## Multi-Layer Perceptron (MLP) Classifier

This example uses MLP (scikit-learn via reticulate) to classify the Iris dataset. Workflow: split train/test, train, predict, and evaluate.

Prerequisites - R packages: daltoolbox, daltoolboxdp - Python accessible via reticulate (scikit-learn installed)

# Installation (if needed)  
#install.packages("daltoolboxdp")

# Loading packages  
library(daltoolbox)  
library(daltoolboxdp)

# Loading Iris dataset  
iris <- datasets::iris

# Training and evaluation with MLP  
  
slevels <- levels(iris$Species)  
  
set.seed(1)  
sr <- sample\_random()  
sr <- train\_test(sr, iris)  
iris\_train <- sr$train  
iris\_test <- sr$test  
  
# Numeric encoding of the target for scikit-learn  
iris\_train$species\_encoded <- as.integer(as.factor(iris\_train$Species))  
iris\_train\_label <- iris\_train[, !names(iris\_train) %in% "Species"]  
  
model <- skcla\_mlp("species\_encoded", slevels, max\_iter = 1000) # increase max\_iter for convergence  
model <- fit(model, iris\_train\_label)  
train\_prediction <- predict(model, iris\_train\_label)  
  
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

iris\_test$species\_encoded <- as.integer(as.factor(iris\_test$Species))  
iris\_test\_label <- iris\_test[, !names(iris\_test) %in% "Species"]  
test\_prediction <- predict(model, iris\_test\_label)  
  
iris\_test\_predictand <- adjust\_class\_label(iris\_test[, "Species"])  
test\_eval <- evaluate(model, iris\_test\_predictand, test\_prediction)  
print(test\_eval$metrics)

## accuracy TP TN FP FN precision recall sensitivity specificity f1  
## 1 0.9666667 11 19 0 0 1 1 1 1 1