SVM parameter tuning

11/30/2021

## Load libraries

library(tidyverse)  
library(kernlab)  
library(tidymodels)  
library(rsample)  
library(doParallel)  
library(caret)

## Load and slipe the data

data <- read.csv("data.csv")  
  
data %>%  
 mutate(target = as.factor(target)) -> data  
glimpse(data)

## Rows: 303  
## Columns: 14  
## $ ï..age <int> 63, 37, 41, 56, 57, 57, 56, 44, 52, 57, 54, 48, 49, 64, 58, 5~  
## $ sex <int> 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1~  
## $ cp <int> 3, 2, 1, 1, 0, 0, 1, 1, 2, 2, 0, 2, 1, 3, 3, 2, 2, 3, 0, 3, 0~  
## $ trestbps <int> 145, 130, 130, 120, 120, 140, 140, 120, 172, 150, 140, 130, 1~  
## $ chol <int> 233, 250, 204, 236, 354, 192, 294, 263, 199, 168, 239, 275, 2~  
## $ fbs <int> 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0~  
## $ restecg <int> 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1~  
## $ thalach <int> 150, 187, 172, 178, 163, 148, 153, 173, 162, 174, 160, 139, 1~  
## $ exang <int> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0~  
## $ oldpeak <dbl> 2.3, 3.5, 1.4, 0.8, 0.6, 0.4, 1.3, 0.0, 0.5, 1.6, 1.2, 0.2, 0~  
## $ slope <int> 0, 0, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 1, 2, 0, 2, 2, 1~  
## $ ca <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0~  
## $ thal <int> 1, 2, 2, 2, 2, 1, 2, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3~  
## $ target <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~

data <- rename(data, age = ï..age) # Rename the variabe to avoid garbage characters

The following code is to split the dataset into training and test sets

set.seed(123) # Set seeds to make it reproducible  
split\_obj <- initial\_split(data = data, prop = 0.7, strata = target)  
train <- training(split\_obj)  
test <- testing(split\_obj)

## i

Create the recipe

rec\_obj <-  
 recipe(target ~ ., data=train) %>%  
 step\_impute\_median(all\_numeric(), -all\_outcomes()) %>%  
 step\_scale(all\_numeric(), - all\_outcomes()) %>%  
 step\_unknown(all\_nominal(), -all\_outcomes()) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes()) %>%  
 prep()  
  
# Bake  
  
train <- bake(rec\_obj, new\_data=train)  
test <- bake(rec\_obj, new\_data=test)

### Define the tuning specification

tuning\_specs <- svm\_poly(  
 cost = tune(), degree = 1  
) %>%  
 set\_mode("classification") %>%  
 set\_engine("kernlab"); tuning\_specs

## Polynomial Support Vector Machine Specification (classification)  
##   
## Main Arguments:  
## cost = tune()  
## degree = 1  
##   
## Computational engine: kernlab

cost <- expand.grid(cost = c(0.25, 0.5, 0.75, 1, 1.25, 1.5))  
cost

## cost  
## 1 0.25  
## 2 0.50  
## 3 0.75  
## 4 1.00  
## 5 1.25  
## 6 1.50

Resampling with v = 3.

cv\_folds <- vfold\_cv(data = train, v = 3); cv\_folds

## # 3-fold cross-validation   
## # A tibble: 3 x 2  
## splits id   
## <list> <chr>  
## 1 <split [140/71]> Fold1  
## 2 <split [141/70]> Fold2  
## 3 <split [141/70]> Fold3

svm\_wf1 <- workflow() %>%  
 add\_model(tuning\_specs) %>% # specifications  
 add\_formula(target ~ .) # formula

### Tuning is faster in parallel

Setting up cores

cores1 <- parallel::detectCores(logical = FALSE)  
cl <- makeCluster(cores1)  
registerDoParallel(cl)

### Start Tuning

svm\_res1 <- svm\_wf1 %>%   
 tune\_grid(resamples = cv\_folds,  
 grid = cost)

### Collect the results and optimized model

svm\_res1 %>%   
 collect\_metrics() %>%  
 arrange(desc(mean))

