In this post, I present some examples of use of crossval on a linear model, and on the popular xgboost and randomForest models. The **error measure** used is [Root Mean Squared Error (RMSE)](https://en.wikipedia.org/wiki/Mean_squared_error), and is currently the only choice implemented.

**Installation**

From R console:

Library("crossval")

**Demo**

We use a simulated dataset for this demo, containing 100 examples, and 5 explanatory variables:

# dataset creation

set.seed(123)

n <- 100 ; p <- 5

X <- matrix(rnorm(n \* p), n, p)

y <- rnorm(n)

**Linear model**

* X contains the explanatory variables
* y is the response
* k is the number of folds in k-fold cross-validation
* repeats is the number of repeats of the k-fold cross-validation procedure

**Linear model** example:

crossval::crossval\_ml(x = X, y = y, k = 5, repeats = 3)

## $folds

## repeat\_1 repeat\_2 repeat\_3

## fold\_1 0.8987732 0.9270326 0.7903096

## fold\_2 0.8787553 0.8704522 1.2394063

## fold\_3 1.0810407 0.7907543 1.3381991

## fold\_4 1.0594537 1.1981031 0.7368007

## fold\_5 0.7593157 0.8913229 0.7734180

##

## $mean

## [1] 0.9488758

##

## $sd

## [1] 0.1902999

##

## $median

## [1] 0.8913229

Linear model example, with **validation set**:

crossval::crossval\_ml(x = X, y = y, k = 5, repeats = 3, p = 0.8)

## $folds

## repeat\_1 repeat\_2 repeat\_3

## fold\_training\_1 1.1256933 0.9144503 0.9746044

## fold\_validation\_1 0.9734644 0.9805410 0.9761265

## fold\_training\_2 1.0124938 0.9652489 0.7257494

## fold\_validation\_2 0.9800293 0.9577811 0.9631389

## fold\_training\_3 0.7695705 1.0091999 0.9740067

## fold\_validation\_3 0.9753250 1.0373943 0.9863062

## fold\_training\_4 1.0482233 0.9194648 0.9680724

## fold\_validation\_4 0.9984861 0.9596531 0.9742874

## fold\_training\_5 0.9210179 1.0455006 0.9886350

## fold\_validation\_5 1.0126038 0.9658146 0.9658412

##

## $mean\_training

## [1] 0.9574621

##

## $mean\_validation

## [1] 0.9804529

##

## $sd\_training

## [1] 0.1018837

##

## $sd\_validation

## [1] 0.02145046

##

## $median\_training

## [1] 0.9740067

##

## $median\_validation

## [1] 0.975325

**Random Forest**

**randomForest** example:

require(randomForest)

# fit randomForest with mtry = 4

crossval::crossval\_ml(x = X, y = y, k = 5, repeats = 3,

fit\_func = randomForest::randomForest, predict\_func = predict,

packages = "randomForest", fit\_params = list(mtry = 4))

## $folds

## repeat\_1 repeat\_2 repeat\_3

## fold\_1 0.9820183 0.9895682 0.8752296

## fold\_2 0.8701763 0.8771651 1.2719188

## fold\_3 1.1869986 0.7736392 1.3521407

## fold\_4 1.0946892 1.1204090 0.7100938

## fold\_5 0.9847612 1.0565001 0.9194678

##

## $mean

## [1] 1.004318

##

## $sd

## [1] 0.1791315

##

## $median

## [1] 0.9847612

randomForest with parameter mtry = 4, and a **validation set**:

crossval::crossval\_ml(x = X, y = y, k = 5, repeats = 2, p = 0.8,

fit\_func = randomForest::randomForest, predict\_func = predict,

packages = "randomForest", fit\_params = list(mtry = 4))

## $folds

## repeat\_1 repeat\_2

## fold\_training\_1 1.0819863 0.9096807

## fold\_validation\_1 0.8413615 0.8415839

## fold\_training\_2 0.9507086 1.0014771

## fold\_validation\_2 0.5631285 0.6545253

## fold\_training\_3 0.7020669 0.9632402

## fold\_validation\_3 0.5090071 0.9129895

## fold\_training\_4 0.8932151 1.0315366

## fold\_validation\_4 0.8299454 0.7147867

## fold\_training\_5 0.9158418 1.1093461

## fold\_validation\_5 0.6438410 0.7644071

##

## $mean\_training

## [1] 0.9559099

##

## $mean\_validation

## [1] 0.7275576

##

## $sd\_training

## [1] 0.1151926

##

## $sd\_validation

## [1] 0.133119

##

## $median\_training

## [1] 0.9569744

##

## $median\_validation

## [1] 0.7395969

**xgboost**

In this case, the response and covariates are named ‘label’ and ‘data’. So (for now), we do this:

# xgboost example -----

require(xgboost)

f\_xgboost <- function(x, y, ...) xgboost::xgboost(data = x, label = y, ...)

Fit xgboost with nrounds = 10:

crossval::crossval\_ml(x = X, y = y, k = 5, repeats = 3,

fit\_func = f\_xgboost, predict\_func = predict,

packages = "xgboost", fit\_params = list(nrounds = 10,

verbose = FALSE))

## $folds

## repeat\_1 repeat\_2 repeat\_3

## fold\_1 0.9487191 1.2019850 0.9160024

## fold\_2 0.9194731 0.8990731 1.2619773

## fold\_3 1.2775092 0.7691470 1.3942022

## fold\_4 1.1893053 1.1250443 0.7173760

## fold\_5 1.1200368 1.1686622 0.9986680

##

## $mean

## [1] 1.060479

##

## $sd

## [1] 0.1965465

##

## $median

## [1] 1.120037

Fit xgboost with `nrounds = 10, and **validation set**:

crossval::crossval\_ml(x = X, y = y, k = 5, repeats = 2, p = 0.8,

fit\_func = f\_xgboost, predict\_func = predict,

packages = "xgboost", fit\_params = list(nrounds = 10,

verbose = FALSE))

## $folds

## repeat\_1 repeat\_2

## fold\_training\_1 1.1063607 1.0350719

## fold\_validation\_1 0.7891655 1.0025217

## fold\_training\_2 1.0117042 1.1723135

## fold\_validation\_2 0.4325200 0.5050369

## fold\_training\_3 0.7074600 1.0101371

## fold\_validation\_3 0.1916094 0.9800865

## fold\_training\_4 0.9131272 1.2411424

## fold\_validation\_4 0.8998582 0.7521359

## fold\_training\_5 0.9462418 1.0543695

## fold\_validation\_5 0.5432650 0.6850912

##

## $mean\_training

## [1] 1.019793

##

## $mean\_validation

## [1] 0.678129

##

## $sd\_training

## [1] 0.147452

##

## $sd\_validation

## [1] 0.2600431

##

## $median\_training

## [1] 1.023388

##

## $median\_validation

## [1] 0.7186136