

Maschinelles Lernen

Maschinelles Lernen in R mit `mlr`

Prof. Dr. Rainer Stollhoff

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

- Die folgenden Übersichten und Dokumentation stammen aus dem *mlr cheatsheet*, Aaron Cooley, <https://posit.co/wp-content/uploads/2022/10/mlr.pdf>
- Der Code wurde mit Hilfe des mlr Package erstellt: <https://CRAN.R-project.org/package=mlr>, Lang, M., et al. (2019). mlr3: A modern object-oriented machine learning framework in R. Journal of Open Source Software, <https://joss.theoj.org/papers/10.21105/joss.01903>

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

Hintergrund: `mlr`

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

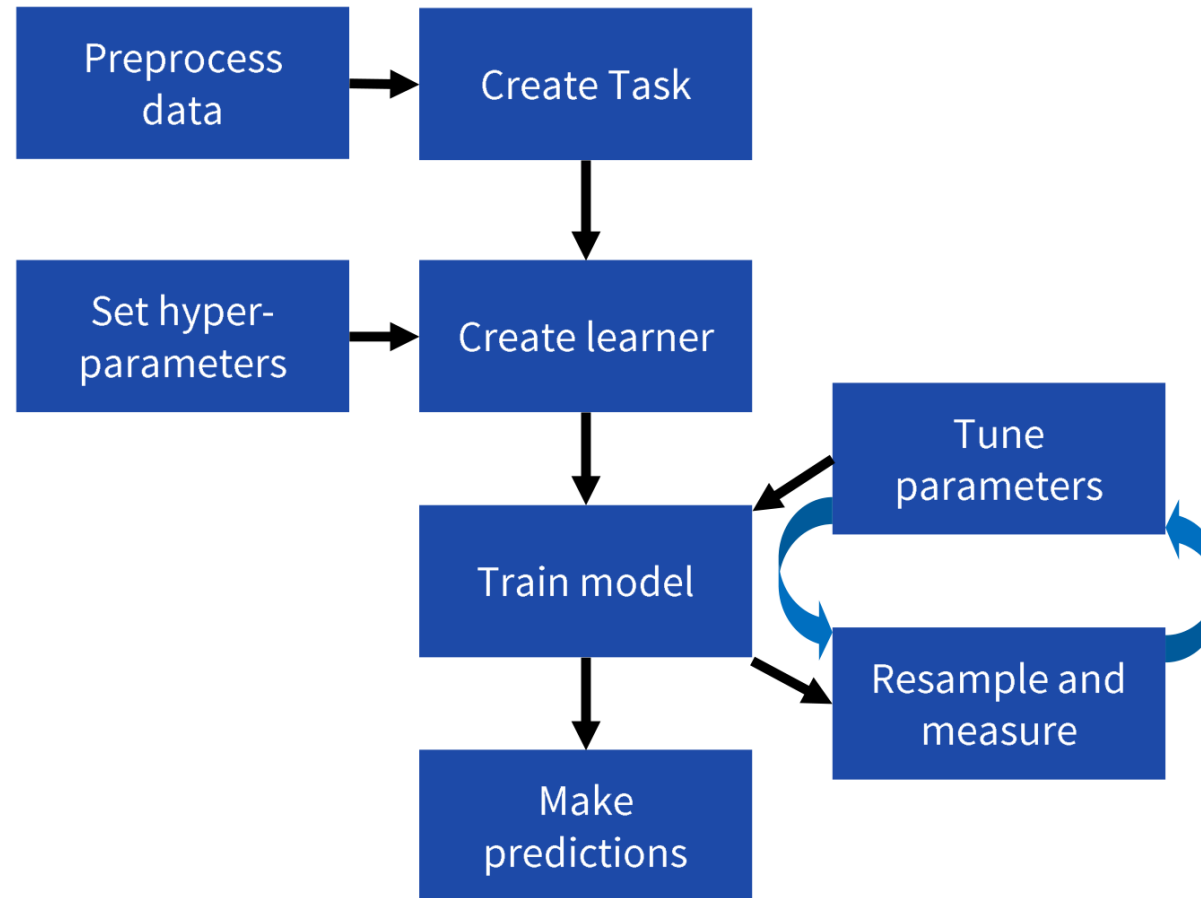
- Einbindung der gängigen ML-Algorithmen aus anderen R Paketen
- Workflows zur Analyse mit einheitlicher Datenformatierung z.B. Benchmarks
- Tools zum Tuning der Modelle, zur Visualisierung,...

