1R Mushroom Example

Name: Perry Koob

eMail: koobp@mst.edu

Course: CS 5402

Date: 2020-06-21

Course: Version 1.3.2

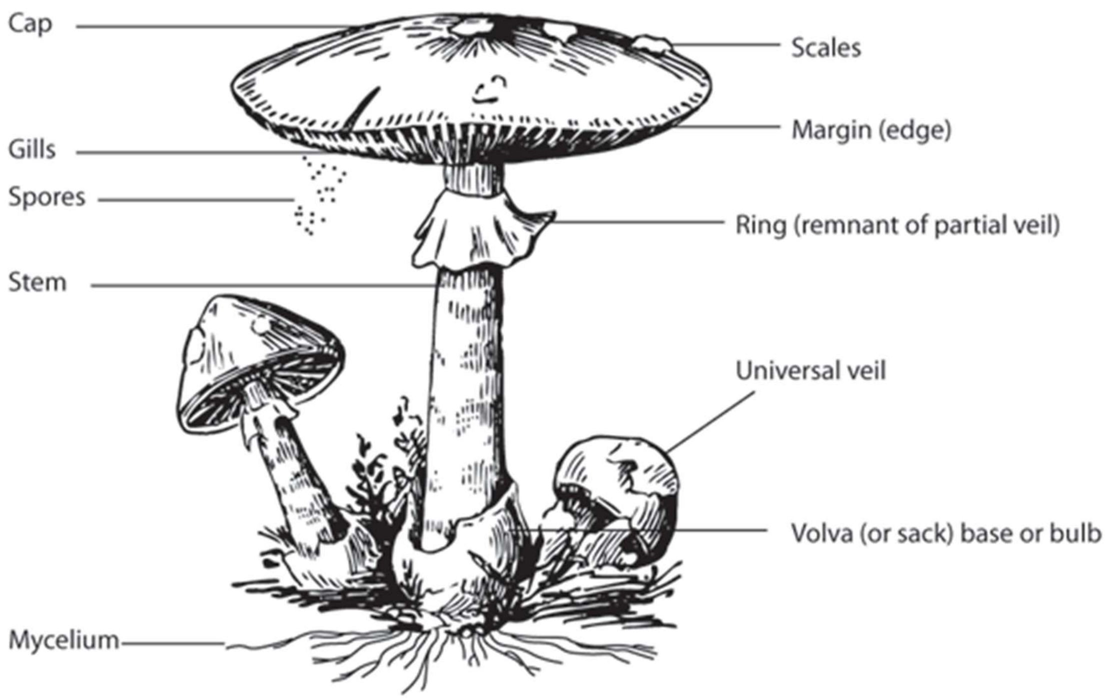
Concept Description:

Train a system from existing data to classify mushrooms as either edible or inedible.

Data Collection:

The data has been provided by Perry B. Koob, not professor or doctor. It is a modified version of the UCI Mushroom data set found here: https://archive.ics.uci.edu/ml/datasets/mushroom

Example Description:



Mushroom Diagram

edible.poisonous

This is the class label.

cap.shape

Nominal attribute that describes the cap shape of the mushroom as:

bell
conical convex
flat
knobbed sunken

cap.surface

Nominal attribute that describes the cap surface of the mushroom as:

fibrous grooves scaly
smooth

cap.color

Nominal attribute that describes the cap color of the mushroom as:

brown
buff
cinnamon green
grey
pink
purple
red
white
yellow

bruises

Nominal attribute, boolean in nature, that describes if the mushroom has bruises.

odor

Nominal attribute that describes the odor of the mushroom as:

almond
anise
creosote fishy
foul
musty
none
pungent spicy

colony

Interval attribute that describes the approximate size of the mushroom colony:

edible

This is a binary class label generated from the edible.poisonous class label. It is a transformation of a nominal Class label, so it is also nominal.The labels are now edible or inedible.

There are no missing values.

