

Data Science ‘Mushrooms’ Capstone Project

András Gelencsér

14 December 2019

Contents

| | |
|------------------------------------|-----------|
| Overview | 2 |
| Methods & analysis | 3 |
| Preparation | 3 |
| Available Data | 4 |
| Feature analysis | 5 |
| The classification model | 19 |
| Decision tree model | 19 |
| Feature based model | 19 |
| Implementation & result | 20 |
| Decision tree model | 20 |
| Feature based model | 20 |
| Conclusion | 21 |

Overview

This document describes the result of the capstone project for the HarvardX Data Science course based on the Mushrooms dataset available in the UCI Machine Learning Repository (Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [<http://archive.ics.uci.edu/ml>]. Irvine, CA: University of California, School of Information and Computer Science.)

The “Mushrooms” dataset was recorded down from The Audubon Society Field Guide to North American Mushrooms (1981). G. H. Lincoff (Pres.), New York: Alfred A. Knopf and is available on the <https://archive.ics.uci.edu/ml/datasets/mushroom> url.

The goal of this project is to develop a model based on data science and machine learning techniques to predict if a mushroom is eatable based on the known attributes.

The dataset contains 8124 records and uses 22 attributes to describe the mushrooms. Each dataset contains additional attribute to flag if the mushroom is edible or not.

I used 80% of the available data for train the model. The remaining 20% will be used as test dataset for evaluating the developed model.

For the developed model is very important to detect poisonous mushrooms to avoid possible health damages or death. Therefore the model performance will be evaluated with the F1 score instead of the overall accuracy. Because the models use the edible prediction as true value, I will try to get a high specificity with the developed models..

The following steps were done during the project for developing the algorithm:

1. Download the data and prepare for the analysis
2. Analyze the data structure
3. Define the classification model based on the result the data analysis
4. Implement and train the model based on the train set
5. Review the model based on the test set

I developed two different prediction model. The first model uses a decision tree, the second model uses a reduced set of the features with high correlation to the edibility.

Both developed models reached an F1 score of 1 with sensitivity 1 and specificity 1. That shows, that both models had a perfect prediction for the test set.

Methods & analysis

Preparation

For the analysis the data we will use the following R packages:

- tidyverse
- caret
- gridExtra
- ggplot2

the following R code loads and install the required packages if needed:

```
if(!require(tidyverse)) install.packages("tidyverse",
                                          repos = "http://cran.us.r-project.org", dependencies = TRUE)
if(!require(caret)) install.packages("caret",
                                     repos = "http://cran.us.r-project.org", dependencies = TRUE)
if(!require(gridExtra)) install.packages("gridExtra",
                                          repos = "http://cran.us.r-project.org", dependencies = TRUE)
if(!require(ggplot2)) install.packages("ggplot2",
                                       repos = "http://cran.us.r-project.org", dependencies = TRUE)
```

At first, we need to prepare the data set for the analysis. The mushrooms dataset contains 8,124 records about different mushrooms.

The following R code downloads the data from the <https://archive.ics.uci.edu/ml/datasets/mushroom> site. To avoid unnecessary data traffic, the data will be downloaded only if not done yet.

```
#create directory for the data file if necessary
if (!dir.exists("mushrooms")){
  dir.create("mushrooms")
}
baseurl = "https://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/"
files_to_download = c("agaricus-lepiota.data", "agaricus-lepiota.names")

for (f in files_to_download){
  #download the files only if not done yet
  if (!file.exists(paste("./mushrooms/",f,sep=''))){
    {
      download.file(paste(baseurl,f,sep=''), paste("./mushrooms/",f,sep=''))
    }
  }
}

f1 <- file("mushrooms/agaricus-lepiota.data")
mushroom_data <- str_split_fixed(readLines(f1), ",", 23)
close(f1)
```

The dataset is stored in the “agaricus-lepiota.data” file. This file contains 23 columns separated via comma. There is no header line in the file. The description file “agaricus-lepiota.names” gives us information about the classes and the 22 attributes in the data file.

So we can extract the data by parsing the data file. The following R code reads the data and stores in a data frame:

```
f1 <- file("mushrooms/agaricus-lepiota.data")
mushroom_data <- str_split_fixed(readLines(f1), ",", 23)
close(f1)
```

```

#set the column names for the features
colnames(mushroom_data) <- c("classes", "cap-shape", "cap-surface", "cap-color",
                             "bruises?", "odor", "gill-attachment",
                             "gill-spacing", "gill-size", "gill-color", "stalk-shape",
                             "stalk-root", "stalk-surface-above-ring",
                             "stalk-surface-below-ring", "stalk-color-above-ring",
                             "stalk-color-below-ring", "veil-type", "veil-color",
                             "ring-number", "ring-type", "spore-print-color",
                             "population", "habitat")

#convert to data frame
mushroom_data <- as.data.frame(mushroom_data)

#remove temporary file variable
rm(fl)

```

For training the classification method we will use 80% of the available data as training set. The remaining data will be used for evaluating the developed method. The following R code creates the train and test set:

```

#initialize random sequenz
set.seed(1, sample.kind = "Rounding")
#create index for train and test set
#20% of the data will be used for the test set
test_idx = createDataPartition(y = mushroom_data$classes, times=1, p=0.2, list=FALSE)
train_data = mushroom_data[-test_idx,]
test_data = mushroom_data[test_idx,]
#remove temporary variables
rm(mushroom_data, test_idx)

```

Available Data

The dataset contains 8,124 records. For the analysis we have data about 6,500 mushrooms (80% of all data). All the features are categorical with different possible values.

