Harvard PH125.9x Capstone: Mushroom Classification Analysis

Mauro Berlanda

April 2020

Introduction

The last assignment of the Datascience Professional Certificate by HarvardX on edx is submitting its own report. The main goal of the project is to prove the ability to clearly communicate the process and the insights gained from an analysis.

We are going to use for this analysis the Mushroom records drawn from The Audubon Society Field Guide to North American Mushrooms (1981). This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family.

> The csv file containing the data was originally downloaded from Kaggle due to its ease of manipulation. The file has been commited in a github repository since Kaggle downloads require authentication. Being unable to retrieve the raw zip file due to a corrupted output (`unzip error -1`), my script is downloading the uncompressed csv file. It does not exceed 365Kb, so it can be requested without any performance or network traffic concern.

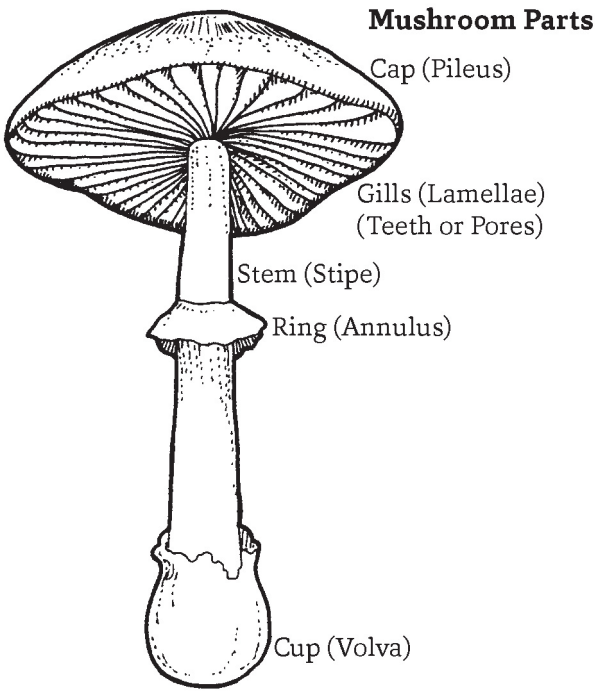
```
file_url <- "https://raw.githubusercontent.com/mberlanda/ph125-9x-data-science-capstone/master/mushroom
csv_filepath <- "./mushrooms.csv"

# Download the csv file if needed
if (!file.exists(csv_filepath)) {
  download.file(file_url, csv_filepath)
}

# Use read.csv to parse the file converting strings to factors
mushrooms <- read.csv(csv_filepath, header=TRUE, sep=",", stringsAsFactors=TRUE)
# Explore the columns and types of the dataset
str(mushrooms)
```

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

```
rm(file_url, csv_filepath)
```

To improve the domain knowledge, you can find below an image illustrating the different parts of a mushroom:



All the attributes are factors and they represent the following abbreviations:

1. cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
2. cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s
3. cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r, pink=p,purple=u,red=e,white=w,yellow=y
4. bruises?: bruises=t,no=f
5. odor: almond=a,anise=l,creosote=c,fishy=y,foul=f, musty=m,none=n,pungent=p,spicy=s
6. gill-attachment: attached=a,descending=d,free=f,notched=n
7. gill-spacing: close=c,crowded=w,distant=d
8. gill-size: broad=b,narrow=n
9. gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g, green=r,orange=o,pink=p,purple=u,red=e, white=w,yellow=y
10. stalk-shape: enlarging=e,tapering=t
11. stalk-root: bulbous=b,club=c,cup=u,equal=e, rhizomorphs=z,rooted=r,missing=?
12. stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s
13. stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s
14. stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y
15. stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y

16. veil-type: partial=p,universal=u
17. veil-color: brown=n,orange=o,white=w,yellow=y
18. ring-number: none=n,one=o,two=t
19. ring-type: cobwebby=c,evanescent=e,flaring=f,large=l, none=n,pendant=p,sheathing=s,zone=z
20. spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r, orange=o,purple=u,white=w,yellow=y
21. population: abundant=a,clustered=c,numerous=n, scattered=s,several=v,solitary=y
22. habitat: grasses=g,leaves=l,meadows=m,paths=p, urban=u,waste=w,woods=d

The classes used for the outcome are `edible` or `poisonous`. In the Kaggle version of the data set there is no `unknown` classification value and missing values have been removed:

```
unique(mushrooms$class)
```

```
## [1] p e
## Levels: e p
```

```
sum(is.na(mushrooms))
```

```
## [1] 0
```

Analysis

The main challenges raised by this data set are:

a. all the features are categorical => the core of the solution will be in the data wrangling part
b. the amount of observation is small (overall around 8k, but we will split it into a training and test sets) => we will need to implement some techniques to avoid overfitting

Data Wrangling

If we try to approach this classification problem as the `tissue_gene_expression` data set distributed by the `dslabs` package, we will quickly realise that the most of utilities used in the PH125.x courses won't be available.

```
# format as list of a features matrix and outcome vector
formatted_mushrooms <- list(
  x = mushrooms %>% select(-class) %>% as.matrix,
  y = mushrooms$class
)

# Partition train and test set
set.seed(22, sample.kind="Rounding")
test_index <- createDataPartition(y=formatted_mushrooms$y, times=1, p=.30, list=FALSE)

train_set <- list(
  x = formatted_mushrooms$x[-test_index,],
  y = formatted_mushrooms$y[-test_index]
)

hclust(train_set$x)
# Error in if (is.na(n) || n > 65536L) stop("size cannot be NA nor exceed 65536") :
# missing value where TRUE/FALSE needed

pca <- prcomp(train_set$x)
# Error in colMeans(x, na.rm = TRUE) : 'x' must be numeric
```

The problem is that the most of libraries cannot work with categorical features only. A solution to this problem can be **encoding the categorical features**.

Even if the most of resources I googled were explaining how to perform this technique with Python (All about Categorical Variable Encoding, How to Encode Categorical Data), there is an excellent R blog post providing a wide overview on the topic.

