 39.989670, (0 missing)
##       stalk.color.below.ring splits as  --LLLLLLR,    improve= 32.330390, (0 missing)
##       cap.color              splits as  RLLLLRLLLL,   improve= 21.700640, (0 missing)
##       ring.number            splits as  -LR,          improve=  9.684872, (0 missing)
##   Surrogate splits:
##       gill.color splits as  -LLLLLLLRLLL, agree=0.989, adj=0.382, (0 split)
##
## Node number 3: 2602 observations
##   predicted class=Poisonous  expected loss=0  P(node) =0.4575347
##     class counts:     0  2602
##    probabilities: 0.000 1.000
##
## Node number 4: 3030 observations
##   predicted class=Edible     expected loss=0.00990099  P(node) =0.5327941
##     class counts:  3000    30
##    probabilities: 0.990 0.010
##
## Node number 5: 55 observations
##   predicted class=Poisonous  expected loss=0  P(node) =0.00967118
##     class counts:     0    55
##    probabilities: 0.000 1.000
```

```r
jpeg("DecisionTree_Mushroom_Sample2.jpg")
fancyRpartPlot(model_new)  # Visual of tree
dev.off()
```




```
## png
##   2
```

```r
# Do the Prediction using Sample 2
predicted=predict(model_new,mush_Df_sampletest1,type="class")
Results<-data.frame(predicted=predicted,Actual=TestClassLabels1)
(table(Results))
```

```
##            Actual
## predicted   Edible Poisonous
##   Edible       597         4
##   Poisonous      0       560
```

```r
# Calculate Accuracy using the ConfusionMatrix. Accuracy comes in at 99%
confusionMatrix(predicted,TestClassLabels1)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Edible Poisonous
##   Edible       597         4
##   Poisonous      0       560
##
##                Accuracy : 0.9966
##                  95% CI : (0.9912, 0.9991)
##     No Information Rate : 0.5142
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9931
##
##  Mcnemar's Test P-Value : 0.1336
##
##             Sensitivity : 1.0000
##             Specificity : 0.9929
##          Pos Pred Value : 0.9933
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5142
##          Detection Rate : 0.5142
##    Detection Prevalence : 0.5177
##       Balanced Accuracy : 0.9965
##
##        'Positive' Class : Edible
```

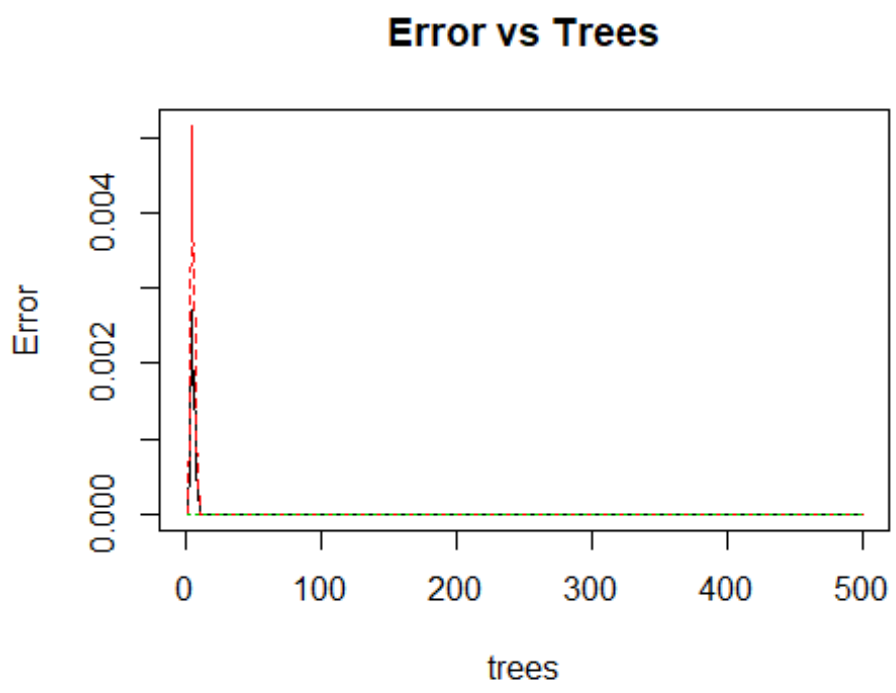
```r
length(TestClassLabels1)
```

```
## [1] 1161
```

```r
#==============================================================================
# Random Forest to predict the importance of Variables and avoid overfitting (Baskar Dakshin)
#==============================================================================

#install.packages("randomForest")
library(randomForest)

# Default model with ntree=500
set.seed(100)
head(mush_train)

##          class cap.shape cap.surface cap.color bruises  odor
## 6538 Poisonous      Flat       Scaly     Brown      No Spicy
## 2605    Edible      Flat     Fibrous     Brown Bruises  None
## 1211    Edible    Convex      Smooth     Brown      No  None
## 1925    Edible    Convex       Scaly       Red Bruises  None
## 6757 Poisonous      Flat      Smooth     Brown      No  Foul
## 5770    Edible   Knobbed       Scaly      Buff Bruises  None
##      gill.attachment gill.spacing gill.size gill.color stalk.shape
## 6538            Free        Close    Narrow       Buff    Tapering
## 2605            Free        Close     Broad      Brown    Tapering
## 1211            Free      Crowded     Broad       Pink    Tapering
## 1925            Free        Close     Broad      White    Tapering
## 6757            Free        Close    Narrow       Buff    Tapering
## 5770            Free        Close     Broad      White   Enlarging
##      stalk.root stalk.surface.above.ring stalk.surface.below.ring
## 6538          ?                   Smooth                   Smooth
## 2605    Bulbous                   Smooth                   Smooth
## 1211      Equal                   Smooth                  fibrous
## 1925    Bulbous                   Smooth                   Smooth
## 6757          ?                   Smooth                    Silky
## 5770          ?                   Smooth                   Smooth
##      stalk.color.above.ring stalk.color.below.ring veil.color ring.number
## 6538                  White                   Pink      White         One
## 2605                   Gray                  White      White         One
## 1211                  White                  White      White         One
## 1925                   Gray                  White      White         One
## 6757                   Pink                  White      White         One
## 5770                  White                  White      White         Two
##       ring.type spore.print.color population habitat
## 6538 Evanescent             White    Several  Leaves
## 2605    Pendant             Brown    Several   Woods
```

```
## 1211 Evanescent             Black   Abundant Grasses
## 1925    Pendant             Brown    Several   Woods
## 6757 Evanescent             White    Several  Leaves
## 5770 Evanescent             White  Clustered   Waste
```

```
rf_model1 = randomForest(class ~ .,
                         data = mush_train,importance=TRUE)


plot(rf_model1,main='Error vs Trees')
```




```
# Evaluate the performance of the model
predict_rf <- predict(rf_model1, newdata = mush_test_nolabels)
confusionMatrix(predict_rf, TestClassLabels)
```

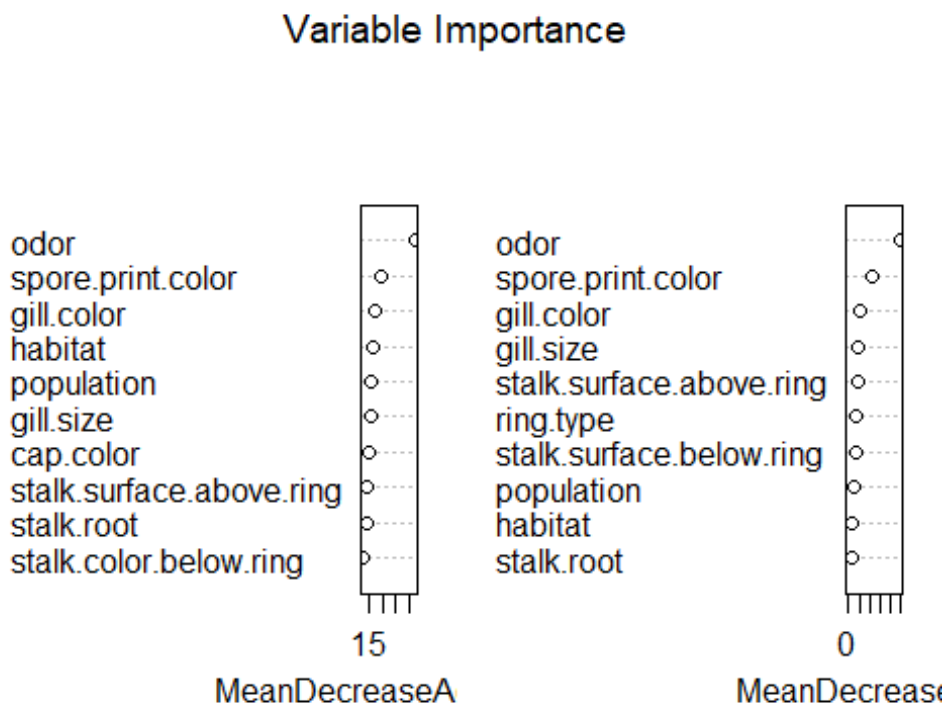
```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Edible Poisonous
##   Edible      1208         0
##   Poisonous      0      1229
##
##                Accuracy : 1
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.5043
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4957
##          Detection Rate : 0.4957
##    Detection Prevalence : 0.4957
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : Edible
##
```

```r
# Plotting the Variables according to its Importance
print(rf_model1)
```

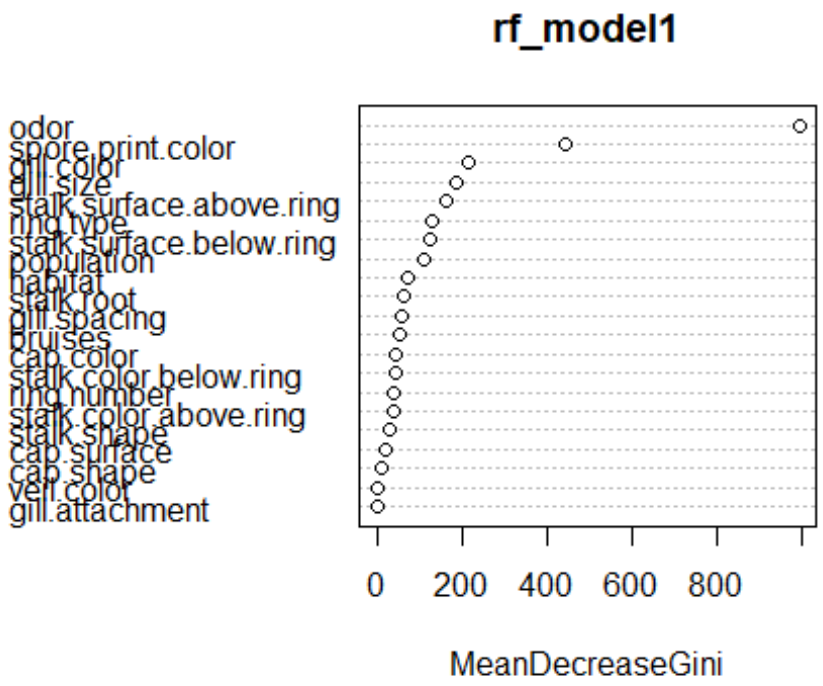
```
##
## Call:
##  randomForest(formula = class ~ ., data = mush_train, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 0%
```

```
## Confusion matrix:
##           Edible Poisonous class.error
## Edible      3000         0           0
## Poisonous      0      2687           0
```

```r
varImpPlot(rf_model1,
           sort = T,
           n.var = 10,
           main = "Variable Importance")
```



```r
varImpPlot(rf_model1,type=2)
```




```r
# Create a dataframe with Variable Importance
var.imp = data.frame(importance(rf_model1,type=2))

# List Variables according to its importance derived from Gini MeanDecrease
var.imp$Variables = row.names(var.imp)
print(var.imp[order(var.imp$MeanDecreaseGini,decreasing = T),])

##                          MeanDecreaseGini                Variables
## odor                           992.132028                     odor
## spore.print.color              440.916978        spore.print.color
## gill.color                     213.891799               gill.color
## gill.size                      184.303537                gill.size
## stalk.surface.above.ring       162.201664 stalk.surface.above.ring
## ring.type                      127.318753                ring.type
## stalk.surface.below.ring       122.329060 stalk.surface.below.ring
## population                     109.036106               population
```

```
## habitat                          73.384490                  habitat
## stalk.root                       63.090165               stalk.root
## gill.spacing                     56.872692             gill.spacing
## bruises                          53.628659                  bruises
## cap.color                        44.466102                cap.color
## stalk.color.below.ring           42.263481   stalk.color.below.ring
## ring.number                      40.907753              ring.number
## stalk.color.above.ring           38.687930   stalk.color.above.ring
## stalk.shape                      30.471673              stalk.shape
## cap.surface                      19.064312              cap.surface
## cap.shape                         8.196628                cap.shape
## veil.color                        2.151543               veil.color
## gill.attachment                   1.414093          gill.attachment
```

```r
# Evaluate the performance of the model1
predict_rf <- predict(rf_model1, newdata = mush_test_nolabels)
confusionMatrix(predict_rf, TestClassLabels)
```

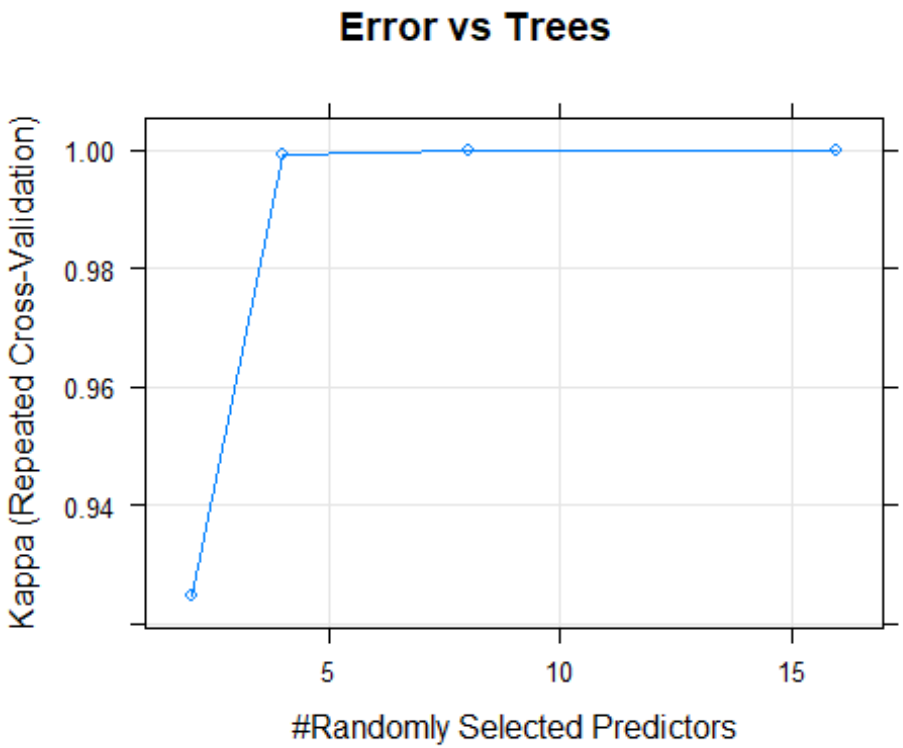
```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Edible Poisonous
##   Edible      1208         0
##   Poisonous      0      1229
##
##                Accuracy : 1
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.5043
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4957
##          Detection Rate : 0.4957
##    Detection Prevalence : 0.4957
##       Balanced Accuracy : 1.0000
```

```
##
##        'Positive' Class : Edible
##
```

```r
# Creating a RF model with Different mtry and ntrees
# This takes a while to run (~15 min) as we are tuning the model with different mtry values
ctrl <- trainControl(method = "repeatedcv", number = 5, repeats = 3)
grid_rf <- expand.grid(.mtry = c(2,4,8,16))
# Kappa was used to select the optimal model using the largest value.
# The final value used for the model was mtry = 4.
rf_model2 <- train(class ~ ., data = mush_train, method = "rf", metric = "Kappa", trControl = ctrl, tuneGrid = grid_rf)
rf_model2
```

```
## Random Forest
##
## 5687 samples
##   21 predictor
##    2 classes: 'Edible', 'Poisonous'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 4549, 4550, 4550, 4549, 4550, 4549, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9625472  0.9245504
##    4    0.9995897  0.9991768
##    8    1.0000000  1.0000000
##   16    1.0000000  1.0000000
##
## Kappa was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 8.
```

```r
plot(rf_model2,main='Error vs Trees')
```




```r
print(rf_model2)

## Random Forest
## 5687 samples
##   21 predictor
##    2 classes: 'Edible', 'Poisonous'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 4549, 4550, 4550, 4549, 4550, 4549, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9625472  0.9245504
##    4    0.9995897  0.9991768
##    8    1.0000000  1.0000000
##   16    1.0000000  1.0000000
```

```
## Kappa was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 8.
```

```r
# Do the prediction
rfPredict=predict(rf_model2,mush_test_nolabels)
Results<-data.frame(predicted=rfPredict,Actual=TestClassLabels)
(table(Results))
```

```
##            Actual
## predicted   Edible Poisonous
##   Edible      1208         0
##   Poisonous      0      1229
```

```r
# Calculate Accuracy using the ConfusionMatrix: Accuracy comes in at 99%
# Random Forest Accuracy comes in at 100% compared to Decision Tree Accuracy of 99.38%
confusionMatrix(rfPredict,TestClassLabels)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Edible Poisonous
##   Edible      1208         0
##   Poisonous      0      1229
##
##                Accuracy : 1
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.5043
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4957
##          Detection Rate : 0.4957
##    Detection Prevalence : 0.4957
##       Balanced Accuracy : 1.0000
##
```

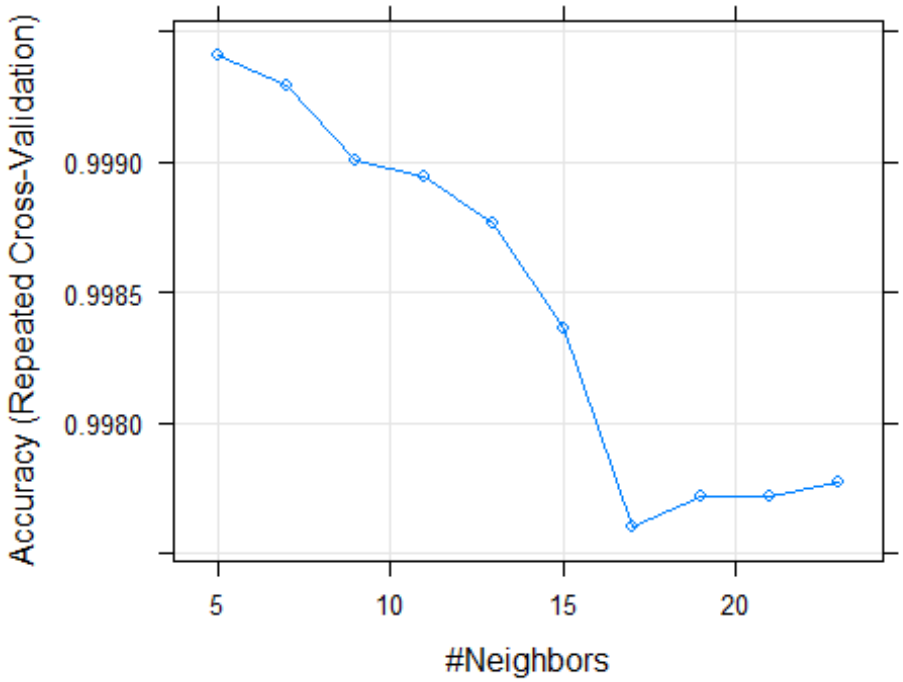
```
##        'Positive' Class : Edible