## # A tibble: 12 x 7  
## cost .metric .estimator mean n std\_err .config   
## <dbl> <chr> <chr> <dbl> <int> <dbl> <chr>   
## 1 0.5 roc\_auc binary 0.885 3 0.0277 Preprocessor1\_Model2  
## 2 0.25 roc\_auc binary 0.883 3 0.0273 Preprocessor1\_Model1  
## 3 1.5 roc\_auc binary 0.883 3 0.0221 Preprocessor1\_Model6  
## 4 0.75 roc\_auc binary 0.883 3 0.0260 Preprocessor1\_Model3  
## 5 1 roc\_auc binary 0.883 3 0.0260 Preprocessor1\_Model4  
## 6 1.25 roc\_auc binary 0.883 3 0.0260 Preprocessor1\_Model5  
## 7 0.5 accuracy binary 0.811 3 0.0324 Preprocessor1\_Model2  
## 8 0.75 accuracy binary 0.811 3 0.0279 Preprocessor1\_Model3  
## 9 1 accuracy binary 0.811 3 0.0279 Preprocessor1\_Model4  
## 10 1.25 accuracy binary 0.811 3 0.0279 Preprocessor1\_Model5  
## 11 1.5 accuracy binary 0.811 3 0.0279 Preprocessor1\_Model6  
## 12 0.25 accuracy binary 0.801 3 0.0320 Preprocessor1\_Model1

final1 <-   
 svm\_res1 %>%  
 select\_best(metric = "accuracy") # best model based on accuracy  
final1

## # A tibble: 1 x 2  
## cost .config   
## <dbl> <chr>   
## 1 0.5 Preprocessor1\_Model2

### Save the final model to an object

model1 <- svm\_wf1 %>%  
 finalize\_workflow(final1) %>%  
 fit(data=train) # fit on train data

### Prediction and evaluation on training data using model 1

p1 <- predict(model1, train, type="prob")  
head(p1)

## # A tibble: 6 x 2  
## .pred\_0 .pred\_1  
## <dbl> <dbl>  
## 1 0.823 0.177  
## 2 0.878 0.122  
## 3 0.205 0.795  
## 4 0.433 0.567  
## 5 0.453 0.547  
## 6 0.787 0.213

p1 <- p1$.pred\_1  
pred1 <- p1 >= 0.5  
pr1 <- as.numeric(pred1)  
confusionMatrix(as.factor(pr1), train$target)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 74 10  
## 1 22 105  
##   
## Accuracy : 0.8483   
## 95% CI : (0.7927, 0.8939)  
## No Information Rate : 0.545   
## P-Value [Acc > NIR] : < 2e-16   
##   
## Kappa : 0.691   
##   
## Mcnemar's Test P-Value : 0.05183   
##   
## Sensitivity : 0.7708   
## Specificity : 0.9130   
## Pos Pred Value : 0.8810   
## Neg Pred Value : 0.8268   
## Prevalence : 0.4550   
## Detection Rate : 0.3507   
## Detection Prevalence : 0.3981   
## Balanced Accuracy : 0.8419   
##   
## 'Positive' Class : 0   
##

### prediction and evaluation on testing data using mdoel1.

p2 <- predict(model1, test, type="prob")  
head(p2)

## # A tibble: 6 x 2  
## .pred\_0 .pred\_1  
## <dbl> <dbl>  
## 1 0.371 0.629  
## 2 0.126 0.874  
## 3 0.129 0.871  
## 4 0.0514 0.949  
## 5 0.171 0.829  
## 6 0.0307 0.969

p2 <- p2$.pred\_1  
pred2 <- p2 >= 0.5  
pr2 <- as.numeric(pred2)  
confusionMatrix(as.factor(pr2), test$target)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 33 3  
## 1 9 47  
##   
## Accuracy : 0.8696   
## 95% CI : (0.7832, 0.9307)  
## No Information Rate : 0.5435   
## P-Value [Acc > NIR] : 2.346e-11   
##   
## Kappa : 0.7341   
##   
## Mcnemar's Test P-Value : 0.1489   
##   
## Sensitivity : 0.7857   
## Specificity : 0.9400   
## Pos Pred Value : 0.9167   
## Neg Pred Value : 0.8393   
## Prevalence : 0.4565   
## Detection Rate : 0.3587   
## Detection Prevalence : 0.3913   
## Balanced Accuracy : 0.8629   
##   
## 'Positive' Class : 0   
##

## iii

Define the tuning specification

tuning\_specs <- svm\_rbf(cost = tune(), rbf\_sigma = tune()) %>%  
 set\_mode("classification") %>%  
 set\_engine("kernlab"); tuning\_specs

## Radial Basis Function Support Vector Machine Specification (classification)  
##   
## Main Arguments:  
## cost = tune()  
## rbf\_sigma = tune()  
##   
## Computational engine: kernlab

Define a grid to vary sigma

sigma <- expand.grid(rbf\_sigma = c(0.1, 1, 2, 3), cost = c(0.25, 0.5, 0.75, 1))

Resampling and setting up the workflow

cv\_folds <- vfold\_cv(data = train, v = 3); cv\_folds

## # 3-fold cross-validation   
## # A tibble: 3 x 2  
## splits id   
## <list> <chr>  
## 1 <split [140/71]> Fold1  
## 2 <split [141/70]> Fold2  
## 3 <split [141/70]> Fold3

