Biostat 203B Homework 5 / XGBoost

Due Mar 20 @ 11:59PM

AUTHOR
Ningke Zhang 705834790

1. Load libraries

```
library(tidymodels)
```

```
── Attaching packages ─────────────────────────────────────── tidymodels 1.3.0 ──

✔ broom        1.0.7     ✔ recipes      1.1.1
✔ dials        1.4.0     ✔ rsample      1.2.1
✔ dplyr        1.1.4     ✔ tibble       3.2.1
✔ ggplot2      3.5.1     ✔ tidyr        1.3.1
✔ infer        1.0.7     ✔ tune         1.3.0
✔ modeldata    1.4.0     ✔ workflows    1.2.0
✔ parsnip      1.3.1     ✔ workflowsets 1.1.0
✔ purrr        1.0.4     ✔ yardstick    1.3.2

── Conflicts ────────────────────────────────────────── tidymodels_conflicts() ──
✖ purrr::discard() masks scales::discard()
✖ dplyr::filter()  masks stats::filter()
✖ dplyr::lag()     masks stats::lag()
✖ recipes::step()  masks stats::step()
```

```
library(dplyr)
library(recipes)
library(workflows)
library(tune)
library(glmnet)
```

```
Loading required package: Matrix


Attaching package: 'Matrix'

The following objects are masked from 'package:tidyr':

    expand, pack, unpack

Loaded glmnet 4.1-8
```

```
library(vip)
```

```
Attaching package: 'vip'

The following object is masked from 'package:utils':

    vi
```

```
library(ranger)
library(future)
library(xgboost)
```

```
Attaching package: 'xgboost'

The following object is masked from 'package:dplyr':

    slice
```

2.Data preprocessing and feature engineering.

```
# read data
mimiciv_icu_cohort <- readRDS("../hw4/mimiciv_shiny/mimic_icu_cohort.rds") |>
  select(-c(intime,
            outtime,
            admittime,
            dischtime,
            deathtime,
            admit_provider_id,
            edregtime,
            edouttime,
            anchor_age,
            anchor_year,
            anchor_year_group,
            last_careunit,
            discharge_location,
            hospital_expire_flag,
            dod,
            los)
         ) |>
  mutate(los_long = as.factor(los_long)) |>
  print(width = Inf)
```

3.Data split

```
set.seed(203)

mimiciv_icu_cohort <- mimiciv_icu_cohort |>
  arrange(subject_id, hadm_id, stay_id) |>
  select(-c(subject_id, hadm_id, stay_id))
mimiciv_icu_cohort <- mimiciv_icu_cohort |> drop_na()
```

```
data_split <- initial_split(mimiciv_icu_cohort,
                            strata = "los_long",
                            prop = 0.5)

icu_other <- training(data_split)
icu_test <- testing(data_split)
```

4.Train logistic regression with elasticnet regularization.

```
# Define the recipe
gb_recipe <-
  recipe(los_long ~ ., data = icu_other) |>
  step_impute_median(all_numeric_predictors()) |>
  step_impute_mode(all_nominal_predictors()) |>
  step_unknown(all_nominal_predictors()) |>
  step_dummy(all_nominal_predictors()) |>
  step_nzv(all_predictors()) |>
  step_normalize(all_numeric_predictors(), -all_outcomes())

# Define the model
gb_mod <-
  boost_tree(
    mode = "classification",
    trees = 600,
    tree_depth = tune(),
    learn_rate = tune()
  ) |>
  set_engine("xgboost")
gb_mod

# Define the workflow
gb_wf <- workflow() |>
  add_recipe(gb_recipe) |>
  add_model(gb_mod)
gb_wf

# Define the grid
param_grid <- grid_regular(
  tree_depth(range = c(3L, 8L)),
  learn_rate(range = c(-3, -0.5), trans = log10_trans()),
  levels = c(5, 5)
)
```

5.Cross-validation

```
set.seed(203)

folds <- vfold_cv(icu_other, v = 5, strata = los_long)
```

```
# fit cross-validation
gb_fit <- gb_wf |>
  tune_grid(
    resamples = folds,
    grid = param_grid,
    metrics = metric_set(roc_auc, accuracy),
    control = control_grid(verbose = TRUE, save_pred = TRUE)
  )
gb_fit

#visualize CV results
gb_fit |>
  collect_metrics() |>
  filter(.metric == "roc_auc") |>
  ggplot(aes(x = learn_rate, y = mean, color = factor(tree_depth),
             group = factor(tree_depth))) +
  geom_point(size = 3, alpha = 0.7) +
  geom_line(linewidth = 1) +
  labs(
    title = "Gradient Boosting: Learning Rate vs AUC",
    x = "Learning Rate",
    y = "Cross-Validation AUC",
    color = "Tree Depth"
  ) +
  scale_x_log10() +
  theme_minimal()
```

6. Model evaluation

```
# select the best model
best_gb <- gb_fit |> select_best(metric = "roc_auc")
print(best_gb)

# finalize the workflow/fit
final_gb_wf <- finalize_workflow(gb_wf, best_gb)

final_gb_fit <- final_gb_wf |> last_fit(data_split)

saveRDS(final_gb_fit, "final_fit_gb_lastfit.rds")

final_gb_model <- final_gb_fit |> extract_workflow() |> extract_fit_parsnip()

final_gb_model |> vip()

saveRDS(final_gb_model, "final_fit_gb.rds")
```