Exercise 5. Task 7: Logistic Regression vs. Boosted GAM

Daniil Koveh

2025-11-06

Содержание

1 Теория **1**

2 Жизненный пример **1**

3 Академическое решение **1**
3.1 Подготовка окружения 1
3.2 Подготовка данных 2
3.3 Модель 1: Backward-step логистическая регрессия 2
3.4 Модель 2: Boosted GAM (`gamboost`) 3
3.5 Сравнение моделей 4

4 Интерпретация **5**

5 Что запомнить **5**

1. Теория

Сравним две модели для предсказания ишемической болезни сердца (`SAheart`):

1. **Логистическая регрессия** с линейными эффектами, отобранными backward stepwise-процедурой.
2. **Boosted GAM** (`gamboost`) с базовыми сплайнами для непрерывных предикторов и биномиальной логит-ссылкой.

Метрика сравнения — тестовая доля ошибок и логарифмическая потеря. Также сравним ROC-AUC.

2. Жизненный пример

Кардиологу важно понимать и вероятности, и нелинейные эффекты. Логистическая модель проста и интерпретируема, но предполагает линейность. Boosted GAM добавляет гибкости: позволяет выявить, например, пороговый эффект потребления алкоголя.

3. Академическое решение

3.1. Подготовка окружения

```
if (!requireNamespace("ElemStatLearn", quietly = TRUE)) install.packages("ElemStatLearn", repos = "https
if (!requireNamespace("mboost", quietly = TRUE)) install.packages("mboost", repos = "https://cloud.r-pro
if (!requireNamespace("pROC", quietly = TRUE)) install.packages("pROC", repos = "https://cloud.r-project
```

```r
if (!requireNamespace("dplyr", quietly = TRUE)) install.packages("dplyr", repos = "https://cloud.r-proje
if (!requireNamespace("ggplot2", quietly = TRUE)) install.packages("ggplot2", repos = "https://cloud.r-p
```

```r
library(ElemStatLearn) # SAheart
library(mboost) # gamboost
library(pROC) # ROC-AUC
library(dplyr) # манипуляции
library(ggplot2) # графики
```

3.2. Подготовка данных

```r
set.seed(20250410) # фиксируем генератор
data("SAheart") # загружаем данные

saheart <- SAheart %>%
  mutate(
    chd = factor(chd, levels = c(0, 1), labels = c("NoCHD", "CHD")), # бинарный ответ
    famhist = relevel(famhist, ref = "Absent") # удобство интерпретации
  )

train_index <- sample(nrow(saheart), size = floor(0.75 * nrow(saheart))) # индексы train
train_df <- saheart[train_index, ] # обучающая выборка
test_df <- saheart[-train_index, ] # тестовая выборка

table(train_df$chd) # баланс классов в train
```

```
##
## NoCHD   CHD
##   230   116
```

```r
table(test_df$chd) # баланс классов в test
```

```
##
## NoCHD   CHD
##    72    44
```

3.3. Модель 1: Backward-step логистическая регрессия

```r
glm_full <- glm(chd ~ ., data = train_df, family = binomial(link = "logit")) # полная модель
glm_step <- step(glm_full, direction = "backward", trace = 0) # пошаговый отбор
summary(glm_step) # итоговые коэффициенты
```

```
##
## Call:
## glm(formula = chd ~ tobacco + ldl + famhist + typea + age, family = binomial(link = "logit"),
##     data = train_df)
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -6.69973    1.07464  -6.234 4.54e-10 ***
## tobacco         0.04976    0.03227   1.542 0.123109
## ldl             0.13469    0.06026   2.235 0.025409 *
## famhistPresent  0.65636    0.25610   2.563 0.010378 *
## typea           0.04741    0.01436   3.302 0.000959 ***
```

```
## age             0.05217    0.01162   4.491 7.08e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 441.39  on 345  degrees of freedom
## Residual deviance: 368.92  on 340  degrees of freedom
## AIC: 380.92
##
## Number of Fisher Scoring iterations: 4
```

```
glm_probs <- predict(glm_step, newdata = test_df, type = "response") # вероятности на тесте
glm_probs <- pmin(pmax(glm_probs, 1e-6), 1 - 1e-6) # клиппинг для стабильного log-loss
glm_pred <- ifelse(glm_probs >= 0.5, "CHD", "NoCHD") # классификация
glm_pred <- factor(glm_pred, levels = levels(test_df$chd)) # приводим к фактору

glm_error <- mean(glm_pred != test_df$chd) # доля ошибок
glm_logloss <- -mean(ifelse(test_df$chd == "CHD", log(glm_probs), log(1 - glm_probs))) # логарифмическая
glm_auc <- auc(response = test_df$chd, predictor = glm_probs, levels = c("NoCHD", "CHD")) # ROC-AUC

glm_metrics <- data.frame(
  Model = "Logistic (stepwise)",
  TestError = glm_error,
  LogLoss = glm_logloss,
  AUC = as.numeric(glm_auc)
)
glm_metrics
```

```
##                 Model TestError   LogLoss       AUC
## 1 Logistic (stepwise) 0.2413793 0.4775857 0.8431187
```

