Classification Demo - Agaricus Lepiota

Tolga Tabanli

2025-10-15

Mushrooms

In this data set, we're going to focus more on the relationship of the exploratory data analysis and how it helps explain the results of the modelling process. We first load the data and add the column names we can read from 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 see there are some lines at the beginning that we should skip, and a line at the end that we should delete. Finally, 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", skip = 9,
                 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")) %>%
  head(-1) %>%
  mutate(across(everything(), as.factor))
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

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.

```
data %>%
  pivot_longer(!edibility, names_to = "predictor", values_to = "value") %>%
  count(predictor, value, edibility) %>%
  group_by(predictor, value) %>%
  mutate(prop = n / sum(n)) %>%
  ggplot(aes(x = value, y = prop, fill = edibility)) +
```

```
  geom_col(position = "dodge") +
  facet_wrap(~ predictor, scale = "free_x") +
  theme(axis.text.x = element_text(angle = 90, vjust = 1, hjust = 1))
```

Data split

```
split_data <- initial_split(data, prop = 0.75)
train <- training(split_data)
test <- testing(split_data)
```

Modelling

We'll compare logistic regression and random forests.

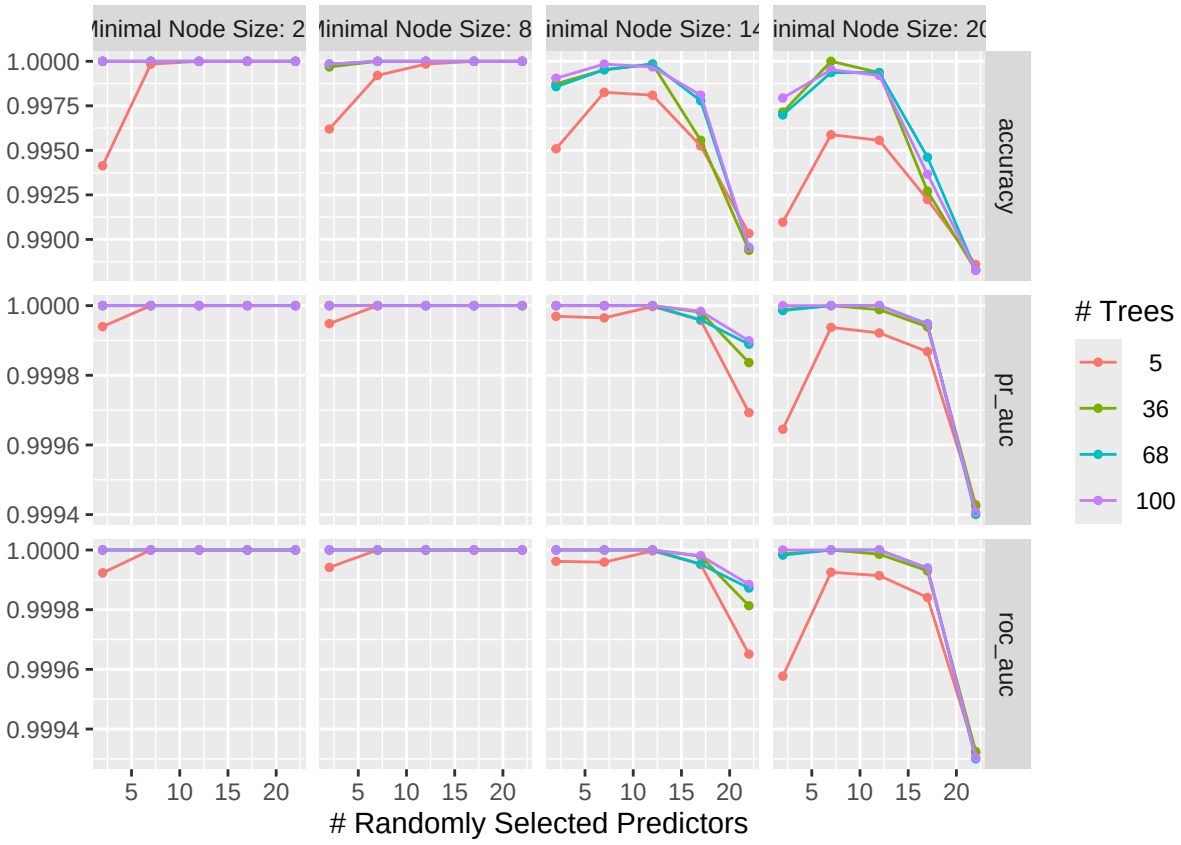
```
# Preprocessing for logistic regression: drop singleton columns
recipe_log <- recipe(edibility ~ ., data = train) %>%
  step_nzv(all_predictors(), unique_cut = 0) %>%
  prep(retain = TRUE)

log_reg <- logistic_reg(mode = "classification", engine = "glm")


rf <- rand_forest(mode = "classification", engine = "ranger",
                  mtry = tune(), trees = tune(), min_n = tune())

# Tuning for random forest parameters
tuned_rf <- tune_grid(rf,
                      preprocessor = edibility ~ .,
                      resamples = vfold_cv(train, v = 10),
                      grid = grid_regular(mtry(range = c(2, 22)),
                                          trees(range = c(5, 100)),
                                          min_n(range = c(20, 200)),
                                          levels = c(5, 4, 4)),
                      metrics = metric_set(yardstick::accuracy, roc_auc, pr_auc))

autoplot(tuned_rf)
```



Finalize and fit

```
final_rf <- finalize_model(rf, select_best(tuned_rf, metric = "accuracy"))
fit_rf <- last_fit(final_rf, split = split_data, preprocessor = edibility ~ .,
                   metrics = metric_set(accuracy, roc_auc, pr_auc))


fit_log <- fit(log_reg, data = bake(recipe_log, new_data = train),
               formula = edibility ~ .)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
test_processed <- bake(recipe_log, new_data = test)
log_preds <- predict(fit_log, new_data = test_processed, type = "prob") %>%
  bind_cols(predict(fit_log, new_data = test_processed, type = "class")) %>%
  bind_cols(test_processed %>% select(edibility))

best_rf <- fit(final_rf, data = test_processed, formula = edibility ~ .)
rf_preds <- predict(best_rf, new_data = test, type = "prob") %>%
  bind_cols(predict(best_rf, new_data = test, type = "class")) %>%
  bind_cols(test %>% select(edibility))
```

Now the performances on test set:

```
rf_metrics <- collect_metrics(fit_rf)

metrics <- metric_set(accuracy, roc_auc, pr_auc)(
  log_preds,
  truth = edibility,
  estimate = .pred_class,
  .pred_EDIBLE
)
metrics
```

```
## # A tibble: 3 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary             1
## 2 roc_auc  binary             1
## 3 pr_auc   binary             1
```

```
metrics <- metric_set(accuracy, roc_auc, pr_auc)(
  rf_preds,
  truth = edibility,
  estimate = .pred_class,
  .pred_EDIBLE
)
metrics
```

```
## # A tibble: 3 x 3
##   .metric  .estimator .estimate
```

```
##   <chr>    <chr>          <dbl>
## 1 accuracy binary             1
## 2 roc_auc  binary             1
## 3 pr_auc   binary             1
```