The most important concept to recall before continuing are:

- Categorical variables can be considered:
  - *Nominal* (e.g. pen/pencil/eraser or cow/dog/cat)
  - *Ordinal* (e.g. excellent/good/bad or fantastic/ok/don't like).
- The three main routes to encode factors string data type are:
  - *Classic Encoders*: e.g. ordinal, OneHot, Binary, Frequency, Hashing ...
  - *Contrast Encoders*: encode data by looking at different levels of features
  - *Bayesian Encoders*: use the target as the foundation of the encoding

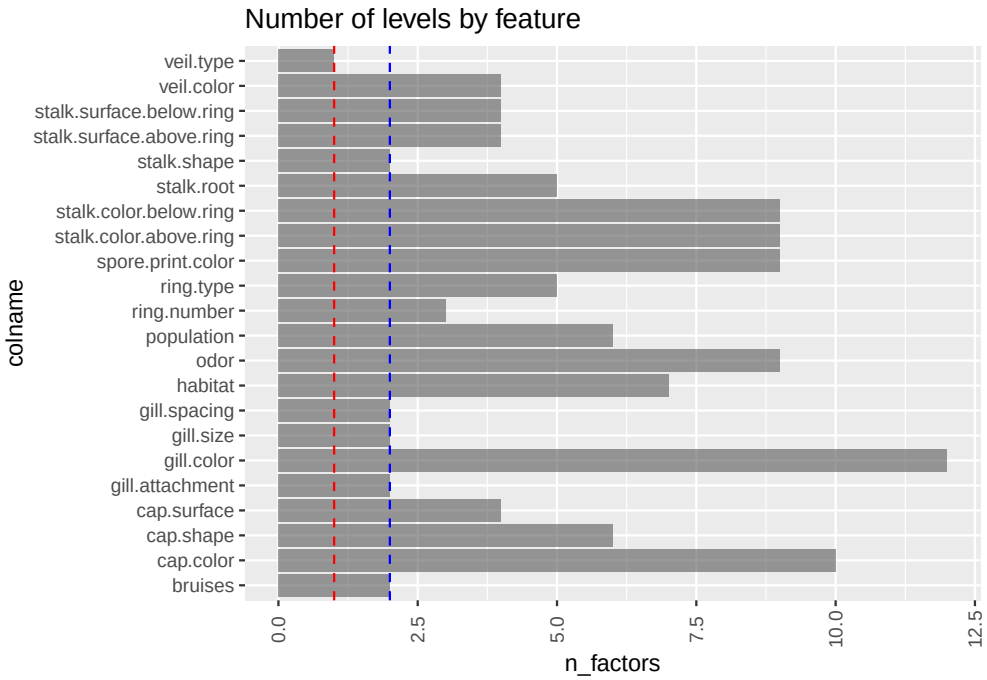


Table 1: Attributes with less than 4 factors

colname	n_factors
ring.number	3
bruises	2
gill.attachment	2
gill.spacing	2
gill.size	2
stalk.shape	2
veil.type	1

We can make these insights actionable with the attributes description provided in the Introduction:

- `veil.type` attribute can be ignored since it has only one level

- `ring.number` and `gill.spacing` attribute is an ordinal variable
- `bruises` and `gill.size` attributes can be considered as a logical/binary vectors
- in general, all the factors having two levels only can be rephrased as a 1/0 attribute
- the other attributes may be considered as nominal variables: colors, shapes etc.

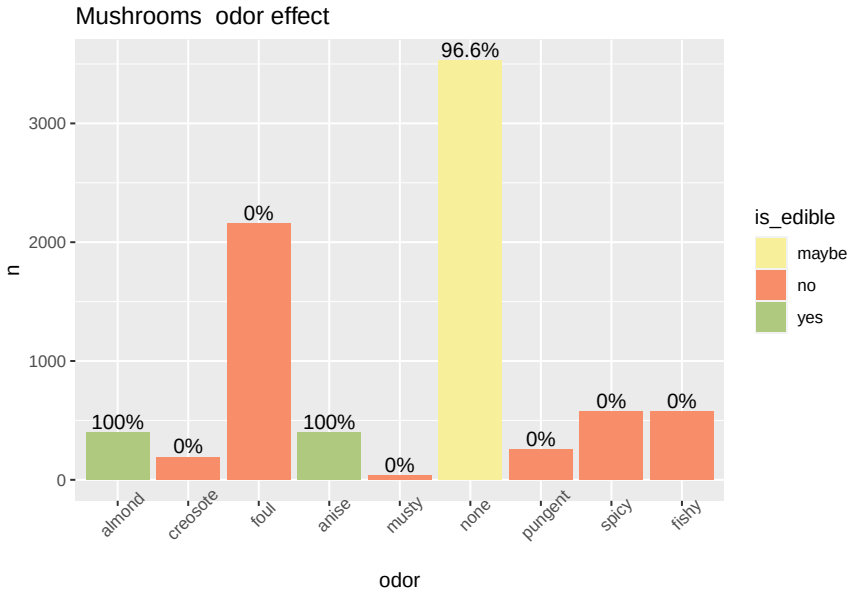
Exploration without any normalization

Before formatting the data, we can only explore the data manually. In fact, we cannot use any library we got familiar with.

Some insights were already gained:

Odor effect

```
mushrooms %>%
  group_by(odor) %>%
  summarize(n=n(), edible=mean(class=="e")) %>%
  mutate(is_edible=ifelse(
        edible==1, "yes", ifelse(edible==0, "no", "maybe"))
  ) %>%
  ggplot(aes(x=odor), group=1) +
  geom_bar(aes(y=n, fill=is_edible), stat="identity") +
  geom_text(aes(y=n, label=paste(100*round(edible, digits=3), "%", sep="")), position=position_dodge(wi
  scale_fill_manual(values=c("#f7ef99", "#f78e69","#afc97e")) +
  scale_x_discrete(labels=c(
    "a"="almond","l"="anise","c"="creosote","y"="fishy","f"="foul",
    "m"="musty","n"="none","p"="pungent","s"="spicy"
  )) +
  ggtitle("Mushrooms  odor effect") +
  theme(axis.text.x = element_text(angle=45))
```

