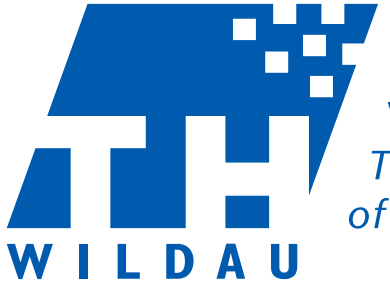


Technische
Hochschule
Wildau [FH]
Technical University
of Applied Sciences

Maschinelles Lernen

Maschinelles Lernen in R mit `mlr`

Prof. Dr. Rainer Stollhoff

Übersicht

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU

- Maschinelles Lernen in R mit `mlr`
  - Datenvorverarbeitung
  - Aufgabenstellung
  - Lernalgorithmen
  - Modellvalidierung
  - Modelltuning
  - Benchmark
- Ausblick
  - `mlr` und `mlr3` in R
  - `scikit-learn` in python

Übersicht

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU

- **Maschinelles Lernen in R mit `mlr`**
  - **Datenvorverarbeitung**
  - **Aufgabenstellung**
  - **Lernalgorithmen**
  - **Modellvalidierung**
  - **Modelltuning**
  - **Benchmark**
- Ausblick
  - `mlr` und `mlr3` in R
  - `scikit-learn` in python

Hintergrund: `mlr`

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU

`mlr` und die Weiterentwicklung `mlr3` bieten ein Interface für Maschinelles Lernen in R

- Einbindung der gängigen ML-Algorithmen aus anderen R Paketen
- Workflows zur Analyse mit einheitlicher Datenformatierung z.B. Benchmarks
- Tools zum Tuning der Modelle, zur Visualisierung,…

