Objective: Demonstrate how to train, validate, and evaluate an MLP (Multilayer Perceptron) model for time-series forecasting with sliding windows, including data preparation, normalization, model fitting, and evaluation with metrics and plots.

# Time Series Regression - MLP  
  
# Installing the package (if needed)  
#install.packages("tspredit")

# Loading the packages  
library(daltoolbox)  
library(tspredit)

# Series for study (generates sliding windows t9..t0)  
  
data(tsd)  
ts <- ts\_data(tsd$y, 10)  
ts\_head(ts, 3)

## t9 t8 t7 t6 t5 t4 t3 t2 t1 t0  
## [1,] 0.0000000 0.2474040 0.4794255 0.6816388 0.8414710 0.9489846 0.9974950 0.9839859 0.9092974 0.7780732  
## [2,] 0.2474040 0.4794255 0.6816388 0.8414710 0.9489846 0.9974950 0.9839859 0.9092974 0.7780732 0.5984721  
## [3,] 0.4794255 0.6816388 0.8414710 0.9489846 0.9974950 0.9839859 0.9092974 0.7780732 0.5984721 0.3816610

# Original series visualization  
library(ggplot2)  
plot\_ts(x=tsd$x, y=tsd$y) + theme(text = element\_text(size=16))



# Train-test split and projection (X, y)  
  
samp <- ts\_sample(ts, test\_size = 5)  
io\_train <- ts\_projection(samp$train)  
io\_test <- ts\_projection(samp$test)

# Preprocessing (global min-max normalization)  
  
preproc <- ts\_norm\_gminmax()

# Training the MLP model  
  
model <- ts\_mlp(ts\_norm\_gminmax(), input\_size=4, size=4, decay=0)  
model <- fit(model, x=io\_train$input, y=io\_train$output)

# Fit evaluation (train)  
  
adjust <- predict(model, io\_train$input)  
adjust <- as.vector(adjust)  
output <- as.vector(io\_train$output)  
ev\_adjust <- evaluate(model, output, adjust)  
ev\_adjust$mse

## [1] 1.452179e-05

# Forecast on test set (5 steps ahead)  
  
prediction <- predict(model, x=io\_test$input[1,], steps\_ahead=5)  
prediction <- as.vector(prediction)  
output <- as.vector(io\_test$output)  
ev\_test <- evaluate(model, output, prediction)  
ev\_test

## $values  
## [1] 0.41211849 0.17388949 -0.07515112 -0.31951919 -0.54402111  
##   
## $prediction  
## [1] 0.41432866 0.17857629 -0.06878185 -0.31271542 -0.53941686  
##   
## $smape  
## [1] 0.03009372  
##   
## $mse  
## [1] 2.698181e-05  
##   
## $R2  
## [1] 0.999767  
##   
## $metrics  
## mse smape R2  
## 1 2.698181e-05 0.03009372 0.999767

# Plot comparing actual vs fit (train) and forecast (test)  
  
yvalues <- c(io\_train$output, io\_test$output)  
plot\_ts\_pred(y=yvalues, yadj=adjust, ypre=prediction) + theme(text = element\_text(size=16))