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

For testing the performance the prediction model we have about 1,600 mushrooms data (20% of all data). The dataset contains only the short 1 character code of the attribute values, but it's not disturbing for the model development.

Feature analysis

As first step we can investigate the distribution of the features corresponding to the edibility classification.

The following code plots this distribution for each feature.

```
feature_distribution_plot <- function(data){
  plots <- list()
  #Plots all the attributes for eatable <-> poisonous
  for (i in 1:(ncol(data)-1))
  {
    summarized_data <- data %>% group_by(classes, .[,i+1]) %>% summarise(n = n())
    names(summarized_data)[2] <- "attr"
    plot <- summarized_data %>% ggplot(aes(attr, classes)) + geom_point(aes(size=n)) +
      xlab(names(data)[i+1]) + ylab("Edibility")
    plots[[i]] <- plot
  }
  rm(summarized_data, i, plot)
  grid.arrange(grobs=plots,ncol=3)
}

feature_distribution_plot(train_data)
```

We can see, that the feature veil-type has only the value 'p' (partial), there is no data record with the veil-type value 'u' (universal). This feature can not provide decision information about the mushrooms, therefore we can remove this feature from the dataset.

On the other side, there are some features with values that can decide the classification. E.g. if we see a mushroom with the value 'f' (foul) for the feature odor, we can classify the mushroom as poisonous. The feature values 'a', 'c', 'f', 'l', 'm', 'p', 's', 'y' for the attribute odor can give us either 'e' or 'p' edibility classification. With this information we can classify 3,689 mushrooms. We can not classify only the mushrooms with the odor value 'n' (none). There are 2,809 mushrooms remaining in the dataset without classification.

We can try to analyse all the mushrooms, where the odor value is 'n'. For this analysis we can remove the attribute odor from the dataset, because we examine further only a specific odor value.

The following code removes the odor attribute and plots the feature distribution for the remaining data:

```
odor_n <- train_data %>% filter(`odor` == 'n') %>% select(-`odor`)
feature_distribution_plot(odor_n)
```

In the reduced data set we see again some features that we can use for classification the mushrooms.

E.g. if we see a mushroom with the value 'k' (black) for the feature spore-print-color, we can classify the mushroom as edible. The feature values 'b', 'h', 'k', 'n', 'o', 'r', 'y' for the feature spore-print-color can give us a classification for the remaining data set. We can not make a classification only if the spore-print-color value is 'w' (white).

We can reuse the same logic like for the feature odor and filter the remaining data with the spore-print-color feature value 'w'.

The following code removes the odor attribute and plots the feature distribution for the remaining data:

