# Is this mushroom edible ?

*Jean-Francois Barthe*

*11/06/2019*

## Contents

**1 EXECUTIVE SUMMARY** **1**

**2 ANALYSIS** **2**
2.1 Getting and tidying the data . . . . . 2
2.2 Data Exploration . . . . . 3
2.3 Logistic Regression . . . . . 4
2.4 K-nearest neighbors . . . . . 5
2.5 Decision Tree . . . . . 6
2.6 Five predictors logistic regression . . . . . 7

**3 RESULTS** **8**

**4 CONCLUSION** **8**

---

# 1 EXECUTIVE SUMMARY

The mushroom database contains about 8.200 types of mushrooms from the most widely consumed family (*Agaricus*), either edible or poisonous. The database gives 22 attributes for each mushroom (cap shape, cap color, veil type, spore print color, habitat, ...). I will try different model to predict whether the mushroom is edible or not, according to the attributes. The first objective is to attain a 100% specificity (positive = poisonous, we don't want false edible !) and if possible a 100% accuracy. The second objective is to find a simple model, that is using a small number of attributes.

**Almost all models achieved perfect accuracy** : logistic regression, k-nearest neighbors and decision tree with a penaly matrix to minimize false edible.

The decision tree shows that **5 attributes among the 22 are enough to get a perfect prediction** : odor, spore.print.color, population, gill.size and habitat

If a mushroom smells fishy, spicy, pungent, musty, foul or creosote, DO NOT EAT IT ! If it doesn't, but the color of its spore print is green or white, do not eat it either.

# 2 ANALYSIS

## 2.1 Getting and tidying the data

The mushroom dataset can be found on the Kaggle website

I couldn't find an easy way to download the dataset directly from the script (restrictions due to Kaggle Terms of Use), so the csv file provided with the script must be copied in the script directory.

```
raw <- read.csv("./mushrooms.csv")
str(raw)
```

```
## '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 ...
```

The dataset is already tidy. The only modification I made for clarity purpose is to use full name instead of one letter for the levels of each factor.

For instance, the code for *cap.shape* is :

```
raw$cap.shape <- mapvalues(raw$cap.shape, from = c("b", "c", "x", "f", "k", "s"),
                           to = c("bell", "conical", "convex", "flat", "knobbed", "sunken"))
```

## 2.2 Data Exploration

Let's have a look at the summary of the dataframe :

```
summary(raw)
```

```
##        class        cap.shape     cap.surface     cap.color
##  edible   :4208   bell   : 452   fibrous:2320   brown  :2284
##  poisonous:3916   conical:   4   grooves:   4   grey   :1840
##                   flat   :3152   smooth :2556   red    :1500
##                   knobbed: 828   scaly  :3244   yellow :1072
##                   sunken :  32                  white  :1040
##                   convex :3656                  buff   : 168
##                                                 (Other): 220
##     bruises          odor      gill.attachment  gill.spacing
##  no     :4748   none   :3528   attached: 210   close  :6812
##  bruises:3376   foul   :2160   free    :7914   crowded:1312
##                 spicy  : 576
##                 fishy  : 576
##                 almond : 400
##                 anise  : 400
##                 (Other): 484
##   gill.size        gill.color      stalk.shape     stalk.root
##  broad :5612   buff     :1728   enlarging:3516   missing:2480
##  narrow:2512   pink     :1492   tapering :4608   bulbous:3776
##                white    :1202                    club   : 556
##                brown    :1048                    equal  :1120
##                grey     : 752                    rooted : 192
##                chocolate: 732
##                (Other)  :1170
##  stalk.surface.above.ring stalk.surface.below.ring stalk.color.above.ring
##  fibrous: 552             fibrous: 600             white  :4464
##  silky  :2372             silky  :2304             pink   :1872
##  smooth :5176             smooth :4936             grey   : 576
##  scaly  :  24             scaly  : 284             brown  : 448
##                                                    buff   : 432
##                                                    orange : 192
##                                                    (Other): 140
##  stalk.color.below.ring   veil.type     veil.color   ring.number
##  white  :4384           partial:8124   n     :  96   none:  36
##  pink   :1872                          orange:  96   one :7488
##  grey   : 576                          white :7924   two : 600
##  brown  : 512                          yellow:   8
##  buff   : 432
##  orange : 192
##  (Other): 156
##       ring.type    spore.print.color     population      habitat
##  evanescent:2776   white    :2388    abundant : 384   woods  :3148
##  flaring   :  48   brown    :1968    clustered: 340   grasses:2148
##  large     :1296   black    :1872    numerous : 400   leaves : 832
##  none      :  36   chocolate:1632    scattered:1248   meadows: 292
##  pendant   :3968   green    :  72    several  :4040   paths  :1144
##                    buff     :  48    solitary :1712   urban  : 368
##                    (Other)  : 144                     waste  : 192
```