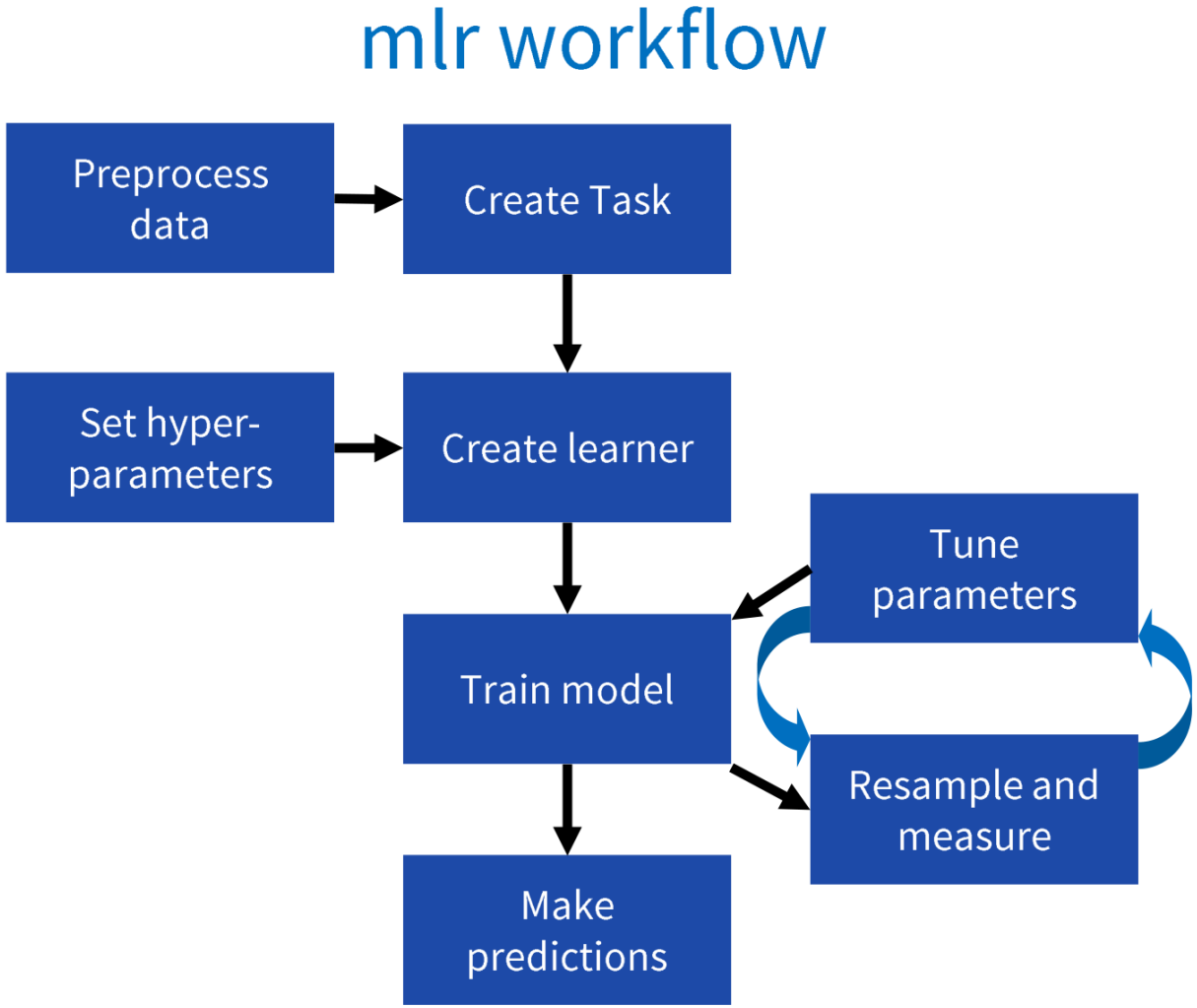
`mlr`

- `mlr` wurde 2013 das erste Mal auf CRAN vorgestellt.
- `mlr` hat seitdem `caret` als Standard-Interface zum Maschinellen Lernen in R abgelöst

flexible Erweiterungen in eigenen Paketen

Interface zu Machine Learning in R – `mlr`

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU



mlr – preprocess data

Preprocessing data

`createDummyFeatures(obj=,target=,method=,cols=)`
Creates (0,1) flags for each non-numeric variable excluding `target`. Can be applied to entire dataset or only specific `cols`

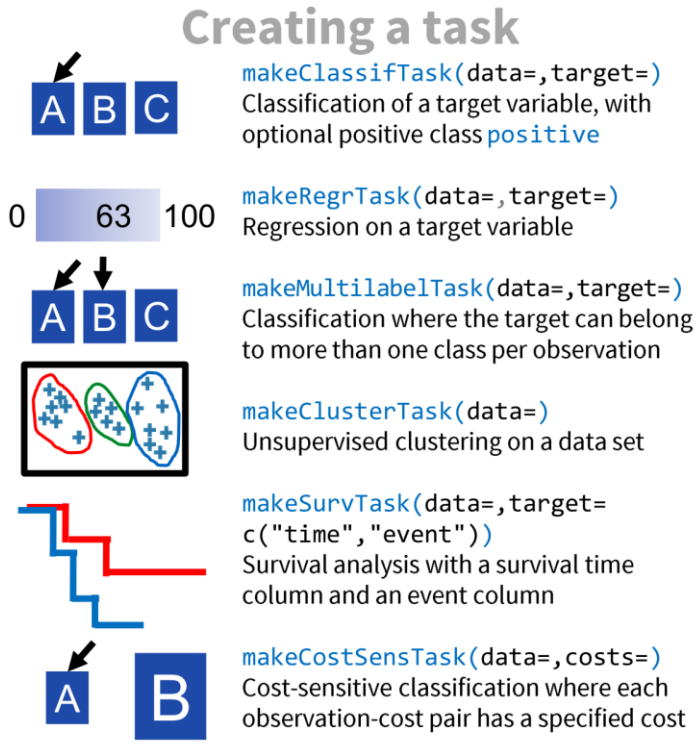
`normalizeFeatures(obj=,target=,method=,cols=, range=,on.constant=)`
Normalizes numerical features according to specified `method`:

- `"center"` (subtract mean)
- `"scale"` (divide by std. deviation)
- `"standardize"` (center and scale)
- `"range"` (linear scale to given range, default `range=c(0,1)`)

```
btw17.wahl.names <- colnames(btw17d)
colnames(btw17d) <- c("Y",paste("X",1:(ncol(btw17d)-1),sep=""))
btw17d$Y <- as.factor(as.numeric(btw17d$Y%in%c(1,5)))
btw17d <- as.data.frame(btw17d)

btw17d.n <-  normalizeFeatures(obj=btw17d,
                               target=colnames(btw17d)[1],
                               cols=colnames(btw17d)[-1],
                               method="standardize")
```

mlr - tasks




```
btw17d.task <- makeClassifTask(data=btw17d.n,
                               target=colnames(btw17d)[1])
```

mlr - learners

`makeLearner(cl=,predict.type=,...,par.vals=)`
Choose an algorithm class to perform the task and determine what that algorithm will predict

- `cl=`name of algorithm, e.g. `"classif.xgboost"` `"regr.randomForest"` `"cluster.kmeans"`
- `predict.type="response"` returns a prediction type that matches the source data; `"prob"` returns a predicted probability for classification problems only; `"se"` returns the a standard error of the prediction for regression problems only. Only certain learners can return `"prob"` and `"se"`
- `par.vals=` takes a list of hyperparameters and passes them to the learner; parameters can also be passed directly (`...`)

