CPTR330 – Homework 3

Pierre Visconti

April 17, 2023

C5.0 Decision Tree Algorithm

Decision tree algorithms are a classification algorithm that uses a tree structure to create a model. The algorithm builds the model using what is known as divide and conquer, which is a method of recursively breaking a problem into sub-problems. This recursive partitioning is what gives the algorithm its name, as when you represent the model visually it looks like a tree. Some strengths of decision trees is that they can provide a model with a resulting structure that is easy for humans to understand, and that they be applied on almost any type of data with often good results. Decision trees also exclude unimportant features unlike other algorithms like naive bayes. Despite these strengths though decision trees do suffer from some negatives, mainly that datasets with a large amount of numeric features could result in an overcomplicated model and the model is prone to overfitting/underfitting.

Step 1 - Collect Data

The data is of gilled mushrooms of 23 species from two different families.The data has a lot of different variables on each mushroom that describes it's physical attributes and characteristics, habitat, color, and anything else that could be used to identify the mushroom. The data then includes one variable with two classes, edible and poisonous which is what we will try to predict. The distribution of edible and poisonous is roughly 50/50 and there are over 8,000 observations. The data comes from records drawn from The Audubon Society Field Guide to North American Mushrooms (1981). G. H. Lincoff (Pres.), New York: Alfred A. Knopf.

Step 2 - Exploring And Preparing The Data

Importing the data and adding column names

```
data = read.csv("agaricus-lepiota.data")
colnames(data) = c("type","cap_shape", "cap-surface", "cap-color", "bruises", "odor", "gill-attachment"
str(data)
```

```
## 'data.frame':    8123 obs. of  23 variables:
##  $ type                    : chr  "e" "e" "p" "e" ...
##  $ cap_shape               : chr  "x" "b" "x" "x" ...
##  $ cap-surface             : chr  "s" "s" "y" "s" ...
##  $ cap-color               : chr  "y" "w" "w" "g" ...
##  $ bruises                 : chr  "t" "t" "t" "f" ...
##  $ odor                    : chr  "a" "l" "p" "n" ...
##  $ gill-attachment         : chr  "f" "f" "f" "f" ...
##  $ gill-spacing            : chr  "c" "c" "c" "w" ...
##  $ gill-size               : chr  "b" "b" "n" "b" ...
##  $ gill-color              : chr  "k" "n" "n" "k" ...
##  $ stalk-shape             : chr  "e" "e" "e" "t" ...
##  $ stalk-root              : chr  "c" "c" "e" "e" ...
```

```
##  $ stalk-surface-above-ring: chr  "s" "s" "s" "s" ...
##  $ stalk-surface-below-ring: chr  "s" "s" "s" "s" ...
##  $ stalk-color-above-ring  : chr  "w" "w" "w" "w" ...
##  $ stalk-color-below-ring  : chr  "w" "w" "w" "w" ...
##  $ veil-type               : chr  "p" "p" "p" "p" ...
##  $ veil-color              : chr  "w" "w" "w" "w" ...
##  $ ring-number             : chr  "o" "o" "o" "o" ...
##  $ ring-type               : chr  "p" "p" "p" "e" ...
##  $ spore-print-color       : chr  "n" "n" "k" "n" ...
##  $ population              : chr  "n" "n" "s" "a" ...
##  $ habitat                 : chr  "g" "m" "u" "g" ...
```

```
data = as.data.frame(unclass(data),stringsAsFactors=TRUE)
str(data)
```

```
## 'data.frame':    8123 obs. of  23 variables:
##  $ type                    : Factor w/ 2 levels "e","p": 1 1 2 1 1 1 1 2 1 1 ...
##  $ cap_shape               : Factor w/ 6 levels "b","c","f","k",..: 6 1 6 6 6 1 1 6 1 6 ...
##  $ cap.surface             : Factor w/ 4 levels "f","g","s","y": 3 3 4 3 4 3 4 4 3 4 ...
##  $ cap.color               : Factor w/ 10 levels "b","c","e","g",..: 10 9 9 4 10 9 9 9 10 10 ...
##  $ bruises                 : Factor w/ 2 levels "f","t": 2 2 2 1 2 2 2 2 2 2 ...
##  $ odor                    : Factor w/ 9 levels "a","c","f","l",..: 1 4 7 6 1 1 4 7 1 4 ...
##  $ 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 2 1 1 1 1 1 1 ...
##  $ gill.size               : Factor w/ 2 levels "b","n": 1 1 2 1 1 1 1 2 1 1 ...
##  $ gill.color              : Factor w/ 12 levels "b","e","g","h",..: 5 6 6 5 6 3 6 8 3 3 ...
##  $ stalk.shape             : Factor w/ 2 levels "e","t": 1 1 1 2 1 1 1 1 1 1 ...
##  $ stalk.root              : Factor w/ 5 levels "?","b","c","e",..: 3 3 4 4 3 3 3 4 3 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 1 5 5 5 5 5 5 ...
##  $ spore.print.color       : Factor w/ 9 levels "b","h","k","n",..: 4 4 3 4 3 3 4 3 3 4 ...
##  $ population              : Factor w/ 6 levels "a","c","n","s",..: 3 3 4 1 3 3 4 5 4 3 ...
##  $ habitat                 : Factor w/ 7 levels "d","g","l","m",..: 2 4 6 2 2 4 4 2 4 2 ...
```