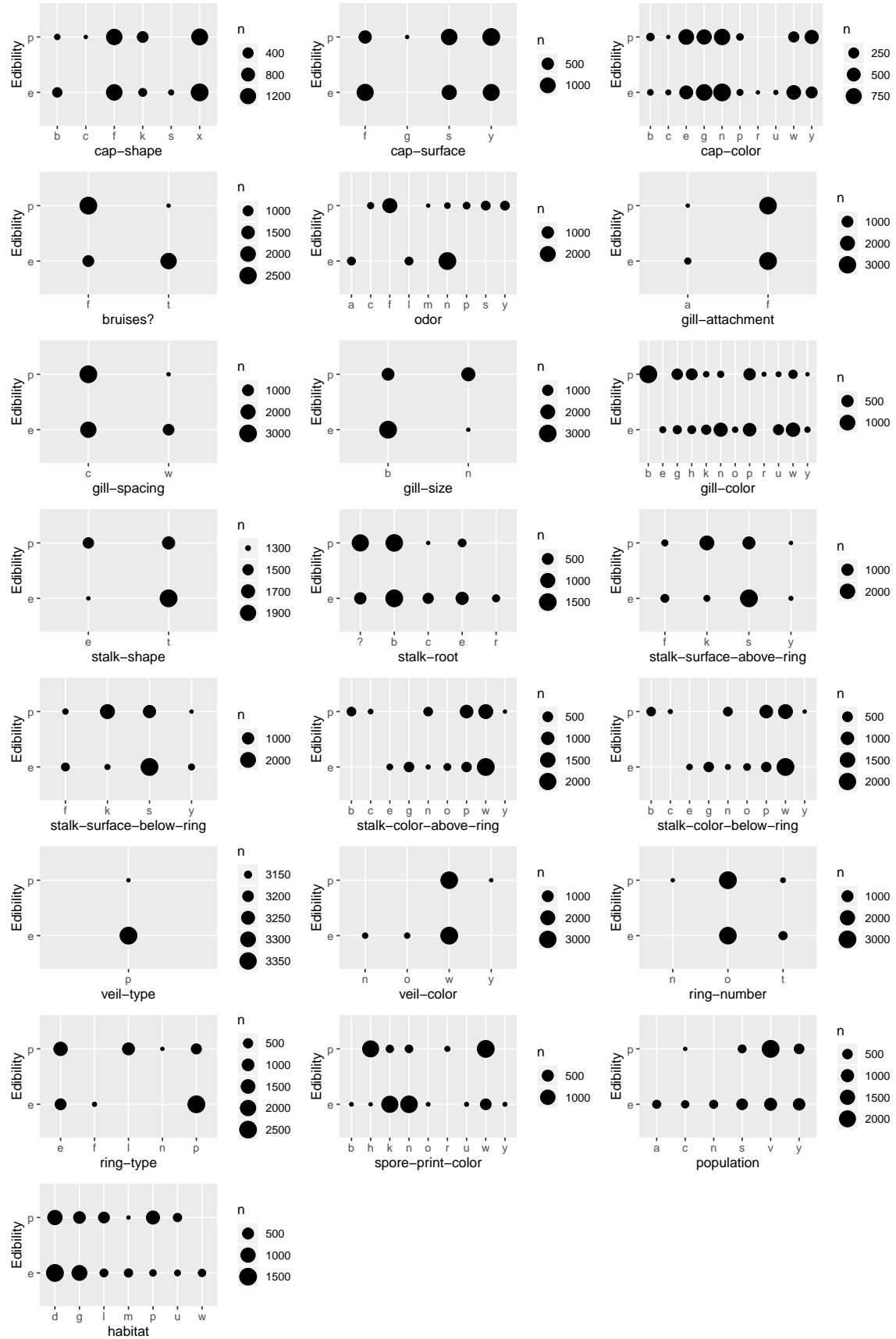


Figure 1: Figure: Features value distribution per edibility

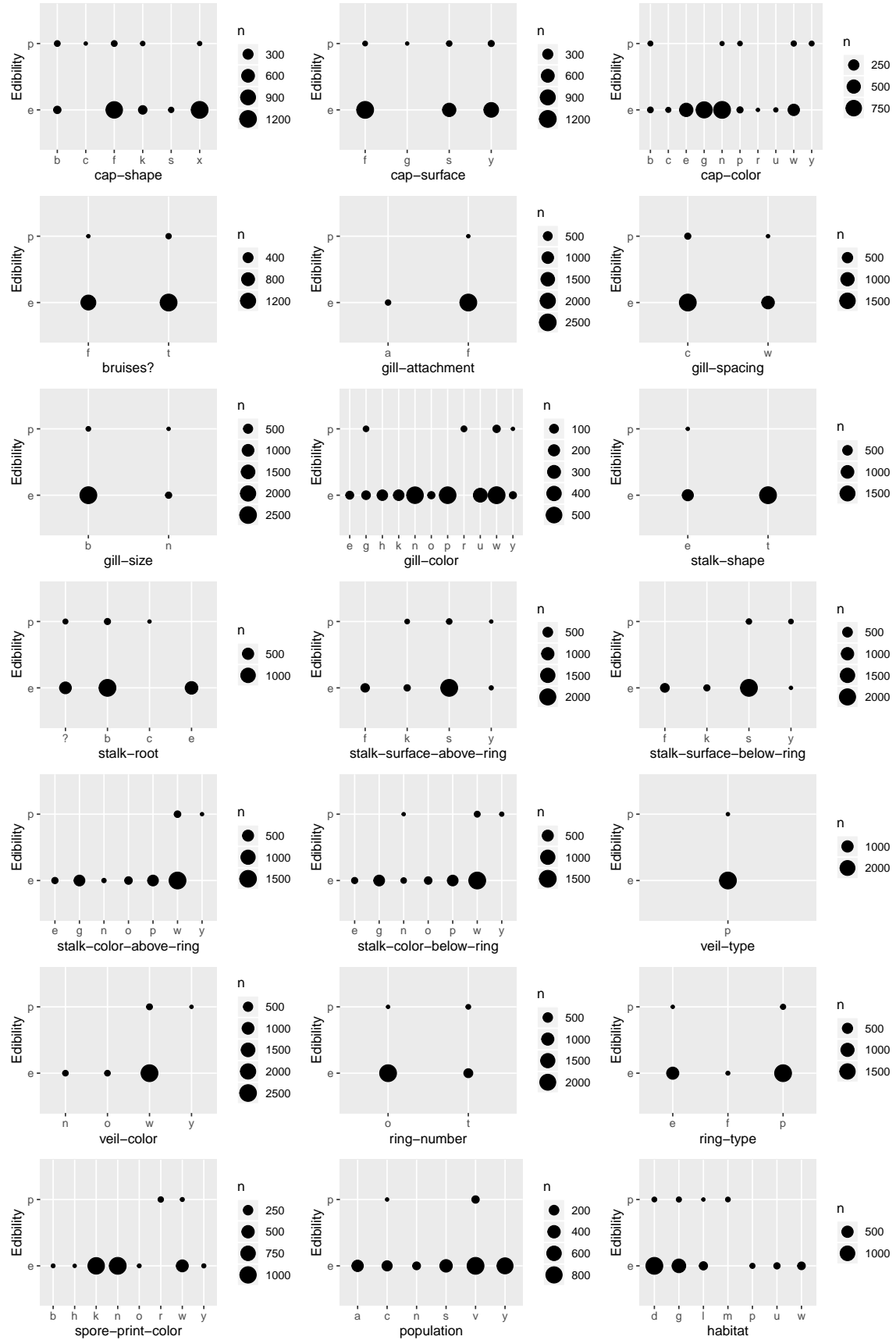
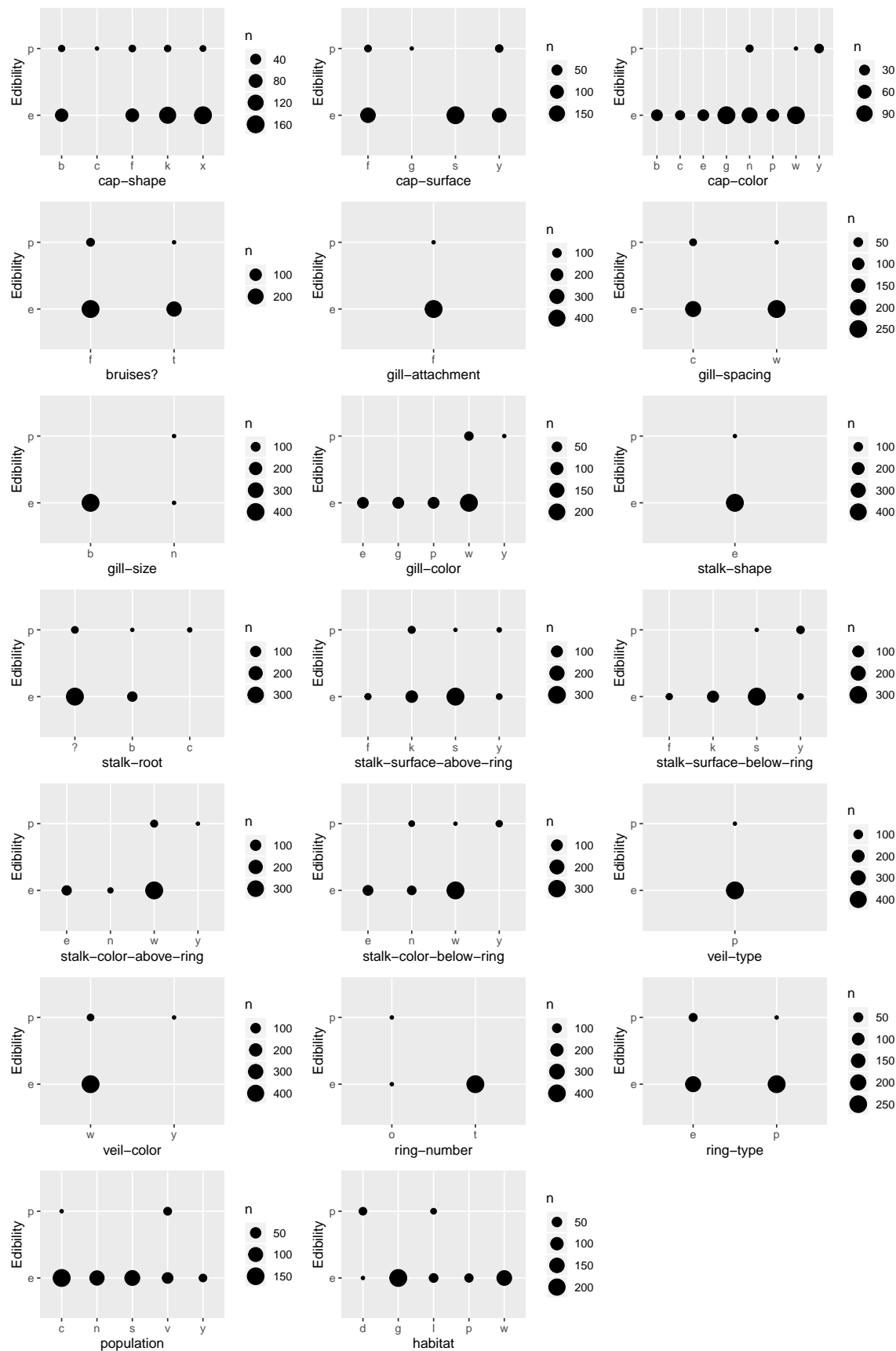