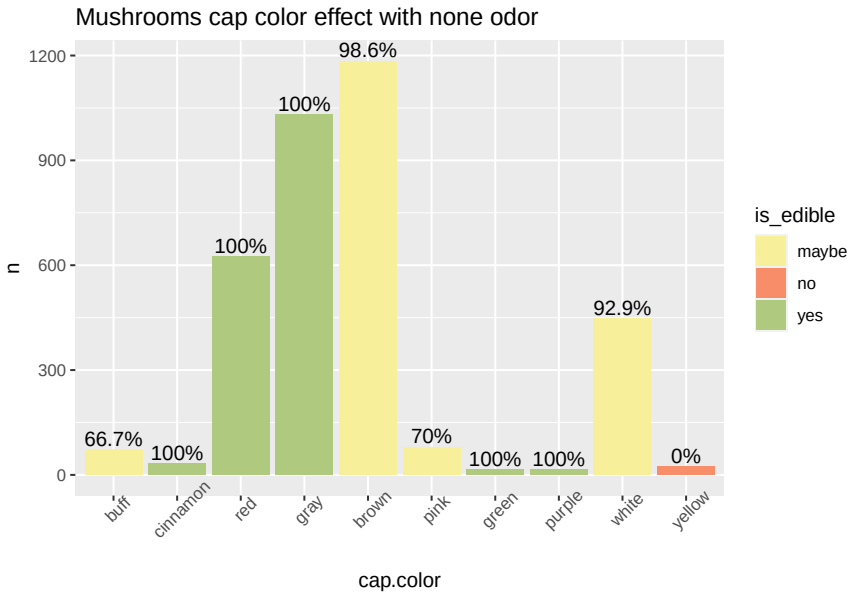


- When odor is almond or anise, the mushroom is alway edible.
- When the mushroom has no odor, it has a 96.6% probability to be edible.
- In any other case, the mushroom is not edible.

It would be worth then to explore investigate the effect of the other features when odor is none.

Cap.color effect

```
mushrooms %>%
  filter(odor =="n") %>%
  group_by(cap.color) %>%
  summarize(n=n(), edible=mean(class=="e")) %>%
  mutate(is_edible=ifelse(
        edible==1, "yes", ifelse(edible==0, "no", "maybe"))
  ) %>%
  ggplot(aes(x=cap.color), group=1) +
  geom_bar(aes(y=n, fill=is_edible), stat="identity") +
  geom_text(aes(y=n, label=paste(100*round(edible, digits=3), "%", sep="")), position=position_dodge(wi
  scale_fill_manual(values=c("#f7ef99", "#f78e69","#afc97e")) +
  scale_x_discrete(labels=c(
    "n"="brown","b"="buff","c"="cinnamon","g"="gray","r"="green",
    "p"="pink","u"="purple","e"="red","w"="white","y"="yellow"
  )) +
  ggtitle("Mushrooms cap color effect with none odor") +
  theme(axis.text.x = element_text(angle=45))
```



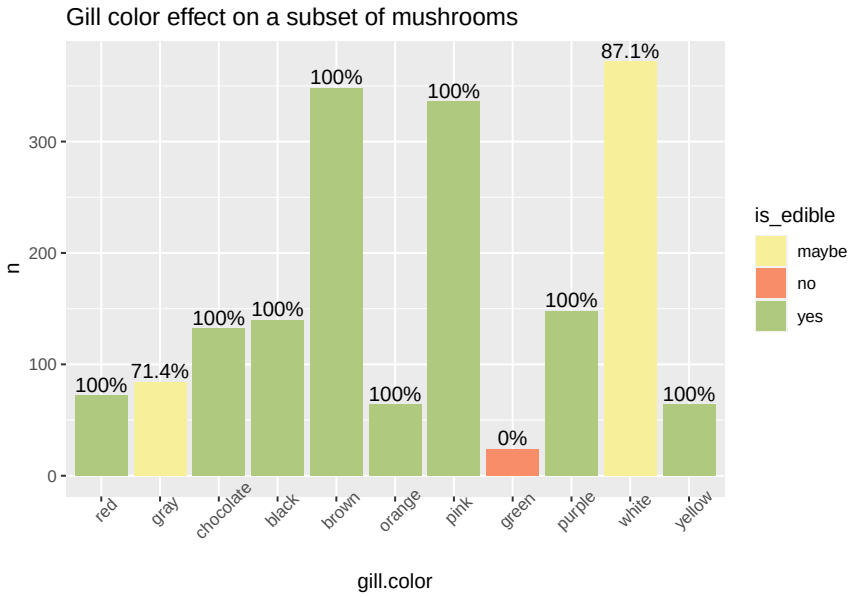
The additional insights analysing the mushrooms with odor `none` by cap color are:

- When cap is cinnamon, red, gray, green or purple the mushrooms are edible
- When cap is yellow, the mushroom is never edible

Gill.color effect

```
mushrooms %>%
  filter(odor =="n") %>%
  filter(cap.color %in% c("n", "b", "p", "w")) %>%
  group_by(gill.color) %>%
  summarize(n=n(), edible=mean(class=="e")) %>%
  mutate(is_edible=ifelse(
        edible==1, "yes", ifelse(edible==0, "no", "maybe"))
  ) %>%
  ggplot(aes(x=gill.color), group=1) +
```

```
  geom_bar(aes(y=n, fill=is_edible), stat="identity") +
  geom_text(aes(y=n, label=paste(100*round(edible, digits=3), "%", sep="")), position=position_dodge(wi
  scale_fill_manual(values=c("#f7ef99", "#f78e69","#afc97e")) +
  scale_x_discrete(labels=c(
    "k"="black","n"="brown","b"="buff","h"="chocolate","g"="gray",
    "r"="green","o"="orange","p"="pink","u"="purple","e"="red",
    "w"="white","y"="yellow"
  )) +
  ggtitle("Gill color effect on a subset of mushrooms") +
  theme(axis.text.x = element_text(angle=45))
```



The insights gained on mushroom with none odor and cap color brown, buff, pink or white are:

- When gill.color is red, chocolate, black, brown, orange, pink, purple or yellow the mushroom is always edible
- When the gill.color is green the mushroom is not edible

We could continue this breakdown to build a model to validate our predictions against the true values. We can use however a set of techniques to manipulate categorical attributes as described above.

Using Classic Encoders

A different (and still easy to understand) approach would be to encode these categorical features using **OneHot** encoding.

A one hot encoding is a representation of categorical variables as binary vectors. This can be perfomed using some libraries as `vtreat` as follows:

```
if(!require(vtreat)) install.packages("vtreat", repos = "http://cran.r-project.org")
tz <- vtreat::designTreatmentsZ(x, colnames(x))
new_x <- vtreat::prepare(tz, x)
ncol(new_x)
colnames(new_x)
detach("package:vtreat", unload = TRUE)
```

The output of this normalization is very effective for creating efficient models but it may lead to some lack of visibility when interpreting the results.