All of the variables in the dataset are categorical and nominal. Although they are all nominal, none of them have a lot of different levels. The data is already sufficiently prepared for the decision tree algorithm, except for the veil-type variable which only has one level and can be removed.

```
# removing veil.type variable
data = data[-17]
str(data)
```

```
## 'data.frame':    8123 obs. of  22 variables:
##  $ type                    : Factor w/ 2 levels "e","p": 1 1 2 1 1 1 1 2 1 1 ...
##  $ cap_shape               : Factor w/ 6 levels "b","c","f","k",..: 6 1 6 6 6 1 1 6 1 6 ...
##  $ cap.surface             : Factor w/ 4 levels "f","g","s","y": 3 3 4 3 4 3 4 4 3 4 ...
##  $ cap.color               : Factor w/ 10 levels "b","c","e","g",..: 10 9 9 4 10 9 9 9 10 10 ...
##  $ bruises                 : Factor w/ 2 levels "f","t": 2 2 2 1 2 2 2 2 2 2 ...
##  $ odor                    : Factor w/ 9 levels "a","c","f","l",..: 1 4 7 6 1 1 4 7 1 4 ...
##  $ 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 2 1 1 1 1 1 1 ...
##  $ gill.size              : Factor w/ 2 levels "b","n": 1 1 2 1 1 1 1 2 1 1 ...
##  $ gill.color             : Factor w/ 12 levels "b","e","g","h",..: 5 6 6 5 6 3 6 8 3 3 ...
##  $ stalk.shape            : Factor w/ 2 levels "e","t": 1 1 1 2 1 1 1 1 1 1 ...
##  $ stalk.root             : Factor w/ 5 levels "?","b","c","e",..: 3 3 4 4 3 3 3 4 3 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.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 1 5 5 5 5 5 5 ...
##  $ spore.print.color      : Factor w/ 9 levels "b","h","k","n",..: 4 4 3 4 3 3 4 3 3 4 ...
##  $ population             : Factor w/ 6 levels "a","c","n","s",..: 3 3 4 1 3 3 4 5 4 3 ...
##  $ habitat                : Factor w/ 7 levels "d","g","l","m",..: 2 4 6 2 2 4 4 2 4 2 ...
```

create training and testing datasets

```
# setting seed
set.seed(201)
# create training and test data by randomly sampling
train.size = round(nrow(data)*0.85) # 85% of the dataset used for training
train.ind = sample(1:nrow(data), train.size)
data_train = data[train.ind,-1]
data_test = data[-train.ind,-1]
# create labels for training and test data
data_train_labels = data[train.ind, 1]
data_test_labels = data[-train.ind, 1]
```

checking distribution of edible/poisonous for each dataset

```
prop.table(table(data_train_labels))
```

```
## data_train_labels
##         e         p
## 0.5193338 0.4806662
```

```
prop.table(table(data_test_labels))
```

```
## data_test_labels
##         e         p
## 0.5106732 0.4893268
```

data is ready to be used for training/testing

Step 3 - Training A Model On The Data

Training the model involves feeding the algorithm our training dataset, from which it will build a model that can then be used to make predictions. The algorithm will produce a model that can be easily interpreted by humans and disregard any non-important features for us. We have multiple parameters that can be changed during the training phase which will impact our final model, such as pruning, adaptive boosting, cost matrix, max layers.

Training the model with default parameters.

```
mushroom_model = C5.0(data_train, data_train_labels)
```

