svm_mejor

Laura Belmonte

2025-12-11

Probamos con el svm radial, sigmoidal y polinomial:

```
library(recipes)
```

```
## Loading required package: dplyr
```

```
## 
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
## 
##     filter, lag
```

```
## The following objects are masked from 'package:base':
## 
##     intersect, setdiff, setequal, union
```

```
## Registered S3 method overwritten by 'future':
##   method               from
##   all.equal.connection parallelly
```

```
## 
## Attaching package: 'recipes'
```

```
## The following object is masked from 'package:stats':
## 
##     step
```

```
library(e1071)
library(ggplot2)
library(caret)
```

```
## Loading required package: lattice
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
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, threshold = 0.20) {

    # -----------------------
    # Grid de hiperparámetros
    # -----------------------
    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, 20),
            gamma = c(0.001, 0.01, 0.1),
            coef0 = c(-1, 0, 1)
        )
    }

    mejor_f1 <- -Inf
    mejor_res <- NULL

    # ===============================
    #  LOOP de búsqueda en el grid
    # ===============================
    for (i in 1:nrow(grid)) {

        params <- grid[i, ]

        # Construir argumentos válidos para svm()
        args <- list(
            formula = Exited ~ .,
            data = train_data,
            kernel = kernel_name,
            cost = params$cost,
            gamma = params$gamma,
            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)

        # Predicción
        pred <- predict(modelo, test_data, probability = TRUE)
        prob <- attr(pred, "probabilities")[, "1"]

        pred_class <- ifelse(prob >= threshold, "1", "0")

        cm <- confusionMatrix(
            factor(pred_class),
            factor(test_data$Exited),
            positive = "1"
        )

        # --- FIX: convertir AUC a numeric ---
        roc_obj <- roc(test_data$Exited, prob, quiet = TRUE)
        auc_val <- as.numeric(auc(roc_obj))

        f1_val <- cm$byClass["F1"]

        # Guardar el mejor
        if (!is.na(f1_val) && f1_val > mejor_f1) {
            mejor_f1 <- f1_val
            mejor_res <- 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),
                F1_Score = f1_val,
                Recall = cm$byClass["Recall"],
                Precision = cm$byClass["Precision"],
                Accuracy = cm$overall["Accuracy"],
                Specificity = cm$byClass["Specificity"],
                AUC = auc_val,
                stringsAsFactors = FALSE
            )
        }
    }

    return(mejor_res)
}
```

```
kernels <- c("radial", "polynomial", "sigmoid")

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

# Combinar resultados (YA NO FALLA)
resultados_finales <- bind_rows(resultados) %>%
    mutate(across(where(is.numeric), round, 4)) %>%
    arrange(desc(F1_Score))
```

```
## Warning: There was 1 warning in ‘mutate()‘.
## i In argument: ‘across(where(is.numeric), round, 4)‘.
## Caused by warning:
## ! The ‘...‘ argument of ‘across()‘ is deprecated as of dplyr 1.1.0.
## Supply arguments directly to ‘.fns‘ through an anonymous function instead.
##
##   # Previously
##   across(a:b, mean, na.rm = TRUE)
##
##   # Now
##   across(a:b, \(x) mean(x, na.rm = TRUE))
```

```
# Mostrar
print(resultados_finales)
```

```
##            Kernel Cost Gamma Degree Coef0 F1_Score Recall Precision Accuracy
## F1...1 polynomial   10  0.10      4     0   0.4400 0.4645    0.4179   0.7624
## F1...2    sigmoid    1  0.01     NA     1   0.3935 0.5735    0.2995   0.6448
## F1...3     radial   20  0.01     NA    NA   0.3768 0.3081    0.4851   0.7952
##        Specificity    AUC
## F1...1      0.8373 0.6661
## F1...2      0.6627 0.6560
## F1...3      0.9178 0.6729
```