In our analysis, we are going to normalize the data manually using different techniques before splitting train and test set. In this way we can keep a strict control on the formatted data structure. In order to make this manipulation easy to reproduce (e.g. in case we want to test our model against additional observations), it will be wrapped in a function called `formatMushrooms`:

```
formatMushrooms <- function(df) {
  # private utility function
  mapValues <- function(v, m) {
    sapply(v, function(x) do.call("switch", prepend(m, x)))
  }

  # Select nominal categorical features with more than 2 categories
  new_df <- df %>%
    select(
      cap.color, cap.shape, cap.surface, gill.attachment, gill.color, habitat, odor,
      population, ring.type, spore.print.color, stalk.color.above.ring,
      stalk.color.below.ring, stalk.root, stalk.surface.above.ring,
      stalk.surface.below.ring, veil.color
    )

  # Nominal variables
  colorsMapping <- list(
    "b"=".buff","c"=".cinnamon","e"=".red","g"=".gray","h"=".chocolate","k"=".black",
    "n"=".brown","o"=".orange","p"=".pink","r"=".green","u"=".purple","w"=".white",
    "y"=".yellow"
  )
  surfacesMapping <- list("f"=".fibrous","g"=".grooves","y"=".scaly","s"=".smooth")

  capShapeMapping <- list("b"=".bell","c"=".conical","x"=".convex","f"=".flat", "k"=".knobbed","s"=".su
  gillAttachmentMapping <- list("a"=".attached","d"=".descending","f"=".free","n"=".notched")
  habitatMapping <- list("g"=".grasses","l"=".leaves","m"=".meadows","p"=".paths", "u"=".urban","w"=".wa
  odorMapping <- list("a"=".almond","l"=".anise","c"=".creosote","y"=".fishy","f"=".foul", "m"=".musty"
  populationMapping <- list("a"=".abundant","c"=".clustered","n"=".numerous", "s"=".scattered","v"=".se
  ringTypeMapping <- list("c"=".cobwebby","e"=".evanescent","f"=".flaring","l"=".large", "n"=".none","p
  stalkRootMapping <- list("b"=".bulbous","c"=".club","u"=".cup","e"=".equal", "z"=".rhizomorphs","r"="

  new_df$cap.color <- mapValues(df$cap.color, colorsMapping) %>% factor
  new_df$gill.color <- mapValues(df$gill.color, colorsMapping) %>% factor
  new_df$spore.print.color <- mapValues(df$spore.print.color, colorsMapping) %>% factor
  new_df$stalk.color.above.ring <- mapValues(df$stalk.color.above.ring, colorsMapping) %>% factor
  new_df$stalk.color.below.ring <- mapValues(df$stalk.color.below.ring, colorsMapping) %>% factor
  new_df$veil.color <- mapValues(df$veil.color, colorsMapping) %>% factor

  new_df$cap.surface <- mapValues(df$cap.surface, surfacesMapping) %>% factor
  new_df$stalk.surface.above.ring <- mapValues(df$stalk.surface.above.ring, surfacesMapping) %>% factor
  new_df$stalk.surface.below.ring <- mapValues(df$stalk.surface.below.ring, surfacesMapping) %>% factor

  new_df$cap.shape <- mapValues(df$cap.shape, capShapeMapping) %>% factor
  new_df$gill.attachment <- mapValues(df$gill.attachment, gillAttachmentMapping) %>% factor
  new_df$habitat <- mapValues(df$habitat, habitatMapping) %>% factor
  new_df$odor <- mapValues(df$odor, odorMapping) %>% factor
  new_df$population <- mapValues(df$population, populationMapping) %>% factor
```

```
  new_df$ring.type <- mapValues(df$ring.type, ringTypeMapping) %>% factor
  new_df$stalk.root <- mapValues(df$stalk.root, stalkRootMapping) %>% factor

  # Cleanup
  rm(
    colorsMapping, surfacesMapping, capShapeMapping, gillAttachmentMapping,
    habitatMapping, odorMapping, populationMapping, ringTypeMapping, stalkRootMapping
  )
  # One Hot encoding: convert nominal variables with more than 2 categories
  # https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/model.matrix
  new_df <- model.matrix(
    ~.-1,
    data = new_df
  )

  # ordinal variables
  # gill.spacing: "c"="close","w"="crowded","d"="distant"
  gillSpacingMapping <- list("c"=0,"w"=1,"d"=2)
  # ring.number: "n"="none","o"="one","t"="two"
  ringNumberMapping <-list("n"=0,"o"=1,"t"=2)

  new_df <- new_df %>%
    as_tibble() %>%
    add_column(
      grill.spacing = mapValues(df$gill.spacing, gillSpacingMapping),
      ring.number = mapValues(df$ring.number, ringNumberMapping),
      bruises = ifelse(df$bruises=="t",1,0),
      gill.size.narrow = ifelse(df$gill.size=="n",1,0),
      stalk.shape.enrlarging = ifelse(df$stalk.shape=="e",1,0)
      # veil.type is ignored in this analysis since it is constant in the data set
      # veil.type.universal = ifelse(df$veil.type=="u",1,0)
    ) %>% as.matrix

  rm(gillSpacingMapping, ringNumberMapping)

  list(
    x = new_df,
    y = factor(df$class == "e") # edible
  )
}
```