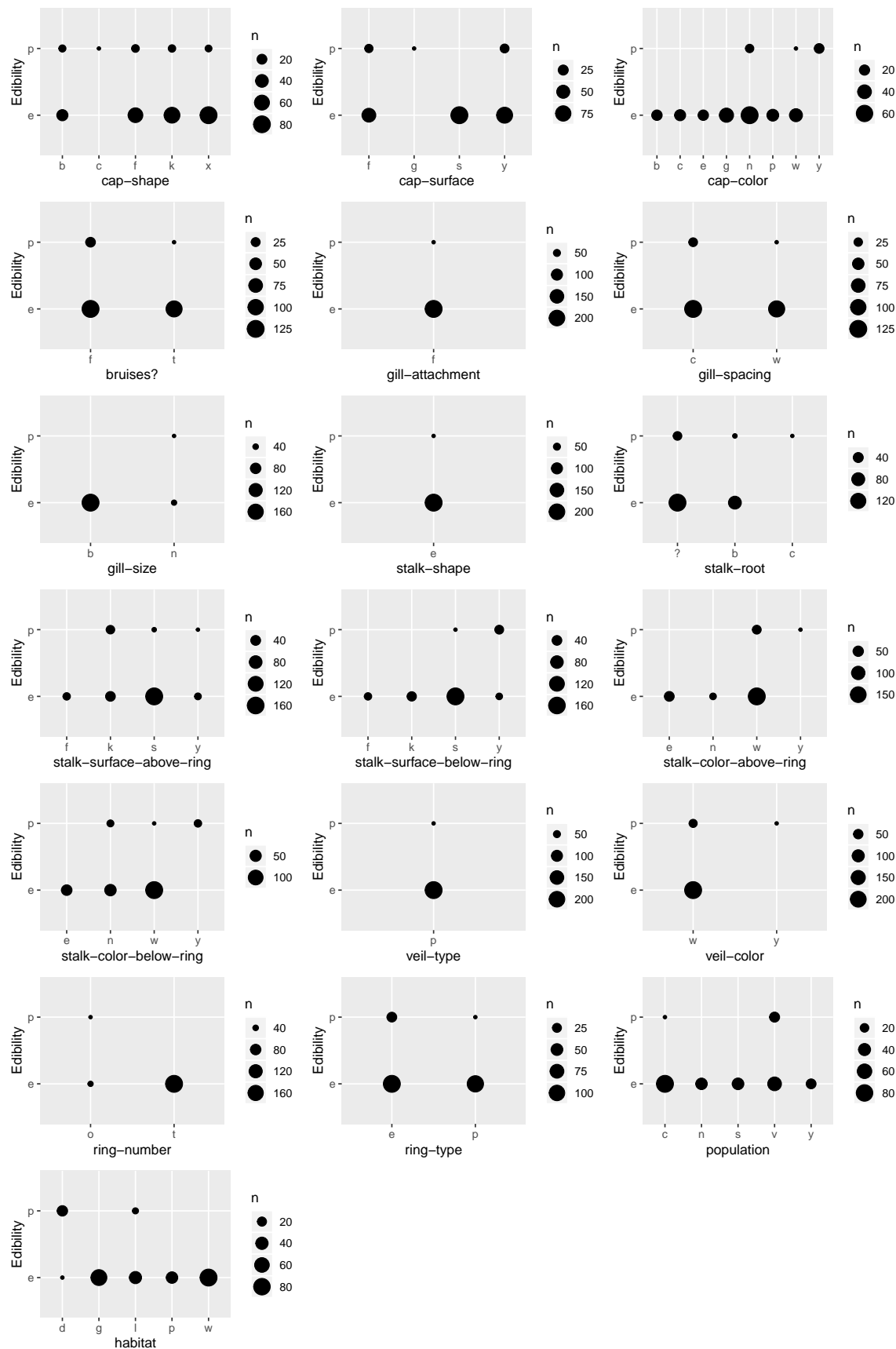
Figure 2: Figure: Features value distribution per edibility step 2

```
spore_print_color_w <- odor_n %>% filter(`spore-print-color` == 'w') %>%  
  select(-`spore-print-color`)  
feature_distribution_plot(spore_print_color_w)
```