Data Import and Wrangling:

```
df <- read.csv(file = '../src-data/mushroom.csv',stringsAsFactors=TRUE)
```

Create a new Class in the training dataset

```
df$edible <- ifelse(df$edible.poisonous == 'edible', "edible", "inedible")
df$edible <- as.factor(df$edible)
```

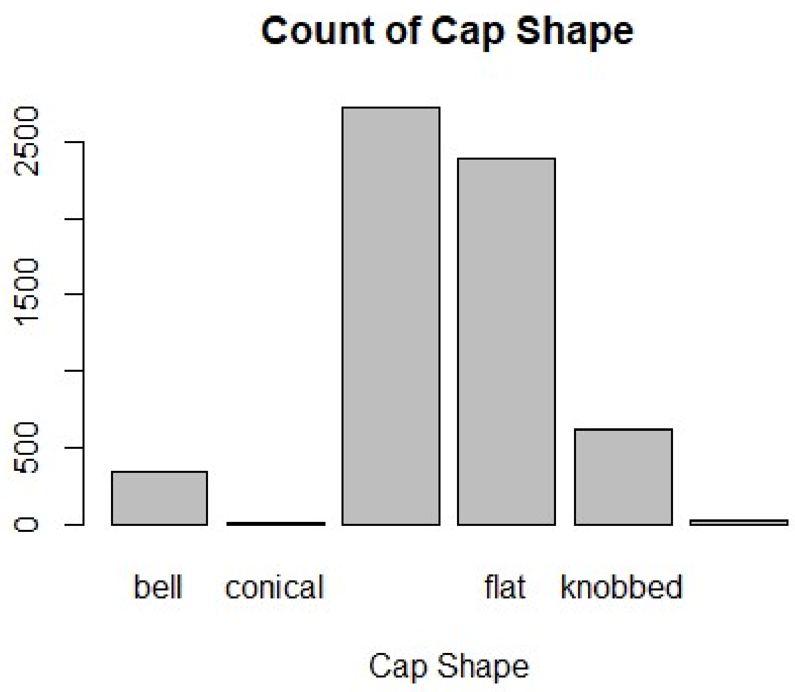
Partition the data into a training set and a test set using a 75/25 split.

```
set.seed(123)
size = floor(0.75*nrow(df))

train_index = sample(seq_len(nrow(df)),size = size)

train = df[train_index,]
test=df[-train_index,]
```

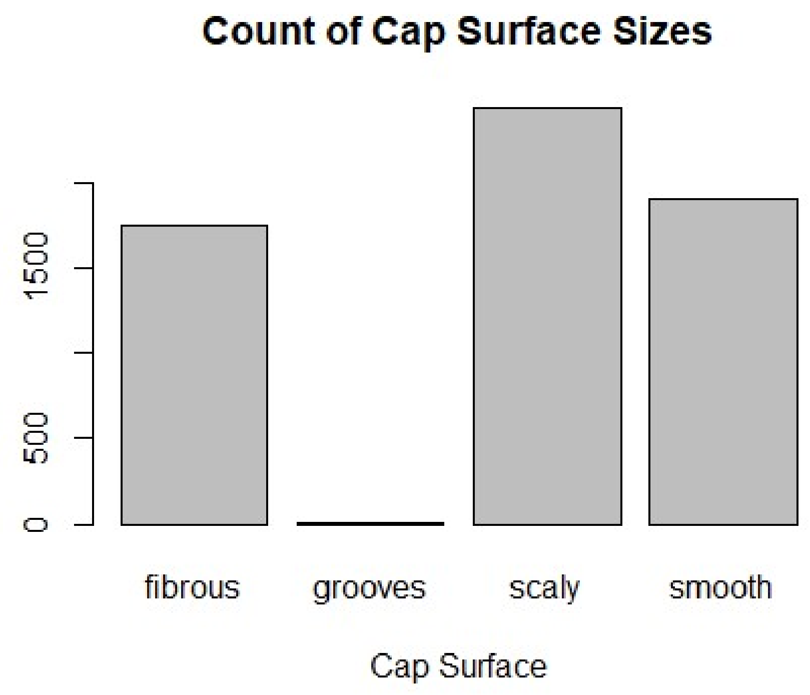
Exploratory Data Analysis:

Looking into what type of measure the attributes are.