The formatted data are:

```
formatted_mushrooms <- formatMushrooms(mushrooms)

object.size(mushrooms)
```

```
## 767792 bytes
```

```
object.size(formatted_mushrooms)
```

```
## 6215944 bytes
```

```
dim(formatted_mushrooms$x)
```

```
## [1] 8124   95
```

```r
sort(colnames(formatted_mushrooms$x))
```

```
##  [1] "bruises"                          "cap.color.black"
##  [3] "cap.color.brown"                  "cap.color.buff"
##  [5] "cap.color.chocolate"              "cap.color.cinnamon"
##  [7] "cap.color.gray"                   "cap.color.green"
##  [9] "cap.color.orange"                 "cap.color.pink"
## [11] "cap.color.red"                    "cap.shape.conical"
## [13] "cap.shape.convex"                 "cap.shape.flat"
## [15] "cap.shape.knobbed"                "cap.shape.sunken"
## [17] "cap.surface.grooves"              "cap.surface.scaly"
## [19] "cap.surface.smooth"               "gill.attachment.descending"
## [21] "gill.color.brown"                 "gill.color.buff"
## [23] "gill.color.chocolate"             "gill.color.cinnamon"
## [25] "gill.color.gray"                  "gill.color.green"
## [27] "gill.color.orange"                "gill.color.pink"
## [29] "gill.color.purple"                "gill.color.red"
## [31] "gill.color.white"                 "gill.size.narrow"
## [33] "grill.spacing"                    "habitat.leaves"
## [35] "habitat.meadows"                  "habitat.paths"
## [37] "habitat.urban"                    "habitat.waste"
## [39] "habitat.woods"                    "odor.anise"
## [41] "odor.creosote"                    "odor.fishy"
## [43] "odor.foul"                        "odor.musty"
## [45] "odor.none"                        "odor.pungent"
## [47] "odor.spicy"                       "population.clustered"
## [49] "population.numerous"              "population.scattered"
## [51] "population.several"               "population.solitary"
## [53] "ring.number"                      "ring.type.evanescent"
## [55] "ring.type.flaring"                "ring.type.large"
## [57] "ring.type.none"                   "spore.print.color.brown"
## [59] "spore.print.color.buff"           "spore.print.color.chocolate"
## [61] "spore.print.color.cinnamon"       "spore.print.color.gray"
## [63] "spore.print.color.orange"         "spore.print.color.pink"
## [65] "spore.print.color.red"            "stalk.color.above.ring.brown"
## [67] "stalk.color.above.ring.buff"      "stalk.color.above.ring.chocolate"
## [69] "stalk.color.above.ring.cinnamon"  "stalk.color.above.ring.gray"
## [71] "stalk.color.above.ring.orange"    "stalk.color.above.ring.pink"
## [73] "stalk.color.above.ring.red"       "stalk.color.below.ring.brown"
## [75] "stalk.color.below.ring.buff"      "stalk.color.below.ring.chocolate"
## [77] "stalk.color.below.ring.cinnamon"  "stalk.color.below.ring.gray"
## [79] "stalk.color.below.ring.orange"    "stalk.color.below.ring.pink"
## [81] "stalk.color.below.ring.red"       "stalk.root.club"
## [83] "stalk.root.cup"                   "stalk.root.equal"
## [85] "stalk.root.rhizomorphs"           "stalk.shape.enrlarging"
## [87] "stalk.surface.above.ring.grooves" "stalk.surface.above.ring.scaly"
## [89] "stalk.surface.above.ring.smooth"  "stalk.surface.below.ring.grooves"
## [91] "stalk.surface.below.ring.scaly"   "stalk.surface.below.ring.smooth"
## [93] "veil.color.cinnamon"              "veil.color.gray"
## [95] "veil.color.red"
```

```r
rm(mushrooms, formatMushrooms)
```

Train and test (validation) set

```
set.seed(22, sample.kind="Rounding")
test_index <- createDataPartition(y=formatted_mushrooms$y, times=1, p=.30, list=FALSE)

train_set <- list(
  x = formatted_mushrooms$x[-test_index,],
  y = formatted_mushrooms$y[-test_index]
)

test_set <- list(
  x = formatted_mushrooms$x[test_index,],
  y = formatted_mushrooms$y[test_index]
)

dim(train_set$x)
```

```
## [1] 5686   95
```

```
rm(formatted_mushrooms, test_index)
```

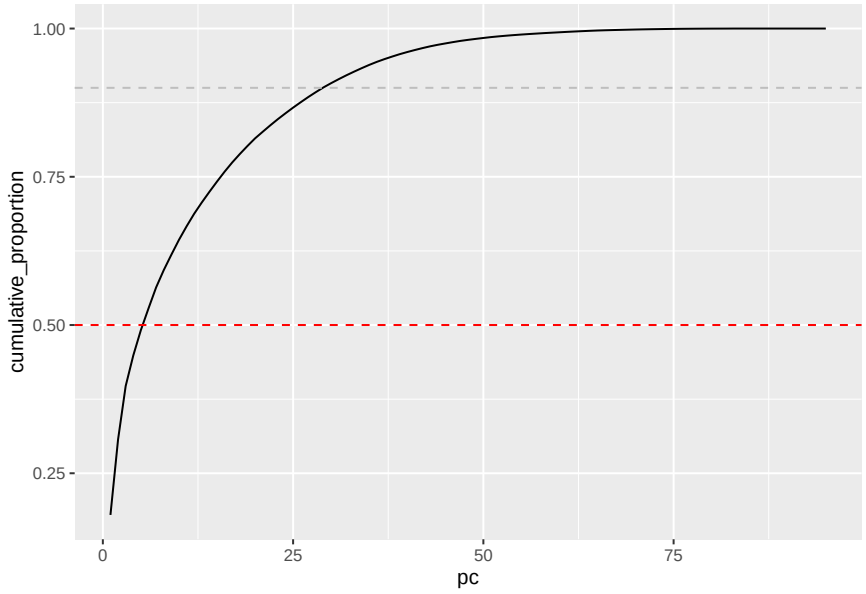
Exploring with Principal Component Analysis

After this manipulation, we have 95 dimensions in our train dataset. We can investigate the possibility of applying one of the dimension reduction techniques illustrated in PH125.x, such as principal component analysis.