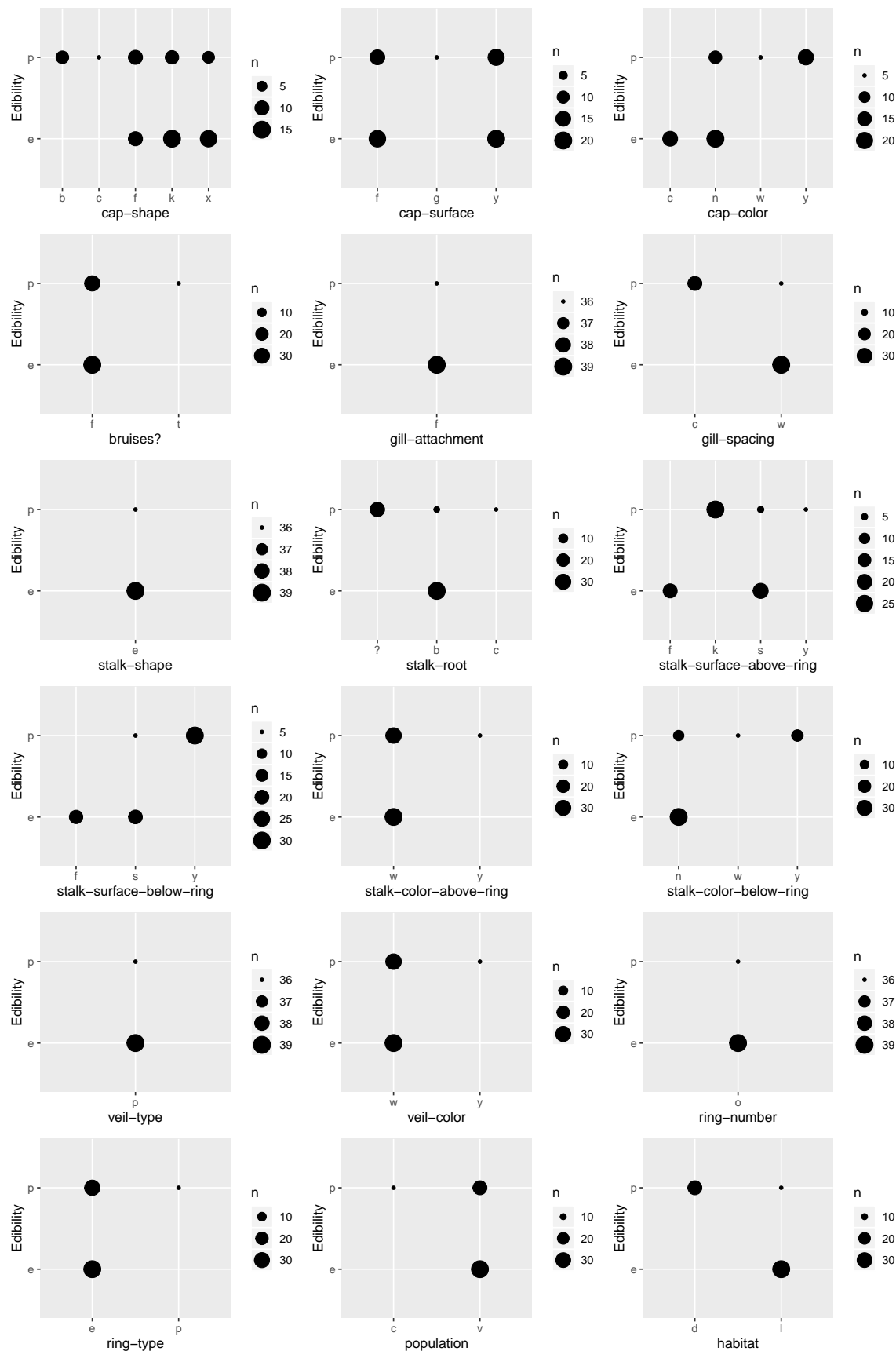
We can repeat this analysis and look for features that we can use for classifying the remaining data set. As next step we can get the gill-color feature and examine only the mushrooms with the gill color 'w' (white).

```
gill_color_w <- spore_print_color_w %>% filter(`gill-color` == 'w') %>%  
  select(-`gill-color`)  
feature_distribution_plot(gill_color_w)
```



