Support Vector Machine

Blake Pappas

2023-12-17

Support Vector Machine in R

Load the following packages:

```r
library(caret)
library(dplyr)
library(e1071)
```

In this exercise, we use the "credit.csv" file. This file concerns credit card applications.

All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data.

The outcome variable is A16, attributes are A1 - A15.

P1: Import the dataset. Split it to 80% training and 20% testing.

```r
credit = read.csv("credit.csv", stringsAsFactors = TRUE)

train_rows = createDataPartition(y = credit$A16,
                                 p = 0.80, list = FALSE)

credit_train = credit[train_rows, ]
credit_test = credit[-train_rows, ]
```

Build a SVM model with a linear kernel

```
svm_model_linear = svm(A16 ~ ., data = credit_train,
                kernel = "linear")

# Make Predictions
pred_svm_model_linear = predict(svm_model_linear, credit_test)

# Performance Evaluation
# Confusion Matrix
confusionMatrix(pred_svm_model_linear, credit_test$A16, mode = "prec_recall", positive = "+")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  -  +
##          - 58  5
##          + 13 54
##
##                Accuracy : 0.8615
##                  95% CI : (0.79, 0.9158)
##     No Information Rate : 0.5462
##     P-Value [Acc > NIR] : 1.598e-14
##
##                   Kappa : 0.7239
##
##  Mcnemar’s Test P-Value : 0.09896
##
##               Precision : 0.8060
##                  Recall : 0.9153
##                      F1 : 0.8571
##              Prevalence : 0.4538
##          Detection Rate : 0.4154
##    Detection Prevalence : 0.5154
##       Balanced Accuracy : 0.8661
##
##        ’Positive’ Class : +
##
```

Build a SVM model with a radial kernel

```
svm_model_radial = svm(A16 ~ ., data = credit_train,
                kernel = "radial")

# Make Predictions
pred_svm_model_radial = predict(svm_model_radial, credit_test)

# Performance Evaluation
# Confusion Matrix
confusionMatrix(pred_svm_model_radial, credit_test$A16, mode = "prec_recall", positive = "+")
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction  -  +
##          - 57  4
##          + 14 55
##
##                Accuracy : 0.8615
##                  95% CI : (0.79, 0.9158)
##     No Information Rate : 0.5462
##     P-Value [Acc > NIR] : 1.598e-14
##
##                   Kappa : 0.7246
##
##  Mcnemar's Test P-Value : 0.03389
##
##               Precision : 0.7971
##                  Recall : 0.9322
##                      F1 : 0.8594
##              Prevalence : 0.4538
##          Detection Rate : 0.4231
##    Detection Prevalence : 0.5308
##       Balanced Accuracy : 0.8675
##
##        'Positive' Class : +
##
```

Use 5-fold cross validation to evaluate each model. Find the model with the best f-measure for class “+”. Report the model and the f-measure.

```r
# Linear Kernel SVM Model
cv = createFolds(y = credit$A16, k = 5)

rmse_cv = c()

for (test_row in cv) {

  credit_train = credit[-test_row, ]
  credit_test = credit[test_row, ]

  svm_model_linear = svm(A16 ~ ., data = credit_train,
                         kernel = "linear")

  pred_svm_model_linear = predict(svm_model_linear, credit_test)

  cm = confusionMatrix(pred_svm_model_linear, credit_test$A16, mode = "prec_recall", positive = "+")

  print(cm$byClass[7])
}
```

```
##        F1
## 0.8976378
##        F1
## 0.8461538
##        F1
## 0.8769231
##        F1
## 0.8181818
##        F1
## 0.8888889
```

```r
# Radial Kernel SVM Model
cv = createFolds(y = credit$A16, k = 5)

rmse_cv = c()

for (test_row in cv) {

  credit_train = credit[-test_row, ]
  credit_test = credit[test_row, ]

  svm_model_radial = svm(A16 ~ ., data = credit_train,
                         kernel = "radial")

  pred_svm_model_radial = predict(svm_model_radial, credit_test)

  cm = confusionMatrix(pred_svm_model_radial, credit_test$A16, mode = "prec_recall", positive = "+")

  print(cm$byClass[7])
}
```

```
##        F1
## 0.8346457
##       F1
## 0.859375
##        F1
## 0.8208955
##        F1
## 0.9365079
##        F1
## 0.8549618
```