HeartAttack Prediction model

koki itagaki

2023-03-31

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr     1.1.1     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

```
library(tidymodels)
```

```
## ── Attaching packages ─────────────────────────────────────── tidymodels
1.0.0 ──
## ✓ broom        1.0.4     ✓ rsample      1.1.1
## ✓ dials        1.2.0     ✓ tune         1.1.0
## ✓ infer        1.0.4     ✓ workflows    1.1.3
## ✓ modeldata    1.1.0     ✓ workflowsets 1.0.1
## ✓ parsnip      1.0.4     ✓ yardstick    1.1.0
## ✓ recipes      1.0.5
## ── Conflicts ────────────────────────────────────────
tidymodels_conflicts() ──
## ✖ scales::discard() masks purrr::discard()
## ✖ dplyr::filter()   masks stats::filter()
## ✖ recipes::fixed()  masks stringr::fixed()
## ✖ dplyr::lag()      masks stats::lag()
## ✖ yardstick::spec() masks readr::spec()
## ✖ recipes::step()   masks stats::step()
## • Use suppressPackageStartupMessages() to eliminate package startup
messages
```

```
library(kknn)
library(ranger)
```

```r
# Import the dataset
heart_data <-
read.csv("/Users/itagakikouki/stat123/project/heart_failure.csv")

# Preprocessing and feature engineering
heart_data <- rename(heart_data,
                     age = Age,
                     sex = Sex,
                     chestpain_type = ChestPainType,
                     resting_bp = RestingBP,
                     cholesterol = Cholesterol,
                     fasting_bs = FastingBS,
                     resting_ecg = RestingECG,
                     max_hr = MaxHR,
                     excercise_angina = ExerciseAngina,
                     old_peak = Oldpeak,
                     st_slope = ST_Slope,
                     heart_disease = HeartDisease) %>%
  mutate(heart_disease = as_factor(heart_disease)) %>%
  mutate(fasting_bs = as_factor(fasting_bs)) %>%
  mutate(sex = as_factor(sex))

# Splitting the data
set.seed(999)
heart_split <- initial_split(heart_data, prop = 0.75, strata = heart_disease)
heart_train <- training(heart_split)
heart_test <- testing(heart_split)

# Recipe with all features
heart_recipe <- recipe(heart_disease ~ ., data = heart_train) %>%
  step_scale(all_numeric_predictors()) %>%
  step_center(all_numeric_predictors())

# Model selection and tuning
models <- list(
  logistic_regression = logistic_reg() %>% set_engine("glm"),
  decision_tree = decision_tree() %>% set_engine("rpart") %>%
set_mode("classification"),
  random_forest = rand_forest() %>% set_engine("ranger", importance =
"impurity") %>% set_mode("classification"),
  knn = nearest_neighbor(weight_func = "rectangular", neighbors = tune()) %>%
set_engine("kknn") %>% set_mode("classification")
)

set.seed(999)
cv_results <- map_dfr(models, function(model) {
  workflow() %>%
    add_recipe(heart_recipe) %>%
    add_model(model) %>%
```

```
    tune_grid(resamples = vfold_cv(heart_train, v = 5, strata =
heart_disease),
              grid = if (inherits(model, "nearest_neighbor"))
tibble(neighbors = seq(from = 1, to = 40)) else 1) %>%
    collect_metrics() %>%
    filter(.metric == "accuracy") %>%
    arrange(desc(mean)) %>%
    slice(1)
}, .id = "model_name")
```

```
## Warning: No tuning parameters have been detected, performance will be
evaluated using the resamples with no tuning. Did you want to [tune()]
parameters?
## No tuning parameters have been detected, performance will be evaluated
using the resamples with no tuning. Did you want to [tune()] parameters?
## No tuning parameters have been detected, performance will be evaluated
using the resamples with no tuning. Did you want to [tune()] parameters?
```

```
best_model_name <- cv_results$model_name[which.max(cv_results$mean)]

# Train the best model
best_model <- models[[best_model_name]]

if (inherits(best_model, "nearest_neighbor")) {
  best_k <- cv_results %>% filter(model_name == best_model_name) %>%
pull(neighbors)
  best_model <- best_model %>% set_args(neighbors = best_k)
}

best_model_fit <- workflow() %>%
  add_recipe(heart_recipe) %>%
  add_model(best_model) %>%
  fit(data = heart_train)

# Model evaluation
heart_predictions <- predict(best_model_fit, heart_test) %>%
  bind_cols(heart_test)

heart_metrics <- heart_predictions %>%
  metrics(truth = heart_disease, estimate = .pred_class)

heart_conf_mat <- heart_predictions %>%
  conf_mat(truth = heart_disease, estimate = .pred_class)

head(heart_predictions)
```

```
## # A tibble: 6 × 13
##   .pred_class   age sex   chest…¹ resti…² chole…³ fasti…⁴ resti…⁵ max_hr
excer…⁶
##   <fct>       <int> <fct> <chr>     <int>   <int> <fct>   <chr>    <int>
```

```
<chr>
## 1 0              40 M     ATA         140     289 0       Normal     172 N
## 2 0              49 F     NAP         160     180 0       Normal     156 N
## 3 0              54 M     NAP         150     195 0       Normal     122 N
## 4 1              37 M     ASY         140     207 0       Normal     130 Y
## 5 0              43 F     ATA         120     201 0       Normal     165 N
## 6 0              36 M     ATA         120     267 0       Normal     160 N
## # … with 3 more variables: old_peak <dbl>, st_slope <chr>, heart_disease
<fct>,
## #   and abbreviated variable names ¹chestpain_type, ²resting_bp, ³
cholesterol,
## #   ⁴fasting_bs, ⁵resting_ecg, ⁶excercise_angina

head(heart_conf_mat)

## $table
##           Truth
## Prediction   0   1
##          0  81   9
##          1  22 118

bind_rows(heart_metrics, precision(heart_predictions, truth= heart_disease,
estimate= .pred_class), recall(heart_predictions, truth= heart_disease,
estimate= .pred_class), f_meas(heart_predictions, truth= heart_disease,
estimate= .pred_class), sensitivity(heart_predictions, truth= heart_disease,
estimate= .pred_class))

## # A tibble: 6 × 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 accuracy    binary         0.865
## 2 kap         binary         0.724
## 3 precision   binary         0.9
## 4 recall      binary         0.786
## 5 f_meas      binary         0.839
## 6 sensitivity binary         0.786
```