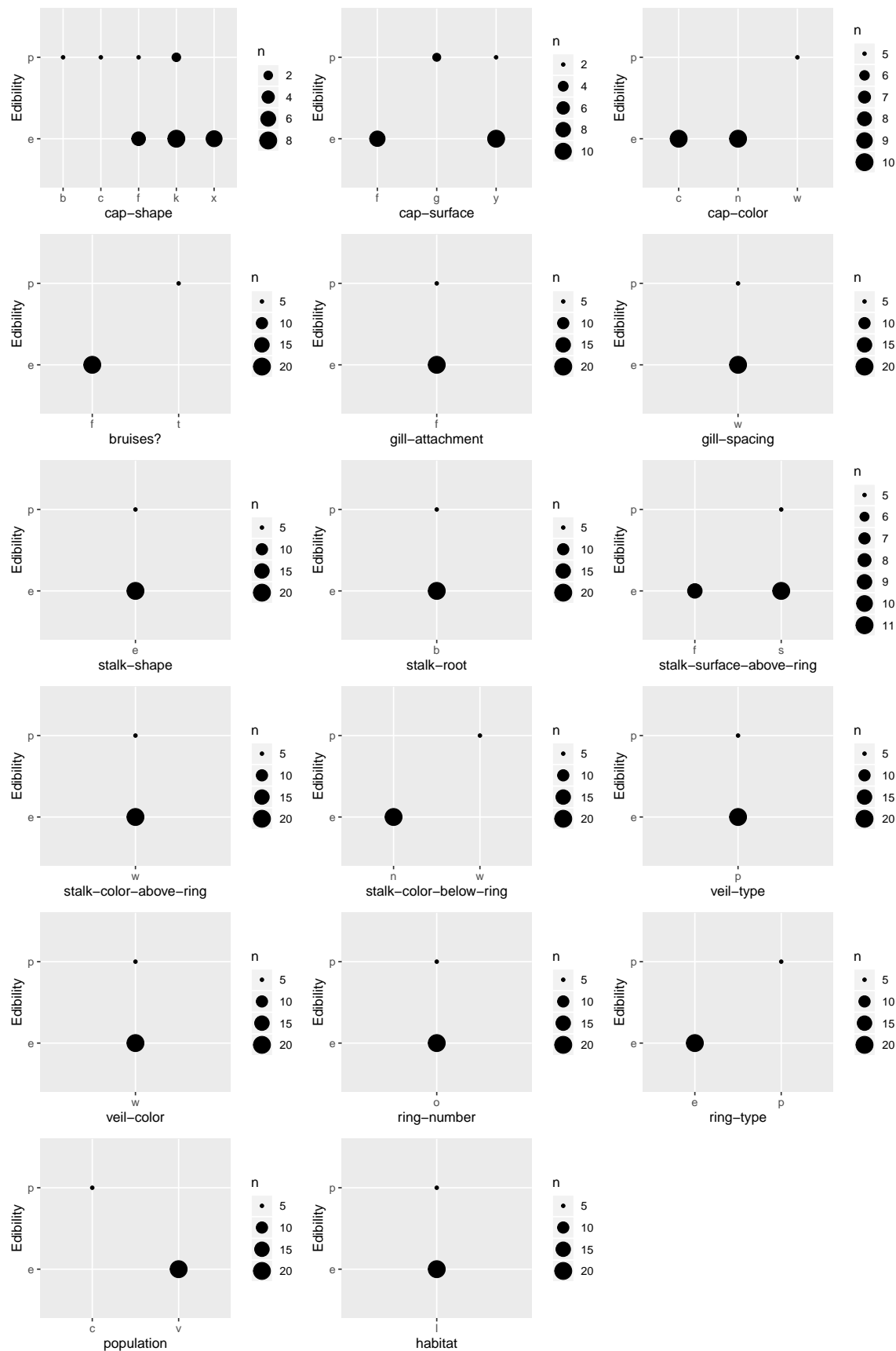
As next step we can get the gill-size feature and examine only the mushrooms with the grill size 'n' (narrow).

```
gill_size_n <- gill_color_w %>% filter(`gill-size` == 'n') %>% select(-`gill-size`)
feature_ditribution_plot(gill_size_n)
```



As next step we can get the stalk-surface-below-ring feature and examine only the mushrooms with the stalk-surface-below-ring 's' (smooth).

```
stalk_surface_below <- gill_size_n %>% filter(`stalk-surface-below-ring` == 's') %>%  
  select(-`stalk-surface-below-ring`)  
feature_distribution_plot(stalk_surface_below)
```



The re-

maining dataset has only 25 mushrooms. As we can see on the plot, now we can use some feature for classifying this remaining mushrooms. E.g. the feature ring-type give us a good classification: if we have a ring type of 'e' (evanescent) than the mushroom is edible. On the other case, if we have a ring-type of 'p' (pedant), the mushroom is poisonous.