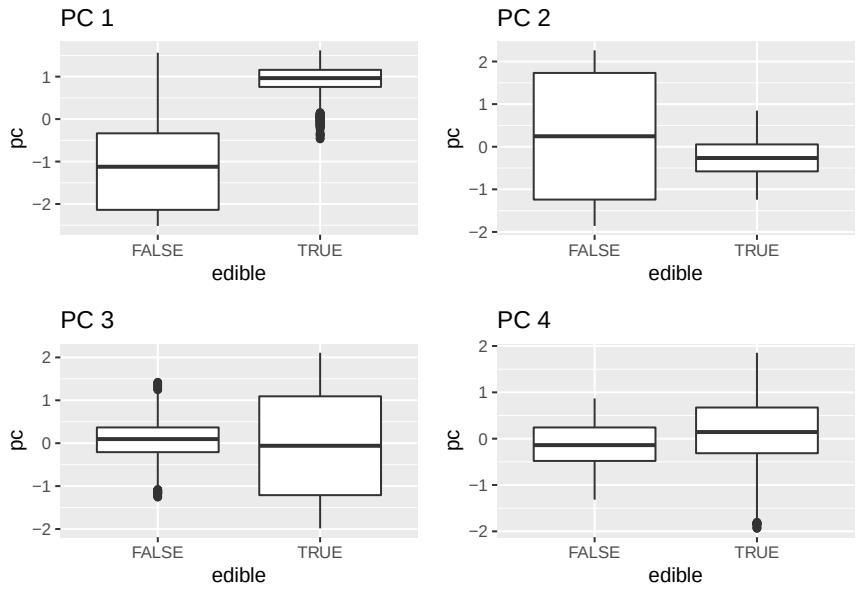
```
pca <- prcomp(with(train_set, sweep(x, 1, rowMeans(x))))

summary(pca)$importance[,1:6]
```

```
##                           PC1   PC2    PC3    PC4    PC5    PC6
## Standard deviation     1.28 1.076 0.9062 0.6844 0.6251 0.5765
## Proportion of Variance 0.18 0.127 0.0902 0.0515 0.0429 0.0365
## Cumulative Proportion  0.18 0.307 0.3972 0.4486 0.4916 0.5281
```

```
tibble(
  pc=1:ncol(summary(pca)$importance),
  cumulative_proportion=summary(pca)$importance[3,]
) %>% ggplot(aes(x=pc)) +
  geom_line(aes(y=cumulative_proportion)) +
  geom_hline(yintercept=0.5, color="red", linetype="dashed") +
  geom_hline(yintercept=0.9, color="grey", linetype="dashed")
```

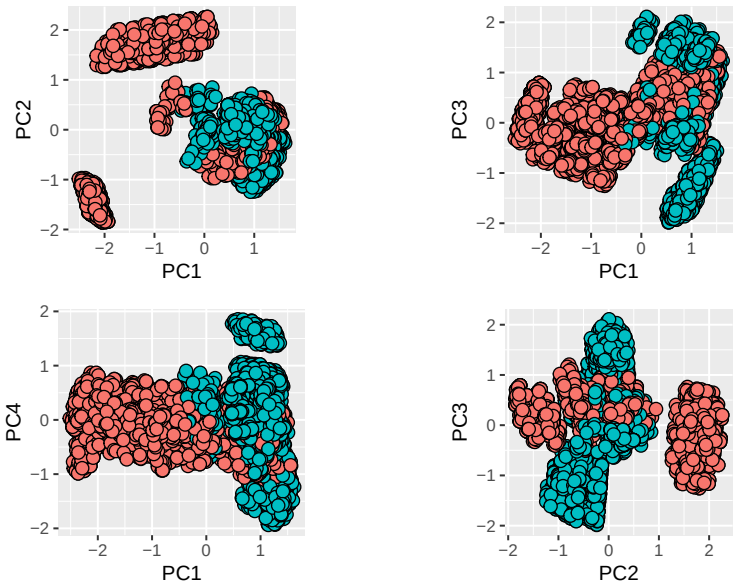



```
gridExtra::grid.arrange(
  grobs=map(1:4, function(i) {
    ggplot(aes(x=edible, y=pc), data=data.frame(pc=pca$x[,i], edible=train_set$y)) +
      geom_boxplot() + ggtitle(paste("PC", i)) +
      theme(plot.margin=unit(x=c(.5, .5, .5, .5), units="lines"))
    }),
  ncol=2
)
```




```
df <- data.frame(pca$x[,1:4], y=train_set$y)

gridExtra::grid.arrange(
  grobs= lapply(
    list(
      df %>% ggplot(aes(PC1, PC2, fill=y)),
```

```
      df %>% ggplot(aes(PC1, PC3, fill=y)),
      df %>% ggplot(aes(PC1, PC4, fill=y)),
      df %>% ggplot(aes(PC2, PC3, fill=y))
    ), function(p) {
      p +
      geom_point(cex=3, pch=21) +
      coord_fixed(ratio=1) +
      theme(
        plot.margin=unit(x=c(.5, .5, .5, .5), units="lines"),
        legend.position="none"
      )
    }
  ),
  ncol=2
)
```




```
rm(pca, df)
```

When used for other dataset analysis, the PCA was able to explain over 0.9 of the cumulative proportion of variance with a couple of attributes. Thanks to this property, it was easy to plot and visualize the clusters of observations.

In this data set more than 25 components will be required to provide a cumuluative proportion of variance. Moreover, for PC1 the interquartile ranges for edible and poisonous are not overlapping (except for a few outliers). In all other dimensions the IQR are overlapped.

After a first analysis, the benefit of including PC in our model does not seem this important. We may reconsider PCA if the accuracy of our analysis is not good enough.

Methods

Since the train set has only 5686 observations, we need to find to avoid overfitting. One effective technique is to perform cross-validation on our models.

This can be done passing the following options to the `caret` train models:

```
train_control <- trainControl(method="cv", number=10)
```

Regarding the evaluation of the models, for regressions we would have used the residual mean squared error defined as

```
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

However, to solve this classification problem, we are going to base our decision on the `accuracy` .

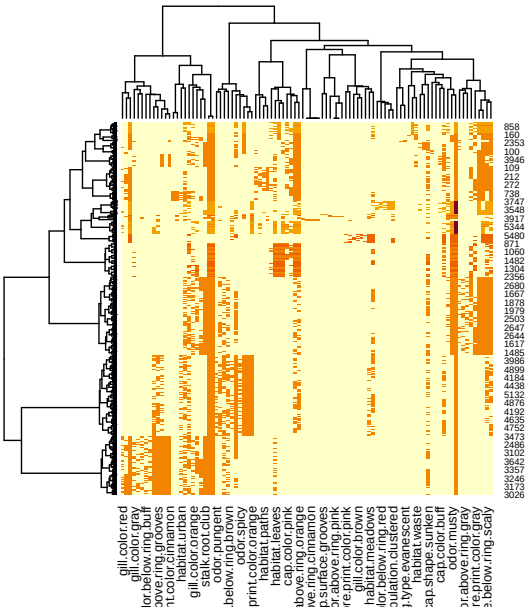
Since the outcome is identify if a mushroom is poisonous or edible (where edible is the positive value), we would tend to prefer a model with the highest **sensitivity** (true positive rate) rather than **specificity** (true negative rate). In fact, a false positive would have worst consequences than a false negative.

The results of models are collected in the same object `models_results` and will be commented in the Results section.

Once we decide the model (or the ensemble) to adopt, we are going to validate it against the validation set.

For every model we are going to list the attributes importance >=50%.