`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`

mlr workflow



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)`)

*mlr cheatsheet, Aaron Cooley,
<https://posit.co/wp-content/uploads/2022/10/mlr.pdf>*

```
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 - 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"

"Adabag Boosting"
"Random forest based on conditional inference trees"
"GLM with Lasso or Elasticnet Regularization (Cross Validated Lambda)"
"Flexible Discriminant Analysis"
"Extremely Randomized Trees"
"Fast k-Nearest Neighbour"
"Gradient Boosting Machine"
"GLM with Lasso or Elasticnet Regularization"
"h2o.gbm"
"k-Nearest Neighbours"
"Propositional Rule Learner"
"k-Nearest Neighbor"
"Linear Discriminant Analysis"
"L1-Regularized Logistic Regression"
"L2-Regularized Logistic Regression"
"Support Vector Classification by Crammer and Singer"
"Least Squares Support Vector Machine"
"Mixture Discriminant Analysis"
"Multinomial Regression"
"Neural Network"
"1-K Classifier"
"Quadratic Discriminant Analysis"
"Random Forest"
"Random Forests"
"Random forests"
"Decision Tree"
"Robust Regularized Linear Discriminant Analysis"
"Shrinkage Discriminant Analysis"
"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.

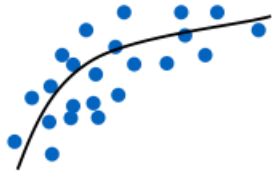
`makeResampleInstance(desc=, task=)` can reduce noise by ensuring the resampling is done identically every time.

`resample(learner=, task=, resampling=, measures=)`

Train and test model according to specified resampling strategy.

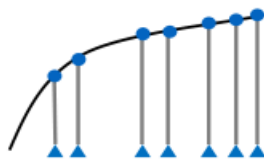
Trainieren, Vorhersagen und Testen

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=)`

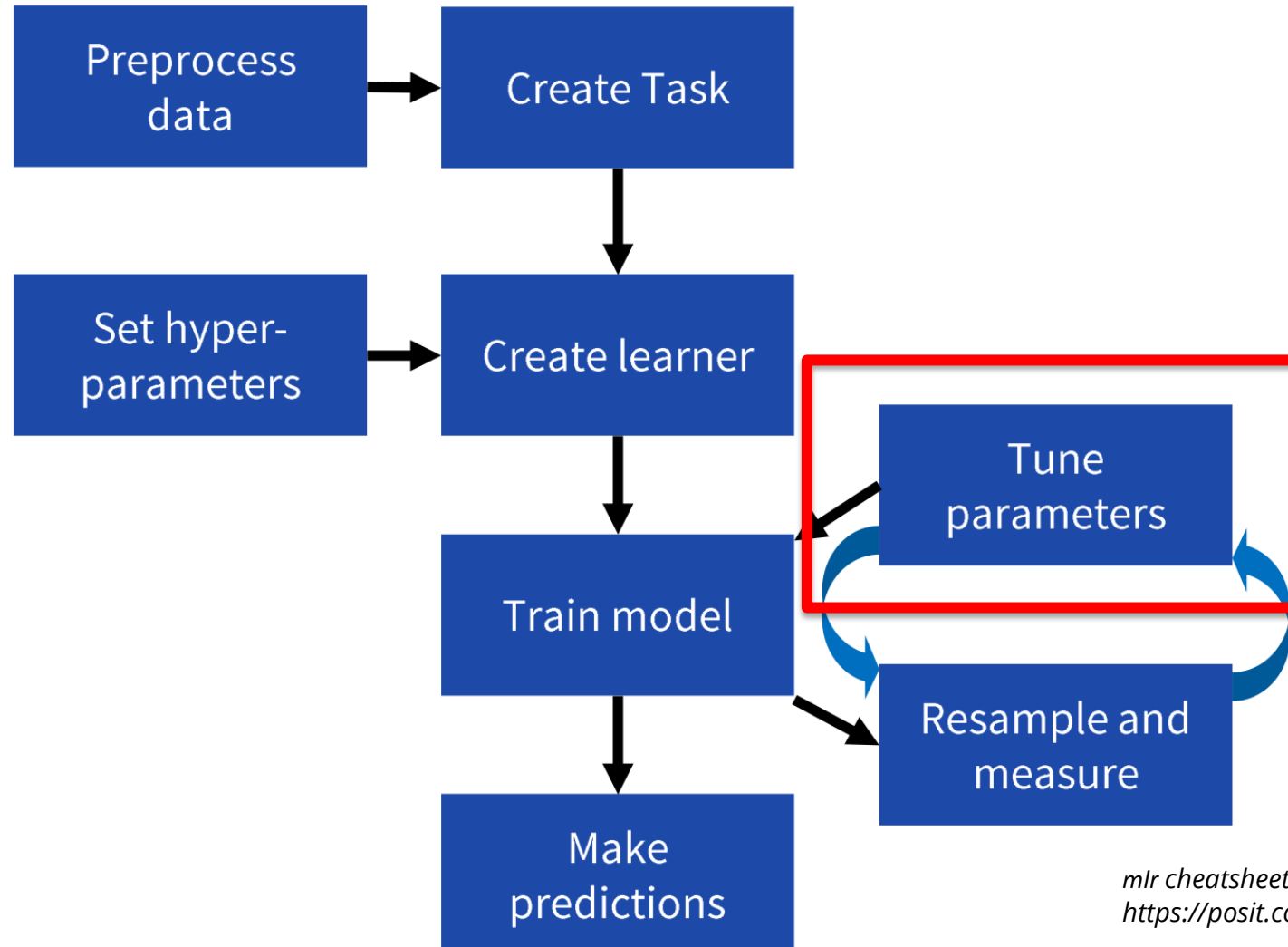
Ergebnisse Benchmark