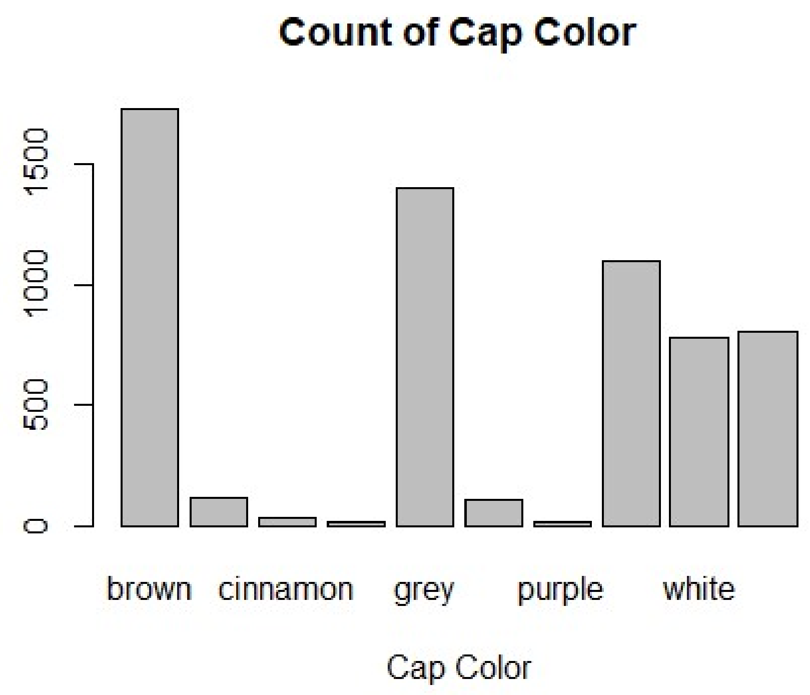
```
summary(df)

##   edible.poisonous   cap.shape     cap.surface     cap.color       bruises
##  edible   :4208    bell    : 452   fibrous:2320   brown   :2284   
bruises:3376
##  poisonous:3916    conical:   4   grooves:   4   grey    :1840   no
:4748
##                    convex :3656   scaly  :3244   red     :1500  
##                    flat    :3152   smooth :2556   yellow :1072  
##                    knobbed: 828                  white   :1040  
##                    sunken :  32                  buff    : 168  
##                                                  (Other): 220  
##       odor          colony            edible
##  none   :3528   Min.    :  1.00   edible   :4208
##  foul   :2160   1st Qu.:  1.00   inedible:3916
##  fishy  : 576   Median :  2.00
##  spicy  : 576   Mean   : 46.47
##  almond : 400   3rd Qu.:  7.00
##  anise  : 400   Max.   :999.00
##  (Other): 484

#Cap Shape
count <- table (train$cap.shape)
barplot(count,main="Count of Cap Shape", xlab="Cap Shape")
```



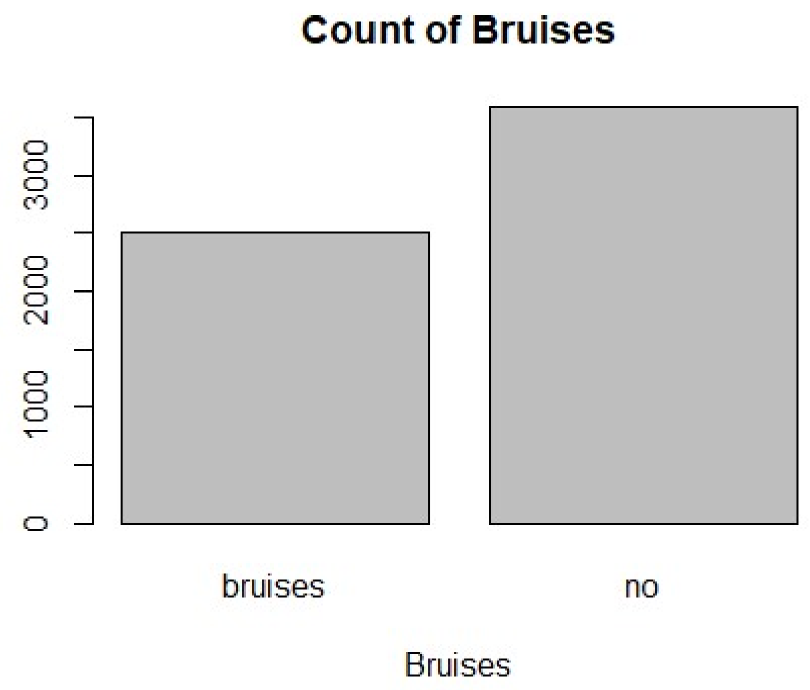
```
#Cap Surface
count <- table (train$cap.surface)
barplot(count,main="Count of Cap Surface Sizes", xlab="Cap Surface")
```



