Exercise 11 Advanced Methods for Regression and Classification

Stefan Merdian

2025-01-12

```
df <- read.csv2("bank.csv")
str(df)
```

```
## 'data.frame':    4521 obs. of  17 variables:
##  $ age      : int  30 33 35 30 59 35 36 39 41 43 ...
##  $ job      : chr  "unemployed" "services" "management" "management" ...
##  $ marital  : chr  "married" "married" "single" "married" ...
##  $ education: chr  "primary" "secondary" "tertiary" "tertiary" ...
##  $ default  : chr  "no" "no" "no" "no" ...
##  $ balance  : int  1787 4789 1350 1476 0 747 307 147 221 -88 ...
##  $ housing  : chr  "no" "yes" "yes" "yes" ...
##  $ loan     : chr  "no" "yes" "no" "yes" ...
##  $ contact  : chr  "cellular" "cellular" "cellular" "unknown" ...
##  $ day      : int  19 11 16 3 5 23 14 6 14 17 ...
##  $ month    : chr  "oct" "may" "apr" "jun" ...
##  $ duration : int  79 220 185 199 226 141 341 151 57 313 ...
##  $ campaign : int  1 1 1 4 1 2 1 2 2 1 ...
##  $ pdays    : int  -1 339 330 -1 -1 176 330 -1 -1 147 ...
##  $ previous : int  0 4 1 0 0 3 2 0 0 2 ...
##  $ poutcome : chr  "unknown" "failure" "failure" "unknown" ...
##  $ y        : chr  "no" "no" "no" "no" ...
```

Preprocessing

```
set.seed(123)

categorical_columns <- c("job", "marital", "education", "default", "housing",
                         "loan", "contact", "month", "poutcome", "y")
df[categorical_columns] <- lapply(df[categorical_columns], as.factor)
df <- na.omit(df)

numeric_columns <- c("age", "balance", "duration", "campaign", "pdays", "previous")
df[numeric_columns] <- scale(df[numeric_columns])

str(df)
```

```
## 'data.frame':    4521 obs. of  17 variables:
##  $ age      : num  -1.056 -0.772 -0.583 -1.056 1.686 ...
```

```
##  $ job      : Factor w/ 12 levels "admin.","blue-collar",..: 11 8 5 5 2 5 7 10 3 8 ...
##  $ marital  : Factor w/ 3 levels "divorced","married",..: 2 2 3 2 2 3 2 2 2 2 ...
##  $ education: Factor w/ 4 levels "primary","secondary",..: 1 2 3 3 2 3 3 2 3 1 ...
##  $ default  : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ balance  : num  0.1211 1.1185 -0.0241 0.0177 -0.4727 ...
##  $ housing  : Factor w/ 2 levels "no","yes": 1 2 2 2 2 1 2 2 2 2 ...
##  $ loan     : Factor w/ 2 levels "no","yes": 1 2 1 2 1 1 1 1 1 2 ...
##  $ contact  : Factor w/ 3 levels "cellular","telephone",..: 1 1 1 3 3 1 1 1 3 1 ...
##  $ day      : int  19 11 16 3 5 23 14 6 14 17 ...
##  $ month    : Factor w/ 12 levels "apr","aug","dec",..: 11 9 1 7 9 4 9 9 9 1 ...
##  $ duration : num  -0.712 -0.169 -0.304 -0.25 -0.146 ...
##  $ campaign : num  -0.577 -0.577 -0.577 0.388 -0.577 ...
##  $ pdays    : num  -0.407 2.989 2.899 -0.407 -0.407 ...
##  $ previous : num  -0.32 2.04 0.27 -0.32 -0.32 ...
##  $ poutcome : Factor w/ 4 levels "failure","other",..: 4 1 1 4 4 1 2 4 4 1 ...
##  $ y        : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
```

Data splitting

```
n <- nrow(df)
train_indices <- sample(1:n, size = floor(2 * n / 3))
train_data <- df[train_indices, ]
test_data <- df[-train_indices, ]
```

a) Apply svm()

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.4.2
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.4.2
```

```
## Loading required package: ggplot2
```

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