You can make multiple learners at once with `makeLearners()`

```
btw17d.learners.all <-
listLearners(btw17d.task,check.packages = F)
```

```
> btw17d.learners.all
                class                                                                                name short.name      package    type installed numerics factors ordered
1  classif.adaboostm1                                                                 ada Boosting M1 adaboostm1        RWeka classif      TRUE     TRUE    TRUE   FALSE
2    classif.boosting                                                                 Adabag Boosting     adabag adabag,rpart classif     FALSE     TRUE    TRUE   FALSE
3         classif.C50                                                                             C50        C50          C50 classif     FALSE     TRUE    TRUE   FALSE
4     classif.cforest                                  Random forest based on conditional inference trees    cforest        party classif     FALSE     TRUE    TRUE    TRUE
5       classif.ctree                                                Conditional Inference Trees      ctree        party classif     FALSE     TRUE    TRUE    TRUE
6    classif.cvglmnet GLM with Lasso or Elasticnet Regularization (Cross Validated Lambda)   cvglmnet       glmnet classif     FALSE     TRUE    TRUE   FALSE
  missings weights prob oneclass twoclass multiclass class.weights featimp oobpreds functionals single.functional    se lcens rcens icens
1    FALSE   FALSE TRUE    FALSE     TRUE       TRUE         FALSE   FALSE    FALSE       FALSE             FALSE FALSE FALSE FALSE FALSE
2     TRUE   FALSE TRUE    FALSE     TRUE       TRUE         FALSE    TRUE    FALSE       FALSE             FALSE FALSE FALSE FALSE FALSE
3     TRUE    TRUE TRUE    FALSE     TRUE       TRUE         FALSE   FALSE    FALSE       FALSE             FALSE FALSE FALSE FALSE FALSE
4     TRUE    TRUE TRUE    FALSE     TRUE       TRUE         FALSE    TRUE    FALSE       FALSE             FALSE FALSE FALSE FALSE FALSE
5     TRUE    TRUE TRUE    FALSE     TRUE       TRUE         FALSE   FALSE    FALSE       FALSE             FALSE FALSE FALSE FALSE FALSE
6    FALSE    TRUE TRUE    FALSE     TRUE       TRUE         FALSE   FALSE    FALSE       FALSE             FALSE FALSE FALSE FALSE FALSE
... (#rows: 59, #cols: 24)
```

mlr - learners

```
> btw17d.learners.all$name
 [1] "ada Boosting M1"
 [3] "C50"
 [5] "Conditional Inference Trees"
 [7] "Deep neural network with weights initialized by DBN"
 [9] "Evolutionary learning of globally optimal trees"
[11] "Featureless classifier"
[13] "Gaussian Processes"
[15] "Geometric Predictive Discriminant Analysis"
[17] "h2o.deeplearning"
[19] "h2o.randomForest"
[21] "J48 Decision Trees"
[23] "k-Nearest Neighbor"
[25] "Support Vector Machines"
[27] "L1-Regularized L2-Loss Support Vector Classification"
[29] "L2-Regularized L1-Loss Support Vector Classification"
[31] "L2-Regularized L2-Loss Support Vector Classification"
[33] "Linear Discriminant Analysis"
[35] "Learning Vector Quantization"
[37] "Multi-Layer Perceptron"
[39] "Naive Bayes"
[41] "Training Neural Network by Backpropagation"
[43] "PART Decision Lists"
[45] "Quadratic Discriminant Analysis"
[47] "Random Forest"
[49] "Regularized Discriminant Analysis"
[51] "Random k-Nearest-Neighbors"
[53] "Regularized Random Forests"
[55] "Deep neural network with weights initialized by Stacked AutoEncoder"
[57] "Sparse Discriminant Analysis"
[59] "eXtreme Gradient Boosting"
[61] "Adabag Boosting"
[63] "Random forest based on conditional inference trees"
[65] "GLM with Lasso or Elasticnet Regularization (Cross Validated Lambda)"
[67] "Flexible Discriminant Analysis"
[69] "Extremely Randomized Trees"
[71] "Fast k-Nearest Neighbour"
[73] "Gradient Boosting Machine"
[75] "GLM with Lasso or Elasticnet Regularization"
[77] "h2o.gbm"
[79] "k-Nearest Neighbours"
[81] "Propositional Rule Learner"
[83] "k-Nearest Neighbor"
[85] "Linear Discriminant Analysis"
[87] "L1-Regularized Logistic Regression"
[89] "L2-Regularized Logistic Regression"
[91] "Support Vector Classification by Crammer and Singer"
[93] "Least Squares Support Vector Machine"
[95] "Mixture Discriminant Analysis"
[97] "Multinomial Regression"
[99] "Neural Network"
[101] "1-R Classifier"
[103] "Quadratic Discriminant Analysis"
[105] "Random Forest"
[107] "Random Forests"
[109] "Random ferns"
[111] "Decision Tree"
[113] "Robust Regularized Linear Discriminant Analysis"
[115] "Shrinkage Discriminant Analysis"
[117] "Support Vector Machines (libsvm)"
```

```
btw17d.learners.all$name
btw17d.learners.subset <- c(52,28,58,14,47,40)
btw17d.learners <-
makeLearners(cl=btw17d.learners.subset$class)
```

Resampling

`makeResampleDesc(method=,...,stratify=)`
`method` must be one of the following:

- `"CV"` (cross-validation, for number of folds use `iters=`)
- `"LOO"` (leave-one-out cross-validation, for folds use `iters=`)
- `"RepCV"` (repeated cross-validation, for number of repetitions use `reps=`, for folds use `folds=`)
- `"Subsample"` (aka Monte-Carlo cross-validation, for iterations use `iters=`, for train % use `split=`)
- `"Bootstrap"` (out-of-bag bootstrap, uses `iters=`)
- `"Holdout"` (for train % use `split=`)

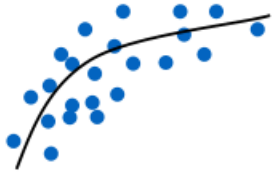
`stratify` keeps target proportions consistent across samples.