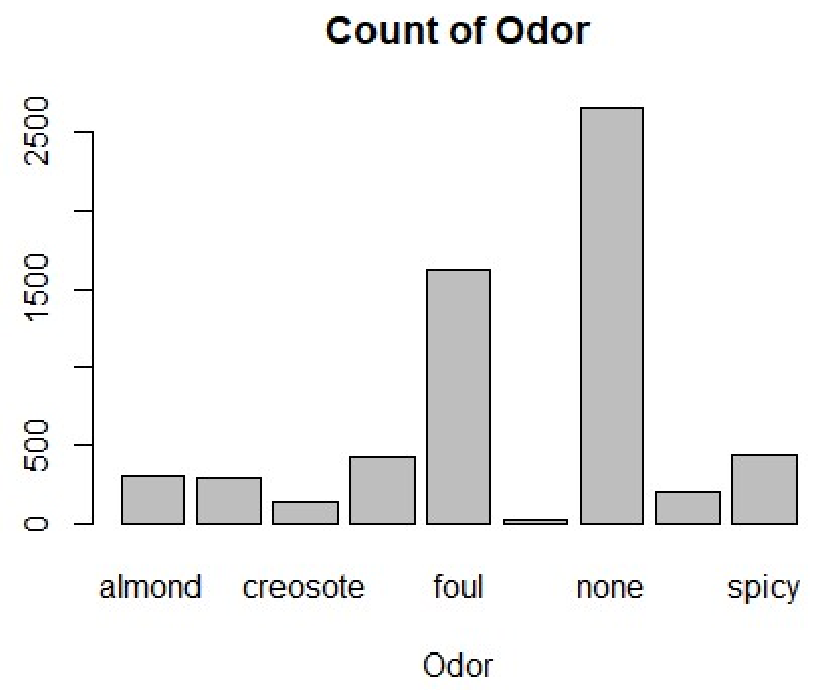
```
#Cap Color
count <- table (train$cap.color)
barplot(count,main="Count of Cap Color", xlab="Cap Color")
```



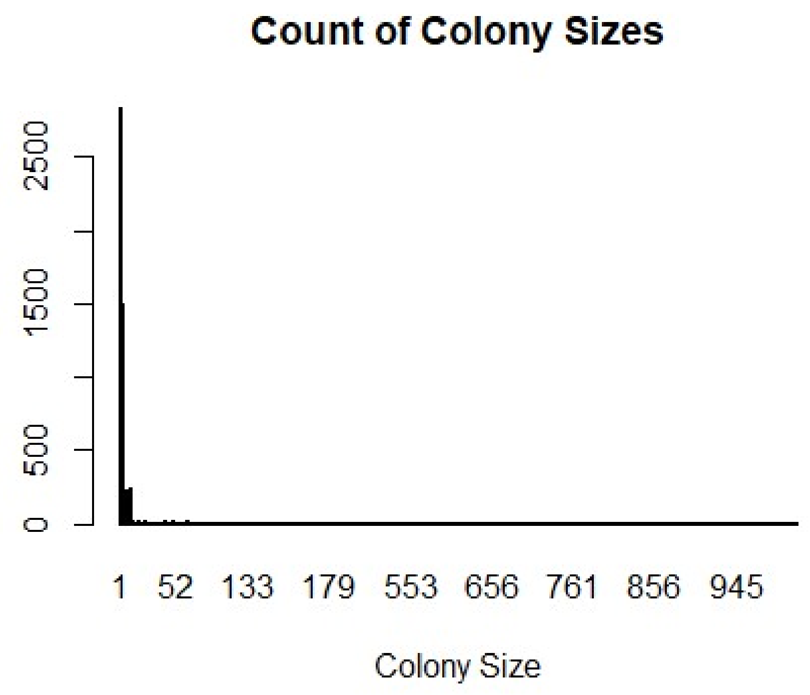
```r
#Bruises
count <- table (train$bruises)
barplot(count,main="Count of Bruises", xlab="Bruises")
```



```r
#Odor
count <- table (train$odor)
barplot(count,main="Count of Odor", xlab="Odor")
```



```
#Determine what the missing attribure label is for odor
summary(train$odor)

##   almond    anise creosote    fishy     foul    musty     none  pungent
##      299      293      140      427     1628       20     2659      197
##    spicy
##      430

#Colony
count <- table (train$colony)
barplot(count,main="Count of Colony Sizes", xlab="Colony Size")
```