Step 4 - Evaluating Model Performance

```
mushroom_model
```

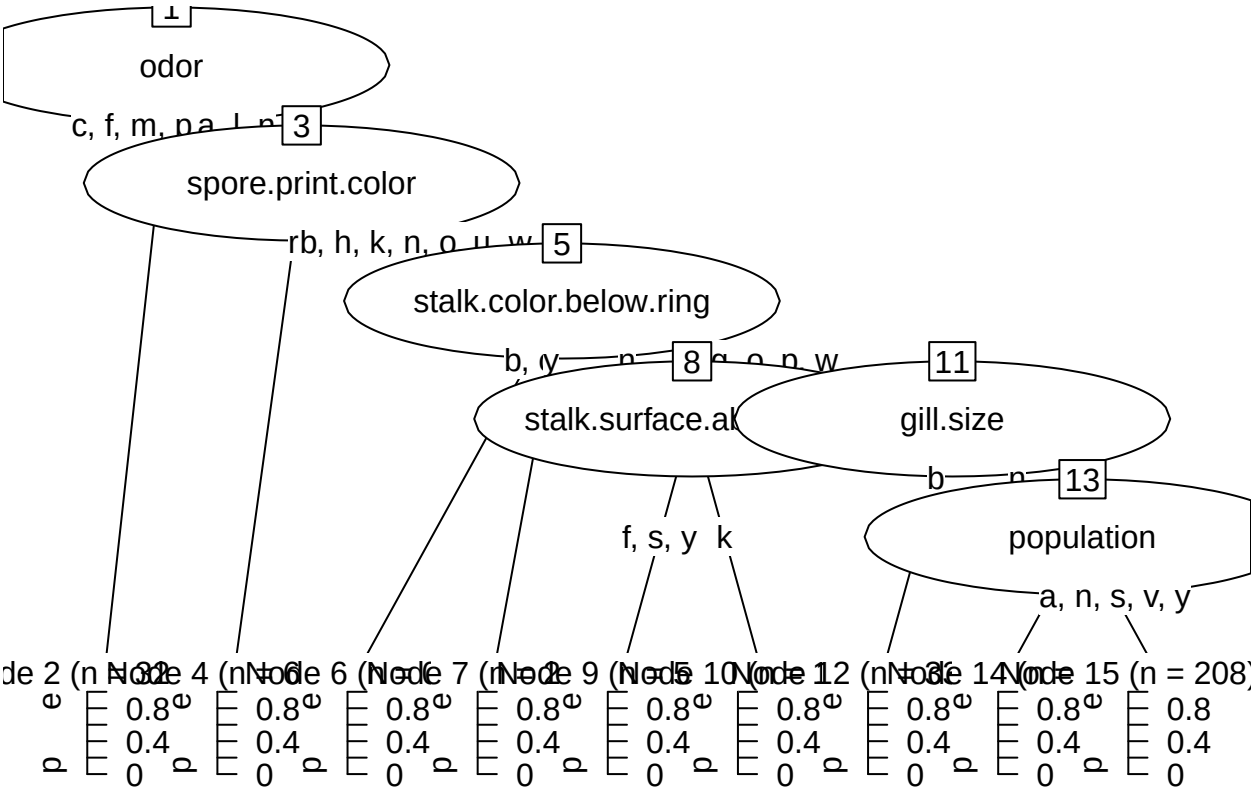
```
##
## Call:
## C5.0.default(x = data_train, y = data_train_labels)
##
## Classification Tree
## Number of samples: 6905
## Number of predictors: 21
##
## Tree size: 9
##
## Non-standard options: attempt to group attributes
```

```
summary(mushroom_model)
```

```
##
## Call:
## C5.0.default(x = data_train, y = data_train_labels)
##
##
## C5.0 [Release 2.07 GPL Edition]      Mon Apr 17 15:08:00 2023
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 6905 cases (22 attributes) from undefined.data
##
## Decision tree:
##
## odor in {c,f,m,p,s,y}: p (3216)
## odor in {a,l,n}:
## :...spore.print.color = r: p (60)
##     spore.print.color in {b,h,k,n,o,u,w,y}:
##     :...stalk.color.below.ring in {b,c}: e (0)
##         stalk.color.below.ring = y: p (21)
##         stalk.color.below.ring = n:
##         :...stalk.surface.above.ring in {f,s,y}: e (55)
##         :   stalk.surface.above.ring = k: p (14)
##         stalk.color.below.ring in {e,g,o,p,w}:
##         :...gill.size = b: e (3323)
##             gill.size = n:
##             :...population = c: p (8)
##                 population in {a,n,s,v,y}: e (208)
##
##
## Evaluation on training data (6905 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##       8    0( 0.0%)   <<
```

```
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##    3586          (a): class e
##          3319    (b): class p
##
##
##  Attribute usage:
##
##  100.00% odor
##   53.43% spore.print.color
##   52.56% stalk.color.below.ring
##   51.25% gill.size
##    3.13% population
##    1.00% stalk.surface.above.ring
##
##
## Time: 0.0 secs
```

```
plot(mushroom_model)
```



The model has a 0% error rate on the training set and built a tree with 6 branches and 9 nodes as can be seen in the decision tree section of the output and the graphical representation. There could be some overfitting that is occurring, which we can check by predicting test dataset.

Checking model performance on test set

```
predict_mushroom = predict(mushroom_model, data_test)
CrossTable(predict_mushroom, data_test_labels,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
```

```
           dnn = c('actual default', 'predicted default'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  1218
##
##
##                 | predicted default
## actual default |         e |         p | Row Total |
## ---------------|-----------|-----------|-----------|
##              e |       622 |         0 |       622 |
##                |     0.511 |     0.000 |           |
## ---------------|-----------|-----------|-----------|
##              p |         0 |       596 |       596 |
##                |     0.000 |     0.489 |           |
## ---------------|-----------|-----------|-----------|
##   Column Total |       622 |       596 |      1218 |
## ---------------|-----------|-----------|-----------|
##
##
```

The model correctly predicted every single object in the testing set for an accuracy of 100% which means that we do indeed have a model with very high accuracy.

Step 5 - Improving Model Performance

While the performance of the original model is very impressive at 100% accuracy, I believe that the model is overly complex with it's 6 branches and 9 nodes, that is a lot of information for a person to remember. The goal of the improvement step will be to simplify the model while still achieving very high accuracy, especially with false negatives. We do not want to identify a mushroom as edible if it is actually poisonous, but we could be a bit more lenient on identifying a mushroom as poisonous if it is actually edible if that helps simplify the model. To simplify the model there are several options possible. A different algorithm could be used which may generate a less complex model, or we could try to prune the model using our current algorithm. If our performance was not as good as our 100% there are parameters that could be tuned to try and improve the accuracy. We could implement adaptive boosting which assigns weights to wrongly classified points and retrains the model giving those points with a higher weight more importance. It will repeat this process for the given number of iterations with the goal of reducing model error. In this case of datasets where certain types of misclassifications are more important than others, we could also give the algorithm a cost matrix. For example with this dataset, misclassifying a poisonous mushroom as edible is clearly worse than misclassifying an edible one as poisonous. So by implementing a cost matrix we would give higher weight to the type of classification that matters the most and train the model with that in mind.

creating testing and training datasets with seed of 770

```
# setting seed
set.seed(770)
# create training and test data by randomly sampling
train.size = round(nrow(data)*0.85) # 85% of the dataset used for training
```

