SVM Classification on Spambase Dataset

Carla Flore

2025-05-20

```
# ---- Load packages ----
library(caret)
library(doParallel)
```

Below, we load the dataset and ensure that the response variable `Class` is encoded as a binary factor.

```
# ---- Set seed and load data ----
set.seed(4500393)
df <- read.csv("spambase.csv")
df$Class <- as.factor(df$Class)  # Ensure binary factor
```

We split the dataset into 70% training and 30% testing data to evaluate generalization performance.

```
# ---- Train-test split (70/30) ----
idx <- sample(1:nrow(df), size = 0.7 * nrow(df), replace = FALSE)
train_raw <- df[idx, ]
test_raw  <- df[-idx, ]
```

SVMs are sensitive to feature scales. We standardize predictors using the training set's mean and variance.

```
# ---- Scale predictors ----
x_train <- scale(train_raw[, -ncol(train_raw)])
x_test  <- scale(test_raw[, -ncol(test_raw)],
                 center = attr(x_train, "scaled:center"),
                 scale = attr(x_train, "scaled:scale"))
```

We reattach the response variable to the scaled predictors for training and testing.

```
# ---- Combine scaled predictors with target ----
train_svm <- as.data.frame(x_train)
train_svm$Class <- train_raw$Class

test_svm <- as.data.frame(x_test)
test_svm$Class <- test_raw$Class
```

We set up parallel processing to accelerate model tuning.

```
# ---- Setup parallel backend ----
cl <- makePSOCKcluster(parallel::detectCores() - 1)
registerDoParallel(cl)
```

We define the tuning grid for cost and gamma (sigma).

```r
# ---- Define tuning grid ----
tune_grid <- expand.grid(
  C = c(0.1, 1, 10),
  sigma = c(0.01, 0.1, 1)
)
```

The best hyperparameters are selected using 10-fold CV and parallel computation.

```r
# ---- Cross-validation tuning with caret + parallel ----
set.seed(4500393)
svm_tuned <- train(
  Class ~ .,
  data = train_svm,
  method = "svmRadial",
  trControl = trainControl(method = "cv", number = 10),
  tuneGrid = tune_grid,
  preProcess = NULL
)
```

We stop the parallel backend.

```r
# ---- Stop cluster ----
stopCluster(cl)
registerDoSEQ()
```

We apply the trained model to the test set and compute performance metrics.

```r
# ---- Predict and evaluate on test set ----
svm_pred <- predict(svm_tuned, newdata = test_svm)
conf_matrix <- confusionMatrix(svm_pred, test_svm$Class)
```

Finally, we print the best parameters and the confusion matrix.

```r
# ---- Print results ----
print(svm_tuned$bestTune)
```

```
##   sigma  C
## 7  0.01 10
```

```r
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 799  43
##          1  45 494
##
##                Accuracy : 0.9363
```

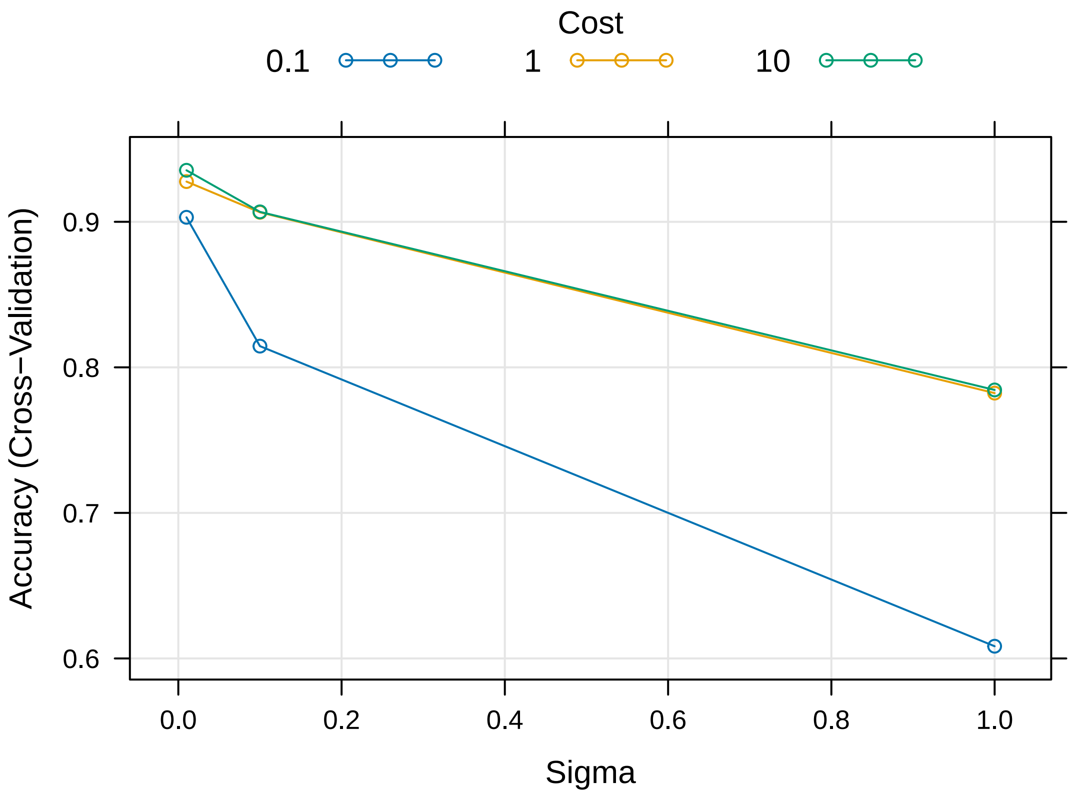
```
##                  95% CI : (0.9221, 0.9486)
##     No Information Rate : 0.6112
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.866
##
##  Mcnemar's Test P-Value : 0.9151
##
##             Sensitivity : 0.9467
##             Specificity : 0.9199
##          Pos Pred Value : 0.9489
##          Neg Pred Value : 0.9165
##              Prevalence : 0.6112
##          Detection Rate : 0.5786
##    Detection Prevalence : 0.6097
##       Balanced Accuracy : 0.9333
##
##        'Positive' Class : 0
##
```

The SVM classifier with a radial basis kernel, tuned using 10-fold cross-validation, achieved its best performance with `C = 10` and `sigma = 0.01`. The model showed excellent accuracy on the test set, achieving 93.6% accuracy. It also exhibited high sensitivity (94.7%) and specificity (91.9%), indicating balanced and robust performance in detecting both spam and non-spam emails. The Kappa value of 0.866 suggests strong agreement beyond chance, and the McNemar's test showed no significant asymmetry in misclassification types.

Table 1: Summary of SVM Model Performance on Test Set

Metric	Value
Accuracy	0.9363
95% CI (lower)	0.9221
95% CI (upper)	0.9486
Kappa	0.8660
Sensitivity	0.9467
Specificity	0.9199
Pos Pred Value	0.9489
Neg Pred Value	0.9165
Balanced Accuracy	0.9333

```r
# ---- Visualize tuning results ----
plot(svm_tuned)
```



Cost
0.1
1
10
Accuracy (Cross−Validation)
0.9
0.8
0.7
0.6
0.0
0.2
0.4
0.6
0.8
1.0
Sigma