#This R script aims to build and evaluate machine learning models for predicting heart disease using patient information. The script follows a detailed process, which can be summarized as follows:

Required libraries are loaded, including tidyverse, tidymodels, kknn, and ranger. The dataset is imported from a CSV file called “heart_failure.csv” and preprocessed by renaming columns for clarity and converting categorical variables using the mutate function.

The dataset is then split into training and testing sets using the initial_split function with 75% for training and 25% for testing, stratified by the heart_disease variable.

A preprocessing recipe is defined using the recipe function, which scales and centers all numeric predictor variables.

Four models are defined for comparison, including logistic regression, decision tree, random forest, and k-NN. These models are then fit and evaluated using the map_dfr function, which applies a function to each model, and the collect_metrics function, which collects performance metrics.

The best model is selected based on the highest average accuracy using the tune_grid function, which performs a grid search to tune hyperparameters, and the cv_results object, which stores the cross-validation results for each model. The best model is trained using the entire training set using the best_model_fit function.

The model is evaluated using the predict function to generate predictions for the test set, and metrics such as precision, recall, F-measure, and sensitivity are calculated using the metrics function and the confusion matrix is generated using the conf_mat function.

Finally, the first few rows of the predictions, confusion matrix, and various metrics are displayed using the head and bind_rows functions.