The observations are almost evenly balanced between edible and poisonous mushrooms. The attributes distributions range from the very rare (e.g. *cap.shape* conical = 4 observations) to the most common (*gill.attachment* free = 7924 observations). All observations share the attribute *veil.type* = partial. Therefore, this factor is not relevant for the model and can be discarded.

```
mushrooms <- select(raw, -veil.type)
rm(raw)
```

The dataset being quite large, we will train our model on 60% of the observations and keep 40 % as a test set.

```
set.seed(1)
index <- createDataPartition(mushrooms$class, times = 1, p = 0.4, list = FALSE)
test_set <- mushrooms[index, ]
train_set <- mushrooms[-index, ]
y <- test_set$class
```

## 2.3 Logistic Regression

The first model we try is a logistic regression with 22 predictors (all attributes but class)

```
glm_train <- glm(class ~ ., data = train_set, family = "binomial", maxit = 100)
y_hat_glm <- predict(glm_train, newdata = test_set, type = "response")
y_hat_glm <- as.factor(ifelse(y_hat_glm > 0.5, "poisonous", "edible"))

cfm <- confusionMatrix(y_hat_glm, y)
results <- tibble(method = "Logistic regression",
                  Specificity = cfm$byClass[["Specificity"]],
                  Sensitivity = cfm$byClass[["Sensitivity"]],
                  Accuracy = cfm$overall[["Accuracy"]])

kable(table(y, y_hat_glm), caption = "Logistic regression", booktabs = TRUE) %>%
  kable_styling(latex_options = "hold_position") %>%
  add_header_above(c(" " = 1, "y_hat" = 2))
```

Table 1: Logistic regression

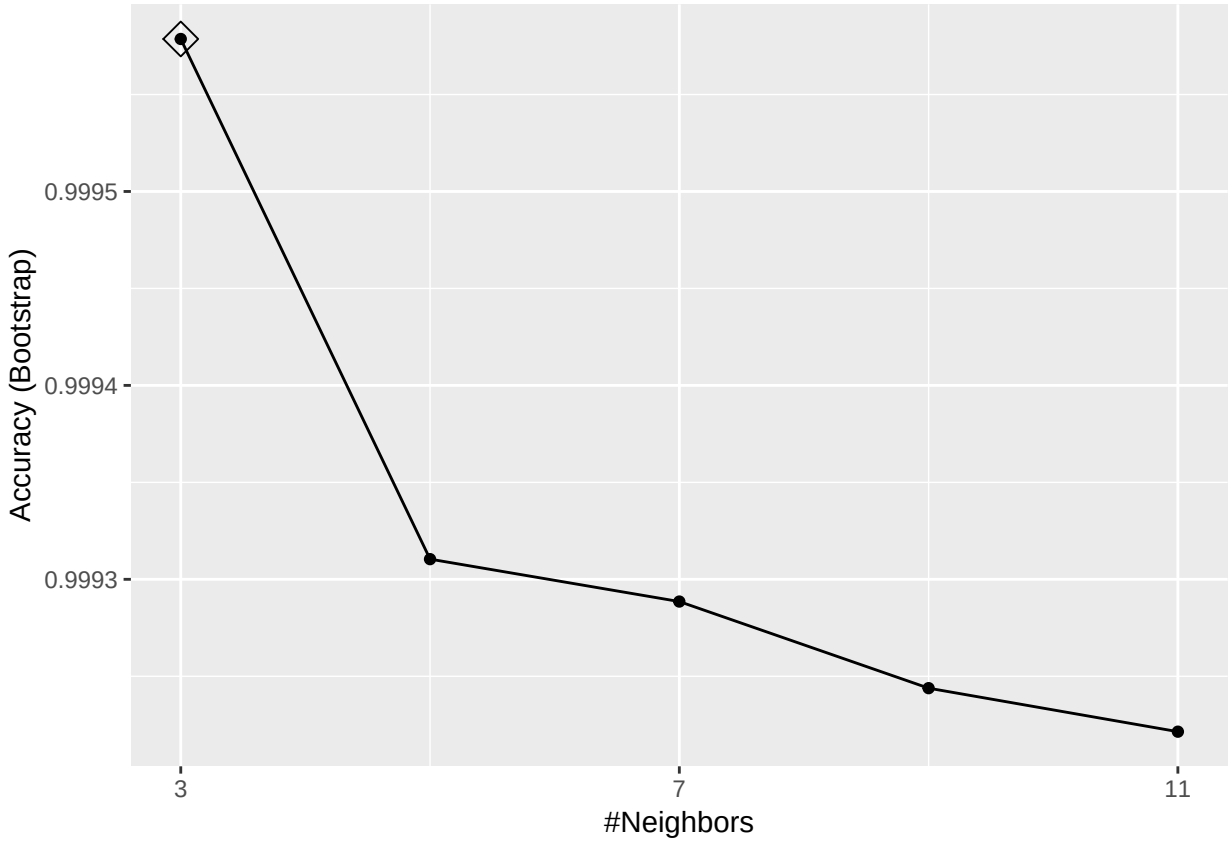
| | y_hat | |
|---|---|---|
| | edible | poisonous |
| edible | 1684 | 0 |
| poisonous | 0 | 1567 |

