svm_mejor

Laura Belmonte

2025-12-11

Probamos con el svm radial, sigmoidal y polinomial:

```r
library(recipes)
library(e1071)
library(ggplot2)
library(caret)
library(pROC)
library(dplyr)

load("~/Documents/GitHub/Mineria/DATA/dataaaaaaaaaaaaaa.RData")
bd <- data_reducida

set.seed(123)

rec <- recipe(Exited ~ ., data = bd) %>%
  step_dummy(all_nominal_predictors(), -group, one_hot = TRUE)

bd <- prep(rec) %>% bake(new_data = NULL)
```

```r
# División train/test

trainbase <- bd[bd$group == "train", ]
trainbase$group <- NULL

testbase <- bd[bd$group == "test", ]
testbase$group <- NULL

ind <- sample(1:nrow(trainbase), 0.7*nrow(trainbase))
train <- trainbase[ind, ]
test  <- trainbase[-ind, ]
```

```r
optimizar_kernel <- function(kernel_name, train_data, test_data,
                             thresholds = seq(0.05, 0.50, 0.01),
                             class_weights = c("0" = 1, "1" = 2)) {

    # -----------------------
    #  Grid por kernel
    # -----------------------
    if (kernel_name == "radial") {
        grid <- expand.grid(
            cost = c(1, 5, 10, 20),
            gamma = c(0.001, 0.01, 0.05, 0.1)
        )
    }

    if (kernel_name == "polynomial") {
        grid <- expand.grid(
            cost = c(1, 5, 10),
            gamma = c(0.001, 0.01, 0.1),
            degree = c(2, 3, 4),
            coef0 = c(0, 1)
        )
    }

    if (kernel_name == "sigmoid") {
        grid <- expand.grid(
            cost = c(1, 5, 10),
            gamma = c(0.001, 0.01, 0.1),
            coef0 = c(-1, 0, 1)
        )
    }

    mejor <- NULL
    mejor_f1 <- -Inf

    for (i in 1:nrow(grid)) {

        params <- grid[i, ]

        # Parámetros dinámicos
        args <- list(
            formula = Exited ~ .,
            data = train_data,
            kernel = kernel_name,
            cost = params$cost,
            gamma = params$gamma,
            class.weights = class_weights,
            probability = TRUE
        )

        if ("degree" %in% names(params)) args$degree <- params$degree
        if ("coef0"  %in% names(params)) args$coef0  <- params$coef0

        modelo <- do.call(svm, args)

        # ---- Probabilidades TRAIN y TEST ----
        prob_train <- attr(predict(modelo, train_data, probability = TRUE), "probabilities")[, "1"]
        prob_test  <- attr(predict(modelo, test_data,  probability = TRUE), "probabilities")[, "1"]

        for (t in thresholds) {

            # Train
            pred_train <- ifelse(prob_train >= t, "1", "0")
            cm_train <- confusionMatrix(
                factor(pred_train),
                factor(train_data$Exited),
                positive = "1"
            )
            f1_train <- cm_train$byClass["F1"]

            # Test
            pred_test <- ifelse(prob_test >= t, "1", "0")
            cm_test <- confusionMatrix(
                factor(pred_test),
                factor(test_data$Exited),
                positive = "1"
            )
            f1_test <- cm_test$byClass["F1"]

            # --- AUC ---
            auc_val <- as.numeric(auc(test_data$Exited, prob_test, quiet = TRUE))

            # -------------------------
            # Guardar mejor según F1_test
            # -------------------------
            if (!is.na(f1_test) && f1_test > mejor_f1) {

                mejor_f1 <- f1_test

                mejor <- data.frame(
                    Kernel = kernel_name,
                    Cost = params$cost,
                    Gamma = params$gamma,
                    Degree = ifelse("degree" %in% names(params), params$degree, NA),
                    Coef0 = ifelse("coef0" %in% names(params), params$coef0, NA),
                    Threshold = t,
                    F1_Train = f1_train,
                    F1_Test = f1_test,
                    Recall_Test = cm_test$byClass["Recall"],
                    Precision_Test = cm_test$byClass["Precision"],
                    Specificity_Test = cm_test$byClass["Specificity"],
                    Accuracy_Test = cm_test$overall["Accuracy"],
                    AUC = auc_val,
                    stringsAsFactors = FALSE
                )
            }
        }
    }

    return(mejor)
}

#Optimización
kernels <- c("radial", "polynomial", "sigmoid")

resultados <- lapply(kernels, function(k) {
    optimizar_kernel(k, train, test)
```

```r
})

final <- bind_rows(resultados) %>%
  arrange(desc(F1_Test))

print(final)
```

```
##            Kernel Cost Gamma Degree Coef0 Threshold  F1_Train   F1_Test
## F1...1 polynomial    1  0.10      4     0      0.17 0.5110861 0.4611006
## F1...2     radial    5  0.05     NA    NA      0.14 0.4952648 0.4569939
## F1...3    sigmoid   10  0.01     NA    -1      0.18 0.4962687 0.4514339
##        Recall_Test Precision_Test Specificity_Test Accuracy_Test       AUC
## F1...1   0.5758294      0.3844937        0.7681764     0.7295238 0.6998197
## F1...2   0.6232227      0.3607682        0.7222884     0.7023810 0.6898481
## F1...3   0.5781991      0.3702580        0.7526818     0.7176190 0.7008845
```