Classification Task 1 - Agaricus Lepiota

Tolga Tabanli

2025-10-15

Intro

In this task, we are going to predict if a mushroom is edible given some of its nominal biological characteristics.

Data set is loaded with the column names (more information can be found in the .names file). We should always look at the raw data file to check for the correctness of the data format and how to handle it. We cast every column as a **factor**, since all the features are categorical and we thus let the data frame know of it as well.

```
library(tidyverse)
library(tidymodels)

data <- read_csv("data/agaricus-lepiota-expanded.data",
                 col_names = c("edibility",
                               "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")) %>%
  mutate(across(everything(), as.factor))
```

Explore the data

One way to understand the relationship of predictors with a nominal outcome is to plot proportions of levels of each feature "stratified" by outcome.

1. Bring the data frame into long format by using `pivot_longer` on all the columns except edibility to collect the variables under a columns named "predictor" and the values under a column named "value".
2. Calculate the proportion of edibility for each value of each predictor. **Hint**: Use `count` on predictor, value, and edibility followed by `group_by` on predictor and value.
3. Finally create a new column named "prop" with `mutate`. Hint: `mutate(prop = n / sum(n))` gives the proportion of each row in its group (already specified by group_by).

```
# TODO
```

Modelling

Data split

Create the data split.

```
# TODO
```

Workflows and tuning

We'll compare a single decision tree (engine: `rpart`) and random forest (engine: `ranger`).

1. Create a recipe to filter out variables with single values by using `step_nzv` and setting its `unique_cut` parameter to 0.
2. Specify a decision tree model with `decision_tree()`. Set the following parameters to `tune()`: `cost_complexity`, `tree_depth`, `min_n`.
3. Specify a random forest model: Set number of trees (`trees`), minimal node size (`min_n`) and the number of randomly selected variables for each split (`mtry`) as hyperparameters to tune with `tune()`.
4. In the random forest's engine, set the `importance` to impurity to later compare the predictors in terms of their importance.
5. Tune the hyperparameters with the seed 2025 and autoplot the results.

```
# Preprocessing for logistic regression: drop singleton columns
# TODO

# === Models ===
# TODO

# === Workflows ===
# TODO

# Tuning
# TODO

# autoplot
# TODO
```

Finalize and fit

Finalize the workflows with best performing hyperparameters.

```
# Select best pars and finalize
# TODO
```

Last fit and Testing

Fit the finalized workflows on the data split and show the test results with `collect_metrics()`.

```
# TODO
```

Optional: We can also visualize the decision tree with the help from rpart package: `rpart.plot::rpart.plot()` on the fitted model (which you get by 1. extracting workflow, 2. extracting fitted parsnip model, 3. `pluck()`ing "fit" field.). Here how it's done:

```
YOUR_FITTED_DECISION_TREE %>%
  extract_workflow() %>%
  extract_fit_parsnip() %>%
  pluck("fit") %>%
  rpart.plot::rpart.plot()
```

Optional: Investigate the importances of variables in the random forest model with `vip`.

```
library(vip)
# TODO
```