```
> 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)
```

mlr workflow



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 cheatsheet, Aaron Cooley,
<https://posit.co/wp-content/uploads/2022/10/mlr.pdf>*

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`

*mlr cheatsheet, Aaron Cooley,
<https://posit.co/wp-content/uploads/2022/10/mlr.pdf>*

```
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` = Reichweite der RBF
- `kernel` = Radiale Basis Funktion (RBF)

mlr – tuning gbm

```
# 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

```
> 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

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
```

Benchmark on tuned values

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

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
```

- 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`

`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`

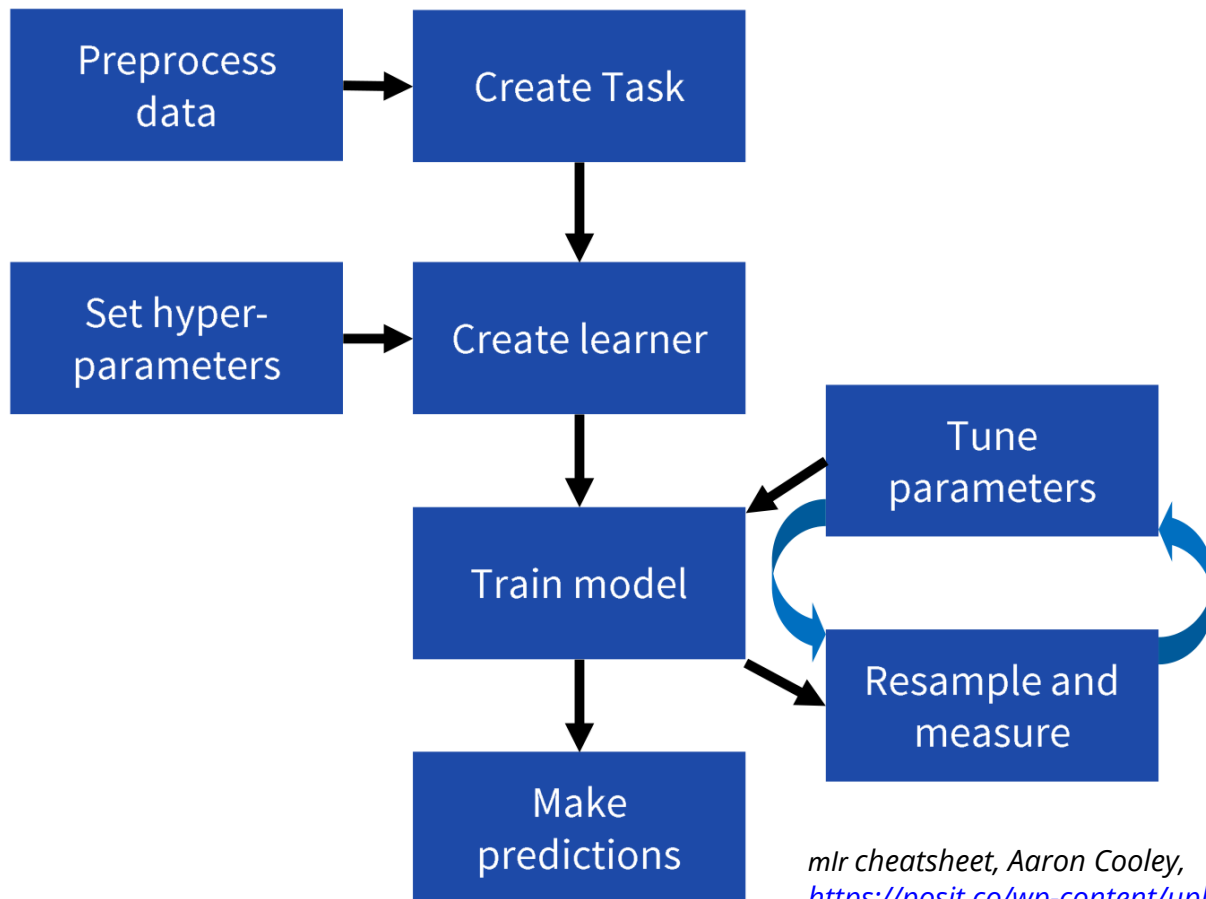
- `mlr` wurde 2013 das erste Mal auf CRAN vorgestellt.
- `mlr` hat seitdem `caret` als Standard-Interface zum Maschinellen Lernen in R abgelöst
- `mlr` wird seit 2019 nicht weiterentwickelt und erhält nur noch Sicherheits-/Kompatibilitätsupdates