#==============================================================================
# K-Nearest Neighbours (Baskar Dakshin)
#==============================================================================
#install.packages("class")
library(class)
#install.packages("gmodels")
library(gmodels)
#install.packages("tictoc")
library(tictoc)

set.seed(101)
tic()

#knn_model <- knn(train=mush_train, test=mushroom,cl=TrainClassLabels,k=11)  # Create KNN model

# Training the Knn model
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3,classProbs = TRUE)
# Training the model -> Will run for 302.954 sec
knn_fit <- train(class~., data=mush_train, method = "knn",
                 trControl=trctrl,
                 preProcess = c("center", "scale"),
                 tuneLength = 10)

toc()

## 383.01 sec elapsed

knn_fit

## k-Nearest Neighbors
##
## 5687 samples
##   21 predictor
##    2 classes: 'Edible', 'Poisonous'
##
## Pre-processing: centered (95), scaled (95)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 5118, 5118, 5118, 5118, 5119, 5118, ...
## Resampling results across tuning parameters:
```

```
##
##   k   Accuracy   Kappa
##    5  0.9994138  0.9988237
##    7  0.9992966  0.9985886
##    9  0.9990032  0.9979993
##   11  0.9989446  0.9978817
##   13  0.9987688  0.9975289
##   15  0.9983586  0.9967052
##   17  0.9975968  0.9951767
##   19  0.9977139  0.9954117
##   21  0.9977139  0.9954117
##   23  0.9977726  0.9955295
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

```r
# Accuracy was used to select the optimal model using the largest value.
# The final value used for the model was k = 7.
summary(knn_fit)
```

```
##             Length Class      Mode
## learn        2     -none-     list
## k            1     -none-     numeric
## theDots      0     -none-     list
## xNames      95     -none-     character
## problemType  1     -none-     character
## tuneValue    1     data.frame list
## obsLevels    2     -none-     character
## param        0     -none-     list
```

```r
plot(knn_fit)
```




```r
# Evaluate the performance of the model
knnPredict<-predict(knn_fit,mush_test_nolabels)




# How Accurately our model is working?
confusionMatrix(knnPredict,TestClassLabels)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction  Edible Poisonous
##   Edible      1208         1
##   Poisonous      0      1228
##
```

```
##                Accuracy : 0.9996
##                  95% CI : (0.9977, 1)
##     No Information Rate : 0.5043
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9992
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 1.0000
##             Specificity : 0.9992
##          Pos Pred Value : 0.9992
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4957
##          Detection Rate : 0.4957
##    Detection Prevalence : 0.4961
##       Balanced Accuracy : 0.9996
##
##        'Positive' Class : Edible
##
```

```r
#==============================================================================
#Support Vector Machines (Baskar)
#==============================================================================



# Create SVM Model with default Values and do the prediction
svm_model <- svm(class~., data=mush_train, type='C-classification', kernel='radial')
# We set the kernel to radial as this data set does not a have a linear plane that can be drawn
summary(svm_model)
```

```
##
## Call:
## svm(formula = class ~ ., data = mush_train, type = "C-classification",
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
```

```
##
## Number of Support Vectors:  516
##
##  ( 261 255 )
##
##
## Number of Classes:  2
##
## Levels:
##  Edible Poisonous
```

```r
test_svm <-predict(svm_model,mush_test_nolabels)  # Predicting with the new SVM model


mean(test_svm==TestClassLabels)  # Percentage of testset predicted correctly by svm
```

```
## [1] 0.9946656
```

```r
Results<-data.frame(predicted=test_svm,Actual=TestClassLabels)
table(Results)  # Confusion matrix of the predictions of the svm and the test data
```

```
##            Actual
## predicted   Edible Poisonous
##   Edible      1208        13
##   Poisonous      0      1216
```

```r
# Perform a SVM tune to get the optimal Cost and Gamma Values-10-fold cross validation
# (Takes a while to run)
svm_tune <- tune(svm, class~., data = mush_train,
                 kernel="radial", ranges=list(cost=10^(-1:2), gamma=c(.5,1,2)))
# Best Parameters:
# cost gamma
#  1   0.5
print(svm_tune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1   0.5
```

```
##
## - best performance: 0
```

```r
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3,classProbs = TRUE)

set.seed(101)
# Support Vector Machines with Linear Kernel and 10 Fold Cross-Validation
model_fit <- train(class~., data=mush_train, method = "svmLinear",
                   trControl=trctrl,
                   preProcess = c("center", "scale"),
                   tuneLength = 10)

print(model_fit);
```

```
## Support Vector Machines with Linear Kernel
##
## 5687 samples
##   21 predictor
##    2 classes: 'Edible', 'Poisonous'
##
## Pre-processing: centered (95), scaled (95)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 5118, 5118, 5118, 5118, 5119, 5118, ...
## Resampling results:
##
##   Accuracy  Kappa
##   1         1
##
## Tuning parameter 'C' was held constant at a value of 1
```

```r
# Classify from our reserved test set.
testing_set_predict = predict(model_fit, newdata = mush_test_nolabels);
# Verifying our model from the classifications.
table(testing_set_predict, TestClassLabels);
```

```
##                    TestClassLabels
## testing_set_predict Edible Poisonous
##           Edible      1208         0
##           Poisonous      0      1229
```

```r
confusionMatrix(testing_set_predict,TestClassLabels)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Edible Poisonous
##   Edible      1208         0
##   Poisonous      0      1229
##
##                Accuracy : 1
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.5043
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4957
##          Detection Rate : 0.4957
##    Detection Prevalence : 0.4957
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : Edible
##
```

```r
# Run the SVM Model one more time with tuned Gamma and Cost Values

svm_model1 <- svm(class~., data=mush_train, type='C-classification', kernel='radial',cost=1,gamma=0.5)


# Classify from our reserved test set.
testing_set_predict1 = predict(svm_model1, newdata = mush_test_nolabels);
```

```r
# Verifying our model from the classifications.
table(testing_set_predict1, TestClassLabels);
```

```
##                      TestClassLabels
## testing_set_predict1 Edible Poisonous
##            Edible      1208         0
##            Poisonous      0      1229
```

```r
confusionMatrix(testing_set_predict1,TestClassLabels)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Edible Poisonous
##   Edible      1208         0
##   Poisonous      0      1229
##
##                Accuracy : 1
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.5043
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4957
##          Detection Rate : 0.4957
##    Detection Prevalence : 0.4957
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : Edible
##
```

```r
#==============================================================================
# Clustering using K-Means,K-mode, Rock, etc. (Baskar Dakshin)
#==============================================================================

#install.packages("factoextra")
library(factoextra)  # for DBSCAN
#install.packages("klaR")
library(klaR)
#install.packages("cba")
library(cba)

# Reading the dataset again
filename="mushrooms.csv"
mushroomDf<- read.csv(filename, header = TRUE, na.strings = "NA")
any(is.na(mushroomDf))
```

```
## [1] FALSE
```

```r
# Removing the Class variable and Veil.type from the dataset
mushroomDf.torun <- subset(mushroomDf, select = -c(class, veil.type))



##########
# Clustering using k-means by one-hot encoding
# This is basically creating dummy variables for each value of the category, for all the variables.
mushroomDf.torun.ohe <- model.matrix(~.-1, data=mushroomDf.torun)

str(mushroomDf.torun.ohe)
```

```
##  num [1:8124, 1:96] 0 0 1 0 0 0 1 1 0 1 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:8124] "1" "2" "3" "4" ...
##   ..$ : chr [1:96] "cap.shapeb" "cap.shapec" "cap.shapef" "cap.shapek" ...
##  - attr(*, "assign")= int [1:96] 1 1 1 1 1 1 2 2 2 3 ...
##  - attr(*, "contrasts")=List of 21
##   ..$ cap.shape               : chr "contr.treatment"
##   ..$ cap.surface             : chr "contr.treatment"
##   ..$ cap.color               : chr "contr.treatment"
##   ..$ bruises                 : chr "contr.treatment"
##   ..$ odor                    : chr "contr.treatment"
##   ..$ gill.attachment         : chr "contr.treatment"
##   ..$ gill.spacing            : chr "contr.treatment"
```

```
##   ..$ gill.size               : chr "contr.treatment"
##   ..$ gill.color              : chr "contr.treatment"
##   ..$ stalk.shape             : chr "contr.treatment"
##   ..$ stalk.root              : chr "contr.treatment"
##   ..$ stalk.surface.above.ring: chr "contr.treatment"
##   ..$ stalk.surface.below.ring: chr "contr.treatment"
##   ..$ stalk.color.above.ring  : chr "contr.treatment"
##   ..$ stalk.color.below.ring  : chr "contr.treatment"
##   ..$ veil.color              : chr "contr.treatment"
##   ..$ ring.number             : chr "contr.treatment"
##   ..$ ring.type               : chr "contr.treatment"
##   ..$ spore.print.color       : chr "contr.treatment"
##   ..$ population              : chr "contr.treatment"
##   ..$ habitat                 : chr "contr.treatment"
```

```r
set.seed(20)  # For reproducibility
# Nstart = 50, indicates R will run 50 different random starting assignments
# and selects the lowest within cluster variation
result.kmean = kmeans(mushroomDf.torun.ohe, 2, nstart = 50, iter.max = 15)
#print(result.kmean)
result.kmean3 <- kmeans(mushroomDf.torun.ohe, centers = 3, nstart = 25)
result.kmean4 <- kmeans(mushroomDf.torun.ohe, centers = 4, nstart = 25)
result.kmean5 <- kmeans(mushroomDf.torun.ohe, centers = 5, nstart = 25)


# Purity of clustering is a simple measure of the accuracy, which is between 0 and 1.
# 0 indicates poor clustering, and 1 indicates perfect clustering

# Purity of Cluster with K=2
result.kmean.mm <- table(mushroomDf$class, result.kmean$cluster)
purity.kmean <- sum(apply(result.kmean.mm, 2, max)) / nrow(mushroomDf.torun)
purity.kmean
```

```
## [1] 0.8953717
```

```r
# Purity of Cluster with K=3
result.kmean3.mm <- table(mushroomDf$class, result.kmean3$cluster)
purity.kmean3 <- sum(apply(result.kmean3.mm, 2, max)) / nrow(mushroomDf.torun)
purity.kmean3
```

```
## [1] 0.8945101
```

```r
# Purity of Cluster with K=4
result.kmean4.mm <- table(mushroomDf$class, result.kmean4$cluster)
purity.kmean4 <- sum(apply(result.kmean4.mm, 2, max)) / nrow(mushroomDf.torun)
purity.kmean4
```

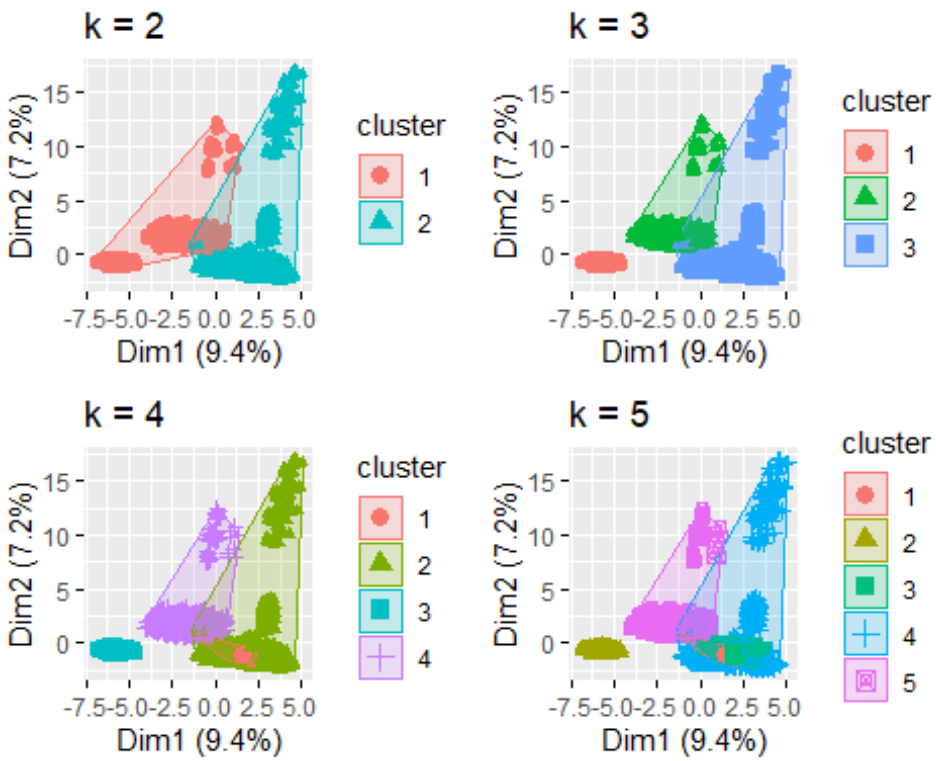
```
## [1] 0.8938946
```

```r
# Purity of Cluster with K=5
result.kmean5.mm <- table(mushroomDf$class, result.kmean5$cluster)
purity.kmean5 <- sum(apply(result.kmean5.mm, 2, max)) / nrow(mushroomDf.torun)
purity.kmean5
```

```
## [1] 0.8936484
```

```r
# Creating plots to compare between different K-Values
p1 <- fviz_cluster(result.kmean, geom = "point", data = mushroomDf.torun.ohe) + ggtitle("k = 2")
p2 <- fviz_cluster(result.kmean3, geom = "point",  data = mushroomDf.torun.ohe) + ggtitle("k = 3")
p3 <- fviz_cluster(result.kmean4, geom = "point",  data = mushroomDf.torun.ohe) + ggtitle("k = 4")
p4 <- fviz_cluster(result.kmean5, geom = "point",  data = mushroomDf.torun.ohe) + ggtitle("k = 5")

#install.packages("gridExtra")
library(gridExtra)
```

```r
grid.arrange(p1, p2, p3, p4, nrow = 2)
```




```r
##########
# Clustering using K-mode with different K-Values

set.seed(20)  # For reproducibility
# Nstart = 50, indicates R will run 50 different random starting assignments
# and selects the lowest within cluster variation
result.kmode <- kmodes(mushroomDf.torun.ohe, 2, iter.max = 50, weighted = FALSE)
#print(result.kmode)
result.kmode.mm <- table(mushroomDf$class, result.kmode$cluster)
result.kmode.mm

##
##        1    2
##   e 3200 1008
##   p 1932 1984
```

```r
purity.kmode <- sum(apply(result.kmode.mm, 2, max)) / nrow(mushroomDf.torun)
purity.kmode
```

```
## [1] 0.6381093
```

```r
result.kmode3 <- kmodes(mushroomDf.torun.ohe,3,iter.max = 50, weighted = FALSE)
#print(result.kmode3)
result.kmode3.mm <- table(mushroomDf$class, result.kmode3$cluster)
result.kmode3.mm
```

```
##
##        1    2    3
##   e 1248   30 2930
##   p  152 3064  700
```

```r
purity.kmode3 <- sum(apply(result.kmode3.mm, 2, max)) / nrow(mushroomDf.torun)
purity.kmode3
```

```
## [1] 0.8914328
```

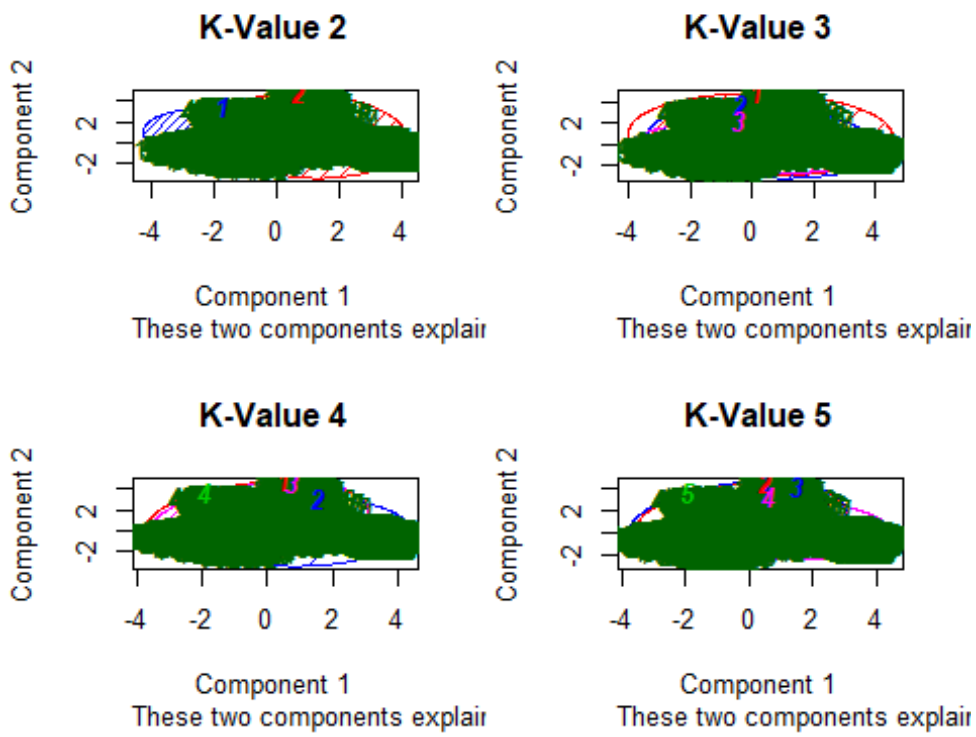
```r
result.kmode4 <- kmodes(mushroomDf.torun.ohe,4,iter.max = 50, weighted = FALSE)
#print(result.kmode4)
result.kmode4.mm <- table(mushroomDf$class, result.kmode4$cluster)
purity.kmode4 <- sum(apply(result.kmode4.mm, 2, max)) / nrow(mushroomDf.torun)
purity.kmode4
```

```
## [1] 0.8838011
```

```r
result.kmode5 <- kmodes(mushroomDf.torun.ohe,5,iter.max = 50, weighted = FALSE)
#print(result.kmode5)
result.kmode5.mm <- table(mushroomDf$class, result.kmode5$cluster)
purity.kmode5 <- sum(apply(result.kmode5.mm, 2, max)) / nrow(mushroomDf.torun)
purity.kmode5
```

```
## [1] 0.8893402
```

```r
# Plots to compare
par(mfrow=c(2,2))
clusplot(mushroomDf.torun, result.kmode$cluster, color=TRUE, shade=TRUE, labels=2, lines=0,main='K-Value 2')
clusplot(mushroomDf.torun, result.kmode3$cluster, color=TRUE, shade=TRUE, labels=2, lines=0,main='K-Value 3')
clusplot(mushroomDf.torun, result.kmode4$cluster, color=TRUE, shade=TRUE, labels=2, lines=0,main='K-Value 4')
clusplot(mushroomDf.torun, result.kmode5$cluster, color=TRUE, shade=TRUE, labels=2, lines=0,main='K-Value 5')
```




```
##########
# Clustering using Rock
mushroom.torun.binary <- as.dummy(mushroomDf.torun)
result.rock <-rockCluster(mushroom.torun.binary, n=5, theta=0.8)

## Clustering:
## computing distances ...
## computing links ...
## computing clusters ...
## rockMerge: terminated with 21 clusters

result.rock.mm<-table(mushroomDf$class, result.rock$cl)
purity.rock <- sum(apply(result.rock.mm, 2, max)) / nrow(mushroomDf.torun)
purity.rock

## [1] 0.9960611
```

```r
##########
# Compute hierarchical clustering
mushroomDf.torun2 <- subset(mushroomDf, select =
c(odor,stalk.color.below.ring,stalk.color.above.ring,habitat,gill.size,gill.color,population,ring.number))
head(mushroomDf.torun2)
```

```
##   odor stalk.color.below.ring stalk.color.above.ring habitat gill.size
## 1    p                      w                      w       u         n
## 2    a                      w                      w       g         b
## 3    l                      w                      w       m         b
## 4    p                      w                      w       u         n
## 5    n                      w                      w       g         b
## 6    a                      w                      w       g         b
##   gill.color population ring.number
## 1          k          s           o
## 2          k          n           o
## 3          n          n           o
## 4          n          s           o
## 5          k          a           o
## 6          n          n           o
```

```r
# Clustering using k-means by one-hot encoding

# This is basically creating dummy variables for each value of the category, for all the variables.
mushroomDf.torun.ohe2 <- model.matrix(~.-1, data=mushroomDf.torun2)
mushroomDf.torun.ohe3<- scale(mushroomDf.torun.ohe2)
head(mushroomDf.torun.ohe3)
```