`makeResampleInstance(desc=,task=)` can reduce noise by ensuring the resampling is done identically every time.

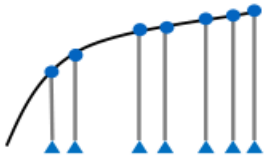
`resample(learner=,task=,resampling=,measures=)`
Train and test model according to specified resampling strategy.

Trainieren, Vorhersagen und Testen

Train a model and predict



`train(learner=,task=)`
Train a model (`WrappedModel`) by applying a learner to a task. By default, the model will train on all observations. The underlying model can be extracted with `getLearnerModel()`



`predict(object=,task=,newdata=)`
Use a trained model to make predictions on a task or dataset. The resulting `pred` object can be viewed with `View(pred)` or accessed by `as.data.frame(pred)`

Measuring performance

`performance(pred=,measures=)`
Calculate performance of predictions according to one or more of several measures (use `listMeasures()` for full list):

- **classif** `acc auc bac ber brier[.scaled] f1 fdr fn fnr fp fpr gmean multiclass[.au1u .aunp .aunu .brier] npv ppv qsr ssr tn tnr tp tpr wkappa`
- **regr** `arsq expvar kendalltau mae mape medae medse mse msle rae rmse rmsle rrse rsq sae spearmanrho sse`
- **cluster** `db dunn G1 G2 silhouette`
- **multilabel** `multilabel[.f1 .subset01 .tpr .ppv .acc .hamloss]`
- **costsens** `mcp meancosts`
- **surv** `cindex`
- **other** `featperc timeboth timepredict timetrain`

For detailed performance data on classification tasks, use:

- `calculateConfusionMatrix(pred=)`
- `calculateROCMeasures(pred=)`

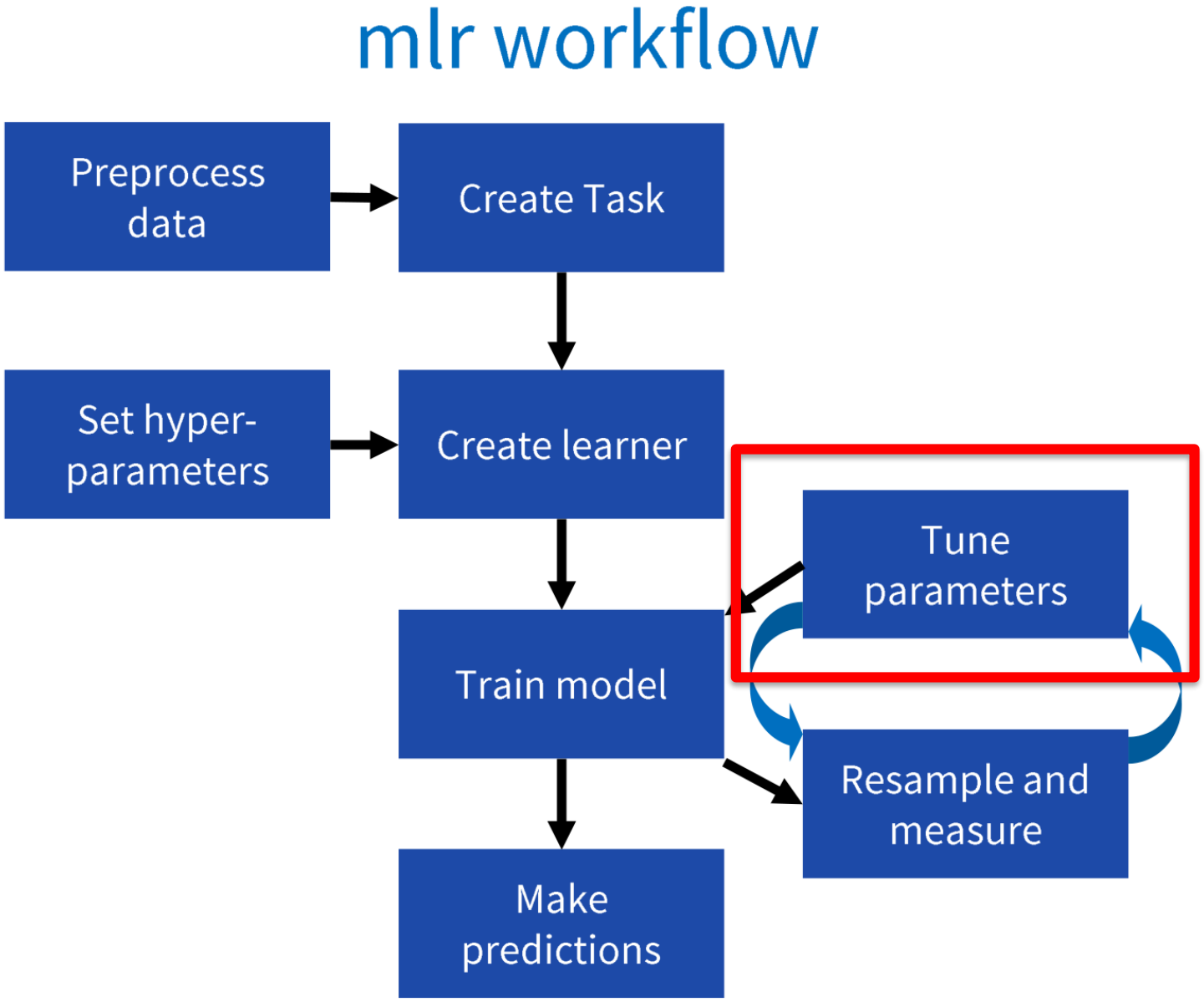
Ergebnisse Benchmark

```
> btw17d.benchmark.direct
   task.id            learner.id acc.test.mean
1 btw17d.n        classif.rpart     0.8025287
2 btw17d.n       classif.logreg     0.8255172
3 btw17d.n          classif.svm     0.8457471
4 btw17d.n          classif.gbm     0.8490805
5 btw17d.n classif.randomForest     0.8391954
6 btw17d.n         classif.nnet     0.8257471
>
```

```
# Benchmark on default values
btw17d.benchmark.direct <- benchmark(learners = btw17d.learners,
                                     tasks = btw17d.task,
                                     resamplings = btw17d.resample,
                                     measures = acc)
```

