About the utility - cla\_tune: performs hyperparameter search for a classifier over ranges defined in ranges. - The example below tunes a cla\_svm varying epsilon, cost, and kernel.

Environment setup.

# Tuning de Classificação   
  
# installation   
#install.packages("daltoolbox")  
  
# loading DAL  
library(daltoolbox)

Sample data and target levels.

# Conjunto de dados para classificação  
  
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

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

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

# Train/test split

Random split for tuning validation.

# preparando amostragem aleatória  
set.seed(1)  
sr <- sample\_random()  
sr <- train\_test(sr, iris)  
iris\_train <- sr$train  
iris\_test <- sr$test  
  
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

# Hyperparameter grid and search training

# Treinamento com busca de hiperparâmetros  
tune <- cla\_tune(cla\_svm("Species", slevels),   
 ranges = list(epsilon=seq(0,1,0.2), cost=seq(20,100,20), kernel = c("linear", "radial", "polynomial", "sigmoid")))  
  
model <- fit(tune, iris\_train)

# Training evaluation with the best configuration

# Avaliação no treino  
train\_prediction <- predict(model, iris\_train)  
  
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 0.9833333 39 81 0 0 1 1 1 1 1

# Test evaluation

# Avaliação no teste  
test\_prediction <- predict(model, iris\_test)  
  
iris\_test\_predictand <- adjust\_class\_label(iris\_test[,"Species"])  
  
# Evaluating # setosa as primary class  
test\_eval <- evaluate(model, iris\_test\_predictand, test\_prediction)  
print(test\_eval$metrics)

## accuracy TP TN FP FN precision recall sensitivity specificity f1  
## 1 0.9333333 11 19 0 0 1 1 1 1 1

# Example grids for other models

# Opções de grids para outros modelos  
# knn  
ranges <- list(k=1:20)  
  
# mlp  
ranges <- list(size=1:10, decay=seq(0, 1, 0.1))  
  
# rf  
ranges <- list(mtry=1:3, ntree=1:10)