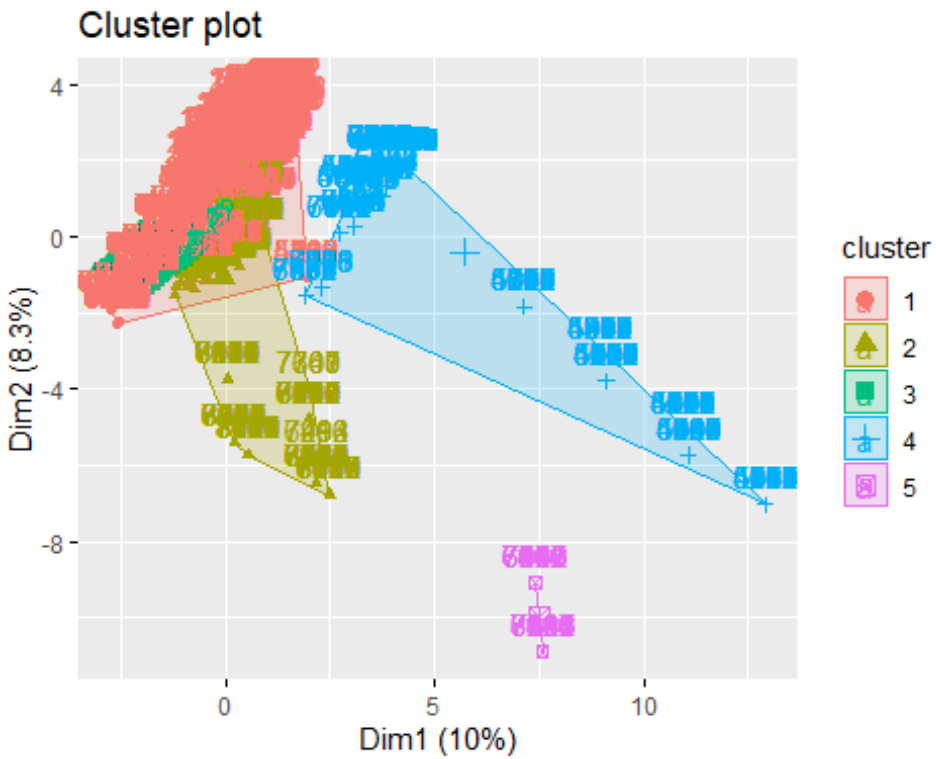
```
##        odora      odorc      odorf      odorl     odorm      odorn
## 1 -0.2275528 -0.1555724 -0.6017711 -0.2275528 -0.066712 -0.8760876
## 2  4.3940440 -0.1555724 -0.6017711 -0.2275528 -0.066712 -0.8760876
## 3 -0.2275528 -0.1555724 -0.6017711  4.3940440 -0.066712 -0.8760876
## 4 -0.2275528 -0.1555724 -0.6017711 -0.2275528 -0.066712 -0.8760876
## 5 -0.2275528 -0.1555724 -0.6017711 -0.2275528 -0.066712  1.1412978
## 6  4.3940440 -0.1555724 -0.6017711 -0.2275528 -0.066712 -0.8760876
##        odorp      odors      odory stalk.color.below.ringc
## 1  5.5435180 -0.2762286 -0.2762286               -0.066712
## 2 -0.1803687 -0.2762286 -0.2762286               -0.066712
## 3 -0.1803687 -0.2762286 -0.2762286               -0.066712
## 4  5.5435180 -0.2762286 -0.2762286               -0.066712
## 5 -0.1803687 -0.2762286 -0.2762286               -0.066712
## 6 -0.1803687 -0.2762286 -0.2762286               -0.066712
```

```
##   stalk.color.below.ringe stalk.color.below.ringg stalk.color.below.ringn
## 1              -0.1093466              -0.2762286              -0.2593336
## 2              -0.1093466              -0.2762286              -0.2593336
## 3              -0.1093466              -0.2762286              -0.2593336
## 4              -0.1093466              -0.2762286              -0.2593336
## 5              -0.1093466              -0.2762286              -0.2593336
## 6              -0.1093466              -0.2762286              -0.2593336
##   stalk.color.below.ringo stalk.color.below.ringp stalk.color.below.ringw
## 1              -0.1555724               -0.547163                0.9235785
## 2              -0.1555724               -0.547163                0.9235785
## 3              -0.1555724               -0.547163                0.9235785
## 4              -0.1555724               -0.547163                0.9235785
## 5              -0.1555724               -0.547163                0.9235785
## 6              -0.1555724               -0.547163                0.9235785
##   stalk.color.below.ringy stalk.color.above.ringc stalk.color.above.ringe
## 1             -0.05442976               -0.066712              -0.1093466
## 2             -0.05442976               -0.066712              -0.1093466
## 3             -0.05442976               -0.066712              -0.1093466
## 4             -0.05442976               -0.066712              -0.1093466
## 5             -0.05442976               -0.066712              -0.1093466
## 6             -0.05442976               -0.066712              -0.1093466
##   stalk.color.above.ringg stalk.color.above.ringn stalk.color.above.ringo
## 1              -0.2762286               -0.241571              -0.1555724
## 2              -0.2762286               -0.241571              -0.1555724
## 3              -0.2762286               -0.241571              -0.1555724
## 4              -0.2762286               -0.241571              -0.1555724
## 5              -0.2762286               -0.241571              -0.1555724
## 6              -0.2762286               -0.241571              -0.1555724
##   stalk.color.above.ringp stalk.color.above.ringw stalk.color.above.ringy
## 1               -0.547163               0.9054234             -0.03139404
## 2               -0.547163               0.9054234             -0.03139404
## 3               -0.547163               0.9054234             -0.03139404
## 4               -0.547163               0.9054234             -0.03139404
## 5               -0.547163               0.9054234             -0.03139404
## 6               -0.547163               0.9054234             -0.03139404
##     habitatg   habitatl  habitatm   habitatp   habitatu   habitatw
## 1 -0.5994944 -0.3377626 -0.193076 -0.4048168  4.5905874 -0.1555724
## 2  1.6678670 -0.3377626 -0.193076 -0.4048168 -0.2178102 -0.1555724
## 3 -0.5994944 -0.3377626  5.178669 -0.4048168 -0.2178102 -0.1555724
## 4 -0.5994944 -0.3377626 -0.193076 -0.4048168  4.5905874 -0.1555724
## 5  1.6678670 -0.3377626 -0.193076 -0.4048168 -0.2178102 -0.1555724
```

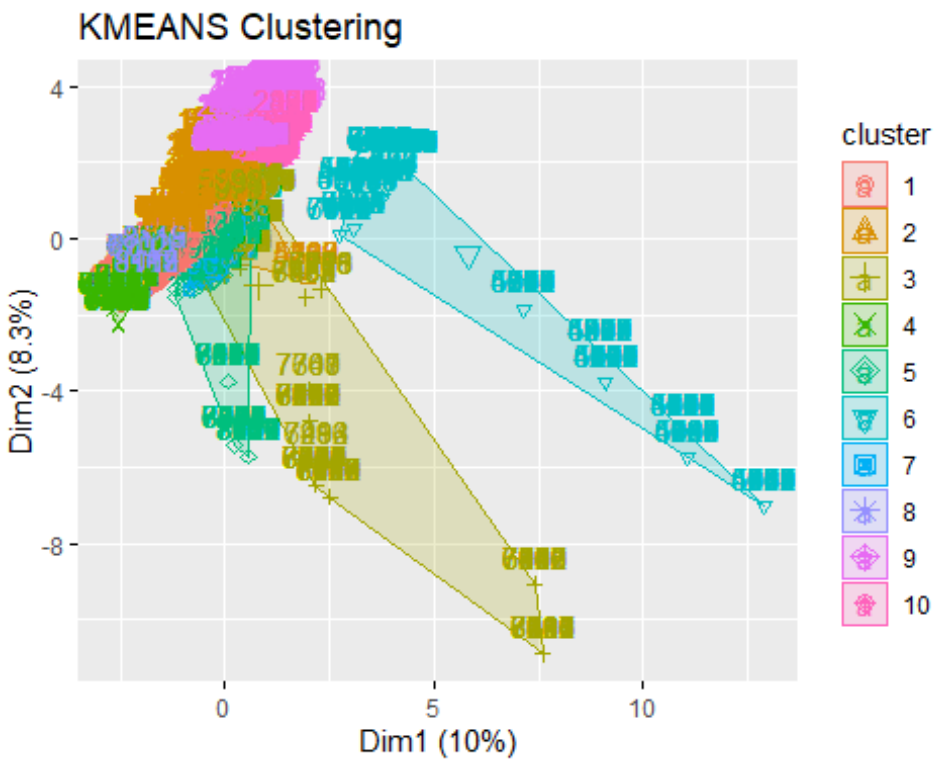
```
## 6  1.6678670 -0.3377626 -0.193076 -0.4048168 -0.2178102 -0.1555724
##   gill.sizen gill.colore gill.colorg gill.colorh gill.colork gill.colorn
## 1  1.4945907  -0.1093466  -0.3193666  -0.3146646   4.3484982  -0.3848221
## 2 -0.6689971  -0.1093466  -0.3193666  -0.3146646   4.3484982  -0.3848221
## 3 -0.6689971  -0.1093466  -0.3193666  -0.3146646  -0.2299361   2.5982835
## 4  1.4945907  -0.1093466  -0.3193666  -0.3146646  -0.2299361   2.5982835
## 5 -0.6689971  -0.1093466  -0.3193666  -0.3146646   4.3484982  -0.3848221
## 6 -0.6689971  -0.1093466  -0.3193666  -0.3146646  -0.2299361   2.5982835
##   gill.coloro gill.colorp gill.colorr gill.coloru gill.colorw gill.colory
## 1  -0.0891037  -0.4742807 -0.05442976  -0.2538848  -0.4166868  -0.1034305
## 2  -0.0891037  -0.4742807 -0.05442976  -0.2538848  -0.4166868  -0.1034305
## 3  -0.0891037  -0.4742807 -0.05442976  -0.2538848  -0.4166868  -0.1034305
## 4  -0.0891037  -0.4742807 -0.05442976  -0.2538848  -0.4166868  -0.1034305
## 5  -0.0891037  -0.4742807 -0.05442976  -0.2538848  -0.4166868  -0.1034305
## 6  -0.0891037  -0.4742807 -0.05442976  -0.2538848  -0.4166868  -0.1034305
##   populationc populationn populations populationv populationy ring.numbero
## 1  -0.2089832  -0.2275528    2.347113  -0.9945373   -0.516688    0.2914197
## 2  -0.2089832   4.3940440   -0.426003  -0.9945373   -0.516688    0.2914197
## 3  -0.2089832   4.3940440   -0.426003  -0.9945373   -0.516688    0.2914197
## 4  -0.2089832  -0.2275528    2.347113  -0.9945373   -0.516688    0.2914197
## 5  -0.2089832  -0.2275528   -0.426003  -0.9945373   -0.516688    0.2914197
## 6  -0.2089832   4.3940440   -0.426003  -0.9945373   -0.516688    0.2914197
##   ring.numbert
## 1   -0.2823739
## 2   -0.2823739
## 3   -0.2823739
## 4   -0.2823739
## 5   -0.2823739
## 6   -0.2823739
```

```r
# Dissimilarity matrix
d <- dist(mushroomDf.torun.ohe2, method = "euclidean")

# Hierarchical clustering using Complete Linkage
hc1 <- hclust(d, method = "complete" )
```

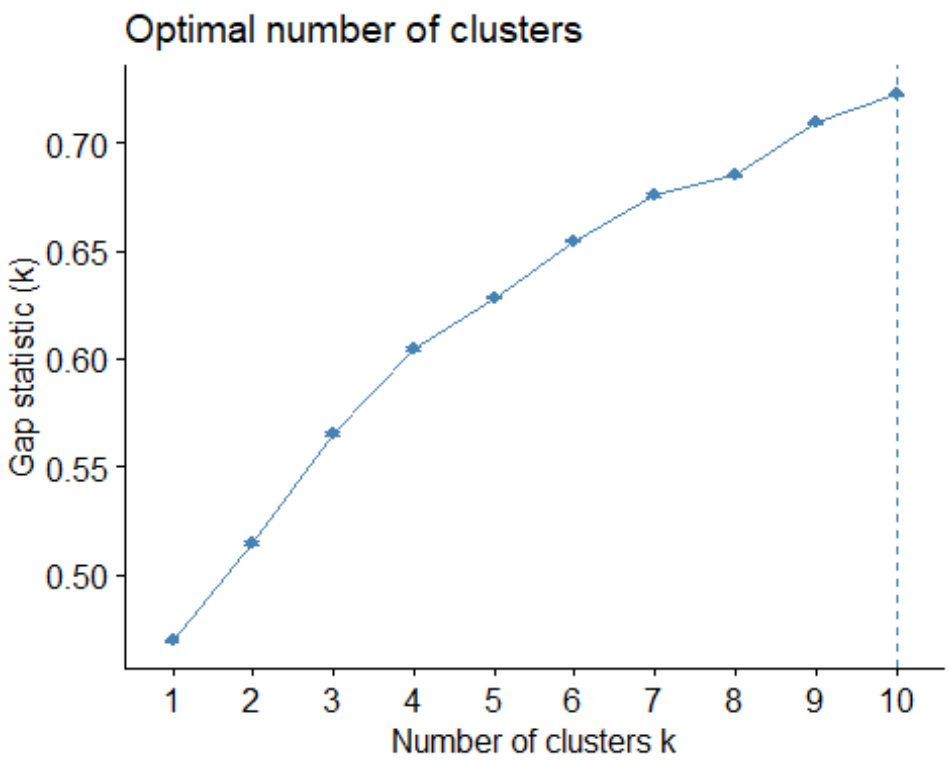
```r
# Cut tree into 4 groups
sub_grp <- cutree(hc1, k = 5)
table(sub_grp)

## sub_grp
##    1    2    3    4    5
## 4426 1766 1296  600   36

fviz_cluster(list(data = mushroomDf.torun.ohe2, cluster = sub_grp))
```



```r
##########
# ENHANCED K-MEANS
# Enhanced k-means clustering


res.km <- eclust(mushroomDf.torun.ohe2, "kmeans", nstart = 25,nboot=5)  # Takes 5 minutes to run
```

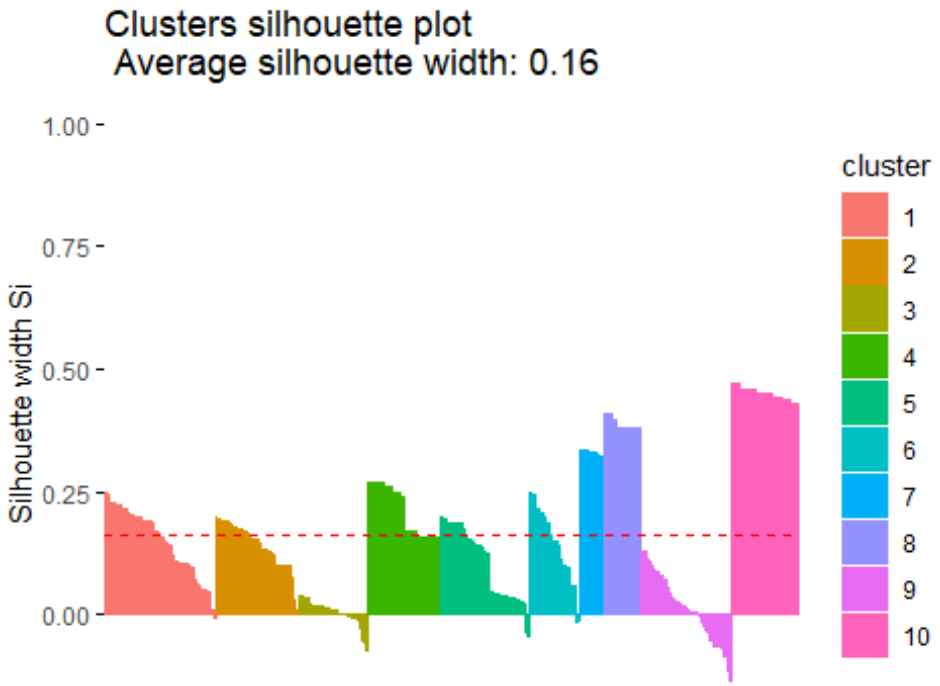


```r
# Gap statistic plot
fviz_gap_stat(res.km$gap_stat)
```