Interface zu Machine Learning in R - `mlr`

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU



`mlr` – tuning

```
Tune using tuneParams(learner=,task=,resampling=,
measures=,par.set=,control=)
```

- `learner` = Verfahren wählen
- `task` = Lernaufgabe identifizieren
- `resampling` = Resamplingverfahren
- `measure` = Gütemaß wählen
- `par.set` = Parameterraum setzen
- `control` = Suchalgorithmus auswählen

mlr - Parameterraum

Set search space using `makeParamSet(make<type>Param())`

- `makeNumericParam(id=,lower=,upper=,trafo=)`
- `makeIntegerParam(id=,lower=,upper=,trafo=)`
- `makeIntegerVectorParam(id=,len=,lower=,upper=,trafo=)`
- `makeDiscreteParam(id=,values=c(...))` (can also be used to test discrete values of numeric or integer parameters)

```
# rpart
getParamSet(btw17d.learners$classif.rpart)
rpart.param <- makeParamSet(
  makeNumericParam("cp",0.001,0.1),
  makeIntegerParam("maxdepth",1,10),
  makeIntegerParam("minsplit",1,10))

# logreg
getParamSet(btw17d.learners$classif.logreg)

# svm
getParamSet(btw17d.learners$classif.svm)
svm.param <-  makeParamSet(
  makeNumericParam("cost",0.01,10),
  makeNumericParam("gamma",.0001,1),
  makeDiscreteParam("kernel",c("radial")))
```

…

mlr - Suchalgorithmus

Set a search algorithm with `makeTuneControl<type>()`

- `Grid(resolution=10L)` Grid of all possible points
- `Random(maxit=100)` Randomly sample search space
- `MBO(budget=)` Use Bayesian model-based optimization
- `Irace(n.instances=)` Iterated racing process
- Other types: `CMAES`, `Design`, `GenSA`

```
rpart.tune <-  tuneParams(
 learner = btw17d.learners$classif.rpart,
 task = btw17d.task,
 cv5,acc,
 par.set = rpart.param,
 control = makeTuneControlMBO(budget=50))
```

mlr – tuning rpart

```
# rpart
getParamSet(btw17d.learners$classif.rpart)
rpart.param <- makeParamSet(
  makeNumericParam("cp",0.001,0.1),
  makeIntegerParam("maxdepth",1,10),
  makeIntegerParam("minsplit",1,10))
rpart.tune <-  tuneParams(learner = btw17d.learners$classif.rpart,
                          task = btw17d.task,
                          cv5,acc,
                          par.set = rpart.param,
                          control = makeTuneControlMBO(budget=50))
setHyperPars(btw17d.learners$classif.rpart,par.vals=rpart.tune$x)
```

- `cp` = Komplexitätsparameter
- `maxdepth` = Baumtiefe
- `minsplit` = Mindestgröße vor Split

mlr – tuning logreg

```
# logreg
getParamSet(btw17d.learners$classif.logreg)
```

- Kein Tuning bei regulärer logreg
- Immer vollständiges Modell

mlr – tuning svm

```
# svm
getParamSet(btw17d.learners$classif.svm)
svm.param <-  makeParamSet(
  makeNumericParam("cost",0.01,10),
  makeNumericParam("gamma",.0001,1),
  makeDiscreteParam("kernel",c("radial")))
svm.tune <- tuneParams(learner = btw17d.learners$classif.svm,
                       task = btw17d.task,
                         cv5,acc,
                         par.set = svm.param,
                         control = makeTuneControlMBO(budget=100))
setHyperPars(btw17d.learners$classif.svm,par.vals=svm.tune$x)
```