Looking at the bar chart we can see Colony Size is not nominal. Since we need nominal values to be able to run 1R, we are going to need to discretize the Colony Size.

```
str(train$colony)

##  int [1:6093] 1 1 855 8 1 2 1 6 1 1 ...
```

Discretizing the Colony attribute

Discretizing Using the Holte method

```
# Holte discretization has not been updated for R version 4
#disc.1r(df)
```

Discretizing Using frequency

Since the Holte method of Discretization is not working for this version, we will use another discretization method. The Frequency based method.

```
discretize(train$colony,onlycuts = TRUE)

## Warning in discretize(train$colony, onlycuts = TRUE): The calculated
breaks are: 1, 1, 2, 999
##   Only unique breaks are used reducing the number of intervals. Look at ?
discretize for details.

## [1]   1   2 999

train$dcolony[train$colony <=2] <- "single"
train$dcolony[train$colony > 2] <- "colony"
```

```
train$dcolony <- as.factor(train$dcolony)

test$dcolony[test$colony <=2] <- "single"
test$dcolony[test$colony > 2] <- "colony"
test$dcolony <- as.factor(test$dcolony)
```

Mining or Analytics:

Getting the 1R rules using a manual method.

```
# Initialize prediction data frame with the train data frame.
prediction <- train
```

Cap Shape

```
# Get the frequency of the Cap.shapes and their frequency by classes values
cap.shape.count<- as.data.frame(table(train$cap.shape))
cap.shape.true <- as.data.frame(table(train[train$edible ==
"edible",]$cap.shape))
cap.shape.false <- as.data.frame(table(train[train$edible ==
"inedible",]$cap.shape))

# Rename the columns so the make sense after the merge
colnames(cap.shape.count) <- c("Label","Freq")
colnames(cap.shape.true) <- c("Label","Edible")
colnames(cap.shape.false) <- c("Label","Inedible")

# Merge the three dataframes together
cap.shape <- merge(cap.shape.count,cap.shape.true, by="Label",all=TRUE)
cap.shape <- merge(cap.shape,cap.shape.false, by="Label",all=TRUE)
cap.shape

##      Label Freq Edible Inedible
## 1     bell  342    310       32
## 2 conical    3      0        3
## 3   convex 2721   1447     1274
## 4     flat 2388   1205     1183
## 5 knobbed  614    170      444
## 6   sunken   25     25        0
```

Base on the frequency of edible and inedible, we would generate the following rule set for the Cap Shape attribute:

bell -> Edible
conical -> Inedible
convex -> Edible
flat -> Edible
knobbed -> Inedible
sunken -> Edible

Now we predict what the Cap Shape rules would classify our training data as.

```
prediction$pred.cap.shape <-NA

prediction$pred.cap.shape[prediction$cap.shape == "bell"] <- "edible"
prediction$pred.cap.shape[prediction$cap.shape == "conical"] <- "inedible"
prediction$pred.cap.shape[prediction$cap.shape == "convex"] <- "edible"
prediction$pred.cap.shape[prediction$cap.shape == "flat"] <- "edible"
prediction$pred.cap.shape[prediction$cap.shape == "knobbed"] <- "inedible"
prediction$pred.cap.shape[prediction$cap.shape == "sunken"] <- "edible"

prediction$cap.shape <-NULL

prediction$pred.cap.shape <- as.factor(prediction$pred.cap.shape)
```

Cap Surface

```
cap.surface.count<- as.data.frame(table(train$cap.surface))
cap.surface.true <- as.data.frame(table(train[train$edible ==
"edible",]$cap.surface))
cap.surface.false <- as.data.frame(table(train[train$edible ==
"inedible",]$cap.surface))

colnames(cap.surface.count) <- c("Label","Freq")
colnames(cap.surface.true) <- c("Label","Edible")
colnames(cap.surface.false) <- c("Label","Inedible")


cap.surface <- merge(cap.surface.count,cap.surface.true, by="Label",all=TRUE)
cap.surface <- merge(cap.surface,cap.surface.false, by="Label",all=TRUE)
cap.surface

##     Label Freq Edible Inedible
## 1 fibrous 1754   1196      558
## 2 grooves    3      0        3
## 3   scaly 2434   1112     1322
## 4  smooth 1902    849     1053
```