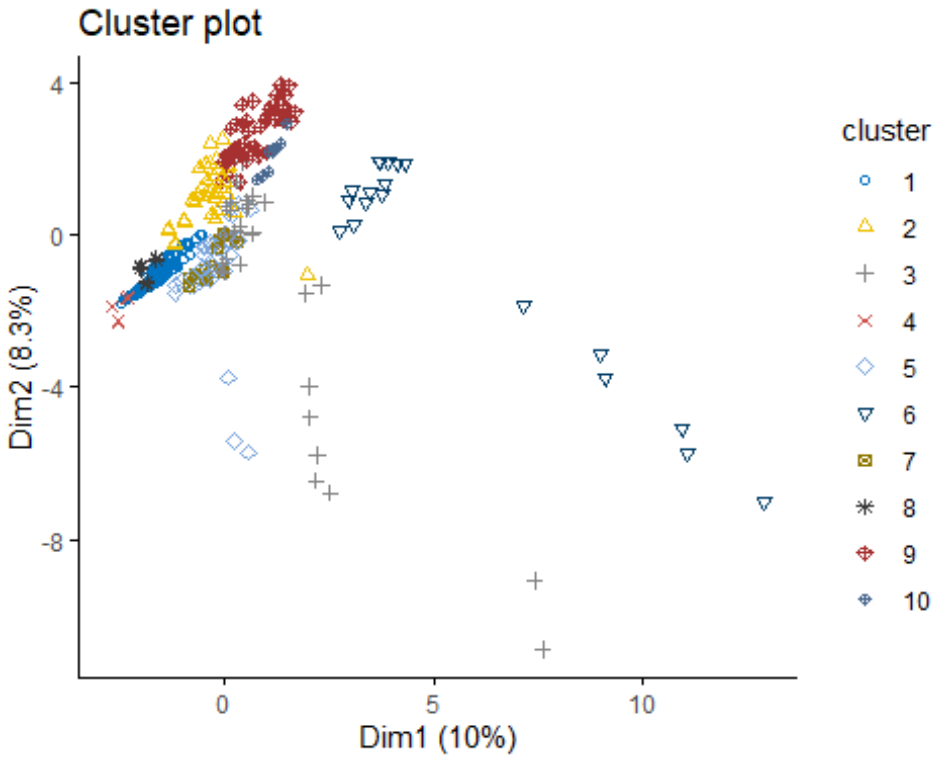


```r
# Silhouette plot
fviz_silhouette(res.km)
```

```
##    cluster size ave.sil.width
## 1        1 1296          0.15
## 2        2  976          0.14
## 3        3  804          0.00
## 4        4  864          0.21
## 5        5 1040          0.10
## 6        6  584          0.14
## 7        7  288          0.33
## 8        8  432          0.39
## 9        9 1072          0.01
## 10      10  768          0.45
```

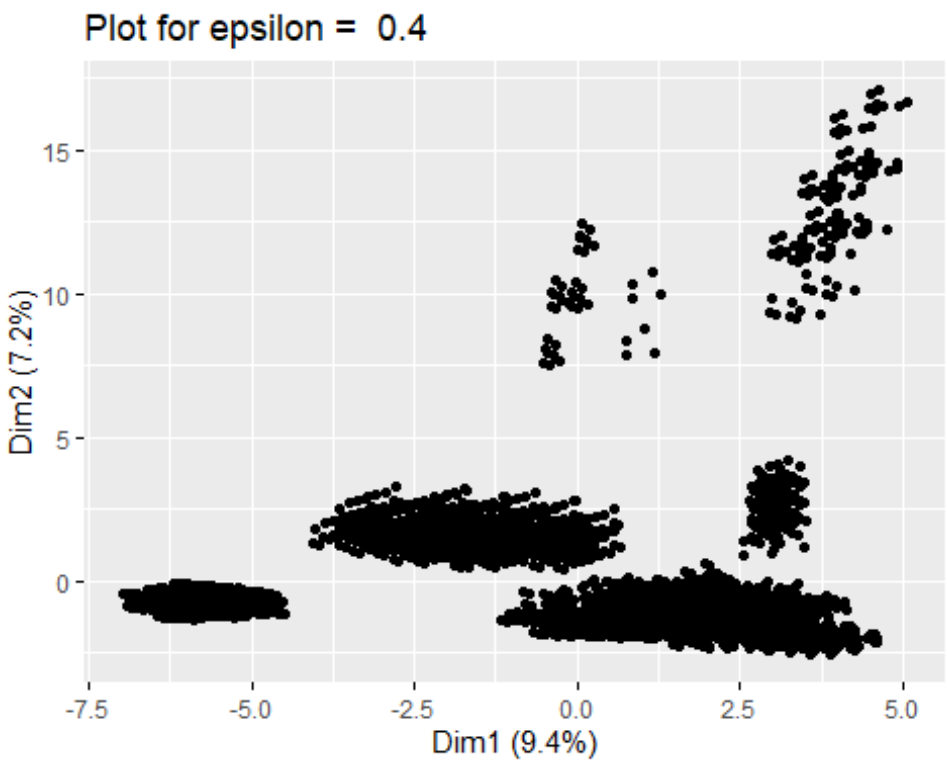


```r
# List of cluster assignments

fviz_cluster(res.km, df,  geom = "point",
             ellipse= FALSE, show.clust.cent = FALSE,
             palette = "jco", ggtheme = theme_classic())
```




```r
##########
# DBSCAN Density-based clustering with columns

#install.packages("dbscan")
library(dbscan)
df <- mushroomDf.torun.ohe

# Create a vector of epsilon values
epsilon_values <- c(1.8, 0.5, 0.4)

# Plot the distribution of distances
kNNdistplot(df, k = 5)
```

```r
# Plot lines at epsilon values
for (e in epsilon_values) {
  abline(h = e, col = "red")
}

for (e in epsilon_values) {
  db_clusters <- dbscan(df, eps=e, minPts=4)
  title <- paste("Plot for epsilon = ", e)
  g <- fviz_cluster(db_clusters, df, ellipse = TRUE, geom = "point",
                    main = title)
  print(g)
}
```



```r
#==============================================================================
# Naive Bayes (Lauren Foltz)
#==============================================================================

# Create another data set without stalk.root or veil.type

# Create training set
str(mush_train)  # Veiltype has already been removed for this data set
```

```
## 'data.frame':    5687 obs. of  22 variables:
##  $ class                   : Factor w/ 2 levels "Edible","Poisonous": 2 1 1 1 2 1 1 1 1 1 ...
##  $ cap.shape               : Factor w/ 6 levels "Bell","Conical",..: 3 3 6 6 3 4 3 6 6 6 ...
##  $ cap.surface             : Factor w/ 4 levels "Fibrous","Grooves",..: 4 1 3 4 3 4 4 1 3 1 ...
##  $ cap.color               : Factor w/ 10 levels "Buff","Cinnamon",..: 5 5 5 3 5 1 3 10 5 9 ...
##  $ bruises                 : Factor w/ 2 levels "No","Bruises": 1 2 1 2 1 2 2 2 1 1 ...
##  $ odor                    : Factor w/ 9 levels "Almond","Creosote",..: 8 6 6 6 3 6 6 4 6 6 ...
##  $ gill.attachment         : Factor w/ 2 levels "Attached","Free": 2 2 2 2 2 2 2 2 1 2 ...
##  $ gill.spacing            : Factor w/ 2 levels "Close","Crowded": 1 1 2 1 1 1 1 2 1 2 ...
##  $ gill.size               : Factor w/ 2 levels "Broad","Narrow": 2 1 1 1 2 1 1 2 1 1 ...
##  $ gill.color              : Factor w/ 12 levels "Buff","Red","Gray",..: 1 6 8 11 1 11 10 11 7 8 ...
##  $ stalk.shape             : Factor w/ 2 levels "Enlarging","Tapering": 2 2 2 2 2 1 2 2 1 2 ...
##  $ stalk.root              : Factor w/ 5 levels "?","Bulbous",..: 1 2 4 2 1 1 2 2 1 4 ...
##  $ stalk.surface.above.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 3 3 3 3 3 3 3 1 ...
##  $ stalk.surface.below.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 1 3 2 3 3 3 3 1 ...
##  $ stalk.color.above.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 8 4 8 4 7 8 8 8 6 8 ...
##  $ stalk.color.below.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 7 8 8 8 8 8 4 8 6 8 ...
##  $ veil.color              : Factor w/ 4 levels "Brown","Orange",..: 3 3 3 3 3 3 3 3 1 3 ...
##  $ ring.number             : Factor w/ 3 levels "None","One","Two": 2 2 2 2 2 3 2 2 2 2 ...
##  $ ring.type               : Factor w/ 5 levels "Evanescent","Flaring",..: 1 5 1 5 1 1 5 5 5 1 ...
##  $ spore.print.color       : Factor w/ 9 levels "Buff","Chocolate",..: 8 4 3 4 8 8 4 7 1 4 ...
##  $ population              : Factor w/ 6 levels "Abundant","Clustered",..: 5 5 1 5 5 2 6 5 2 1 ...
##  $ habitat                 : Factor w/ 7 levels "Woods","Grasses",..: 3 1 2 1 3 7 1 1 3 2 ...
```

```r
mush_train2<-mush_train[,-c(12)]  # Removing stalk.root
str(mush_train2)
```

```
## 'data.frame':    5687 obs. of  21 variables:
##  $ class                   : Factor w/ 2 levels "Edible","Poisonous": 2 1 1 1 2 1 1 1 1 1 ...
##  $ cap.shape               : Factor w/ 6 levels "Bell","Conical",..: 3 3 6 6 3 4 3 6 6 6 ...
##  $ cap.surface             : Factor w/ 4 levels "Fibrous","Grooves",..: 4 1 3 4 3 4 4 1 3 1 ...
##  $ cap.color               : Factor w/ 10 levels "Buff","Cinnamon",..: 5 5 5 3 5 1 3 10 5 9 ...
##  $ bruises                 : Factor w/ 2 levels "No","Bruises": 1 2 1 2 1 2 2 2 1 1 ...
##  $ odor                    : Factor w/ 9 levels "Almond","Creosote",..: 8 6 6 6 3 6 6 4 6 6 ...
##  $ gill.attachment         : Factor w/ 2 levels "Attached","Free": 2 2 2 2 2 2 2 2 1 2 ...
##  $ gill.spacing            : Factor w/ 2 levels "Close","Crowded": 1 1 2 1 1 1 1 2 1 2 ...
##  $ gill.size               : Factor w/ 2 levels "Broad","Narrow": 2 1 1 1 2 1 1 2 1 1 ...
##  $ gill.color              : Factor w/ 12 levels "Buff","Red","Gray",..: 1 6 8 11 1 11 10 11 7 8 ...
##  $ stalk.shape             : Factor w/ 2 levels "Enlarging","Tapering": 2 2 2 2 2 1 2 2 1 2 ...
##  $ stalk.surface.above.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 3 3 3 3 3 3 3 1 ...
##  $ stalk.surface.below.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 1 3 2 3 3 3 3 1 ...
##  $ stalk.color.above.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 8 4 8 4 7 8 8 8 6 8 ...
##  $ stalk.color.below.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 7 8 8 8 8 8 4 8 6 8 ...
##  $ veil.color              : Factor w/ 4 levels "Brown","Orange",..: 3 3 3 3 3 3 3 3 1 3 ...
##  $ ring.number             : Factor w/ 3 levels "None","One","Two": 2 2 2 2 2 3 2 2 2 2 ...
##  $ ring.type               : Factor w/ 5 levels "Evanescent","Flaring",..: 1 5 1 5 1 1 5 5 5 1 ...
##  $ spore.print.color       : Factor w/ 9 levels "Buff","Chocolate",..: 8 4 3 4 8 8 4 7 1 4 ...
##  $ population              : Factor w/ 6 levels "Abundant","Clustered",..: 5 5 1 5 5 2 6 5 2 1 ...
##  $ habitat                 : Factor w/ 7 levels "Woods","Grasses",..: 3 1 2 1 3 7 1 1 3 2 ...
```

```r
dim(mush_train2)
```

```
## [1] 5687   21
```

```r
# Create testing set
str(mush_test)  # Veiltype has already been removed for this data set
```

```
## 'data.frame':    2437 obs. of  22 variables:
##  $ class                   : Factor w/ 2 levels "Edible","Poisonous": 1 2 1 1 1 1 2 1 2 1 ...
##  $ cap.shape               : Factor w/ 6 levels "Bell","Conical",..: 1 6 1 6 6 5 6 1 6 6 ...
##  $ cap.surface             : Factor w/ 4 levels "Fibrous","Grooves",..: 3 4 4 4 1 1 3 3 4 4 ...
##  $ cap.color               : Factor w/ 10 levels "Buff","Cinnamon",..: 9 9 9 10 5 4 5 10 9 10 ...
##  $ bruises                 : Factor w/ 2 levels "No","Bruises": 2 2 2 2 1 1 2 2 2 2 ...
##  $ odor                    : Factor w/ 9 levels "Almond","Creosote",..: 4 7 4 4 6 6 7 4 7 4 ...
##  $ 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": 1 2 1 1 1 2 2 1 2 1 ...
##  $ gill.color              : Factor w/ 12 levels "Buff","Red","Gray",..: 6 6 6 3 6 5 6 3 5 6 ...
##  $ 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",..: 3 4 3 3 4 4 4 3 4 3 ...
##  $ stalk.surface.above.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.surface.below.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 3 3 1 3 3 3 3 3 ...
##  $ stalk.color.above.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ stalk.color.below.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ 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 "Buff","Chocolate",..: 4 3 4 4 3 4 3 4 4 4 ...
##  $ population              : Factor w/ 6 levels "Abundant","Clustered",..: 3 4 4 3 1 6 4 3 4 3 ...
##  $ habitat                 : Factor w/ 7 levels "Woods","Grasses",..: 4 6 4 2 2 6 2 4 6 4 ...
```

```r
mush_test2<-mush_test[,-c(12)]  # Removing stalk.root
str(mush_test2)
```

```
## 'data.frame':    2437 obs. of  21 variables:
##  $ class                   : Factor w/ 2 levels "Edible","Poisonous": 1 2 1 1 1 1 2 1 2 1 ...
##  $ cap.shape               : Factor w/ 6 levels "Bell","Conical",..: 1 6 1 6 6 5 6 1 6 6 ...
##  $ cap.surface             : Factor w/ 4 levels "Fibrous","Grooves",..: 3 4 4 4 1 1 3 3 4 4 ...
##  $ cap.color               : Factor w/ 10 levels "Buff","Cinnamon",..: 9 9 9 10 5 4 5 10 9 10 ...
##  $ bruises                 : Factor w/ 2 levels "No","Bruises": 2 2 2 2 1 1 2 2 2 2 ...
##  $ odor                    : Factor w/ 9 levels "Almond","Creosote",..: 4 7 4 4 6 6 7 4 7 4 ...
##  $ 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": 1 2 1 1 1 2 2 1 2 1 ...
##  $ gill.color              : Factor w/ 12 levels "Buff","Red","Gray",..: 6 6 6 3 6 5 6 3 5 6 ...
##  $ stalk.shape             : Factor w/ 2 levels "Enlarging","Tapering": 1 1 1 1 2 1 1 1 1 1 ...
##  $ stalk.surface.above.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.surface.below.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 3 3 1 3 3 3 3 3 ...
##  $ stalk.color.above.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ stalk.color.below.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ 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 "Buff","Chocolate",..: 4 3 4 4 3 4 3 4 4 4 ...
##  $ population              : Factor w/ 6 levels "Abundant","Clustered",..: 3 4 4 3 1 6 4 3 4 3 ...
##  $ habitat                 : Factor w/ 7 levels "Woods","Grasses",..: 4 6 4 2 2 6 2 4 6 4 ...
```

```r
dim(mush_train2)
```

```
## [1] 5687   21
```

```r
# Create training with no labels
mush_train2_no_labels<-mush_train2[,-c(1)]  # Remove class column
str(mush_train2_no_labels)
```

```
## 'data.frame':    5687 obs. of  20 variables:
##  $ cap.shape               : Factor w/ 6 levels "Bell","Conical",..: 3 3 6 6 3 4 3 6 6 6 ...
##  $ cap.surface             : Factor w/ 4 levels "Fibrous","Grooves",..: 4 1 3 4 3 4 4 1 3 1 ...
##  $ cap.color               : Factor w/ 10 levels "Buff","Cinnamon",..: 5 5 5 3 5 1 3 10 5 9 ...
##  $ bruises                 : Factor w/ 2 levels "No","Bruises": 1 2 1 2 1 2 2 2 1 1 ...
##  $ odor                    : Factor w/ 9 levels "Almond","Creosote",..: 8 6 6 6 3 6 6 4 6 6 ...
##  $ gill.attachment         : Factor w/ 2 levels "Attached","Free": 2 2 2 2 2 2 2 2 1 2 ...
##  $ gill.spacing            : Factor w/ 2 levels "Close","Crowded": 1 1 2 1 1 1 1 2 1 2 ...
##  $ gill.size               : Factor w/ 2 levels "Broad","Narrow": 2 1 1 1 2 1 1 2 1 1 ...
##  $ gill.color              : Factor w/ 12 levels "Buff","Red","Gray",..: 1 6 8 11 1 11 10 11 7 8 ...
##  $ stalk.shape             : Factor w/ 2 levels "Enlarging","Tapering": 2 2 2 2 2 1 2 2 1 2 ...
##  $ stalk.surface.above.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 3 3 3 3 3 3 3 1 ...
##  $ stalk.surface.below.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 1 3 2 3 3 3 3 1 ...
##  $ stalk.color.above.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 8 4 8 4 7 8 8 8 6 8 ...
##  $ stalk.color.below.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 7 8 8 8 8 8 4 8 6 8 ...
##  $ veil.color              : Factor w/ 4 levels "Brown","Orange",..: 3 3 3 3 3 3 3 3 1 3 ...
##  $ ring.number             : Factor w/ 3 levels "None","One","Two": 2 2 2 2 2 3 2 2 2 2 ...
##  $ ring.type               : Factor w/ 5 levels "Evanescent","Flaring",..: 1 5 1 5 1 1 5 5 5 1 ...
##  $ spore.print.color       : Factor w/ 9 levels "Buff","Chocolate",..: 8 4 3 4 8 8 4 7 1 4 ...
##  $ population              : Factor w/ 6 levels "Abundant","Clustered",..: 5 5 1 5 5 2 6 5 2 1 ...
##  $ habitat                 : Factor w/ 7 levels "Woods","Grasses",..: 3 1 2 1 3 7 1 1 3 2 ...
```

```r
dim(mush_train2_no_labels)
```

```
## [1] 5687   20
```