```
set.seed(1234, sample.kind="Rounding")
heatmap(
  train_set$x,
  distfun=function(x) as.dist(1-cor(t(x))),
  hclustfun=function(x) hclust(x, method="ward.D2")
)
```



Generalized Linear Model

https://en.wikipedia.org/wiki/Generalized_linear_model

```
start_time <- Sys.time()
train_glm <- train(
  train_set$x,
  factor(train_set$y),
  trControl = train_control,
```

```
  method = "glm"
)
end_time <- Sys.time()
duration <- as.numeric(difftime(end_time, start_time, units="secs"))
y_hat <- predict(train_glm, train_set$x)
conf_matrix <- confusionMatrix(data=y_hat, reference=train_set$y)

models_results <- data.frame(
   method= "glm",
   accuracy=conf_matrix$overall["Accuracy"],
   sensitivity=conf_matrix$byClass["Sensitivity"],
   specificity=conf_matrix$byClass["Specificity"],
   duration=duration,
   memory=as.numeric(object.size(train_glm))
)

vv <- varImp(train_glm)
tibble(name=rownames(vv$importance), importance=round(vv$importance[[1]], digits=1)) %>%
  filter(importance >= 50.0) %>% arrange(desc(importance)) %>% knitr::kable()
```

name	importance
spore.print.color.orange	100.0
stalk.root.rhizomorphs	70.0
spore.print.color.gray	68.7
spore.print.color.red	68.7
odor.anise	68.2
spore.print.color.brown	66.4
spore.print.color.chocolate	66.4
spore.print.color.pink	66.4
spore.print.color.buff	66.3
gill.color.buff	58.3
ring.type.evanescent	51.5

```
rm(start_time, end_time, duration, y_hat, conf_matrix, vv)
```

Linear Discriminant Analysis

https://en.wikipedia.org/wiki/Linear_discriminant_analysis

```
start_time <- Sys.time()
train_lda <- train(
  train_set$x,
  factor(train_set$y),
  trControl = train_control,
  preProcess = c("center"),
  method = "lda"
)
end_time <- Sys.time()
duration <- as.numeric(difftime(end_time, start_time, units="secs"))
y_hat <- predict(train_lda, train_set$x)
conf_matrix <- confusionMatrix(data=y_hat, reference=train_set$y)
```

```
models_results <- bind_rows(
  models_results,
  data.frame(
   method= "lda",
   accuracy=conf_matrix$overall["Accuracy"],
   sensitivity=conf_matrix$byClass["Sensitivity"],
   specificity=conf_matrix$byClass["Specificity"],
   duration=duration,
   memory=as.numeric(object.size(train_lda))
  )
)

vv <- varImp(train_lda)
tibble(name=rownames(vv$importance), importance=round(vv$importance[[1]], digits=1)) %>%
  filter(importance >= 50.0) %>% arrange(desc(importance)) %>% knitr::kable()
```

name	importance
odor.musty	100.0
ring.type.none	71.8
odor.creosote	71.5
stalk.surface.above.ring.grooves	69.8
stalk.surface.below.ring.grooves	67.6
bruises	65.1
gill.size.narrow	64.4
stalk.surface.above.ring.scaly	62.6
gill.color.buff	58.0
population.several	57.9
stalk.surface.below.ring.scaly	54.5
spore.print.color.cinnamon	50.9

```
rm(start_time, end_time, duration, y_hat, conf_matrix, vv)
```

Generalized Additive Model using LOESS

https://en.wikipedia.org/wiki/Generalized_additive_model

```
set.seed(3, sample.kind = "Rounding")
start_time <- Sys.time()
train_loess <- train(
  train_set$x,
  factor(train_set$y),
  trControl = train_control,
  method = "gamLoess"
)
end_time <- Sys.time()
duration <- as.numeric(difftime(end_time, start_time, units="secs"))
y_hat <- predict(train_loess, train_set$x)
conf_matrix <- confusionMatrix(data=y_hat, reference=train_set$y)

models_results <- bind_rows(
  models_results,
  data.frame(
```

```
    method= "loess",
    accuracy=conf_matrix$overall["Accuracy"],
    sensitivity=conf_matrix$byClass["Sensitivity"],
    specificity=conf_matrix$byClass["Specificity"],
    duration=duration,
    memory=as.numeric(object.size(train_loess))
  )
)

rm(start_time, end_time, duration, y_hat, conf_matrix)
```

k-Nearest Neighbors

https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

```
set.seed(2015, sample.kind = "Rounding")
start_time <- Sys.time()
train_knn <- train(
  train_set$x,
  factor(train_set$y),
  trControl = train_control,
  tuneGrid = data.frame(k=c(2)),
  method = "knn"
)
end_time <- Sys.time()
duration <- as.numeric(difftime(end_time, start_time, units="secs"))
y_hat <- predict(train_knn, train_set$x)
conf_matrix <- confusionMatrix(data=y_hat, reference=train_set$y)

models_results <- bind_rows(
  models_results,
  data.frame(
    method= "knn",
    accuracy=conf_matrix$overall["Accuracy"],
    sensitivity=conf_matrix$byClass["Sensitivity"],
    specificity=conf_matrix$byClass["Specificity"],
    duration=duration,
    memory=as.numeric(object.size(train_knn))
  )
)

vv <- varImp(train_knn)
tibble(name=rownames(vv$importance), importance=round(vv$importance[[1]], digits=1)) %>%
  filter(importance >= 50.0) %>% arrange(desc(importance)) %>% knitr::kable()
```

name	importance
odor.musty	100.0
ring.type.none	71.8
odor.creosote	71.5
stalk.surface.above.ring.grooves	69.8
stalk.surface.below.ring.grooves	67.6
bruises	65.1
gill.size.narrow	64.4

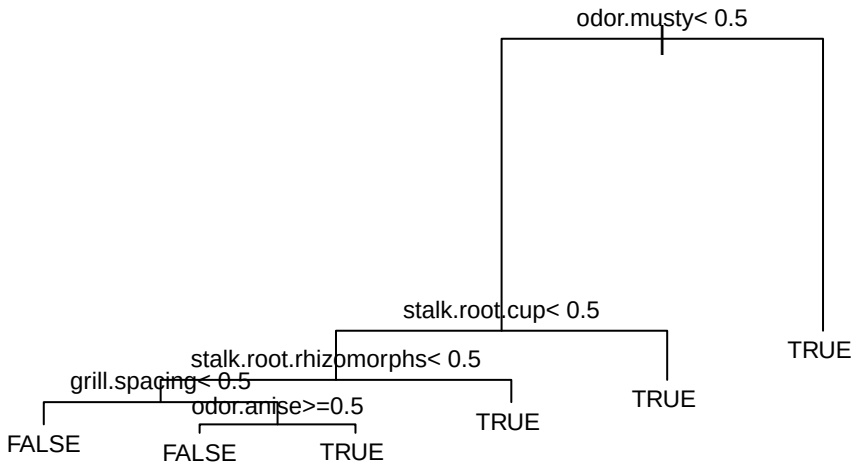
name	importance
stalk.surface.above.ring.scaly	62.6
gill.color.buff	58.0
population.several	57.9
stalk.surface.below.ring.scaly	54.5
spore.print.color.cinnamon	50.9