The logistic regression model gives a perfect prediction on the test set.

## 2.4 K-nearest neighbors

Let's try a kNN model with k values between 3 and 11 :

```
knn_train <- train(class ~ ., method = "knn",
                   data = train_set,
                   tuneGrid = data.frame(k = seq(3, 11, 2)))
ggplot(knn_train, highlight = TRUE) + scale_x_continuous(breaks = c(3,7,11))
```




The best accuracy is achieved with k = 3.

```
y_hat_knn <- predict(knn_train, newdata = test_set, type = "raw")

kable(table(y, y_hat_knn), caption = "3-nearest neighbors", booktabs = TRUE) %>%
  kable_styling(latex_options = "hold_position") %>%
  add_header_above(c(" " = 1, "y_hat" = 2))
```

Table 2: 3-nearest neighbors

| | y_hat | |
|---|---|---|
| | edible | poisonous |
| edible | 1684 | 0 |
| poisonous | 0 | 1567 |

The 3-nearest neighbors model also gives a perfect prediction on the test set.

## 2.5 Decision Tree

```
rpart_train <- rpart(class ~ ., data = train_set, method = "class")
y_hat_rpart <- predict(rpart_train, newdata = test_set, type="class")

kable(table(y, y_hat_rpart), caption = "Decision tree", booktabs = TRUE) %>%
  kable_styling(latex_options = "hold_position") %>%
  add_header_above(c(" " = 1, "y_hat" = 2))
```

Table 3: Decision tree

| | y_hat | |
|---|---|---|
| | edible | poisonous |
| edible | 1684 | 0 |
| poisonous | 13 | 1554 |

13 mushrooms are predicted edible when they are indeed poisonous : NOT GOOD !!

In order to minimize false negatives (false edibles), let's add a penalty matrix :

```
penalty.matrix <- matrix(c(0,10,1,0), nrow = 2, ncol = 2)
rpart2_train <- rpart(class ~ ., data = train_set, method = "class", parms = list(loss = penalty.matrix
y_hat_rpart2 <- predict(rpart2_train, newdata = test_set, type="class")

kable(table(y, y_hat_rpart2), caption = "Decision tree with penalty", booktabs = TRUE) %>%
  kable_styling(latex_options = "hold_position") %>%
  add_header_above(c(" " = 1, "y_hat" = 2))
```

Table 4: Decision tree with penalty

| | y_hat | |
|---|---|---|
| | edible | poisonous |
| edible | 1684 | 0 |
| poisonous | 0 | 1567 |