Base on the frequency of edible and inedible, we would generate the following rule set for the Cap Surface attribute:

fibrous -> Edible
grooves -> Inedible
scaly -> Inedible
smooth -> Inedible

Now we predict what the Cap Surface rules would classify our training data as.

```
prediction$pred.cap.surface <-NA

prediction$pred.cap.surface[prediction$cap.surface == "fibrous"] <- "edible"
prediction$pred.cap.surface[prediction$cap.surface == "grooves"] <-
```

```
"inedible"
prediction$pred.cap.surface[prediction$cap.surface == "scaly"] <- "inedible"
prediction$pred.cap.surface[prediction$cap.surface == "smooth"] <- "inedible"

prediction$cap.surface <-NULL

prediction$pred.cap.surface <- as.factor(prediction$pred.cap.surface)
```

Cap Color

```
cap.color.count<- as.data.frame(table(train$cap.color))
cap.color.true <- as.data.frame(table(train[train$edible ==
"edible",]$cap.color))
cap.color.false <- as.data.frame(table(train[train$edible ==
"inedible",]$cap.color))

colnames(cap.color.count) <- c("Label","Freq")
colnames(cap.color.true) <- c("Label","Edible")
colnames(cap.color.false) <- c("Label","Inedible")


cap.color <- merge(cap.color.count,cap.color.true, by="Label",all=TRUE)
cap.color <- merge(cap.color,cap.color.false, by="Label",all=TRUE)
cap.color

##       Label Freq Edible Inedible
## 1     brown 1729    945      784
## 2      buff  118     34       84
## 3  cinnamon   32     25        7
## 4     green   16     16        0
## 5      grey 1400    788      612
## 6      pink  106     40       66
## 7    purple    9      9        0
## 8       red 1098    454      644
## 9     white  781    550      231
## 10   yellow  804    296      508
```

Base on the frequency of edible and inedible, we would generate the following rule set for the Cap Color attribute:

brown -> Edible
buff -> Inedible
cinnamon -> Edible
green -> Edible
grey -> Edible
pink -> Inedible
purple -> Edible
red -> Inedible

white -> Edible
yellow -> Inedible

Now we predict what the Cap Color rules would classify our training data as.

```
prediction$pred.cap.color <- NA

prediction$pred.cap.color[prediction$cap.color=="brown"] <- "edible"
prediction$pred.cap.color[prediction$cap.color=="buff"] <- "inedible"
prediction$pred.cap.color[prediction$cap.color=="cinnamon"] <- "edible"
prediction$pred.cap.color[prediction$cap.color=="green"] <- "edible"
prediction$pred.cap.color[prediction$cap.color=="grey"] <- "edible"
prediction$pred.cap.color[prediction$cap.color=="pink"] <- "inedible"
prediction$pred.cap.color[prediction$cap.color=="purple"] <- "edible"
prediction$pred.cap.color[prediction$cap.color=="red"] <- "inedible"
prediction$pred.cap.color[prediction$cap.color=="white"] <- "edible"
prediction$pred.cap.color[prediction$cap.color=="yellow"] <- "inedible"

prediction$cap.color <-  NULL
prediction$pred.cap.color <- as.factor(prediction$pred.cap.color)
```

Bruises

```
bruises.count<- as.data.frame(table(train$bruises))
bruises.true <- as.data.frame(table(train[train$edible ==
"edible",]$bruises))
bruises.false <- as.data.frame(table(train[train$edible ==
"inedible",]$bruises))

colnames(bruises.count) <- c("Label","Freq")
colnames(bruises.true) <- c("Label","Edible")
colnames(bruises.false) <- c("Label","Inedible")


bruises <- merge(bruises.count,bruises.true, by="Label",all=TRUE)
bruises <- merge(bruises,bruises.false, by="Label",all=TRUE)
bruises

##     Label Freq Edible Inedible
## 1 bruises 2506   2044      462
## 2      no 3587   1113     2474
```

