modeltuner workflow

fitted model

Requirements:
- **formula** & **data** arguments, optionally **weights**
- *call* stored in object
- `predict()` returns a vector of length `nrow(data)`

Examples:
lm, **glm**, **lmrob**, (**g**)**lmer**, **gam**, **rpart**, …
In modeltuner pkg: **fm_xgb**, **fm_glmnet**, **fm_knn**

↓ `model(x, label=`*label*`)`

class **model**
or
class **multimodel**

Contains data and fitting call, but not necessarily the model fit
→ `fit()` restores original fitted model
→ `predict()` returns same output as for fitted object

Collection of >1 model objects
Same response in all models
Two ways to create a multimodel:
→ `c(model1, model2, ...)`
→ `multimodel(mymodel, parameter=`*values*`, ...)`

↓ `cv(x, nfold=10,...)`
Takes a (multi-)model and runs a cross-validation
Same cv-groups used for all models if **x** is a multimodel

class **cv**

Output of cross-validation for ≥1 model objects
→ `cv_predict()` Extract out-of-sample (oos) predictions
→ `cv_performance(x, metric, ...)`
Computes in-sample and oos performance
Metrics for continuous response: **rmse**, **mae**, **medae**
Metrics for binary response: **logLoss**, **classification_error**
→ `plot()` Scatter plot of response vs. oos predictions

↓ `tune(x, metric, ...)`
selects model having the smallest test error
or
↓ `extract_model(x,which)`
manually select a model (usually based on `cv()` results)

class **model**

↓ `fit(x)`

fitted model

Iteratively fitted models: fm_xgb, fm_glmnet

fitted model

Contains a ***sequence of models*** *("iterations")* of increasing structural complexity.
- `fm_xgb`: iterations arise from successively adding trees
- `fm_glmnet`: iterations correspond to different degrees of regularization (parameter `lambda`)

→ `evaluation_log()` training error only

↓ `model(x, label=`*label*`)`

class **model**
or
class **multimodel**

→ `predict()` returns predictions for last (most complex) model
→ `fit()` restores fitted model from last iteration (most complex model)
→ `evaluation_log()` training error only

↓ `cv(x, nfold=10, metric, iter = crit_min())`
*Criterion for **preferred iteration***
Alternative criteria: `crit_first()`
`crit_last()`
`crit_iter(iter)`
`crit_overfit(ratio=0.9)`
`crit_se(factor=1)`
Default values
Multiple criteria, e.g. `c(crit_min(),crit_last())`

class **cv**

cv-object contains
- training and test errors for *all iterations* according to `metric` (*evaluation log*)
- individual predictions for the *preferred iterations only*.

→ `cv_performance(x, metric, ...)` *uses (first) preferred iteration*
→ `cv_predict(x)` *uses (first) preferred iteration*
→ `evaluation_log(x, metric*, ...)` *training & test errors*

→ `set_iter(x, iter=crit_xy(), ...)`
Changes criterion for preferred iterations
→ `set_metric(x, metric=metric, ...)*`
Changes the default metric
Alternative metric: requires that `cv()` has been called with **keep_fits=TRUE**
→ `expand_iter(x, ...)`
Converts a cv with *one model* and *multiple preferred iterations* to a cv with *several (identical) models* and *one preferred iteration each*.

↓ `tune(x,...)`
or
↓ `extract_model(x,which)`

class **model**

↓ `fit(x)` Returns model according to (first) *preferred iteration*.

fitted model