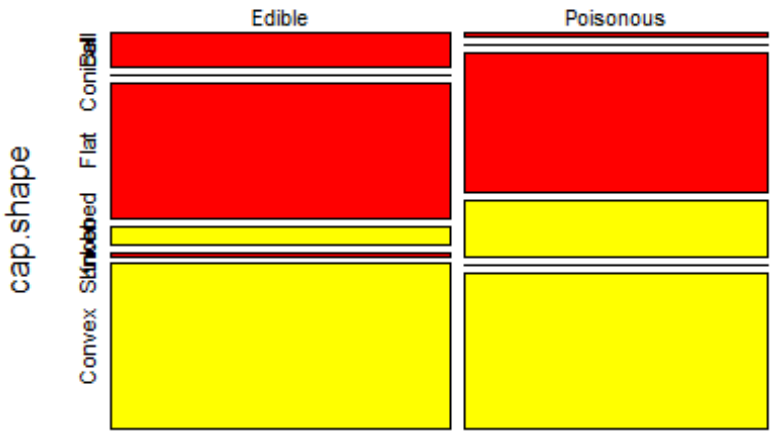
```r
# Create testing with no labels
mush_test2_no_labels<-mush_test2[,-c(1)]  # Remove class column
str(mush_test2_no_labels)
```

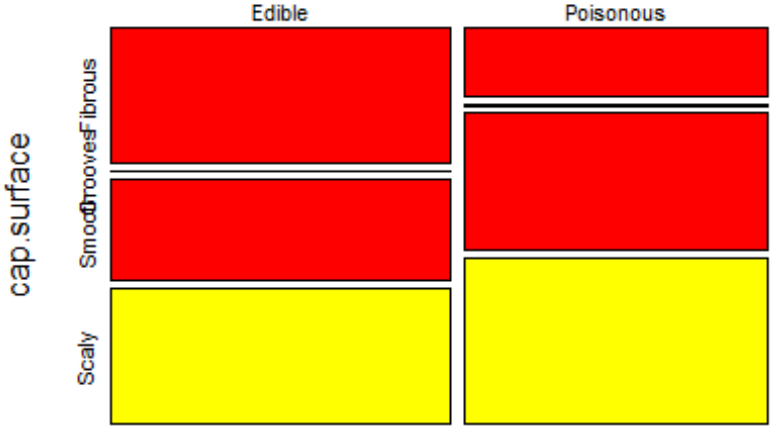
```
## 'data.frame':    2437 obs. of  20 variables:
##  $ cap.shape               : Factor w/ 6 levels "Bell","Conical",..: 1 6 1 6 6 5 6 1 6 6 ...
##  $ cap.surface             : Factor w/ 4 levels "Fibrous","Grooves",..: 3 4 4 4 1 1 3 3 4 4 ...
##  $ cap.color               : Factor w/ 10 levels "Buff","Cinnamon",..: 9 9 9 10 5 4 5 10 9 10 ...
##  $ bruises                 : Factor w/ 2 levels "No","Bruises": 2 2 2 2 1 1 2 2 2 2 ...
##  $ odor                    : Factor w/ 9 levels "Almond","Creosote",..: 4 7 4 4 6 6 7 4 7 4 ...
##  $ 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": 1 2 1 1 1 2 2 1 2 1 ...
##  $ gill.color              : Factor w/ 12 levels "Buff","Red","Gray",..: 6 6 6 3 6 5 6 3 5 6 ...
##  $ stalk.shape             : Factor w/ 2 levels "Enlarging","Tapering": 1 1 1 1 2 1 1 1 1 1 ...
##  $ stalk.surface.above.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.surface.below.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 3 3 1 3 3 3 3 3 ...
##  $ stalk.color.above.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ stalk.color.below.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ 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 "Buff","Chocolate",..: 4 3 4 4 3 4 3 4 4 4 ...
##  $ population              : Factor w/ 6 levels "Abundant","Clustered",..: 3 4 4 3 1 6 4 3 4 3 ...
##  $ habitat                 : Factor w/ 7 levels "Woods","Grasses",..: 4 6 4 2 2 6 2 4 6 4 ...
```

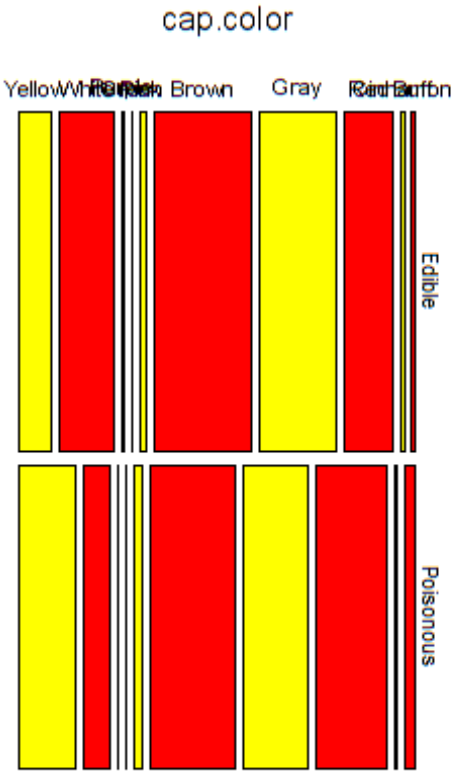
```r
dim(mush_train2_no_labels)
```

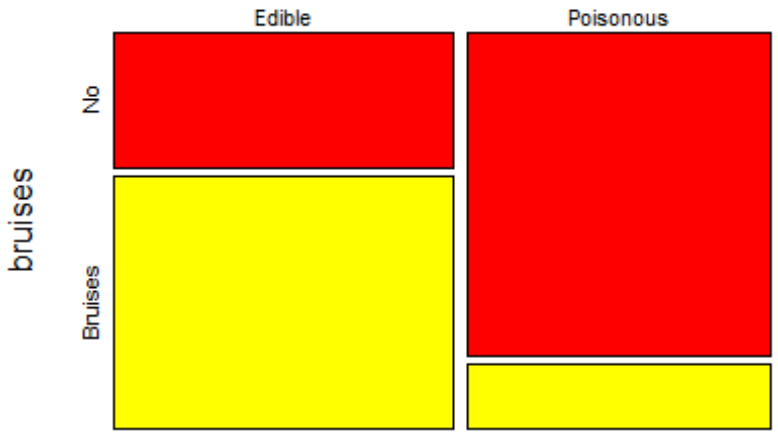
```
## [1] 5687   20
```

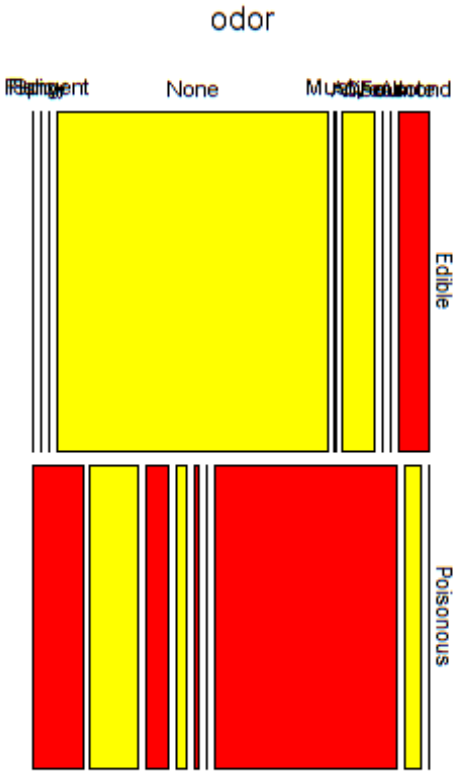
```r
# NB Model #1 (using Naive Bayes package)

nb <- naive_bayes( class ~. , data= mush_train2 , laplace=1, na.action = na.pass )  # Create model with train set
nb_prediction <- predict ( nb ,  mush_test2 )  # Create prediction with test set

## Warning: predict.naive_bayes(): More features in the newdata are provided
## as there are probability tables in the object. Calculation is performed
## based on features to be found in the tables.

(cm = table(nb_prediction , TestClassLabels))  # Get confusion matrix

##              TestClassLabels
## nb_prediction Edible Poisonous
##     Edible      1203        85
##     Poisonous      5      1144

model_accuracy = sum(diag(cm))/sum(cm)  # Calculate accuracy
a<-(model_accuracy_p<- paste(round((model_accuracy)*100,digits=2),"%",sep=""))  # Convert to a percent
cat("The accuracy of the Naive Bayes model is ", model_accuracy_p )

## The accuracy of the Naive Bayes model is  96.31%

# Visualize Naive Bayes
# This gives a separate vis per variable
plot(nb, legend.box = TRUE)
```