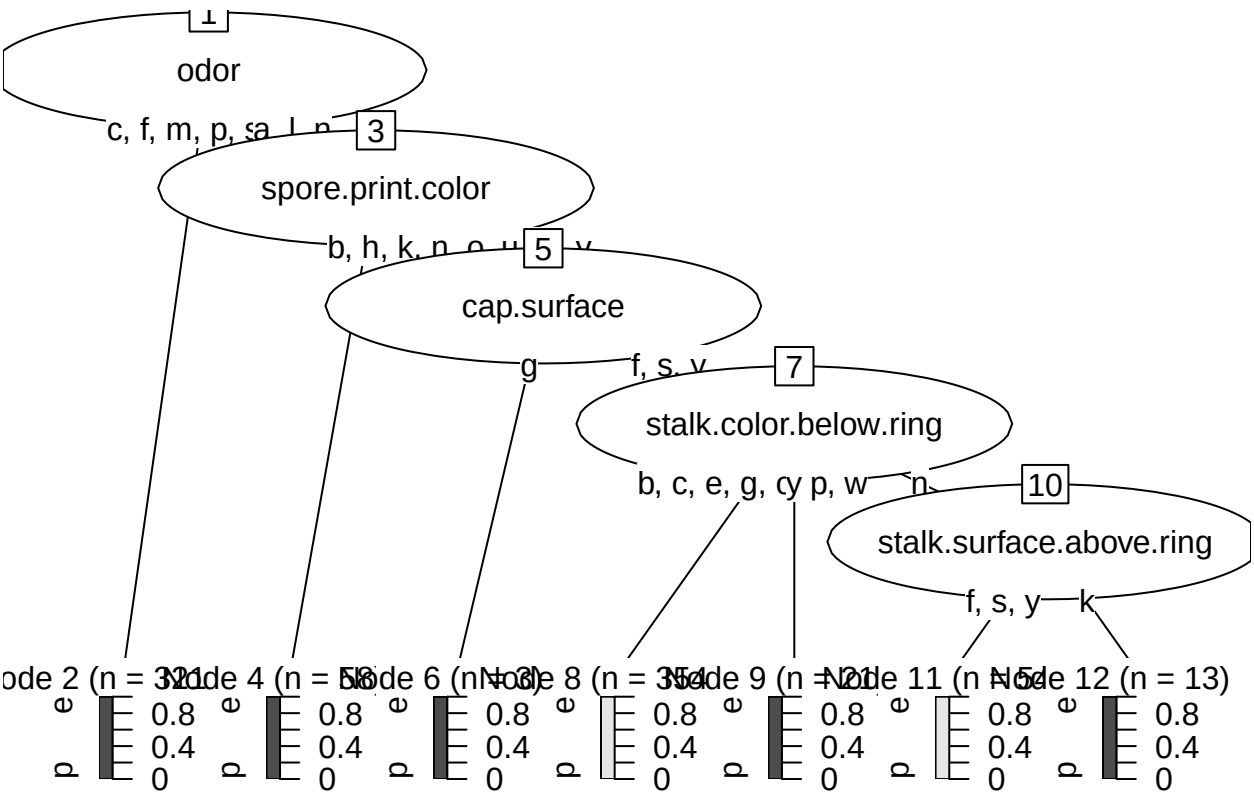
```
train.ind = sample(1:nrow(data), train.size)
data_train = data[train.ind,-1]
data_test = data[-train.ind,-1]
# create labels for training and test data
data_train_labels = data[train.ind, 1]
data_test_labels = data[-train.ind, 1]
```

checking accuracy of original model with this new seed

```
mushroom_model = C5.0(data_train, data_train_labels)
predict_mushroom = predict(mushroom_model, data_test)
CrossTable(predict_mushroom, data_test_labels,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  1218
##
##
##                | predicted default
## actual default |         e |         p | Row Total |
## ---------------|-----------|-----------|-----------|
##              e |       611 |         1 |       612 |
##                |     0.502 |     0.001 |           |
## ---------------|-----------|-----------|-----------|
##              p |         0 |       606 |       606 |
##                |     0.000 |     0.498 |           |
## ---------------|-----------|-----------|-----------|
##   Column Total |       611 |       607 |      1218 |
## ---------------|-----------|-----------|-----------|
##
##
```

```
plot(mushroom_model)
```



With the seed set to 770, our original model predicted 1217 out of 1218 objects correctly for an accuracy of 0.99. The model has 5 branches and 7 nodes so it is significantly less complex than with the original seed.

adding adaptive boosting with 10 iterations

```
mushroom_model = C5.0(data_train, data_train_labels, trials=10)
predict_mushroom = predict(mushroom_model, data_test)
CrossTable(predict_mushroom, data_test_labels,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  1218
##
##
##                | predicted default
## actual default |         e |         p | Row Total |
## ---------------|-----------|-----------|-----------|
##              e |       611 |         0 |       611 |
##                |     0.502 |     0.000 |           |
## ---------------|-----------|-----------|-----------|
##              p |         0 |       607 |       607 |
```

```
##                 |     0.000 |     0.498 |           |
## ----------------|-----------|-----------|-----------|
##   Column Total  |       611 |       607 |      1218 |
## ----------------|-----------|-----------|-----------|
##
##
```

```
mushroom_model
```

```
##
## Call:
## C5.0.default(x = data_train, y = data_train_labels, trials = 10)
##
## Classification Tree
## Number of samples: 6905
## Number of predictors: 21
##
## Number of boosting iterations: 10
## Average tree size: 6.5
##
## Non-standard options: attempt to group attributes
```

```
summary(mushroom_model)
```

```
##
## Call:
## C5.0.default(x = data_train, y = data_train_labels, trials = 10)
##
##
## C5.0 [Release 2.07 GPL Edition]      Mon Apr 17 15:08:04 2023
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 6905 cases (22 attributes) from undefined.data
##
## -----  Trial 0:  -----
##
## Decision tree:
##
## odor in {c,f,m,p,s,y}: p (3210)
## odor in {a,l,n}:
## :...spore.print.color = r: p (58)
##     spore.print.color in {b,h,k,n,o,u,w,y}:
##     :...cap.surface = g: p (3)
##         cap.surface in {f,s,y}:
##         :...stalk.color.below.ring in {b,c,e,g,o,p,w}: e (3546/3)
##             stalk.color.below.ring = y: p (21)
##             stalk.color.below.ring = n:
##             :...stalk.surface.above.ring in {f,s,y}: e (54)
##                 stalk.surface.above.ring = k: p (13)
##
## -----  Trial 1:  -----
##
## Decision tree:
```