- `cost` = Kostenparameter Schlupfvariablen
- `gamma` = Rechweite der RBF
- `kernel` = Radiale Basis Funktion (RBF)

mlr – tuning gbm

```r
# gbm
getParamSet(btw17d.learners$classif.gbm)
gbm.param <-  makeParamSet(
  makeNumericParam("shrinkage",0.001,1),
  makeIntegerParam("interaction.depth",1,10),
  makeIntegerParam("n.trees",100,1000),
  makeDiscreteParam("distribution","bernoulli")
)
gbm.tune <- tuneParams(learner = btw17d.learners$classif.gbm,
                       task = btw17d.task,
                       cv5,acc,
                       par.set = gbm.param,
                       control = makeTuneControlMBO(budget=50))
setHyperPars(btw17d.learners$classif.gbm,par.vals=gbm.tune$x)
```

- `shrinkage` = Vorhersagen näher bei p=50%
- `interaction.depth` = Baumtiefe
- `n.trees` = Anzahl Bäume
- `distribution` = `Bernoulli`

mlr – tuning randomForest

```
# randomForest
getParamSet(btw17d.learners$classif.randomForest)
rf.param <-  makeParamSet(
  makeIntegerParam("mtry",1,10),
  makeIntegerParam("ntree",100,1000)

)
rf.tune <- tuneParams(learner = btw17d.learners$classif.randomForest,
                      task = btw17d.task,
                      cv5,acc,
                      par.set = rf.param,
                      control = makeTuneControlMBO(budget=50))
setHyperPars(btw17d.learners$classif.randomForest,par.vals=rf.tune$x)
```

- `mtry` = Anzahl der Zufallsvarianten je Split
- `n.trees` = Anzahl Bäume

mlr – tuning nnet

```
# neuralnet
getParamSet(btw17d.learners$classif.nnet)
nnet.param <-  makeParamSet(
  makeIntegerParam("size",1,20)
)
nnet.tune <- tuneParams(learner = btw17d.learners$classif.nnet,
                        task = btw17d.task,
                        cv5,acc,
                        par.set = nnet.param,
                        control = makeTuneControlMBO(budget=20))
setHyperPars(btw17d.learners$classif.nnet,par.vals=nnet.tune$x)
```

- `size` = Anzahl der Hidden Nodes

Tuned Benchmark

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU

```
> btw17d.benchmark.direct
   task.id            learner.id acc.test.mean
1 btw17d.n        classif.rpart     0.8025287
2 btw17d.n       classif.logreg     0.8255172
3 btw17d.n          classif.svm     0.8457471
4 btw17d.n          classif.gbm     0.8490805
5 btw17d.n classif.randomForest     0.8391954
6 btw17d.n        classif.nnet      0.8257471
> btw17d.benchmark.tuned
   task.id            learner.id acc.test.mean
1 btw17d.n        classif.rpart     0.8428736
2 btw17d.n       classif.logreg     0.8225287
3 btw17d.n          classif.svm     0.8528736
4 btw17d.n          classif.gbm     0.8629885
5 btw17d.n classif.randomForest     0.8464368
6 btw17d.n        classif.nnet      0.8159770
>
```

```
# Benchmark on tuned values
btw17d.benchmark.tuned <- benchmark(learners = btw17d.learners,
                                    tasks = btw17d.task,
                                    resamplings = btw17d.resample,
                                    measures = acc)
```

Tuned Benchmark

Technische Hochschule Wildau
Technical University of Applied Sciences

2019

```
> btw17d.benchmark.direct
   task.id           learner.id acc.test.mean
1 btw17d.n       classif.rpart     0.8025287
2 btw17d.n      classif.logreg     0.8255172
3 btw17d.n         classif.svm     0.8457471
4 btw17d.n         classif.gbm     0.8490805
5 btw17d.n classif.randomForest     0.8391954
6 btw17d.n       classif.nnet      0.8257471
> btw17d.benchmark.tuned
   task.id           learner.id acc.test.mean
1 btw17d.n       classif.rpart     0.8428736
2 btw17d.n      classif.logreg     0.8225287
3 btw17d.n         classif.svm     0.8528736
4 btw17d.n         classif.gbm     0.8629885
5 btw17d.n classif.randomForest     0.8464368
6 btw17d.n       classif.nnet      0.8159770
```