Base on the frequency of edible and inedible, we would generate the following rule set for the Cap Color attribute:

bruises -> Edible
no -> Inedible

Now we predict what the Bruises rules would classify our training data as.

```
prediction$pred.bruises <- NA

prediction$pred.bruises[prediction$bruises=="bruises"] <- "edible"
prediction$pred.bruises[prediction$bruises=="no"] <- "inedible"


prediction$bruises <-  NULL
prediction$pred.bruises <- as.factor(prediction$pred.bruises)
```

Odor

```
odor.count<- as.data.frame(table(train$odor))
odor.true <- as.data.frame(table(train[train$edible == "edible",]$odor))
odor.false <- as.data.frame(table(train[train$edible == "inedible",]$odor))

colnames(odor.count) <- c("Label","Freq")
colnames(odor.true) <- c("Label","Edible")
colnames(odor.false) <- c("Label","Inedible")

odor <- merge(odor.count,odor.true, by="Label",all=TRUE)
odor <- merge(odor,odor.false, by="Label",all=TRUE)
odor

##      Label Freq Edible Inedible
## 1   almond  299    299        0
## 2    anise  293    293        0
## 3 creosote  140      0      140
## 4    fishy  427      0      427
## 5     foul 1628      0     1628
## 6    musty   20      0       20
## 7     none 2659   2565       94
## 8  pungent  197      0      197
## 9    spicy  430      0      430
```

Base on the frequency of edible and inedible, we would generate the following rule set for the Odor attribute:

almond -> edible
anise -> edible
creosote -> inedible
fishy -> inedible
foul -> inedible
musty -> inedible
none -> edible
pungent -> inedible
spicy -> inedible

Now we predict what the Odor rules would classify our training data as.

```
prediction$pred.odor <- NA

prediction$pred.odor[prediction$odor=="almond"] <- "edible"
prediction$pred.odor[prediction$odor=="anise"] <- "edible"
prediction$pred.odor[prediction$odor=="creosote"] <- "inedible"
prediction$pred.odor[prediction$odor=="fishy"] <- "inedible"
prediction$pred.odor[prediction$odor=="foul"] <- "inedible"
prediction$pred.odor[prediction$odor=="musty"] <- "inedible"
prediction$pred.odor[prediction$odor=="none"] <- "edible"
prediction$pred.odor[prediction$odor=="pungent"] <- "inedible"
prediction$pred.odor[prediction$odor=="spicy"] <- "inedible"

prediction$odor <-  NULL
prediction$pred.odor <- as.factor(prediction$pred.odor)
```

The error rate will be determined for each Rule Set to determine which one is better.

```
# Determine when the rule predict incorrectly
prediction$pred.cap.shape.correct = ifelse(prediction$edible ==
prediction$pred.cap.shape,0,1)
prediction$pred.cap.surface.correct = ifelse(prediction$edible ==
prediction$pred.cap.surface,0,1)
prediction$pred.cap.color.correct = ifelse(prediction$edible ==
prediction$pred.cap.color,0,1)
prediction$pred.bruises.correct = ifelse(prediction$edible ==
prediction$pred.bruises,0,1)
prediction$pred.odor.correct = ifelse(prediction$edible ==
prediction$pred.odor,0,1)

# Calculate the error rate
print(paste("Error Rate of Cap Shape
Rules",sum(prediction$pred.cap.shape.correct)/nrow(prediction)))

## [1] "Error Rate of Cap Shape Rules 0.436402429016905"

print(paste("Error Rate of Cap Surface
Rules",sum(prediction$pred.cap.surface.correct) /nrow(prediction)))

## [1] "Error Rate of Cap Surface Rules 0.413425242081077"

print(paste("Error Rate of Cap Color
Rules",sum(prediction$pred.cap.color.correct)/nrow(prediction)))

## [1] "Error Rate of Cap Color Rules 0.403413753487609"

print(paste("Error Rate of Bruises
Rules",sum(prediction$pred.bruises.correct)/nrow(prediction)))

## [1] "Error Rate of Bruises Rules 0.258493353028065"

print(paste("Error Rate of Cap Odor
Rules",sum(prediction$pred.odor.correct)/nrow(prediction)))
```

```
## [1] "Error Rate of Cap Odor Rules 0.0154275397997702"
```

The Rule Set with the best error rate is the Odor Rules Set, which has an error rate of 0.015.