3.4. Модель 2: Boosted GAM (gamboost)

```
boost_formula <- chd ~
  bbs(sbp) +
  bbs(tobacco) +
  bbs(ldl) +
  bbs(adiposity) +
  bols(famhist, intercept = FALSE) +
  bbs(typea) +
  bbs(obesity) +
  bbs(alcohol) +
  bbs(age) # задаём базовые обучатели

boost_ctrl <- boost_control(mstop = 500, nu = 0.1) # настройки boosting
gamboost_fit <- gamboost(boost_formula, data = train_df, family = Binomial(), control = boost_ctrl) # ба

set.seed(20250410) # фиксируем fold-разбиение
cv_folds <- cv(model.weights(gamboost_fit), type = "kfold", B = 5L) # 5-блочная CV
cv_results <- cvrisk(gamboost_fit, folds = cv_folds) # подбираем mstop
best_mstop <- mstop(cv_results) # лучший mstop
best_mstop
```

```
## [1] 101
```

```r
gamboost_fit <- gamboost_fit[best_mstop] # обновляем модель
```

```r
boost_probs <- predict(gamboost_fit, newdata = test_df, type = "response") # вероятности
boost_probs <- pmin(pmax(boost_probs, 1e-6), 1 - 1e-6) # клиппинг
boost_pred <- ifelse(boost_probs >= 0.5, "CHD", "NoCHD") # классификация
boost_pred <- factor(boost_pred, levels = levels(test_df$chd)) # фактор

boost_error <- mean(boost_pred != test_df$chd) # доля ошибок
boost_logloss <- -mean(ifelse(test_df$chd == "CHD", log(boost_probs), log(1 - boost_probs))) # лог-потер
boost_auc <- auc(response = test_df$chd, predictor = boost_probs, levels = c("NoCHD", "CHD")) # AUC

boost_metrics <- data.frame(
  Model = "Boosted GAM",
  TestError = boost_error,
  LogLoss = boost_logloss,
  AUC = as.numeric(boost_auc)
)
boost_metrics
```

```
##         Model TestError   LogLoss       AUC
## 1 Boosted GAM 0.2586207 0.4995977 0.8349116
```