cap.shape
Convex
Knobbed
Flat
Conical
Bell
Edible
Poisonous

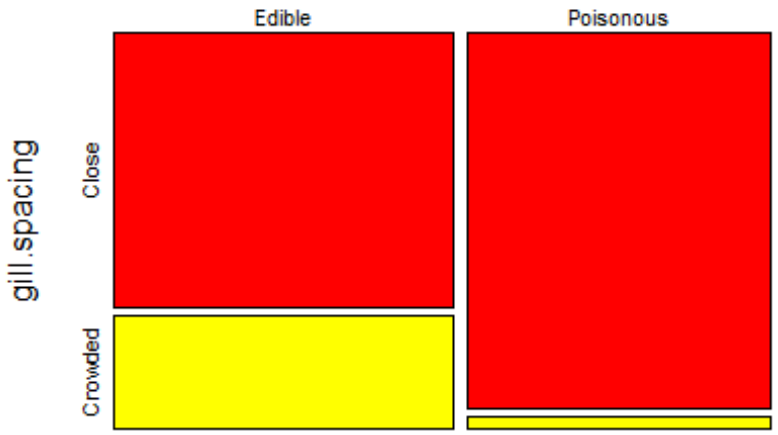

Edible
Poisonous
cap.surface
Fibrous
Grooves
Smooth
Scaly

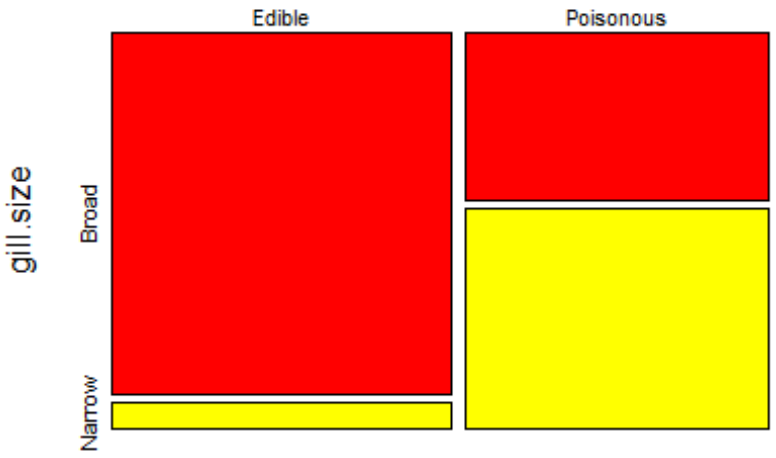

cap.color
Yellow
Brown
Gray
Edible
Poisonous

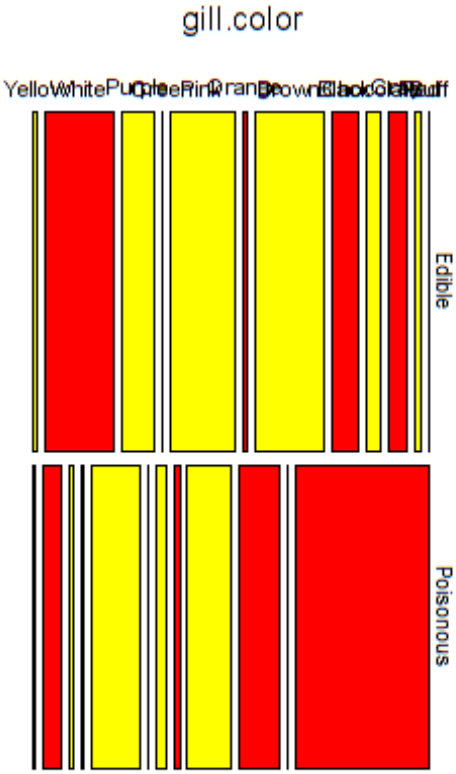

Edible
Poisonous
No
Bruises
bruises


odor
Edible
Poisonous
None

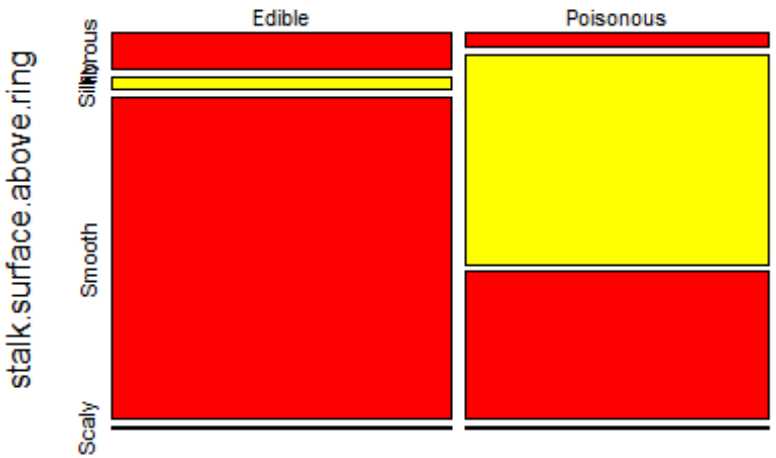

Edible
Poisonous
Attached
Free
gill.attachment

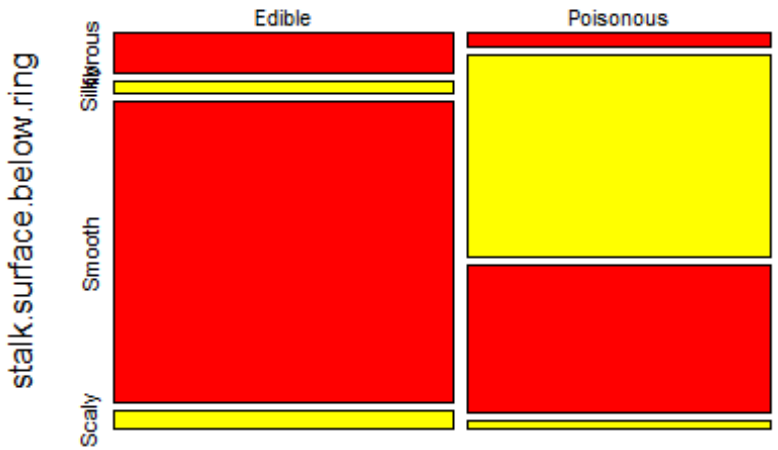

Edible
Poisonous
gill.spacing
Close
Crowded

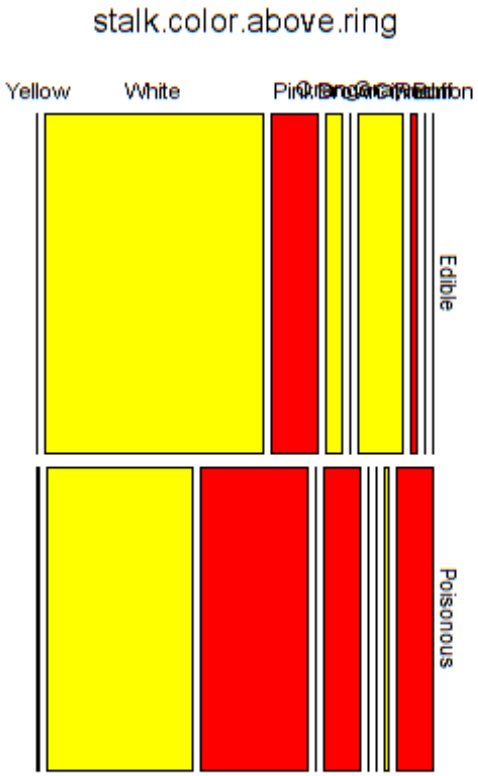

gill.size
Narrow
Broad
Edible
Poisonous

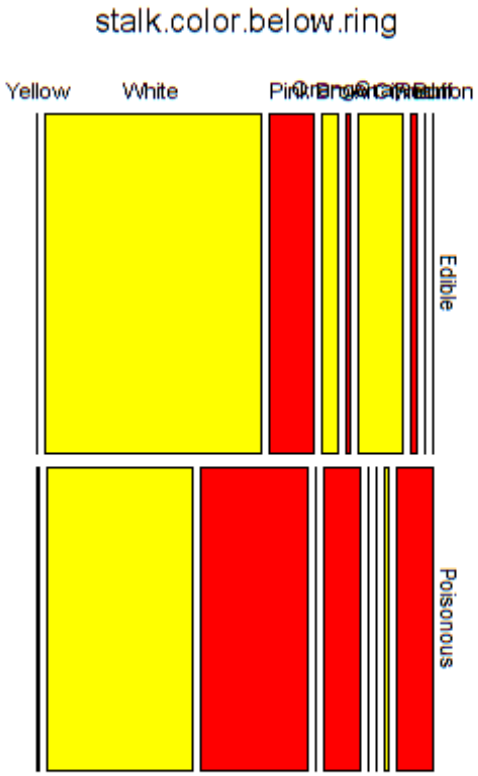

gill.color
Edible
Poisonous

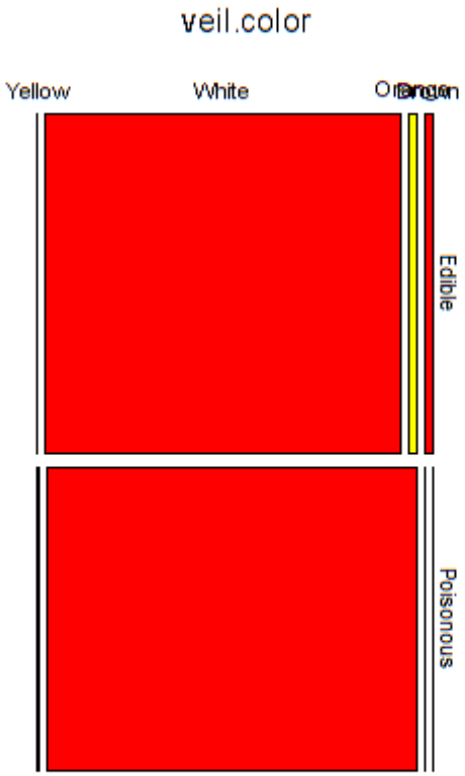

Edible
Poisonous
Enlarging
Tapering
stalk.shape

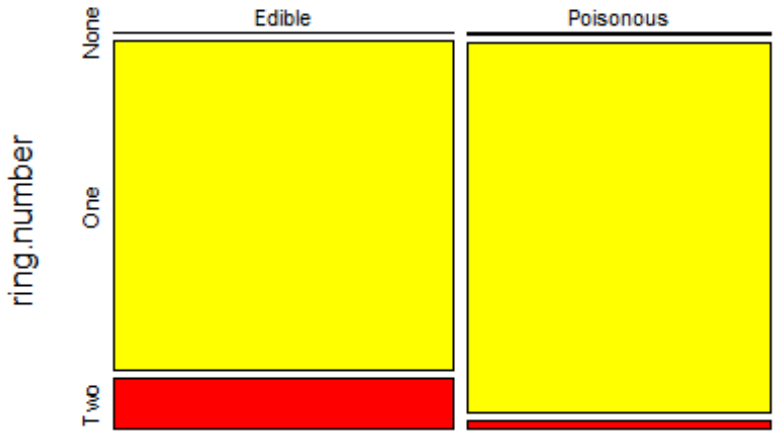

Edible
Poisonous
stalk.surface.above.ring
Silky
Fibrous
Smooth
Scaly

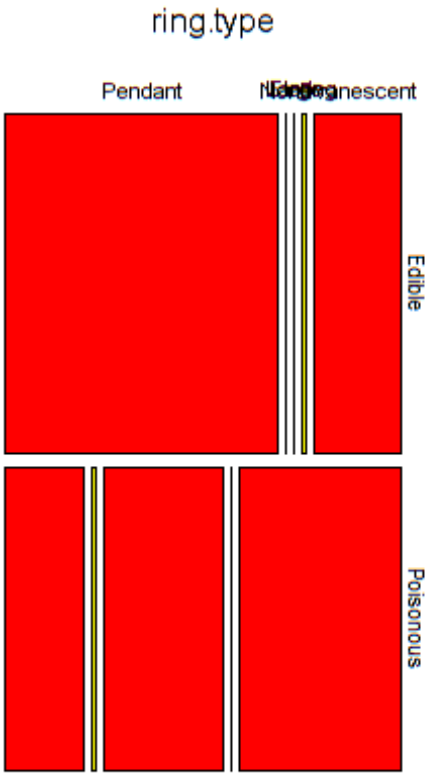

Edible
Poisonous
stalk.surface.below.ring
Silky
Fibrous
Smooth
Scaly

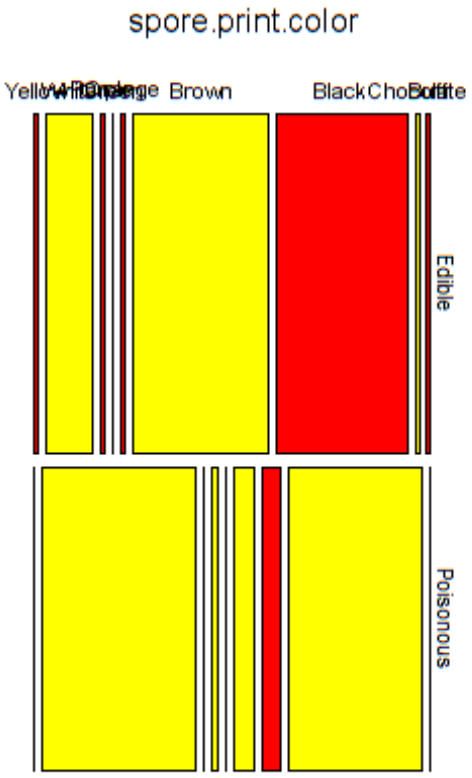

stalk.color.above.ring
Yellow
White
Pink
Edible
Poisonous

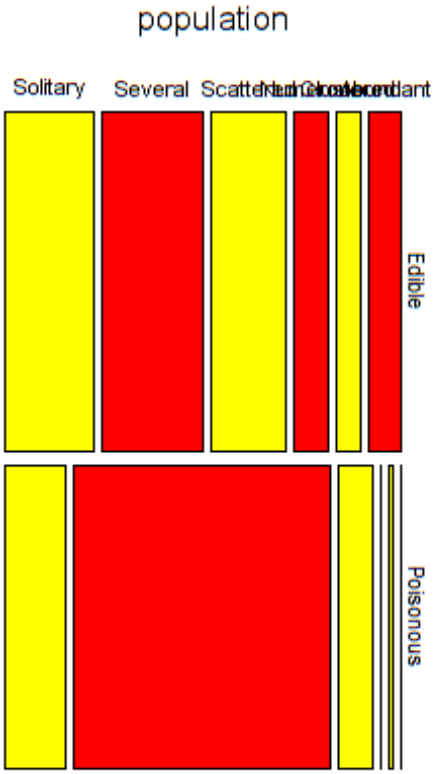

stalk.color.below.ring
Yellow
White
Pink
Edible
Poisonous

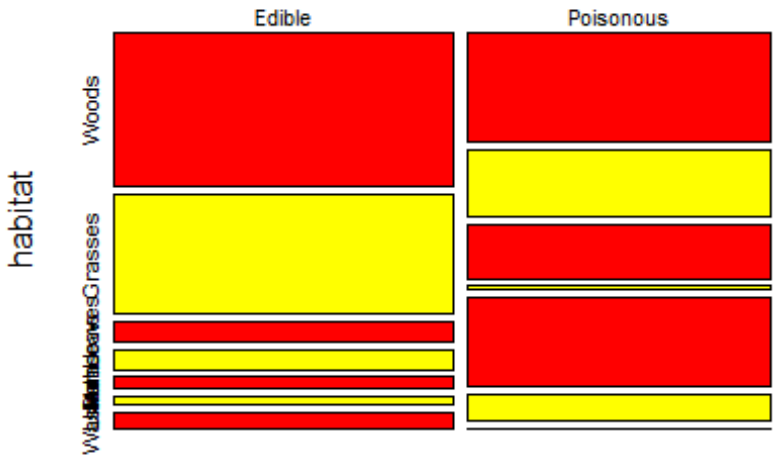

veil.color
Yellow
White
Orange
Brown
Edible
Poisonous


Edible
Poisonous
None
One
Two
ring.number


ring.type
Pendant
Evanescent
Edible
Poisonous


spore.print.color
Yellow
Brown
Black
Buff
Edible
Poisonous


population
Solitary
Several
Scattered
Edible
Poisonous




```
# Look at information gain with Entropy
#install.packages("CORElearn")
library(CORElearn)
Method.CORElearn <- CORElearn::attrEval(mush_train2$class ~ ., data=mush_train2,  estimator = "GainRatio")
max(Method.CORElearn)

## [1] 0.3913577

which.max(Method.CORElearn)

## odor
##    5
```

```r
# NB Model #2 (using e1071 package)
NB_e1071<-naiveBayes (class~., data=mush_train2, na.action = na.pass)
NB_e1071_Pred <- predict(NB_e1071, mush_test2_no_labels)  # Takes a while
(cm = table(NB_e1071_Pred, TestClassLabels))  # Get confusion matrix

##              TestClassLabels
## NB_e1071_Pred Edible Poisonous
##     Edible      1203       116
##     Poisonous      5      1113

model_accuracy = sum(diag(cm))/sum(cm)  # Calculate accuracy
model_accuracy_p<- paste(round((model_accuracy)*100,digits=2),"%",sep="")  # Convert to a percent
cat("The accuracy of the Naive Bayes model made with package e1071 is ", model_accuracy_p )

## The accuracy of the Naive Bayes model made with package e1071 is  95.03%

#plot(NB_e1071, ylab = "Density", main = "Naive Bayes Plot")  # Commented out due to error below.
# Error in xy.coords(x, y, xlabel, ylabel, log) : 'x' is a list, but does not have components 'x' and 'y'



#==============================================================================
# Association Rule Mining (Lauren Foltz)
#==============================================================================
# Important: This model was included last because the Arules package does not play nicely with other packages.
# For example, this package masks "recode" and "inspect"

#install.packages("arules")
library("arules")

# The following object is masked from package:dplyr: recode
# The following object is masked from package:tm:inspect
# The following objects are masked from package:base: abbreviate, write

#install.packages("arulesViz")
library("arulesViz")
```

```r
# Adjust Data

# Make a copy of data frame with veil.type removed. Call it "armDF"
armDF <- m2
str(armDF)
```

```
## 'data.frame':    8124 obs. of  22 variables:
##  $ class                   : Factor w/ 2 levels "Edible","Poisonous": 2 1 1 2 1 1 1 1 2 1 ...
##  $ cap.shape               : Factor w/ 6 levels "Bell","Conical",..: 6 6 1 6 6 6 1 1 6 1 ...
##  $ cap.surface             : Factor w/ 4 levels "Fibrous","Grooves",..: 3 3 3 4 3 4 3 4 4 3 ...
##  $ cap.color               : Factor w/ 10 levels "Buff","Cinnamon",..: 5 10 9 9 4 10 9 9 9 10 ...
##  $ bruises                 : Factor w/ 2 levels "No","Bruises": 2 2 2 2 1 2 2 2 2 2 ...
##  $ odor                    : Factor w/ 9 levels "Almond","Creosote",..: 7 1 4 7 6 1 1 4 7 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 "Buff","Red","Gray",..: 5 5 6 6 5 6 3 6 8 3 ...
##  $ 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",..: 4 3 3 4 4 3 3 3 4 3 ...
##  $ stalk.surface.above.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.surface.below.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.color.above.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ stalk.color.below.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ 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 "Buff","Chocolate",..: 3 4 4 3 4 3 3 4 3 3 ...
##  $ population              : Factor w/ 6 levels "Abundant","Clustered",..: 4 3 3 4 1 3 3 4 5 4 ...
##  $ habitat                 : Factor w/ 7 levels "Woods","Grasses",..: 6 2 4 6 2 2 4 4 2 4 ...
```

```r
# Remove column for stalk.root because it has "?" in some cells
which( colnames(armDF)=="stalk.root" ) # 12 means stalk.root is column 12
```

```
## [1] 12
```

```r
armDF <-armDF[-c(12)]
which( colnames(armDF)=="stalk.root" ) # 0 means stalk.root is no longer present
```

```
## integer(0)
```

```r
# Make a copy and remove the label, which is the first column
armDF_unlabeled <-armDF[-c(1)]
str(armDF_unlabeled)
```

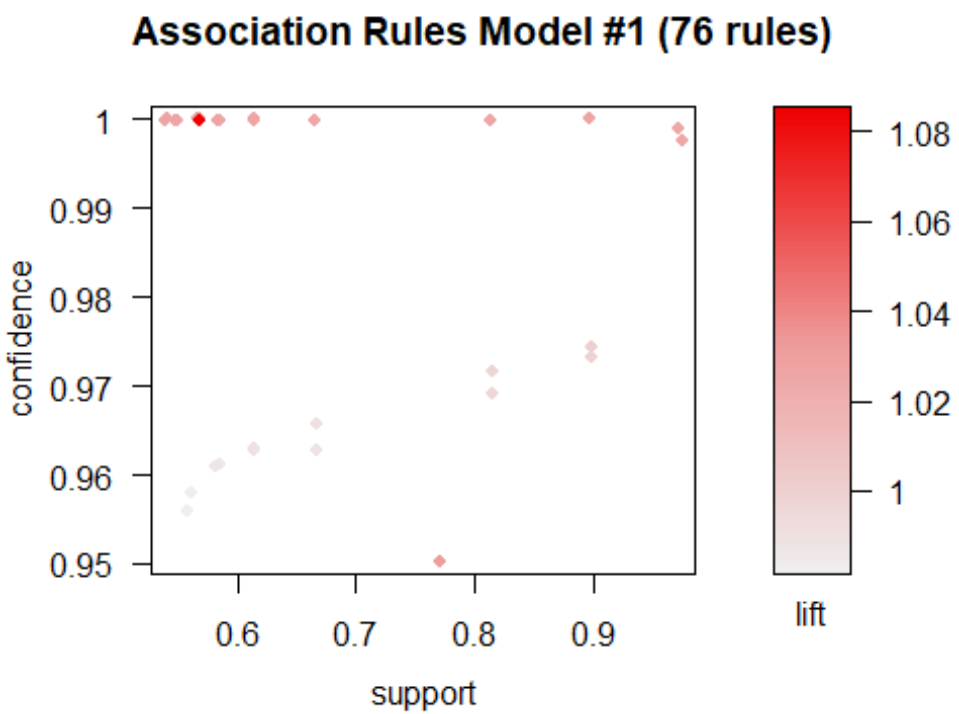
```
## 'data.frame':    8124 obs. of  20 variables:
##  $ cap.shape               : Factor w/ 6 levels "Bell","Conical",..: 6 6 1 6 6 6 1 1 6 1 ...
##  $ cap.surface             : Factor w/ 4 levels "Fibrous","Grooves",..: 3 3 3 4 3 4 3 4 4 3 ...
##  $ cap.color               : Factor w/ 10 levels "Buff","Cinnamon",..: 5 10 9 9 4 10 9 9 9 10 ...
##  $ bruises                 : Factor w/ 2 levels "No","Bruises": 2 2 2 2 1 2 2 2 2 2 ...
##  $ odor                    : Factor w/ 9 levels "Almond","Creosote",..: 7 1 4 7 6 1 1 4 7 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 "Buff","Red","Gray",..: 5 5 6 6 5 6 3 6 8 3 ...
##  $ stalk.shape             : Factor w/ 2 levels "Enlarging","Tapering": 1 1 1 1 2 1 1 1 1 1 ...
##  $ stalk.surface.above.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.surface.below.ring: Factor w/ 4 levels "fibrous","Silky",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ stalk.color.above.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ stalk.color.below.ring  : Factor w/ 9 levels "Buff","Cinnamon",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ 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 "Buff","Chocolate",..: 3 4 4 3 4 3 3 4 3 3 ...
##  $ population              : Factor w/ 6 levels "Abundant","Clustered",..: 4 3 3 4 1 3 3 4 5 4 ...
##  $ habitat                 : Factor w/ 7 levels "Woods","Grasses",..: 6 2 4 6 2 2 4 4 2 4 ...
```

```
################
# Arm Model 1  #
################

# Explore unlabeled data
rules<-arules::apriori(armDF_unlabeled,parameter = list(supp=0.50, conf = 0.95,minlen=2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.95    0.1    1 none FALSE            TRUE       5     0.5      2
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
```

```
##
## Absolute minimum support count: 4062
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[111 item(s), 8124 transaction(s)] done [0.01s].
## sorting and recoding items ... [11 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [76 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
rules <- rules[!is.redundant(rules)]  # Remove redundant rules
options(digits=3)
# Produced 76 rules

plot(rules, main = "Association Rules Model #1 (76 rules)")  # Add title

## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```



```r
# Sort by confidence & inspect
rules_conf<-sort (rules, decreasing = TRUE, by='confidence')
inspect(rules_conf[1:10])

# Top 10 rules:
#      lhs                                                    rhs                    support confidence lift count
# [1]  {stalk.color.below.ring=White}                      => {gill.attachment=Free} 0.540   1          1.03 4384
# [2]  {stalk.color.below.ring=White}                      => {veil.color=White}     0.540   1          1.03 4384
# [3]  {stalk.color.above.ring=White}                      => {gill.attachment=Free} 0.549   1          1.03 4464
# [4]  {stalk.color.above.ring=White}                      => {veil.color=White}     0.549   1          1.03 4464
# [5]  {stalk.shape=Tapering}                              => {ring.number=One}      0.567   1          1.08 4608
# [6]  {stalk.shape=Tapering}                              => {gill.attachment=Free} 0.567   1          1.03 4608
# [7]  {stalk.shape=Tapering}                              => {veil.color=White}     0.567   1          1.03 4608
# [8]  {gill.attachment=Free,stalk.surface.below.ring=Smooth} => {veil.color=White}     0.584   1          1.03 4744
# [9]  {stalk.surface.below.ring=Smooth,veil.color=White}     => {gill.attachment=Free} 0.584   1          1.03 4744
# [10] {gill.attachment=Free,stalk.surface.above.ring=Smooth} => {veil.color=White}     0.613   1          1.03 4984
```

```r
# Sort by support
rules_supp <- sort(rules, decreasing = TRUE, by="supp")
inspect(rules_supp[1:10])

# Top 10 rules:
#      lhs                                          rhs                    support confidence lift  count
# [1]  {gill.attachment=Free}                    => {veil.color=White}     0.973   0.999      1.024 7906
# [2]  {veil.color=White}                        => {gill.attachment=Free} 0.973   0.998      1.024 7906
# [3]  {ring.number=One}                         => {gill.attachment=Free} 0.898   0.974      1.000 7296
# [4]  {ring.number=One}                         => {veil.color=White}     0.897   0.973      0.998 7288
# [5]  {veil.color=White,ring.number=One}        => {gill.attachment=Free} 0.897   1.000      1.027 7288
# [6]  {gill.spacing=Close}                      => {veil.color=White}     0.815   0.972      0.996 6620
# [7]  {gill.spacing=Close}                      => {gill.attachment=Free} 0.813   0.969      0.995 6602
# [8]  {gill.attachment=Free,gill.spacing=Close} => {veil.color=White}     0.813   1.000      1.025 6602
# [9]  {gill.attachment=Free,gill.spacing=Close} => {ring.number=One}      0.772   0.950      1.031 6272
# [10] {gill.size=Broad}                         => {veil.color=White}     0.667   0.966      0.990 5420
```

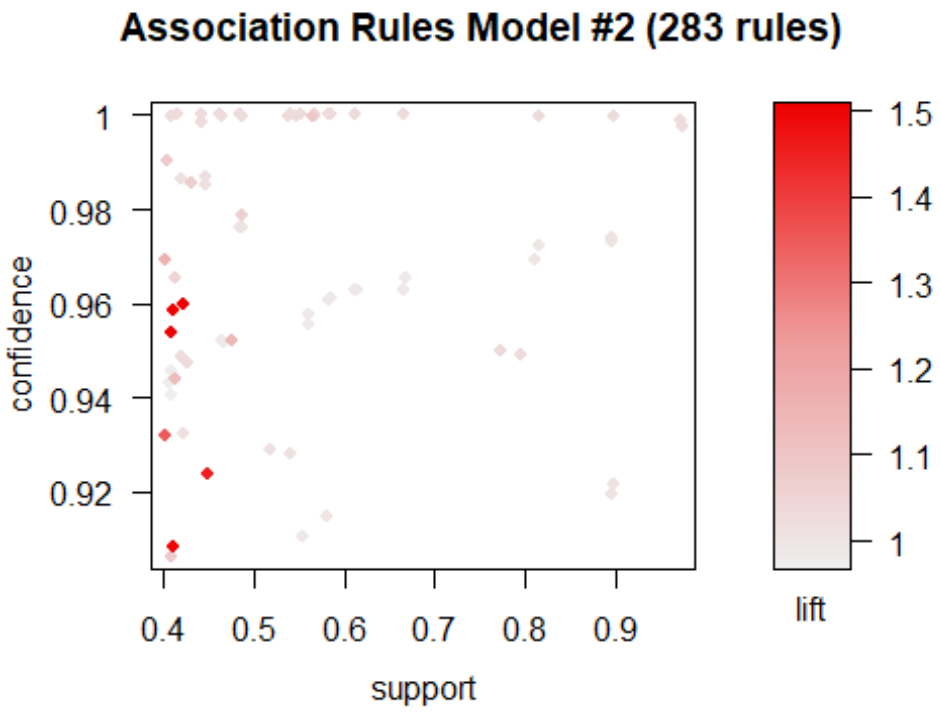
```r
# Sort by lift
rules_lift <- sort(rules, decreasing = TRUE,by="lift")
inspect(rules_lift[1:10])