`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

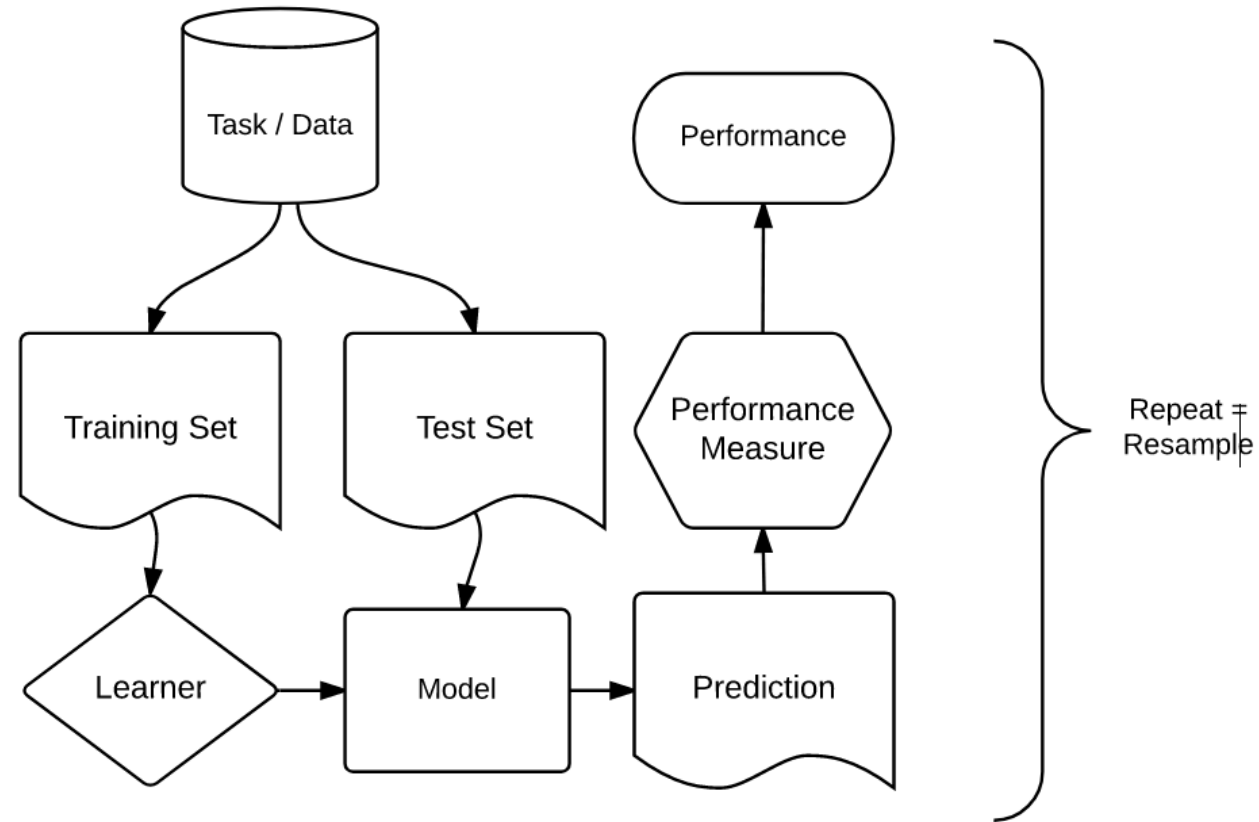
Interface zu Machine Learning in R – `mlr` und `mlr3`

mlr workflow



mlr cheatsheet, Aaron Cooley,
<https://posit.co/wp-content/uploads/2022/10/mlr.pdf>

mlr3 workflow



Bischl, B., et al., Flexible and Robust Machine Learning Using mlr3 in R,
<https://mlr3book.ml-org.com>

Maschinelles Lernen in Python

- `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/>