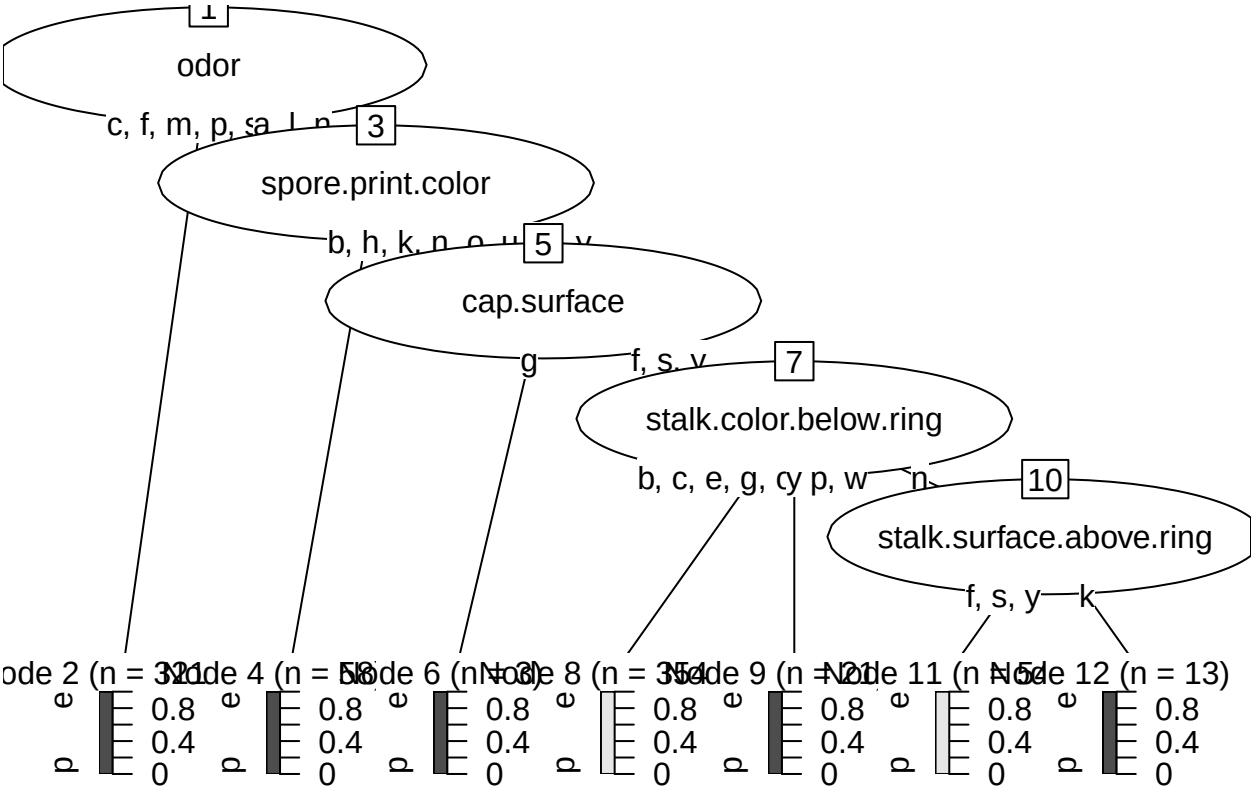
```
##
## spore.print.color in {b,k,n,o,u,y}:
## :...odor in {a,f,l,m,n,s,y}: e (2299.8)
## :   odor in {c,p}: p (268.5)
## spore.print.color in {h,r,w}:
## :...ring.number = t: e (381.8/43.5)
##     ring.number in {n,o}:
##     :...ring.type in {e,l,n,p}: p (3924.1/29.3)
##         ring.type = f: e (30.8)
##
## -----  Trial 2:  -----
##
## Decision tree:
##
## spore.print.color in {b,k,n,o,u,y}:
## :...odor in {a,f,l,m,n,s,y}: e (1731)
## :   odor in {c,p}: p (202.1)
## spore.print.color in {h,r,w}:
## :...stalk.color.below.ring in {g,o}: p (0)
##     stalk.color.below.ring in {e,n}: e (976.9/212.8)
##     stalk.color.below.ring in {b,c,p,w,y}:
##     :...ring.type = f: e (23.1)
##         ring.type in {e,l,n,p}:
##         :...population = a: p (0)
##             population in {n,s}: e (208.3/67.7)
##             population in {c,v,y}:
##             :...habitat in {d,g,l,m,p,u}: p (3719.5/14.7)
##                 habitat = w: e (44)
##
## -----  Trial 3:  -----
##
## Decision tree:
##
## spore.print.color in {b,k,n,o,u,y}:
## :...odor in {a,f,l,m,n,s,y}: e (1317.6)
## :   odor in {c,p}: p (153.8)
## spore.print.color in {h,r,w}:
## :...cap.color in {c,n,r,u}: e (978.2/358.7)
##     cap.color in {b,e,g,p,w,y}:
##     :...population = a: p (0)
##         population = n: e (52.4)
##         population in {c,s,v,y}:
##         :...habitat in {d,g,l,m,p,u}: p (4349.6/107.6)
##             habitat = w: e (53.3)
##
## -----  Trial 4:  -----
##
## Decision tree:
##
## spore.print.color in {b,k,n,o,u,y}: e (1130.2/118.2)
## spore.print.color in {h,r,w}:
## :...odor in {a,c,f,l,m,p,s,y}: p (3411.7)
##     odor = n:
##     :...population = a: p (0)
```

```
##         population in {n,s,y}: e (379.6)
##         population in {c,v}:
##         :...ring.type in {l,n}: p (0)
##             ring.type in {e,f}: e (558.1/72.4)
##             ring.type = p:
##             :...habitat in {d,g,l,m,u,w}: p (1377.5)
##                 habitat = p: e (48)
##
## -----  Trial 5:  -----
##
## Decision tree:
##
## odor in {c,f,m,p,s,y}: p (4190.6)
## odor in {a,l,n}:
## :...population in {a,n,s,y}: e (801.5)
##     population in {c,v}:
##     :...cap.color in {b,p,w,y}: p (1208.1/63.3)
##         cap.color in {c,e,g,n,r,u}: e (704.8/111.9)
##
## -----  Trial 6:  -----
##
## Decision tree:
##
## odor in {c,f,m,p,s,y}: p (3170.2)
## odor in {a,l,n}:
## :...stalk.surface.below.ring = y: p (347.1/35.4)
##     stalk.surface.below.ring in {f,k,s}:
##     :...spore.print.color = r: p (352.9)
##         spore.print.color in {b,k,o,y}: e (276.4)
##         spore.print.color in {h,n,u,w}:
##         :...habitat in {d,g,m,p,u,w}: e (2078.2)
##             habitat = l: p (680.1/244.1)
##
## -----  Trial 7:  -----
##
## Decision tree:
##
## odor in {c,f,m,p,s,y}: p (2411.1)
## odor in {a,l,n}:
## :...stalk.color.below.ring = y: p (107)
##     stalk.color.below.ring in {b,c,g,p}: e (143.9)
##     stalk.color.below.ring in {e,n,o,w}:
##     :...cap_shape in {b,c}: p (542.4/161)
##         cap_shape = s: e (4.2)
##         cap_shape in {f,k,x}:
##         :...spore.print.color = r: p (143.5)
##             spore.print.color in {b,o,y}: e (12.7)
##             spore.print.color in {h,k,n,u,w}:
##             :...stalk.surface.above.ring in {f,s,y}: e (3402.6/110.7)
##                 stalk.surface.above.ring = k: p (137.7/43.1)
##
## -----  Trial 8:  -----
##
## Decision tree:
```

```
##
## odor in {c,f,m,p,s,y}: p (1837.1)
## odor in {a,l,n}:
## :...cap_shape = c: p (88.2)
##     cap_shape = s: e (3.2)
##     cap_shape in {b,f,k,x}:
##     :...stalk.color.below.ring = y: p (77.6)
##         stalk.color.below.ring in {b,c,g,p}: e (109.7)
##         stalk.color.below.ring in {e,n,o,w}:
##         :...spore.print.color in {b,h,k,n,o,u,w,y}: e (4584.8/304.8)
##             spore.print.color = r: p (204.5)
##
## -----  Trial 9:  -----
##
## Decision tree:
##
## odor in {a,l}: e (1585.9)
## odor in {c,f,m,n,p,s,y}:
## :...ring.number = t: e (1160/155.7)
##     ring.number in {n,o}:
##     :...veil.color in {n,o}: e (256.2)
##         veil.color in {w,y}:
##         :...cap.color = c: e (140/6.2)
##             cap.color in {e,r,u}: p (105.9/43.7)
##             cap.color in {b,g,n,p,w,y}:
##             :...stalk.surface.below.ring = f: e (172.7/45.6)
##                 stalk.surface.below.ring in {k,s,y}:
##                 :...stalk.surface.above.ring = f: e (73.8/24.5)
##                     stalk.surface.above.ring in {k,s,y}: p (3410.5/145.1)
##
##
## Evaluation on training data (6905 cases):
##
## Trial        Decision Tree
## -----      ----------------
##    Size      Errors
##
##    0      7    3( 0.0%)
##    1      5   97( 1.4%)
##    2      7  523( 7.6%)
##    3      6  931(13.5%)
##    4      6  392( 5.7%)
##    5      4  187( 2.7%)
##    6      6  262( 3.8%)
##    7      9  432( 6.3%)
##    8      7   17( 0.2%)
##    9      8 2005(29.0%)
## boost              0( 0.0%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##    3597          (a): class e
##          3308    (b): class p
```

```
##
##
##  Attribute usage:
##
##  100.00% odor
##  100.00% spore.print.color
##   94.82% stalk.color.below.ring
##   94.82% population
##   91.28% cap.color
##   90.12% ring.number
##   89.04% stalk.surface.below.ring
##   82.75% veil.color
##   75.51% stalk.surface.above.ring
##   70.82% habitat
##   53.51% cap_shape
##   52.67% cap.surface
##   50.20% ring.type
##
##
## Time: 0.0 secs
```

```
plot(mushroom_model)
```