# Top 10 rules:
#      lhs                                                   rhs                     support confidence lift count
# [1]  {stalk.shape=Tapering}                             => {ring.number=One}      0.567   1.00       1.08 4608
# [2]  {gill.attachment=Free,gill.spacing=Close}          => {ring.number=One}      0.772   0.95       1.03 6272
# [3]  {stalk.color.below.ring=White}                     => {gill.attachment=Free} 0.540   1.00       1.03 4384
# [4]  {stalk.color.above.ring=White}                     => {gill.attachment=Free} 0.549   1.00       1.03 4464
# [5]  {stalk.shape=Tapering}                             => {gill.attachment=Free} 0.567   1.00       1.03 4608
# [6]  {stalk.surface.below.ring=Smooth,veil.color=White} => {gill.attachment=Free} 0.584   1.00       1.03 4744
# [7]  {stalk.surface.above.ring=Smooth,veil.color=White} => {gill.attachment=Free} 0.613   1.00       1.03 4984
# [8]  {veil.color=White,ring.number=One}                 => {gill.attachment=Free} 0.897   1.00       1.03 7288
# [9]  {stalk.color.below.ring=White}                     => {veil.color=White}     0.540   1.00       1.03 4384
# [10] {stalk.color.above.ring=White}                     => {veil.color=White}     0.549   1.00       1.03 4464
```

```r
################
# Arm Model 2  #
################

# What happens if I reduce confidence to 0.90 and support to 0.40?
rules<-arules::apriori(armDF_unlabeled,parameter = list(supp=0.40, conf = 0.90,minlen=2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.9    0.1    1 none FALSE            TRUE       5     0.4      2
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 3249
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[111 item(s), 8124 transaction(s)] done [0.01s].
## sorting and recoding items ... [17 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 done [0.00s].
## writing ... [283 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
rules <- rules[!is.redundant(rules)]  # Remove redundant rules
options(digits=3)
# Produced 283 rules

plot(rules, main = "Association Rules Model #2 (283 rules)")  # Add title
```

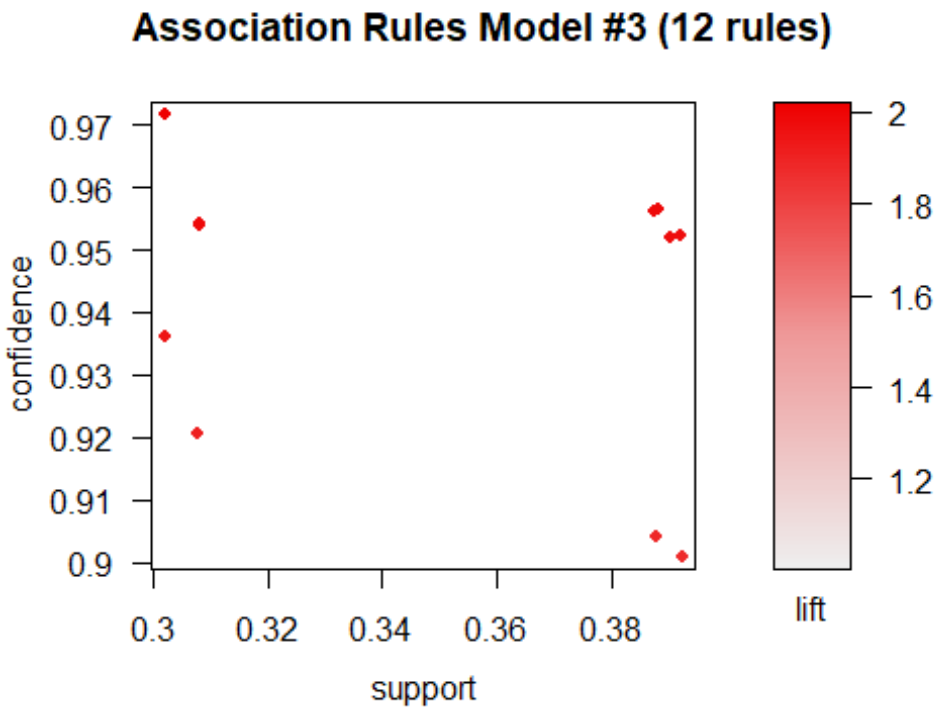
```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```




```
# We got more rules, and the lift range increased
```

```r
# Sort by confidence & inspect
rules_conf<-sort (rules, decreasing = TRUE, by='confidence')
rules <- rules[!is.redundant(rules)]  # Remove redundant rules
inspect(rules_conf[1:10])

# Top 10 rules:
#      lhs                                rhs                    support confidence lift count
# [1]  {bruises=Bruises}              => {gill.attachment=Free} 0.416   1          1.03 3376
# [2]  {bruises=Bruises}              => {veil.color=White}     0.416   1          1.03 3376
# [3]  {stalk.color.below.ring=White} => {gill.attachment=Free} 0.540   1          1.03 4384
# [4]  {stalk.color.below.ring=White} => {veil.color=White}     0.540   1          1.03 4384
# [5]  {stalk.color.above.ring=White} => {gill.attachment=Free} 0.549   1          1.03 4464
# [6]  {stalk.color.above.ring=White} => {veil.color=White}     0.549   1          1.03 4464
# [7]  {stalk.shape=Tapering}         => {ring.number=One}      0.567   1          1.08 4608
# [8]  {stalk.shape=Tapering}         => {gill.attachment=Free} 0.567   1          1.03 4608
# [9]  {stalk.shape=Tapering}         => {veil.color=White}     0.567   1          1.03 4608
# [10] {odor=None,veil.color=White}   => {gill.attachment=Free} 0.410   1          1.03 3328




# Sort by support
rules_supp <- sort(rules, decreasing = TRUE, by="supp")
inspect(rules_supp[1:10])

# Top 10 rules:
#      lhs                                          rhs                    support confidence lift  count
# [1]  {gill.attachment=Free}                    => {veil.color=White}     0.973   0.999      1.024 7906
# [2]  {veil.color=White}                        => {gill.attachment=Free} 0.973   0.998      1.024 7906
# [3]  {ring.number=One}                         => {gill.attachment=Free} 0.898   0.974      1.000 7296
# [4]  {gill.attachment=Free}                    => {ring.number=One}      0.898   0.922      1.000 7296
# [5]  {ring.number=One}                         => {veil.color=White}     0.897   0.973      0.998 7288
# [6]  {veil.color=White}                        => {ring.number=One}      0.897   0.920      0.998 7288
# [7]  {veil.color=White,ring.number=One}        => {gill.attachment=Free} 0.897   1.000      1.027 7288
# [8]  {gill.spacing=Close}                      => {veil.color=White}     0.815   0.972      0.996 6620
# [9]  {gill.spacing=Close}                      => {gill.attachment=Free} 0.813   0.969      0.995 6602
# [10] {gill.attachment=Free,gill.spacing=Close} => {veil.color=White}     0.813   1.000      1.025 6602
```

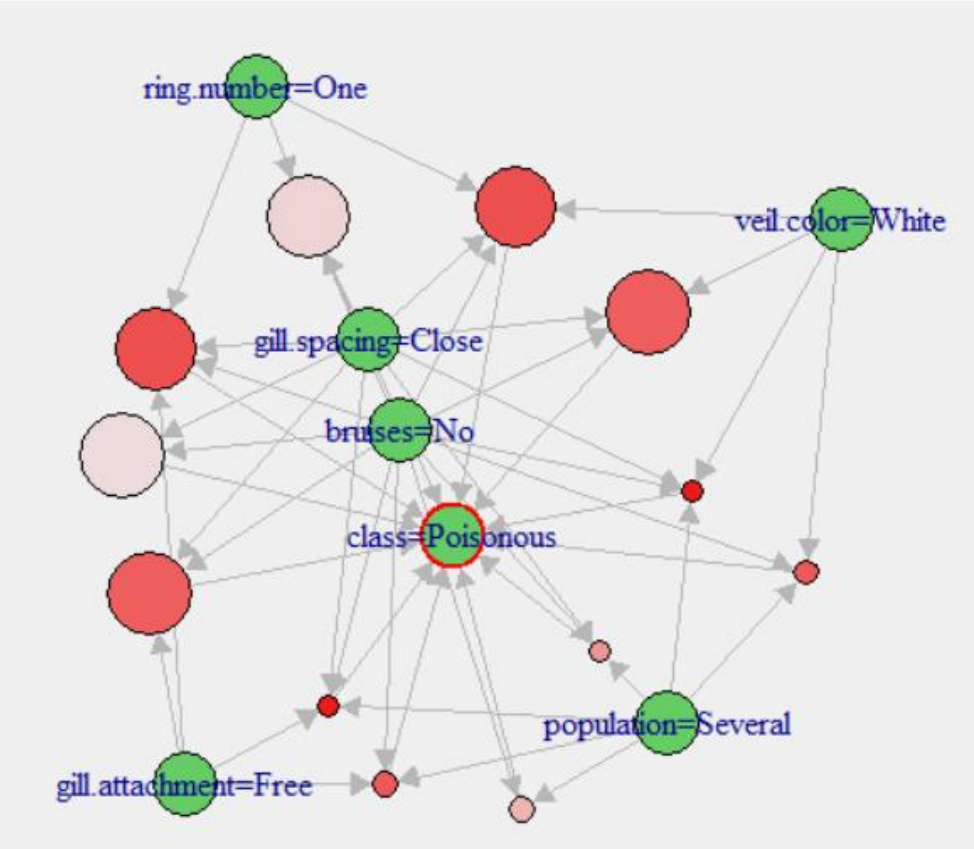
```r
# Sort by lift
rules_lift <- sort(rules, decreasing = TRUE,by="lift")
inspect(rules_lift[1:10])

# Top 10 rules:
#     lhs                                                    rhs                                 support conf.  lift count
# [1] {ring.number=One,ring.type=Pendant}                 => {stalk.surface.above.ring=Smooth} 0.420  0.960   1.51 3416
# [2] {stalk.surface.below.ring=Smooth,ring.type=Pendant} => {stalk.surface.above.ring=Smooth} 0.410  0.959   1.50 3328
# [3] {gill.spacing=Close,ring.type=Pendant}              => {stalk.surface.above.ring=Smooth} 0.409  0.954   1.50 3320
# [4] {stalk.surface.above.ring=Smooth,ring.type=Pendant} => {stalk.surface.below.ring=Smooth} 0.410  0.908   1.49 3328
# [5] {ring.type=Pendant}                                 => {stalk.surface.above.ring=Smooth} 0.451  0.923   1.45 3664
# [6] {odor=None}                                         => {gill.size=Broad}                 0.405  0.932   1.35 3288
# [7] {bruises=Bruises}                                   => {gill.spacing=Close}              0.403  0.969   1.16 3272
# [8] {population=Several}                                => {gill.spacing=Close}              0.474  0.952   1.14 3848
# [9] {ring.number=One,ring.type=Pendant}                 => {gill.spacing=Close}              0.414  0.944   1.13 3360
# [10] {stalk.shape=Tapering}                              => {ring.number=One}                 0.567  1.000   1.08 4608





################
# Arm Model 3  #
################
# Now let's use labeled data and set the class as the rhs, to find out what's most associated with poison/edible
# Support of .40 gave no rules, so reduce it to .30
rules<-arules::apriori(data = armDF, parameter = list(supp=0.30, conf = 0.90, minlen=2),
                       appearance = list(default="lhs",rhs="class=Poisonous"),
                       control = list(verbose = F))
options(digits=3)
rules <- rules[!is.redundant(rules)]  # Remove redundant rules
rules

## set of 12 rules

# Produced 12 rules
plot(rules, main = "Association Rules Model #3 (12 rules)")  # Add title

## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```



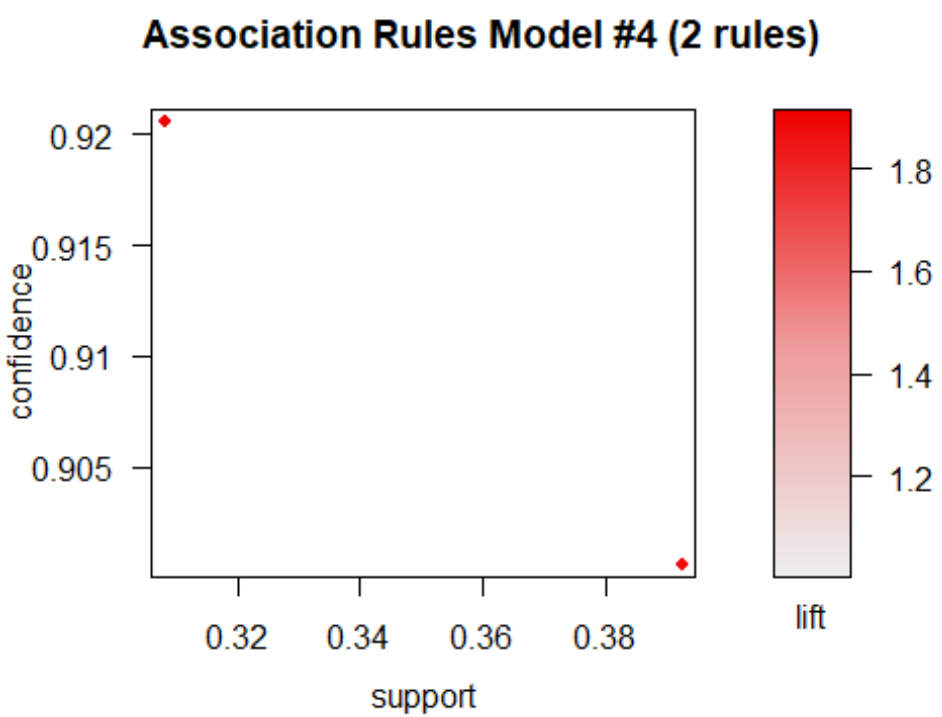

```
# Lift looks strong

# Sort by confidence & inspect
rules_conf<-sort (rules, decreasing = TRUE, by='confidence')
inspect(rules_conf[1:10])

# Sort by support
rules_supp <- sort(rules, decreasing = TRUE, by="supp")
inspect(rules_supp[1:10])

# Sort by lift
rules_lift <- sort(rules, decreasing = TRUE,by="lift")
inspect(rules_lift[1:10])
```

```
# This produced some pretty complicated rules that centered around the same 6 attributes
#   lhs
rhs                support confidence lift count
#  {bruises=No, gill.attachment=Free,gill.spacing=Close,                    population=Several}
=> {class=Poisonous} 0.302   0.972      2.02 2456
#  {bruises=No,                      gill.spacing=Close, veil.color=White,  population=Several}
=> {class=Poisonous} 0.302   0.972      2.02 2456
#  {bruises=No,                      gill.spacing=Close,                    population=Several}
=> {class=Poisonous} 0.302   0.936      1.94 2456
#  {bruises=No, gill.attachment=Free,gill.spacing=Close,                                          ring.number=One}
=> {class=Poisonous} 0.388   0.956      1.98 3152
#  {bruises=No,                      gill.spacing=Close, veil.color=White,                        ring.number=One}
=> {class=Poisonous} 0.388   0.956      1.98 3152
#  {bruises=No, gill.attachment=Free,gill.spacing=Close}
=> {class=Poisonous} 0.390   0.952      1.97 3170
#  {bruises=No,                      gill.spacing=Close, veil.color=White}
=> {class=Poisonous} 0.392   0.952      1.98 3188
#  {bruises=No, gill.attachment=Free,                                       population=Several}
=> {class=Poisonous} 0.308   0.954      1.98 2504
#  {bruises=No,                                          veil.color=White,  population=Several}
=> {class=Poisonous} 0.308   0.954      1.98 2504
```

```r
# PLOT POISONOUS
plot(rules, method="graph", engine="interactive")
```

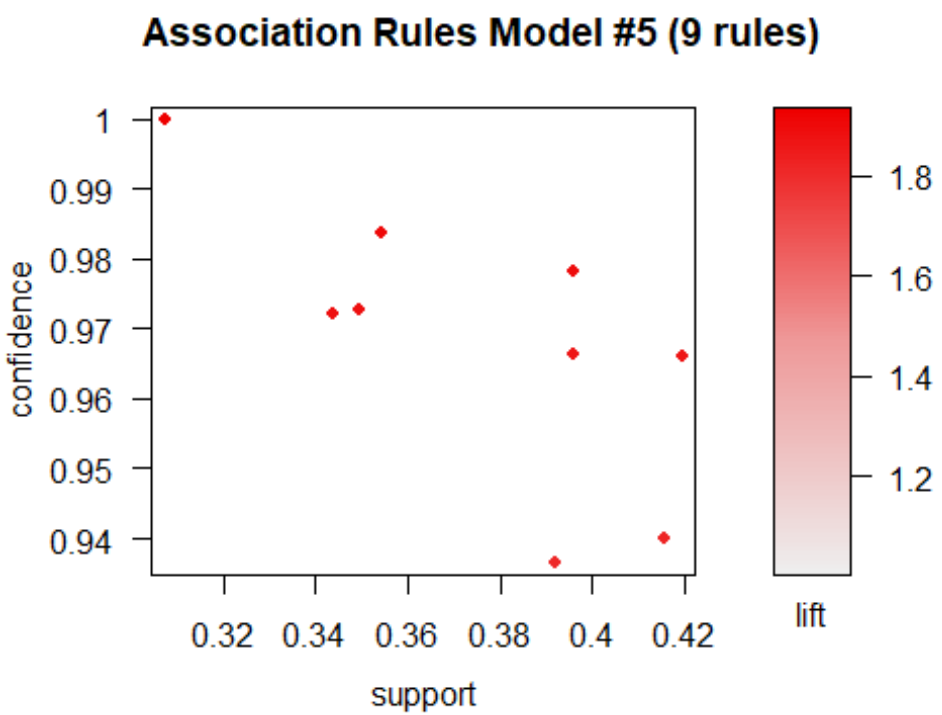


```r
################
# Arm Model 4  #
################
# Let's reduce rule length to get simpler rules
# maxlen = 2 produced no rules, so go with 3
rules<-arules::apriori(data = armDF, parameter = list(supp=0.30, conf = 0.90, minlen=2, maxlen=3),
                       appearance = list(default="lhs",rhs="class=Poisonous"),
                       control = list(verbose = F))
options(digits=3)
rules <- rules[!is.redundant(rules)]  # Remove redundant rules
rules

## set of 2 rules

# Produced 2 rules

plot(rules, main = "Association Rules Model #4 (2 rules)")  # Add title
```



```r
# Lift looks strong
inspect(rules)

# gill.spacing had higher support, but population had stronger confidence.
# I don't think "population several" is very useful though,
# so let's go with "bruises = No" and "Gill Spacing = Close" as the most important variables.