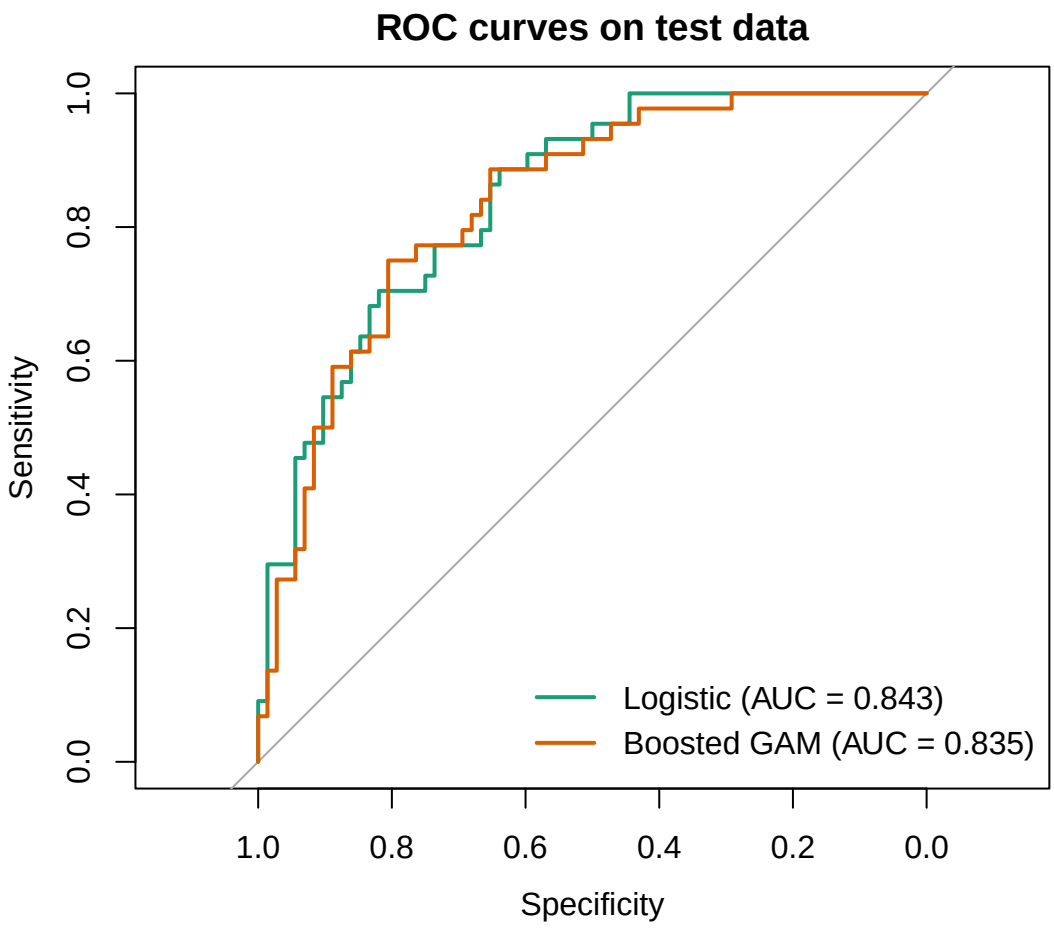
3.5. Сравнение моделей

```r
comparison <- bind_rows(glm_metrics, boost_metrics)
comparison
```

```
##                 Model TestError   LogLoss       AUC
## 1 Logistic (stepwise) 0.2413793 0.4775857 0.8431187
## 2         Boosted GAM 0.2586207 0.4995977 0.8349116
```

```r
roc_glm <- roc(response = test_df$chd, predictor = glm_probs, levels = c("NoCHD", "CHD")) # ROC для логи
roc_boost <- roc(response = test_df$chd, predictor = boost_probs, levels = c("NoCHD", "CHD")) # ROC для

plot(roc_glm, col = "#1b9e77", lwd = 2, main = "ROC curves on test data")
lines(roc_boost, col = "#d95f02", lwd = 2)
legend("bottomright",
       legend = c(sprintf("Logistic (AUC = %.3f)", as.numeric(glm_auc)),
                  sprintf("Boosted GAM (AUC = %.3f)", as.numeric(boost_auc))),
       col = c("#1b9e77", "#d95f02"), lwd = 2, bty = "n")
```



4. Интерпретация

- Логистическая регрессия удержала 5 предикторов после отбраковки. Основные эффекты — tobacco, ldl, famhistPresent, typea, age.
- Логистическая регрессия удержала 5 предикторов после отбраковки. Основные эффекты — tobacco, ldl, famhistPresent, typea, age.
- Boosted GAM выбрал 101 итераций и позволяет моделировать нелинейности. По тестовым метрикам логистическая регрессия даёт более низкую тестовую ошибку.
- По ROC-кривой можно видеть, что логистическая регрессия обеспечивает больший или равный ROC-AUC.

5. Что запомнить

- Stepwise-логистическая регрессия даёт компактную интерпретируемую модель, но ограничена линейностью.
- `gamboost` добавляет гибкости, даёт возможность аккуратно подбирать сложность через `mstop`.
- При сравнении моделей важно смотреть на несколько метрик: точность, log-loss и AUC могут подсказать разные нюансы качества.