About the method - cla\_svm: Support Vector Machine for classification, maximizing the margin between classes. - Common hyperparameters: cost (penalty), epsilon (insensitive-margin width), and kernel (e.g., linear, radial, polynomial, sigmoid).

Environment setup: install and load the package.

# Classification using Support Vector Machine  
  
# installation   
#install.packages("daltoolbox")  
  
# loading DAL  
library(daltoolbox)

Load sample data (iris) and initial inspection.

iris <- datasets::iris  
head(iris)

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
## 1 5.1 3.5 1.4 0.2 setosa  
## 2 4.9 3.0 1.4 0.2 setosa  
## 3 4.7 3.2 1.3 0.2 setosa  
## 4 4.6 3.1 1.5 0.2 setosa  
## 5 5.0 3.6 1.4 0.2 setosa  
## 6 5.4 3.9 1.7 0.4 setosa

Identify the target Species levels.

# extracting the levels for the dataset  
slevels <- levels(iris$Species)  
slevels

## [1] "setosa" "versicolor" "virginica"

# Building train and test samples via random sampling

Random train/test split.

# Construindo amostras (treino e teste) por amostragem aleatória  
set.seed(1)  
sr <- sample\_random()  
sr <- train\_test(sr, iris)  
iris\_train <- sr$train  
iris\_test <- sr$test

Check class distribution after the split.

tbl <- rbind(table(iris[,"Species"]),   
 table(iris\_train[,"Species"]),   
 table(iris\_test[,"Species"]))  
rownames(tbl) <- c("dataset", "training", "test")  
head(tbl)

## setosa versicolor virginica  
## dataset 50 50 50  
## training 39 38 43  
## test 11 12 7

# Model training

Train SVM: tune cost, epsilon, and optionally kernel.

# Treinamento do modelo  
model <- cla\_svm("Species", slevels, epsilon=0.0, cost=20.000) # kernel padrão; ajuste conforme necessário  
model <- fit(model, iris\_train)

# Training evaluation

Predict and compute metrics.

# Verificando ajuste no treino  
train\_prediction <- predict(model, iris\_train)  
  
# Avaliação do modelo (treino)  
iris\_train\_predictand <- adjust\_class\_label(iris\_train[,"Species"])  
train\_eval <- evaluate(model, iris\_train\_predictand, train\_prediction)  
print(train\_eval$metrics)

## accuracy TP TN FP FN precision recall sensitivity specificity f1  
## 1 0.975 39 81 0 0 1 1 1 1 1

# Test evaluation

Predict and compute metrics.

# Teste do modelo  
test\_prediction <- predict(model, iris\_test)  
  
iris\_test\_predictand <- adjust\_class\_label(iris\_test[,"Species"])  
  
# Avaliação no teste  
 test\_eval <- evaluate(model, iris\_test\_predictand, test\_prediction)  
print(test\_eval$metrics)

## accuracy TP TN FP FN precision recall sensitivity specificity f1  
## 1 1 11 19 0 0 1 1 1 1 1