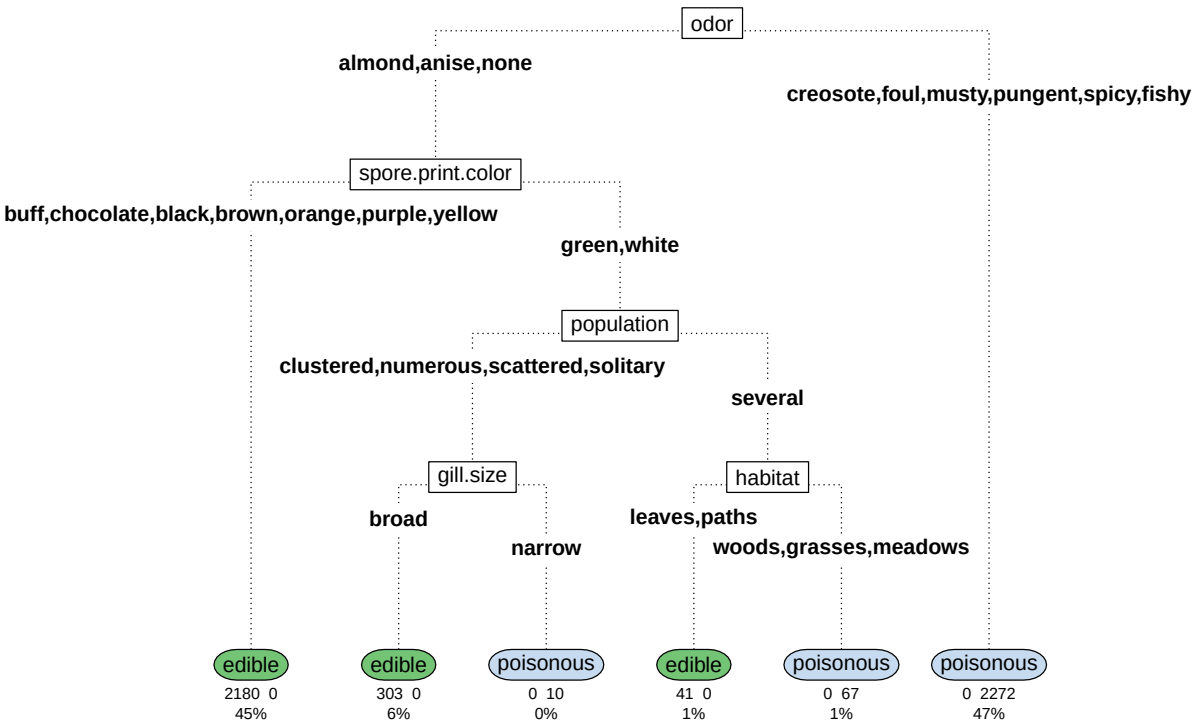
This time the prediction on the test is perfect. Let's check our decision tree on the full set :

```
y_hat <- predict(rpart2_train, newdata = mushrooms, type = "class")
kable(table(mushrooms$class, y_hat), caption = "DTwP, full set", booktabs = TRUE) %>%
  kable_styling(latex_options = "hold_position") %>%
  add_header_above(c(" " = 1, "y_hat" = 2))
```

Table 5: DTwP, full set

| | y_hat | |
|---|---|---|
| | edible | poisonous |
| edible | 4208 | 0 |
| poisonous | 0 | 3916 |

**Our decision tree predicts correctly any mushroom in our dataset.** Let's have a look at the tree :




The decision tree can correctly classify all observations with only **5 attributes** : odor, spore.print.color, population, gill.size and habitat. Let see if logistic regression can achieve 100% accuracy with these 5 predictors.

## 2.6 Five predictors logistic regression

```
form <- as.formula("class ~ odor + spore.print.color + population + gill.size + habitat")
glm5P_train <- glm(form, data = train_set, family = "binomial", maxit = 100)
y_hat_glm5P <- predict(glm5P_train, newdata = test_set, type = "response")
y_hat_glm5P <- as.factor(ifelse(y_hat_glm5P > 0.5, "poisonous", "edible"))
kable(table(y, y_hat_glm5P), caption = "5-predictors logistic regression", booktabs = TRUE) %>%
  kable_styling(latex_options = "hold_position") %>%
  add_header_above(c(" " = 1, "y_hat" = 2))
```

Table 6: 5-predictors logistic regression

| | y_hat | |
|---|---|---|
| | edible | poisonous |
| edible | 1684 | 0 |
| poisonous | 0 | 1567 |

The 5 predictors are indeed enough to achieve 100% accuracy.

# 3 RESULTS

| method | Specificity | Sensitivity | Accuracy |
|---|---|---|---|
| Logistic regression | 1.000 | 1 | 1.000 |
| 3-nearest neighbors | 1.000 | 1 | 1.000 |
| Decision tree | 0.992 | 1 | 0.996 |
| Decision tree with penalty | 1.000 | 1 | 1.000 |
| 5-predictors logistic regression | 1.000 | 1 | 1.000 |

---

# 4 CONCLUSION

Five attributes are enough to decide if a mushroom in our dataset is edible or poisonous. If you go shrooming, the golden rule is to **use your nose first.** If the mushroom smells fishy, spicy, pungent, musty, foul or creosote, discard it ! If it doesn't, then look at the color of the spore print. If it's green or white, then you'd better throw it away as well. It's probably edible (72% of this population is) but to be sure you would have to establish its population, its habitat and its gill size. Better safe than sorry !