So we choose the the Odor Rules Set as out One Rule.

Getting the 1R rules using the OneR package.

```
df_train <- train
df_train$edible.poisonous <-  NULL

model <- OneR(df_train)

## Warning in OneR.data.frame(df_train): data contains unused factor levels

model_prediction <- predict(model,test)
```

Evaluation:

We can evaluate the health of the two models using the training data set and Confusion Matrices obtained from both models. The Accuracy rate and the Error rate will allow us to compare the models to each other.

```
#prepare the list of classes from the test data for evaluation
reference <- as.data.frame(test$edible)
colnames(reference) <- c("class")
reference <- as.factor(reference$class)

#prepare the list of predictions from the test data for evaluation
testing <- as.data.frame(test[c("odor")])
colnames(testing) <- c("odor")

testing$pred[testing$odor=="almond"] <- "edible"
testing$pred[testing$odor=="anise"] <- "edible"
testing$pred[testing$odor=="creosote"] <- "inedible"
testing$pred[testing$odor=="fishy"] <- "inedible"
testing$pred[testing$odor=="foul"] <- "inedible"
testing$pred[testing$odor=="musty"] <- "inedible"
testing$pred[testing$odor=="none"] <- "edible"
testing$pred[testing$odor=="pungent"] <- "inedible"
testing$pred[testing$odor=="spicy"] <- "inedible"

testing$pred <- as.factor(testing$pred)

confusionMatrix(testing$pred, reference)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction edible inedible
##   edible     1051       26
##   inedible      0      954
##                                          
##                Accuracy : 0.9872          
##                  95% CI : (0.9813, 0.9916)
##     No Information Rate : 0.5175          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                          
##                   Kappa : 0.9743          
##                                          
##  Mcnemar's Test P-Value : 9.443e-07       
##                                          
##             Sensitivity : 1.0000          
##             Specificity : 0.9735          
##          Pos Pred Value : 0.9759          
##          Neg Pred Value : 1.0000          
##              Prevalence : 0.5175          
##          Detection Rate : 0.5175          
##    Detection Prevalence : 0.5303          
##       Balanced Accuracy : 0.9867          
##                                          
##        'Positive' Class : edible          
##
```

```
eval_model(model_prediction, test)
```

```
## 
## Confusion matrix (absolute):
##           Actual
## Prediction colony single  Sum
##     colony    259     88  347
##     single    352   1332 1684
##     Sum       611   1420 2031
## 
## Confusion matrix (relative):
##           Actual
## Prediction colony single  Sum
##     colony   0.13   0.04 0.17
##     single   0.17   0.66 0.83
##     Sum      0.30   0.70 1.00
## 
## Accuracy:
## 0.7834 (1591/2031)
## 
## Error rate:
## 0.2166 (440/2031)
## 
## Error rate reduction (vs. base rate):
## 0.2799 (p-value < 2.2e-16)
```

The accuracy rate of the 1R Rules generated by hand is 98% and generated by the oneR package is 78%. So both are good models.

Results:

Since both the hand generated 1R classifier and the oneR package generated the same results, a 1R classifier based on the Odor attribute, and they both have a acceptaible accuracy with training and test data, we can feel confident that our model is a strong one.

Our final model is…

If the odor of the mushroom is:

almond -> edible
anise -> edible
creosote -> inedible
fishy -> inedible
foul -> inedible
musty -> inedible
none -> edible
pungent -> inedible
spicy -> inedible

References:

https://stackoverflow.com/questions/7739578/merge-data-frames-based-on-rownames-in-r https://www.datanovia.com/en/lessons/rename-data-frame-columns-in-r/
https://www.dummies.com/programming/r/how-to-convert-tables-to-a-data-frame-in-r/
https://wilkelab.org/classes/SDS348/2016_spring/labs/remove_variables.html
https://stackoverflow.com/questions/2854625/select-only-rows-if-its-value-in-a-particular-column-is-less-than-its-value-in-t
https://stackoverflow.com/questions/7072159/how-do-you-remove-columns-from-a-data-frame https://stats.stackexchange.com/questions/5253/how-do-i-get-the-number-of-rows-of-a-data-frame-in-r
https://www.rdocumentation.org/packages/caret/versions/3.45
https://www.rdocumentation.org/packages/arules/versions/1.6-8/topics/discretize