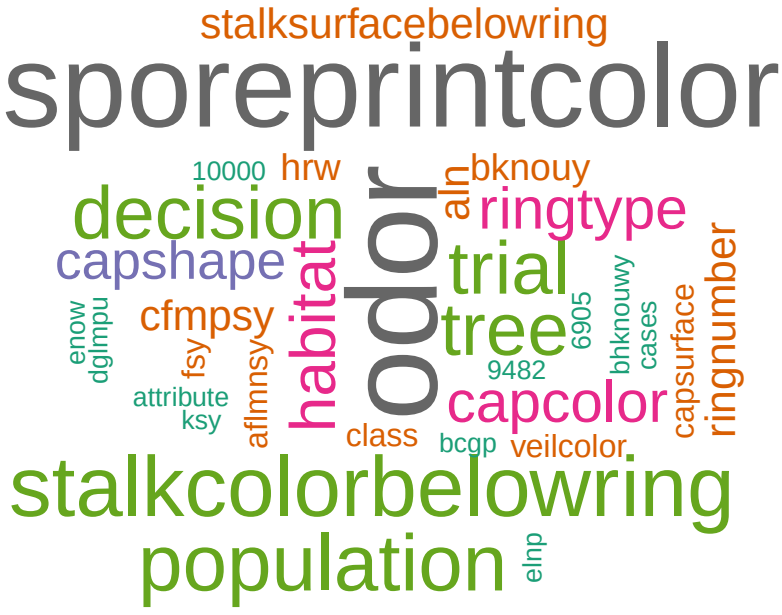

```
# wordcloud output showing most important features
wordcloud(words = mushroom_model$output, min.freq = 2, random.order=FALSE, rot.per=0.35, colors=brewer.p
```

```
## Loading required namespace: tm
```

```
## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation
## drops documents
```

```
## Warning in tm_map.SimpleCorpus(corpus, function(x) tm::removeWords(x,
## tm::stopwords())): transformation drops documents
```

```
## Warning in wordcloud(words = mushroom_model$output, min.freq = 2, random.order =
## FALSE, : stalksurfaceabovering could not be fit on page. It will not be plotted.
```



