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)  
  
# 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 ¹​chestpain\_type, ²​resting\_bp, ³​cholesterol,  
## # ⁴​fasting\_bs, ⁵​resting\_ecg, ⁶​excercise\_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.