2020

```
> btw17d.benchmark.direct
   task.id           learner.id acc.test.mean
1 btw17d.n       classif.rpart     0.8594253
2 btw17d.n      classif.logreg     0.8428736
3 btw17d.n         classif.svm     0.8663218
4 btw17d.n         classif.gbm     0.8763218
5 btw17d.n classif.randomForest     0.8695402
6 btw17d.n       classif.nnet      0.8760920
> btw17d.benchmark.tuned
   task.id           learner.id acc.test.mean
1 btw17d.n       classif.rpart     0.8795402
2 btw17d.n      classif.logreg     0.8562069
3 btw17d.n         classif.svm     0.8697701
4 btw17d.n         classif.gbm     0.8762069
5 btw17d.n classif.randomForest     0.8664368
6 btw17d.n       classif.nnet      0.8662069
```

```
# Benchmark on tuned values
btw17d.benchmark.tuned <- benchmark(learners = btw17d.learners,
                                    tasks = btw17d.task,
                                    resamplings = btw17d.resample,
                                    measures = acc)
```

Übersicht

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU

- Maschinelles Lernen in R mit `mlr`
  - Datenvorverarbeitung
  - Aufgabenstellung
  - Lernalgorithmen
  - Modellvalidierung
  - Modelltuning
  - Benchmark
- **Ausblick**
  - **`mlr` und `mlr3` in R**
  - **`scikit-learn` in python**

Hintergrund: `mlr` und `mlr3`

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU

`mlr` und die Weiterentwicklung `mlr3` bieten ein Interface für Maschinelles Lernen in R

- Einbindung der gängigen ML-Algorithmen aus anderen R Paketen
- Workflows zur Analyse mit einheitlicher Datenformatierung z.B. Benchmarks
- Tools zum Tuning der Modelle, zur Visualisierung,…

`mlr`

- `mlr` wurde 2013 das erste Mal auf CRAN vorgestellt.
- `mlr` hat seitdem `caret` als Standard-Interface zum Maschinellen Lernen in R abgelöst
- `mlr` wird seit 2019 nicht weiterentwickelt und erhält nur noch Sicherheits-/Kompatibilitätsupdates

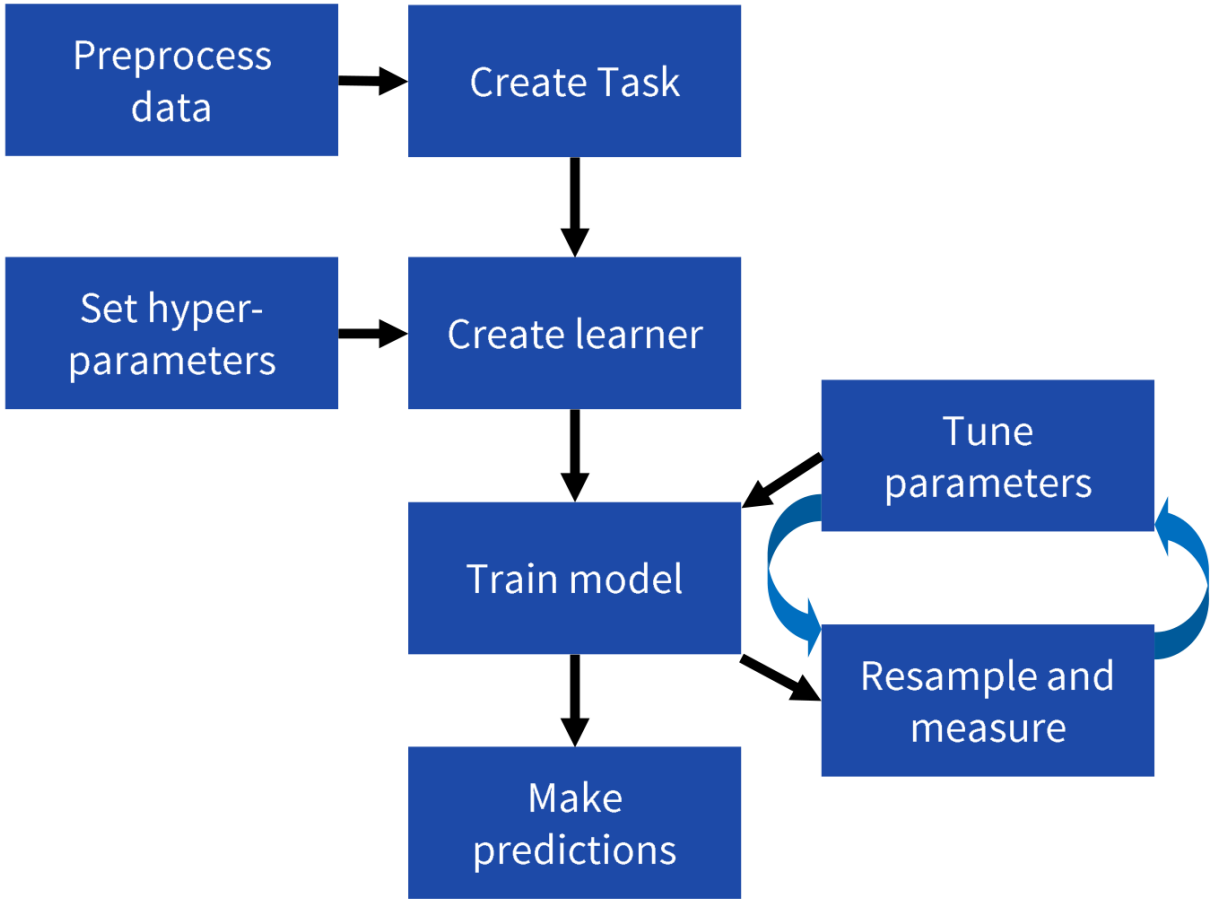
`mlr3`

- Seit 2019 wird an einem neuen Paket `mlr3` gearbeitet:
  https://joss.theoj.org/papers/10.21105/joss.01903
- Basis sind `R6` Klassen und `data.table` Datentabellen
- Idee: Reduktion auf die Kerninhalte, flexible Erweiterungen in eigenen Paketen