With the steps about we defined a decision tree for classifying all the mushrooms in the train set. We can use this information to develop a classification model for the mushrooms.

The analyse above give us an other important information: some fatures are more relevant for the classification some features gives no additional information for the method. That means, the different features have different information weight for the classification.

We have to try to calculate a correlation between the features and the edibility. For the unordered categorical features we can use the uncertainty coefficient as correlation score. (See: https://en.wikipedia.org/wiki/Uncertainty_coefficient)

The uncertainty score builds on the conditinal entropy form the information theory.

The entropy for a single feature is defined as follows:

$$H(X) = - \sum_x P_X(x) \log P_X(x)$$

The conditional entropy of X given Y is defined as follows:

$$H(X|Y) = - \sum_{x,y} P_{X,Y}(x,y) \log P_{X|Y}(x|y)$$

The uncertainty coefficient of the two features can be calculated as follows:

$$U(X|Y) = \frac{H(X) - H(X|Y)}{H(X)}$$

We can implement an R function to calculate the entropy, conditional entropy and the uncertainty coefficient.

The following R code implements the calculation functions:

```
entropy <- function(dataset, colX){

  ret <- 0
  summarized_data <- dataset %>% group_by(.[,colX]) %>% summarise(n = n())
  rowCount <- nrow(dataset)
  level_items <- levels(dataset[,colX])
  for (item in level_items){
    prob <- summarized_data %>% filter(.[,1] == item) %>% pull(n) / rowCount
    ret <- ret + prob * log(prob,base = 2)
  }
  return (-ret)
}

cond_entropy <- function(dataset, colX, colY){
  ret <- 0
  rowCount <- nrow(dataset)
  summarized_y <- dataset %>% group_by(.[,colY]) %>% summarise(n = n())
  names(summarized_y)[1] <- "Y"
  level_itemsy <- levels(summarized_y$Y)

  for (itemy in level_itemsy){
    proby <- summarized_y %>% filter(Y == itemy) %>% pull(n) / rowCount
  }
}
```



```

summarized_x <- dataset %>% filter(.[,colY] == itemy) %>% group_by(.[,colX]) %>%
  summarise(n = n())
names(summarized_x)[1] <- "X"
level_itemsx <- levels(summarized_x$X)
for (itemx in level_itemsx){
  filtered <- summarized_x %>% filter(X == itemx)
  if (nrow(filtered) > 0 && probxy != 0){
    probxy <- filtered$n / rowCount
    probx_at_y <- probxy / proby
    ret <- ret + probxy * log(probx_at_y, base = 2)
  }
}
}
return (-ret)
}

uncertainty <- function(dataset, colX, colY){
  entr <- entropy(dataset,colX)
  cond_entr <- cond_entropy(dataset, colX, colY)
  if (entr == 0){
    return(0)
  }
  return ( (entr - cond_entr) / entr)
}