With the implementation of adaptive boosting we have increased the accuracy to 100% and lowered our error rate which is the goal of adaptive boosting. For this specific dataset, this was probably not necessary since we had already achieved incredibly low error with the original model and there was little room for improvement in terms of prediction accuracy.

building the model with a cost matrix

```
# create dimensions for a cost matrix
matrix_dimensions <- list(c("e", "p"), c("e", "p"))
names(matrix_dimensions) <- c("predicted", "actual")
matrix_dimensions
```

```
## $predicted
## [1] "e" "p"
##
## $actual
## [1] "e" "p"
```

```
# build the matrix
error_cost <- matrix(c(0, 1, 2, 0), nrow = 2, dimnames = matrix_dimensions)
error_cost
```

```
##          actual
## predicted e p
##         e 0 2
##         p 1 0
```

```
# apply the cost matrix to the tree
mushroom_model = C5.0(data_train, data_train_labels, costs=error_cost)
predict_mushroom = predict(mushroom_model, data_test)
CrossTable(predict_mushroom, data_test_labels,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  1218
##
##
##                 | predicted default
## actual default |         e |         p | Row Total |
## ---------------|-----------|-----------|-----------|
##              e |       611 |         0 |       611 |
##                |     0.502 |     0.000 |           |
## ---------------|-----------|-----------|-----------|
##              p |         0 |       607 |       607 |
##                |     0.000 |     0.498 |           |
## ---------------|-----------|-----------|-----------|
##   Column Total |       611 |       607 |      1218 |
## ---------------|-----------|-----------|-----------|
##
##
```

```
mushroom_model
```