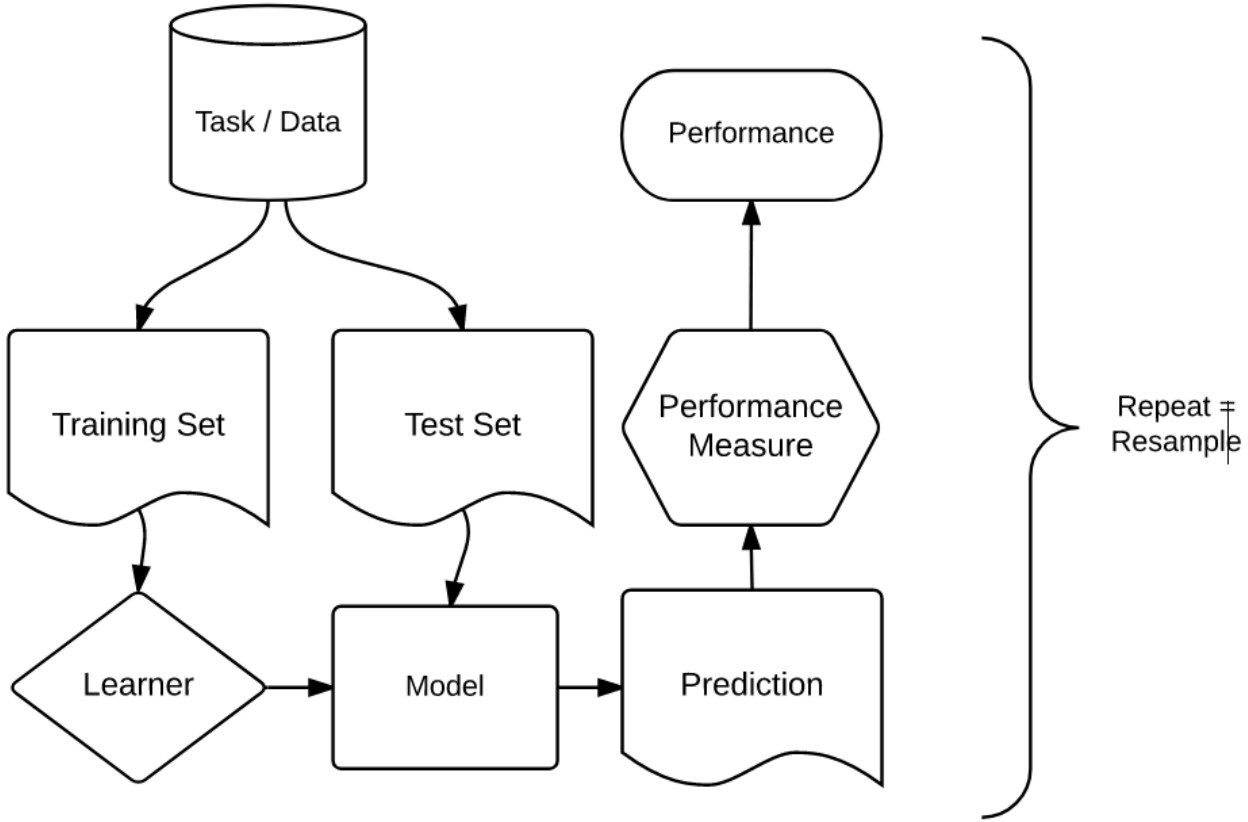
Interface zu Machine Learning in R – `mlr` und `mlr3`

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU

mlr workflow



mlr3 workflow



Maschinelles Lernen in Python

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU

- `scikit-learn` bietet in `python` ein Interface zu ML-Algorithmen
  https://scikit-learn.org
- `scikit-learn` und `mlr` bieten vergleichbare Funktionalitäten
  https://blog.exxactcorp.com/scikitlearn-vs-mlr-for-machine-learning/
- Deutschsprachige Tutorials/Ressourcen für `scikit-learn`
  - https://www.statworx.com/de/blog/data-science-in-python-der-einstieg-in-machine-learning-mit-scikit-learn/
  - https://www.heise.de/developer/artikel/Oliver-Zeigermann-Interaktive-Einfuehrung-in-Machine-Learning-mit-Scikit-Learn-3996819.html
  - https://www.g-webservice.de/python-machine-learning-tutorial/