# lhs                                rhs               support confidence lift count
# [1] {bruises=No,population=Several} => {class=Poisonous} 0.308   0.921      1.91 2504
# [2] {bruises=No,gill.spacing=Close} => {class=Poisonous} 0.392   0.901      1.87 3188
```

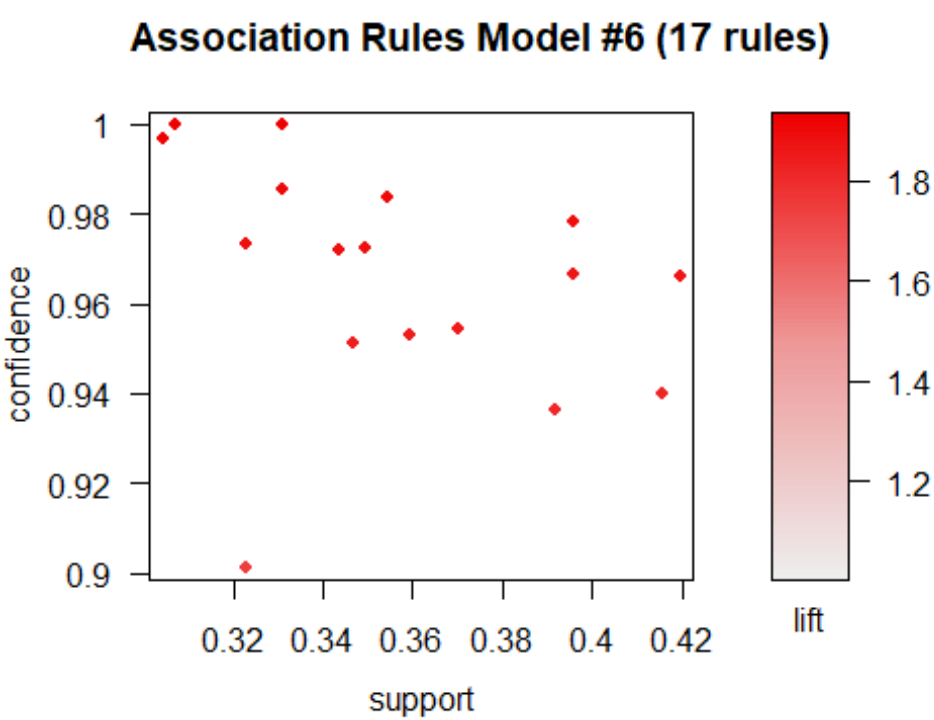
```r
################
# Arm Model 5  #
################

# Now let's set the rhs to edible
rules<-arules::apriori(data = armDF, parameter = list(supp=0.30, conf = 0.90, minlen=2, maxlen =3),
                       appearance = list(default="lhs",rhs="class=Edible"),
                       control = list(verbose = F))
options(digits=3)
rules <- rules[!is.redundant(rules)]  # Remove redundant rules
rules

## set of 9 rules

plot(rules, main = "Association Rules Model #5 (9 rules)")  # Add title
```



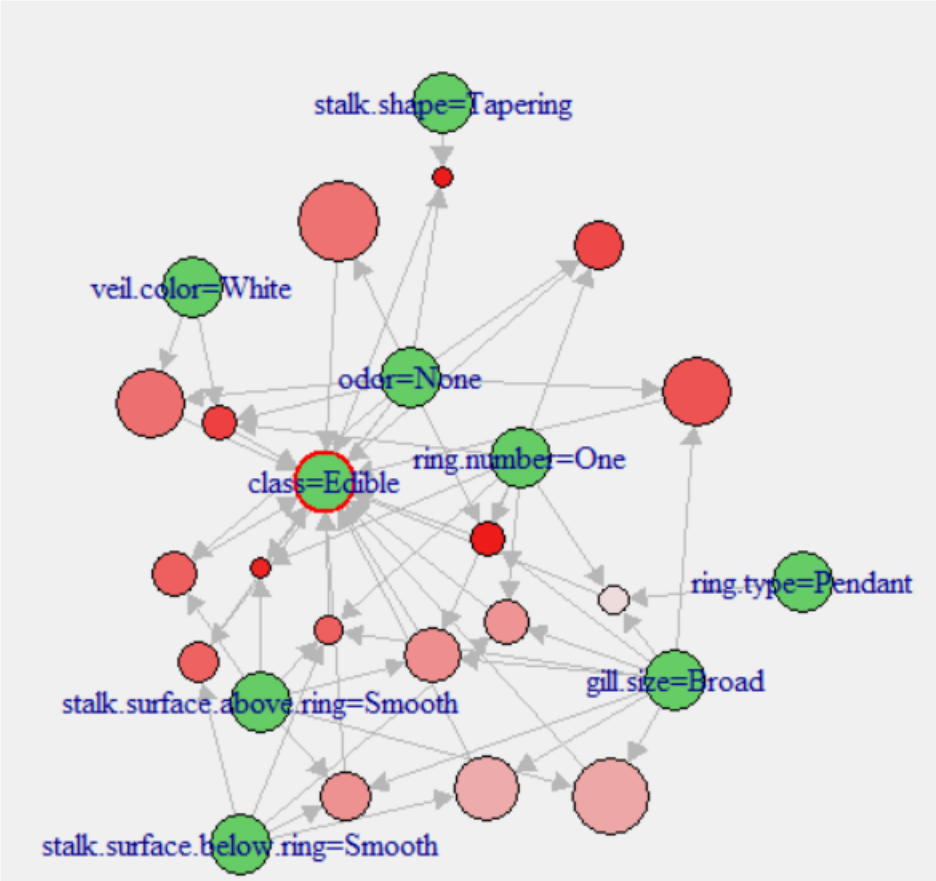

```r
# Lift looks strong
```

```r
inspect(rules)

# We have 9 rules. "Odor= None" seems to have a strong association.
# lhs                                                  rhs           support confidence lift count
# {odor=None}                                       => {class=Edible} 0.419   0.966      1.86 3408
# {odor=None,stalk.shape=Tapering}                  => {class=Edible} 0.307   1.000      1.93 2496
# {odor=None,stalk.surface.below.ring=Smooth}       => {class=Edible} 0.344   0.972      1.88 2792
# {odor=None,stalk.surface.above.ring=Smooth}       => {class=Edible} 0.350   0.973      1.88 2840
# {odor=None,gill.size=Broad}                       => {class=Edible} 0.396   0.978      1.89 3216
# {odor=None,ring.number=One}                       => {class=Edible} 0.355   0.984      1.90 2880
# {odor=None,veil.color=White}                      => {class=Edible} 0.396   0.966      1.87 3216
# {gill.size=Broad,stalk.surface.below.ring=Smooth} => {class=Edible} 0.392   0.936      1.81 3184
# {gill.size=Broad,stalk.surface.above.ring=Smooth} => {class=Edible} 0.416   0.940      1.81 3376
```

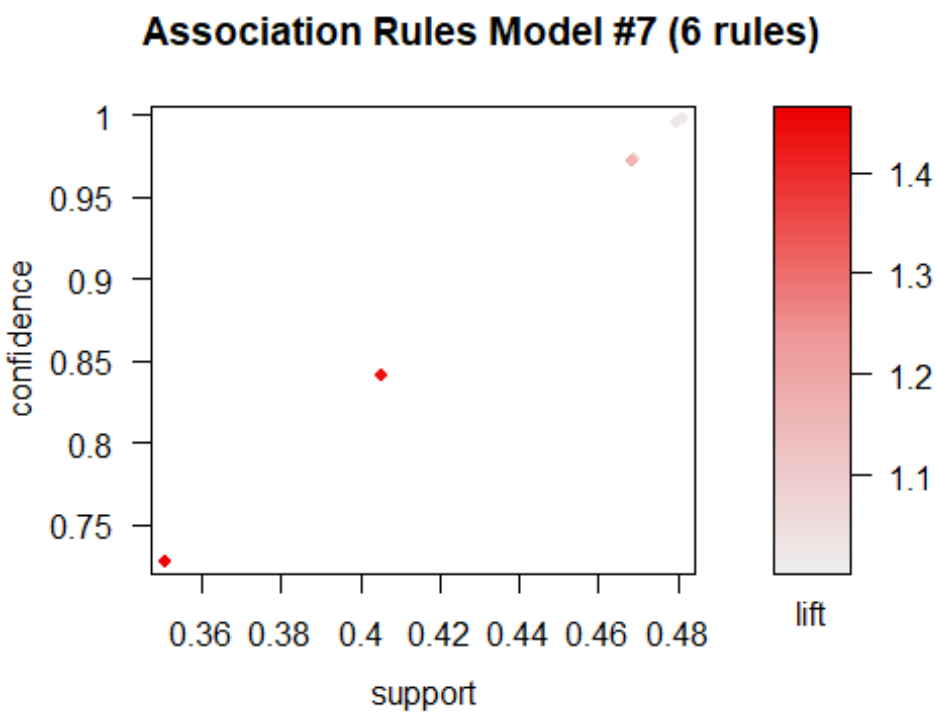
```r
################
# Arm Model 6  #
################

## Make a slightly larger model of Edible for an interactive plot
rules<-arules::apriori(data = armDF, parameter = list(supp=0.30, conf = 0.90, minlen=2),
                       appearance = list(default="lhs",rhs="class=Edible"),
                       control = list(verbose = F))
rules <- rules[!is.redundant(rules)] ## Remove redundant rules
rules ## 17 rules

## set of 17 rules

plot(rules, main = "Association Rules Model #6 (17 rules)") ## Add title
```



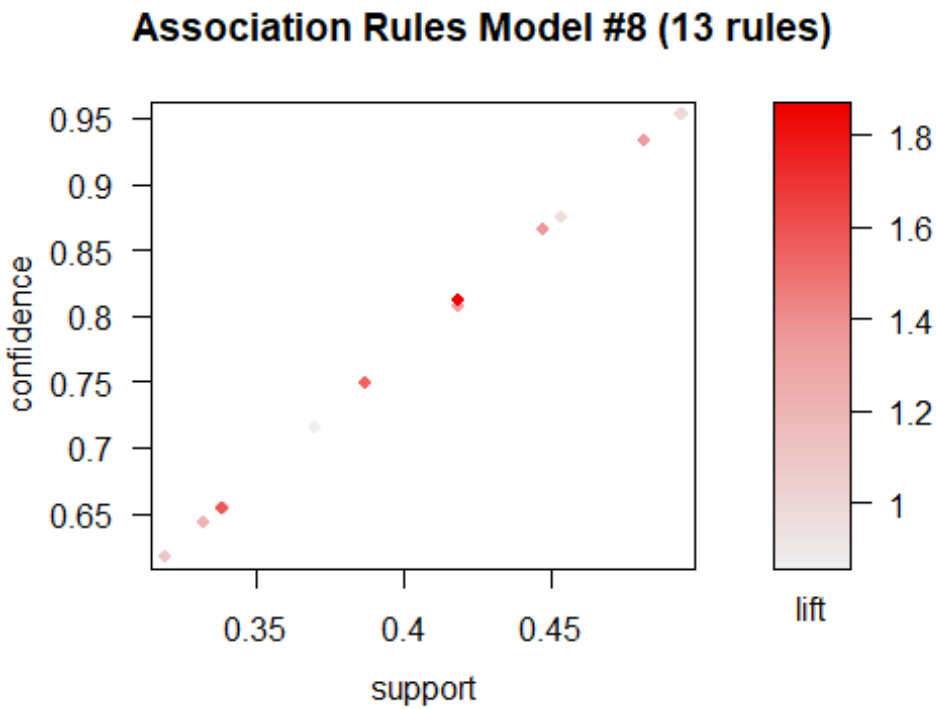
```r
## PLOT EDIBLE
plot(rules, method="graph", engine="interactive")
```



```r
################
# Arm Model 7  #
################
# Per Instructions, set the LHS
# I'll try setting LHS to Poisonous
rules<-arules::apriori(data = armDF, parameter = list(supp=0.01, conf = 0.60, minlen=2),
                       appearance = list(lhs = "class=Poisonous",default = "rhs"),
                       control = list(verbose = F))
options(digits=3)
rules <- rules[!is.redundant(rules)]  # Remove redundant rules
rules

## set of 6 rules

# Produced 6 rules

plot(rules, main = "Association Rules Model #7 (6 rules)")  # Add title
```



```r
# Sort by confidence & inspect
rules_conf<-sort (rules, decreasing = TRUE, by='confidence')
inspect(rules_conf)

#      lhs                    rhs                     support confidence lift count
# [1] {class=Poisonous} => {veil.color=White}     0.481   0.998      1.02 3908
# [2] {class=Poisonous} => {gill.attachment=Free} 0.480   0.995      1.02 3898
# [3] {class=Poisonous} => {ring.number=One}      0.469   0.972      1.06 3808
# [4] {class=Poisonous} => {gill.spacing=Close}   0.468   0.971      1.16 3804
# [5] {class=Poisonous} => {bruises=No}           0.405   0.841      1.44 3292
# [6] {class=Poisonous} => {population=Several}   0.351   0.727      1.46 2848

# Sort by support
rules_supp <- sort(rules, decreasing = TRUE, by="supp")
inspect(rules_supp)


# Sort by lift
rules_lift <- sort(rules, decreasing = TRUE,by="lift")
inspect(rules_lift)





################
# Arm Model 8  #
################
# Now try setting LHS to Edible
rules<-arules::apriori(data = armDF, parameter = list(supp=0.01, conf = 0.60, minlen=2),
                       appearance = list(lhs = "class=Edible",default = "rhs"),
                       control = list(verbose = F))
options(digits=3)
rules <- rules[!is.redundant(rules)]  # Remove redundant rules
rules

## set of 13 rules

# Produced 13 rules

plot(rules, main = "Association Rules Model #8 (13 rules)")  # Add title

## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```




```r
# Sort by confidence & inspect
rules_conf<-sort (rules, decreasing = TRUE, by='confidence')
inspect(rules_conf[1:10])

# lhs                     rhs                                support confidence lift  count
# [1]  {class=Edible} => {gill.attachment=Free}             0.494   0.954      0.980 4016
# [2]  {class=Edible} => {veil.color=White}                 0.494   0.954      0.978 4016
# [3]  {class=Edible} => {gill.size=Broad}                  0.483   0.932      1.349 3920
# [4]  {class=Edible} => {ring.number=One}                  0.453   0.875      0.949 3680
# [5]  {class=Edible} => {stalk.surface.above.ring=Smooth} 0.448   0.865      1.358 3640
# [6]  {class=Edible} => {odor=None}                        0.419   0.810      1.865 3408
# [7]  {class=Edible} => {stalk.surface.below.ring=Smooth} 0.419   0.808      1.330 3400
# [8]  {class=Edible} => {ring.type=Pendant}                0.388   0.749      1.534 3152
# [9]  {class=Edible} => {gill.spacing=Close}               0.370   0.715      0.853 3008
# [10] {class=Edible} => {bruises=Bruises}                  0.339   0.654      1.574 2752
```

```r
# Sort by support
rules_supp <- sort(rules, decreasing = TRUE, by="supp")
inspect(rules_supp[1:10])

# lhs                     rhs                                support confidence lift  count
# [1]  {class=Edible} => {gill.attachment=Free}            0.494   0.954      0.980 4016
# [2]  {class=Edible} => {veil.color=White}                0.494   0.954      0.978 4016
# [3]  {class=Edible} => {gill.size=Broad}                 0.483   0.932      1.349 3920
# [4]  {class=Edible} => {ring.number=One}                 0.453   0.875      0.949 3680
# [5]  {class=Edible} => {stalk.surface.above.ring=Smooth} 0.448   0.865      1.358 3640
# [6]  {class=Edible} => {odor=None}                       0.419   0.810      1.865 3408
# [7]  {class=Edible} => {stalk.surface.below.ring=Smooth} 0.419   0.808      1.330 3400
# [8]  {class=Edible} => {ring.type=Pendant}               0.388   0.749      1.534 3152
# [9]  {class=Edible} => {gill.spacing=Close}              0.370   0.715      0.853 3008
# [10] {class=Edible} => {bruises=Bruises}                 0.339   0.654      1.574 2752

# Sort by lift
rules_lift <- sort(rules, decreasing = TRUE,by="lift")
inspect(rules_lift[1:10])

# lhs               rhs                                support confidence lift count
# [1]  {class=Edible} => {odor=None}                       0.419   0.810      1.86 3408
# [2]  {class=Edible} => {bruises=Bruises}                 0.339   0.654      1.57 2752
# [3]  {class=Edible} => {ring.type=Pendant}               0.388   0.749      1.53 3152
# [4]  {class=Edible} => {stalk.surface.above.ring=Smooth} 0.448   0.865      1.36 3640
# [5]  {class=Edible} => {gill.size=Broad}                 0.483   0.932      1.35 3920
# [6]  {class=Edible} => {stalk.surface.below.ring=Smooth} 0.419   0.808      1.33 3400
# [7]  {class=Edible} => {stalk.color.below.ring=White}    0.333   0.643      1.19 2704
# [8]  {class=Edible} => {stalk.color.above.ring=White}    0.339   0.654      1.19 2752
# [9]  {class=Edible} => {stalk.shape=Tapering}            0.319   0.616      1.09 2592
# [10] {class=Edible} => {gill.attachment=Free}            0.494   0.954      0.98 4016



################################################################################
# End of project content
################################################################################
```

```
# Data Notes

# Mushroom Data can be found on both Kaggle and Machine Learning Repository
  # https://www.kaggle.com/uciml/mushroom-classification
  # https://archive.ics.uci.edu/ml/datasets/mushroom

# Book #1
  # Title: Mushrooms: how to grow them a practical treatise on mushroom culture for profit and pleasure
  # Author/Year: William Falconer, 1892
  # can be found on Project Gutenberg: https://www.gutenberg.org/ebooks/24944

# Book #2:
  # Title: The Mushroom Cultivator: A Practical Guide to  Growing Mushrooms at Home
  # Author/Year: by Paul Stamets and J.S. Chilton, 1983
```