```
svm_model <- svm(y ~ ., data = train_data, kernel = "radial")
predictions <- predict(svm_model, newdata = test_data)  # Predict on test_data
conf_matrix <- table(Predicted = predictions, Actual = test_data$y)  # Use test_data$y

print("Confusion Matrix:")
```

```
## [1] "Confusion Matrix:"
```

```r
print(conf_matrix)
```

```
##          Actual
## Predicted   no  yes
##       no  1326  151
##       yes   12   18
```

```r
conf_metrics <- confusionMatrix(as.factor(predictions), as.factor(test_data$y))
balanced_accuracy <- mean(conf_metrics$byClass[c("Sensitivity", "Specificity")])

print(paste("Balanced Accuracy:", round(balanced_accuracy, 4)))
```

```
## [1] "Balanced Accuracy: 0.5488"
```

b) Parameter tuning

```r
gamma_values <- c(0.01, 0.1, 1, 10)
cost_values <- c(1, 10, 100, 1000)

set.seed(123)
tuning_result <- tune.svm(y ~ ., data = train_data,
                          kernel = "radial",
                          gamma = gamma_values,
                          cost = cost_values)

print(tuning_result)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma cost
##   0.01  100
##
## - best performance: 0.1001969
```

```r
print("Best Parameters:")
```

```
## [1] "Best Parameters:"
```

```r
print(tuning_result$best.parameters)
```

```
##   gamma cost
## 9  0.01  100
```

The best Paramters for our setup are:

- gamma: 0.01
- const: 100

c) Use best model

```
svm_bestPara <- svm(y~.,data = train_data,
                          kernel = "radial",
                          gamma = as.numeric(tuning_result$best.parameters[1]),
                          cost = as.numeric(tuning_result$best.parameters[2]))

predictions <- predict(svm_bestPara, newdata = test_data)
conf_matrix <- table(Predicted = predictions, Actual = test_data$y)

print("Confusion Matrix:")
```

```
## [1] "Confusion Matrix:"
```

```
print(conf_matrix)
```

```
##          Actual
## Predicted   no  yes
##       no  1299  116
##       yes   39   53
```

```
conf_metrics <- confusionMatrix(as.factor(predictions), as.factor(test_data$y))
balanced_accuracy <- mean(conf_metrics$byClass[c("Sensitivity", "Specificity")])

print(paste("Balanced Accuracy:", round(balanced_accuracy, 4)))
```

```
## [1] "Balanced Accuracy: 0.6422"
```

We used now the best parameters, calculated in advanced and indeed it improved. From: 0.5488 To : 0.6422

d) Improve the misclassification error

```
class_weights <- list(no = 1, yes = table(train_data$y)["no"] / table(train_data$y)["yes"])

custom_error_fun <- function(true, predicted) {
  cm <- caret::confusionMatrix(as.factor(predicted), as.factor(true))
  1 - mean(cm$byClass[c("Sensitivity", "Specificity")])
}

tuning_result <- tune(
  svm,
  y ~ .,
  data = train_data,
  kernel = "radial",
  ranges = list(gamma = c(0.01, 0.1, 1), cost = c(1, 10, 100)),
  tunecontrol = tune.control(error.fun = custom_error_fun),
  class.weights = class_weights
)
print(tuning_result$best.parameters)
```

```
##   gamma cost
## 1  0.01    1
```

```
svm_best_model <- svm(
  y ~ .,
  data = train_data,
  kernel = "radial",
  gamma = as.numeric(tuning_result$best.parameters$gamma),
  cost = as.numeric(tuning_result$best.parameters$cost),
  class.weights = class_weights
)

predictions <- predict(svm_best_model, newdata = test_data)
conf_matrix <- table(Predicted = predictions, Actual = test_data$y)

print("Confusion Matrix:")
```

```
## [1] "Confusion Matrix:"
```

```
print(conf_matrix)
```

```
##          Actual
## Predicted   no  yes
##       no  1125   40
##       yes  213  129
```

```
conf_metrics <- caret::confusionMatrix(as.factor(predictions), as.factor(test_data$y))
balanced_accuracy <- mean(conf_metrics$byClass[c("Sensitivity", "Specificity")])

print(paste("Balanced Accuracy:", round(balanced_accuracy, 4)))
```

```
## [1] "Balanced Accuracy: 0.8021"
```

Did the balanced accuracy improve?

Yes it improved a lot. From: 0.6442 To: 0.8021