Identifying poisonous mushrooms with rule learners

Nithya Sarabudla

02-25-2024

Step 1 – collecting data

Step 2 – exploring and preparing the data

```r
# Load the mushroom dataset into R, converting strings to factors to utilize R's factor functionality
mushrooms <- read.csv("/cloud/home/r2519596/mushrooms.csv", stringsAsFactors = TRUE)
# Display the structure of the dataset to understand its features and types
str(mushrooms)
```

```
## 'data.frame':    8124 obs. of  23 variables:
##  $ type                    : 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 ...
```

This command reveals that the dataset contains 8124 observations of 23 variables, which include both the target variable (type) and predictive features (e.g., cap_shape, cap_color).

```r
# Remove the 'veil_type' variable from the dataset as it only contains one level, making it non-informa
mushrooms$veil_type <- NULL
# Examine the distribution of the target variable 'type' to check the balance between classes
table(mushrooms$type)
```

```
##
##    e    p
## 4208 3916
```

The veil_type variable is dropped because it does not vary across samples, thus offering no value for the prediction task. The table produces a count of edible and poisonous mushrooms, highlighting the dataset's class balance.

Step 3 – training a model on the data

```r
# Load the OneR package
library(OneR)
# Train a 1R classifier on the mushroom data, attempting to predict 'type' based on all other features
mushroom_1R <- OneR(type ~ ., data = mushrooms)
```

The 1R algorithm identifies the most predictive feature of the target class and constructs a rule based on this feature.

Step 4 – evaluating model performance

```r
# Generate predictions for the mushroom dataset using the trained 1R model
mushroom_1R_pred <- predict(mushroom_1R, mushrooms)
# Compare the actual types against the predicted types to evaluate the model's accuracy
table(actual = mushrooms$type, predicted = mushroom_1R_pred)
```

```
##       predicted
## actual    e    p
##      e 4208    0
##      p  120 3796
```

This confusion matrix shows how well the 1R model performed, specifically highlighting any misclassifications.The evaluation of the 1R classifier revealed a critical flaw: while it accurately identified all edible mushrooms, it misclassified 120 poisonous ones as edible, posing a significant safety risk. Despite its reliance on a single feature, the classifier performed reasonably well by leveraging odor as the distinguishing factor. However, in real-world applications where lives are at stake, such close accuracy is insufficient. To address this, additional rules are needed to enhance classification accuracy and ensure user safety, especially for field guide publishers who face potential liability issues if their readers fall ill due to misclassifications

Step 5 – improving model performance

```r
# Load the RWeka package for accessing Java-based machine learning algorithms, including JRip
library(RWeka)
```

```
##
## Attaching package: 'RWeka'

## The following object is masked from 'package:OneR':
##
##     OneR
```

```r
mushroom_JRip <- JRip(type ~ ., data = mushrooms)
# Train a JRip classifier on the mushroom data, which identifies a set of rules for predicting 'type'
mushroom_JRip
```

```
## JRIP rules:
## ===========
##
## (odor = f) => type=p (2160.0/0.0)
## (gill_size = n) and (gill_color = b) => type=p (1152.0/0.0)
## (gill_size = n) and (odor = p) => type=p (256.0/0.0)
## (odor = c) => type=p (192.0/0.0)
## (spore_print_color = r) => type=p (72.0/0.0)
## (stalk_surface_below_ring = y) and (stalk_surface_above_ring = k) => type=p (68.0/0.0)
## (habitat = l) and (cap_color = w) => type=p (8.0/0.0)
## (stalk_color_above_ring = y) => type=p (8.0/0.0)
##  => type=e (4208.0/0.0)
##
## Number of Rules : 9
```

JRip, a Java-based implementation of the RIPPER algorithm, is used for generating a more sophisticated set of rules.The JRip() classifier generated eight rules to distinguish between edible and poisonous mushrooms, which can be interpreted as if-else statements. For example, if the odor is foul, the mushroom is poisonous. Each rule covers specific instances and aims to minimize misclassifications. Interestingly, these rules achieved 100 percent accuracy, with no misclassifications observed. By iteratively identifying distinguishing features, the classifier successfully separated homogeneous segments of mushrooms, ultimately achieving perfect classification. This success highlights the distinct characteristics of mushroom varieties, enabling accurate classification and ensuring user safety.