```
rm(start_time, end_time, duration, y_hat, conf_matrix, vv)
```

Classification and Regression Tree

https://en.wikipedia.org/wiki/Decision_tree_learning

```
set.seed(2019, sample.kind = "Rounding")
start_time <- Sys.time()
train_rpart <- train(
  train_set$x,
  factor(train_set$y),
  trControl = train_control,
  tuneGrid = data.frame(cp=seq(0.01,0.1,0.01)),
  method = "rpart"
)
end_time <- Sys.time()
duration <- as.numeric(difftime(end_time, start_time, units="secs"))
y_hat <- predict(train_rpart, train_set$x)
conf_matrix <- confusionMatrix(data=y_hat, reference=train_set$y)

plot(train_rpart$finalModel, margin = 0.1)
text(train_rpart$finalModel, cex = 0.75)
```




```
models_results <- bind_rows(
  models_results,
  data.frame(
    method= "rpart",
    accuracy=conf_matrix$overall["Accuracy"],
    sensitivity=conf_matrix$byClass["Sensitivity"],
    specificity=conf_matrix$byClass["Specificity"],
    duration=duration,
    memory=as.numeric(object.size(train_rpart))
  )
)

rm(start_time, end_time, duration, y_hat, conf_matrix)
```

Random Forest

https://en.wikipedia.org/wiki/Random_forest

```
set.seed(2016, sample.kind = "Rounding")
start_time <- Sys.time()
train_rf <- train(
  train_set$x,
  factor(train_set$y),
  trControl = train_control,
  tuneGrid = data.frame(mtry = seq(3,7,2)),
  method = "rf"
```

```
)
end_time <- Sys.time()
duration <- as.numeric(difftime(end_time, start_time, units="secs"))
y_hat <- predict(train_rf, train_set$x)
conf_matrix <- confusionMatrix(data=y_hat, reference=train_set$y)

models_results <- bind_rows(
  models_results,
  data.frame(
    method= "rf",
    accuracy=conf_matrix$overall["Accuracy"],
    sensitivity=conf_matrix$byClass["Sensitivity"],
    specificity=conf_matrix$byClass["Specificity"],
    duration=duration,
    memory=as.numeric(object.size(train_rf))
  )
)

vv <- varImp(train_rf)
tibble(name=rownames(vv$importance), importance=round(vv$importance[[1]], digits=1)) %>%
  filter(importance >= 50.0) %>% arrange(desc(importance)) %>% knitr::kable()
```

name	importance
odor.musty	100
odor.creosote	52
gill.size.narrow	52

```
rm(start_time, end_time, duration, y_hat, conf_matrix, vv)
```

Results

```
models_results %>% knitr::kable(caption="Models results")
```

Table 6: Models results

method	accuracy	sensitivity	specificity	duration	memory
glm	1.000	1.000	1	15.97	21386328
lda	1.000	0.999	1	3.96	4713392
loess	1.000	1.000	1	11.80	21907784
knn	1.000	1.000	1	9.02	9113056
rpart	0.981	0.961	1	2.12	10568968
rf	1.000	1.000	1	374.63	7440296

As mentionned in the Methods subsection, the purpose of this analysis is to provide a prediction model in order to tell if a mushroom is edible or poisonous.

The most of the model trained have an accuracy of 1 - all except for `rpart`. Its tuning has not been optimized in order to plot an interpretable tree. `lda` has a sensitivity, so following its prediction someone could be poisoned when the model is predicting that the mushroom is edible.

The predictions provided by `glm`, `knn` and `rf` are equivalent and all corrected. We can either create a ensemble choosing by majority vote the correct solution using the three models, or simply pick `knn` model since it is the fastest to train and using a reasonable amount of memory.

```
confusionMatrix(predict(train_knn, test_set$x), test_set$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE  1175    0
##      TRUE      0 1263
##
##                Accuracy : 1
##                  95% CI : (0.998, 1)
##     No Information Rate : 0.518
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.000
##             Specificity : 1.000
##          Pos Pred Value : 1.000
##          Neg Pred Value : 1.000
##              Prevalence : 0.482
##          Detection Rate : 0.482
##    Detection Prevalence : 0.482
##       Balanced Accuracy : 1.000
##
##        'Positive' Class : FALSE
##
```

Conclusions

For the latest assignment of PH125.x specialization, I was initially considering some trending topic such as Covid-19 data sets (I found multiple versions on Kraggle). However, I do not have a medical background to provide some meaningful insights (I always keep in mind the examples listed in several websites about spurious correlations). The instruction of the WHO have not been followed in the same way across all the countries, so the analysis would have been very weak.

When I was a child, I used to collect wild mushrooms with my grand parents. I've always been a bit scared of eating a poisonous one even if I trusted my grandpa. For this project, I wanted to check if machine learning is confirming my grand parents tips.

My grandparents "algorithms" - as far as I remember - was:

- touch the gill attachment (prefer when it's *spongy*)
- smell the odor
- check the stalk and the rings

This seems to be pretty much coherent with the result of the knn model we used for the solution.

The most interesting takeways of this analysis were about the techniques to encode data and avoid overfitting. Even the rudimental benchmark done on the methods could be a first step for additional developments.

We may extend the scope of this research including:

- validation of the model against more recent observation (the data set used is almost 40 years old)
- inclusion of more mushroom families other than Agaricus and Lepiota
- evaluation of different ways to encode categorical features