```

We can plot the uncertainty coefficient between all the features as a matrix. The following R code creates the plot:

```

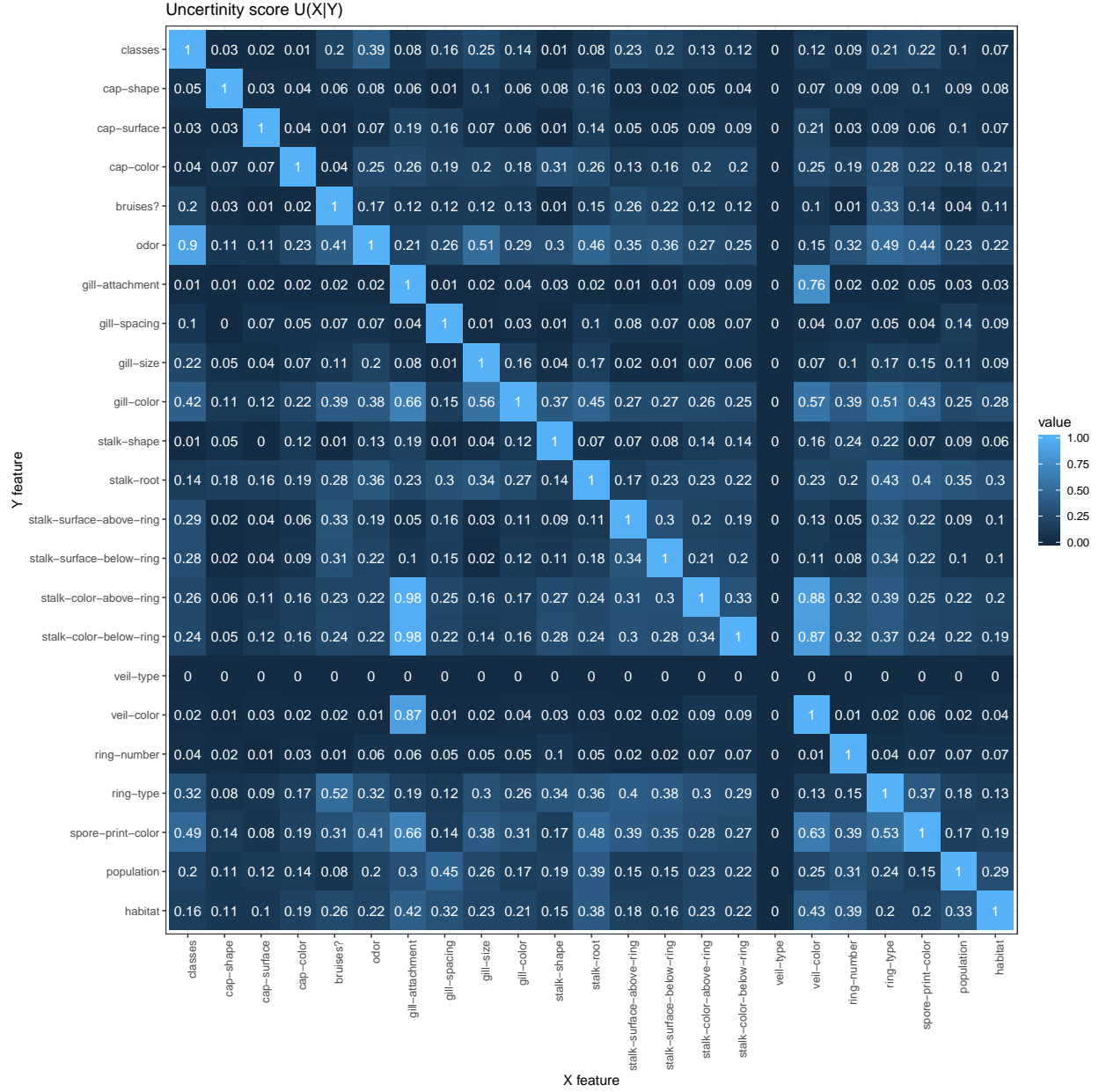
uncertainty_plot <- function(dataset){
  #calculate uncertainty for all feature pairs
  uncertainty_df <- data_frame(X=character(), idx=numeric(), Y=character(),
                              idy=numeric(), value=numeric())

  labels <- names(dataset)
  for (i in 1:ncol(dataset))
  {
    for(j in 1:ncol(dataset))
    {
      uncertainty_df <- bind_rows(uncertainty_df, data_frame(Y = labels[j], idy=j ,
                                                             X=labels[i], idx=i, value=uncertainty(train_data,i,j)))
    }
  }

  #plot the uncertainty with colors
  uncertainty_df %>% ggplot(aes(x=reorder(X,idx), y=reorder(Y,-idy), fill=value)) +
    geom_tile() + geom_text(aes(label=round(value,2)), color='white') +
    theme_bw() + theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
    xlab('X feature') + ylab('Y feature') + ggtitle('Uncertainty score U(X|Y)')
}

uncertainty_plot(train_data)

```



This plot shows the uncertainty coefficient between all feature pairs. The light blue values show a high uncertainty coefficient (high correlation) the dark blue values shows a low uncertainty coefficient (low correlation) The value 1 in the matrix shows a perfect correlation, therefore the matrix diagonal elements are 1. (The diagonal elements show the correlation of the features with himself, therefore there is a perfect correlation)

We see, that the feature veil-type has 0 correlation with all other features. In the first data analysis we saw, that the feature veil-type has only one value and has no prediction information, that causes the 0 correlation.

For the classification model we examine the first column of the matrix. This column shows the correlation between edibility and other feature. As we see, the feature odor has the highest correlation with the edibility. The other feature have only a correlation below 0.5.

We can use this information for developing a classification model that use only the features with high correlation with the edibility.

The classification model

The classification model will be evaluated with the F_1 score. This score includes not only the overall accuracy, it considers the prediction specificity and sensitivity too.

The F_1 score is the harmonic average of the precision and recall :

$$F_1 = \frac{1}{\frac{1}{2}(\frac{1}{recall} + \frac{1}{precision})}$$

where recall is the true positiv rate (sensitivity) and precision is the positiv prediction value (specificity).

The model should predict the edibility of the mushrooms, therefore it's more important to make the right prediction for the poisonous mushrooms as the edible. Therefore if two models have the same F_1 score, the model with the higher true negativ rate (specificity) will be declared as better model.

After the data analysis we can develop two different model: * model based on simple decision rules * model based on the feature value combinations

In the following two chapter i will describe the two models.

Decision tree model

As we saw before, we can develop a prediction model based on the feature distribution between the edible and poisonus classes. A very easy way to define a decision tree is to examine the generated charts like we did it before.

On the other side, this method is fixed on the current train data set. If the dataset changes, we must to make the analysis again manually.

Therefore we need to develop a method to create a decision tree automatically. For this method we can use the same logik as we did:

1. create the edibility distribution for all features for the dataset
2. search for features, that have values with only either 'e' or 'p' edibility. If we don't find a feature, than we can't improve our decision tree anymore
3. if we found some feature in step 2, we select one of them and add to the decision tree
4. filter out all reords from the dataset that match the feature values with distinct edibility values
5. if there is no remaining record, than we reached a full decision tree for the train set
6. if there are remaining records, we start at step 1 again, but we use only the remaining record for the next loop

With this method we can find a decision tree for the train set, that we can use for predict the edibility for the test dataset.

For the prediction, we can go throug the features in the prediction tree and check if the feature in the actual record matches either the feature values with dictinct 'e' or 'p' edibility. If we find a match, we use the corresponding edibility value as prediction. If there is no match for all the feature in the prediction tree, we predict 'p' edibility. This is, because, we want to be sure not to predict edible for a possible poisonous mushroom.

Feature based model

Implementation & result

Decision tree model

Feature based model

Conclusion