# Workflow for tuning  
svm\_wf3 <- workflow() %>%  
 add\_model(tuning\_specs) %>% # specs  
 add\_formula(target ~ .) # formula

### Start Tuning

svm\_res3 <- svm\_wf3 %>%   
 tune\_grid(resamples = cv\_folds,  
 grid = sigma)

### Collect the results and optimized model

svm\_res3 %>%   
 collect\_metrics() %>%  
 arrange(desc(mean))

## # A tibble: 32 x 8  
## cost rbf\_sigma .metric .estimator mean n std\_err .config   
## <dbl> <dbl> <chr> <chr> <dbl> <int> <dbl> <chr>   
## 1 0.75 0.1 roc\_auc binary 0.899 3 0.0291 Preprocessor1\_Model09  
## 2 1 0.1 roc\_auc binary 0.898 3 0.0280 Preprocessor1\_Model13  
## 3 0.5 0.1 roc\_auc binary 0.898 3 0.0322 Preprocessor1\_Model05  
## 4 0.25 0.1 roc\_auc binary 0.895 3 0.0335 Preprocessor1\_Model01  
## 5 1 0.1 accuracy binary 0.820 3 0.0254 Preprocessor1\_Model13  
## 6 1 1 roc\_auc binary 0.812 3 0.0132 Preprocessor1\_Model14  
## 7 0.25 1 roc\_auc binary 0.812 3 0.0137 Preprocessor1\_Model02  
## 8 0.5 1 roc\_auc binary 0.812 3 0.0137 Preprocessor1\_Model06  
## 9 0.75 1 roc\_auc binary 0.812 3 0.0137 Preprocessor1\_Model10  
## 10 0.75 0.1 accuracy binary 0.810 3 0.0209 Preprocessor1\_Model09  
## # ... with 22 more rows

final3 <-   
 svm\_res3 %>%  
 select\_best(metric = "accuracy") # best model based on accuracy  
final3

## # A tibble: 1 x 3  
## cost rbf\_sigma .config   
## <dbl> <dbl> <chr>   
## 1 1 0.1 Preprocessor1\_Model13

### Save the final model to an object

model3 <- svm\_wf3 %>%  
 finalize\_workflow(final3) %>%  
 fit(data=train) # fit on train data

### Prediction and evaluation on training data using model 3

p5 <- predict(model3, train, type="prob")  
head(p5)

## # A tibble: 6 x 2  
## .pred\_0 .pred\_1  
## <dbl> <dbl>  
## 1 0.945 0.0550  
## 2 0.984 0.0158  
## 3 0.657 0.343   
## 4 0.869 0.131   
## 5 0.558 0.442   
## 6 0.945 0.0550

p5 <- p5$.pred\_1  
pred5 <- p5 >= 0.5  
pr5 <- as.numeric(pred5)  
confusionMatrix(as.factor(pr5), train$target)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 90 3  
## 1 6 112  
##   
## Accuracy : 0.9573   
## 95% CI : (0.9206, 0.9803)  
## No Information Rate : 0.545   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9138   
##   
## Mcnemar's Test P-Value : 0.505   
##   
## Sensitivity : 0.9375   
## Specificity : 0.9739   
## Pos Pred Value : 0.9677   
## Neg Pred Value : 0.9492   
## Prevalence : 0.4550   
## Detection Rate : 0.4265   
## Detection Prevalence : 0.4408   
## Balanced Accuracy : 0.9557   
##   
## 'Positive' Class : 0   
##

### Prediction and evaluation on training data using model 2

p6 <- predict(model3, test, type="prob")  
head(p6)

## # A tibble: 6 x 2  
## .pred\_0 .pred\_1  
## <dbl> <dbl>  
## 1 0.735 0.265  
## 2 0.0287 0.971  
## 3 0.0392 0.961  
## 4 0.0218 0.978  
## 5 0.0358 0.964  
## 6 0.0352 0.965

p6 <- p6$.pred\_1  
pred6 <- p6 >= 0.5  
pr6 <- as.numeric(pred6)  
confusionMatrix(as.factor(pr6), test$target)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 37 9  
## 1 5 41  
##   
## Accuracy : 0.8478   
## 95% CI : (0.7579, 0.9142)  
## No Information Rate : 0.5435   
## P-Value [Acc > NIR] : 5.991e-10   
##   
## Kappa : 0.6957   
##   
## Mcnemar's Test P-Value : 0.4227   
##   
## Sensitivity : 0.8810   
## Specificity : 0.8200   
## Pos Pred Value : 0.8043   
## Neg Pred Value : 0.8913   
## Prevalence : 0.4565   
## Detection Rate : 0.4022   
## Detection Prevalence : 0.5000   
## Balanced Accuracy : 0.8505   
##   
## 'Positive' Class : 0   
##