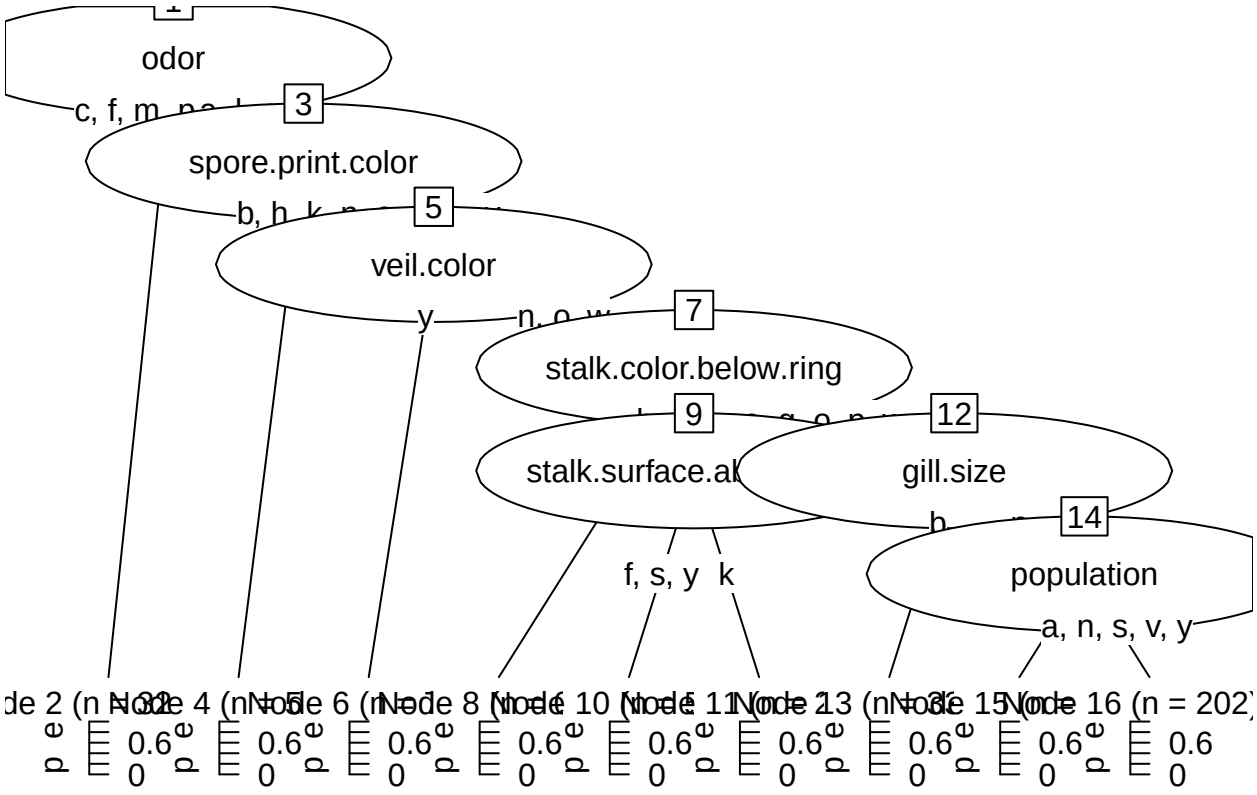
```
##
## Call:
## C5.0.default(x = data_train, y = data_train_labels, costs = error_cost)
##
## Classification Tree
## Number of samples: 6905
## Number of predictors: 21
##
## Tree size: 9
##
## Non-standard options: attempt to group attributes
##
## Cost Matrix:
##          actual
## predicted e p
##         e 0 2
##         p 1 0
```

```
summary(mushroom_model)
```

```
##
## Call:
## C5.0.default(x = data_train, y = data_train_labels, costs = error_cost)
##
##
## C5.0 [Release 2.07 GPL Edition]      Mon Apr 17 15:08:06 2023
## -------------------------------
```

```
##
## Class specified by attribute `outcome'
##
## Read 6905 cases (22 attributes) from undefined.data
## Read misclassification costs from undefined.costs
##
## Decision tree:
##
## odor in {c,f,m,p,s,y}: p (3210)
## odor in {a,l,n}:
## :...spore.print.color = r: p (58)
##     spore.print.color in {b,h,k,n,o,u,w,y}:
##     :...veil.color = y: p (7)
##         veil.color in {n,o,w}:
##         :...stalk.color.below.ring in {b,c}: e (0)
##             stalk.color.below.ring in {n,y}:
##             :...stalk.surface.above.ring in {f,s,y}: e (54)
##             :   stalk.surface.above.ring = k: p (27)
##             stalk.color.below.ring in {e,g,o,p,w}:
##             :...gill.size = b: e (3341)
##                 gill.size = n:
##                 :...population = c: p (6)
##                     population in {a,n,s,v,y}: e (202)
##
##
## Evaluation on training data (6905 cases):
##
##         Decision Tree
##    ----------------------
##    Size      Errors   Cost
##
##       8    0( 0.0%)   0.00   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##    3597          (a): class e
##          3308    (b): class p
##
##
##  Attribute usage:
##
##  100.00% odor
##   53.51% spore.print.color
##   52.67% veil.color
##   52.57% stalk.color.below.ring
##   51.40% gill.size
##    3.01% population
##    1.17% stalk.surface.above.ring
##
##
## Time: 0.0 secs
```

```
plot(mushroom_model)
```




```
# wordcloud output showing most important features
wordcloud(words = mushroom_model$output, min.freq = 2, random.order=FALSE, rot.per=0.35, colors=brewer.p
```

```
## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation
## drops documents
```

```
## Warning in tm_map.SimpleCorpus(corpus, function(x) tm::removeWords(x,
## tm::stopwords())): transformation drops documents
```

```
## Warning in wordcloud(words = mushroom_model$output, min.freq = 2, random.order =
## FALSE, : stalkcolorbelowring could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = mushroom_model$output, min.freq = 2, random.order =
## FALSE, : stalksurfaceabovering could not be fit on page. It will not be plotted.
```

A cost matrix was built where a misclassification of edible when it is actually poisonous was given a higher weight of 2, and a misclassification of poisonous when it is in fact edible was given a lower weight of 1. This would have a higher impact if our original model was not as effective, especially if it reported a lot of false negatives where the poisonous mushroom was misclassified. That being said, the model did perform better than our original model and got 100% accuracy compared to the 99% before. Note that this model did not use adaptive boosting yet still achieved the same performance with the cost matrix. The performance increase though is overshadowed by the increased complexity of the model which is something we did not observe from the adaptive boosting model. For this specific dataset a cost matrix was not necessary since we already had zero important false negatives, but could be necessary for another dataset where importantant false negatives or false positives are made even if it means building a more complex model.

Autograding

```
.AutograderMyTotalScore()
```

```
##
## Step 1:      0/1
## Step 2:      0/1
## Step 3:      0/1
## Step 4:      0/1
## Step 5:      0/1
## Total Score: 0/5
```