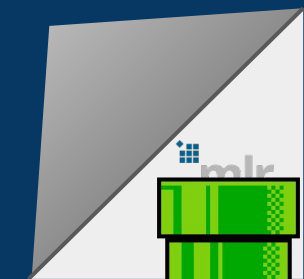
mlr3pipelines

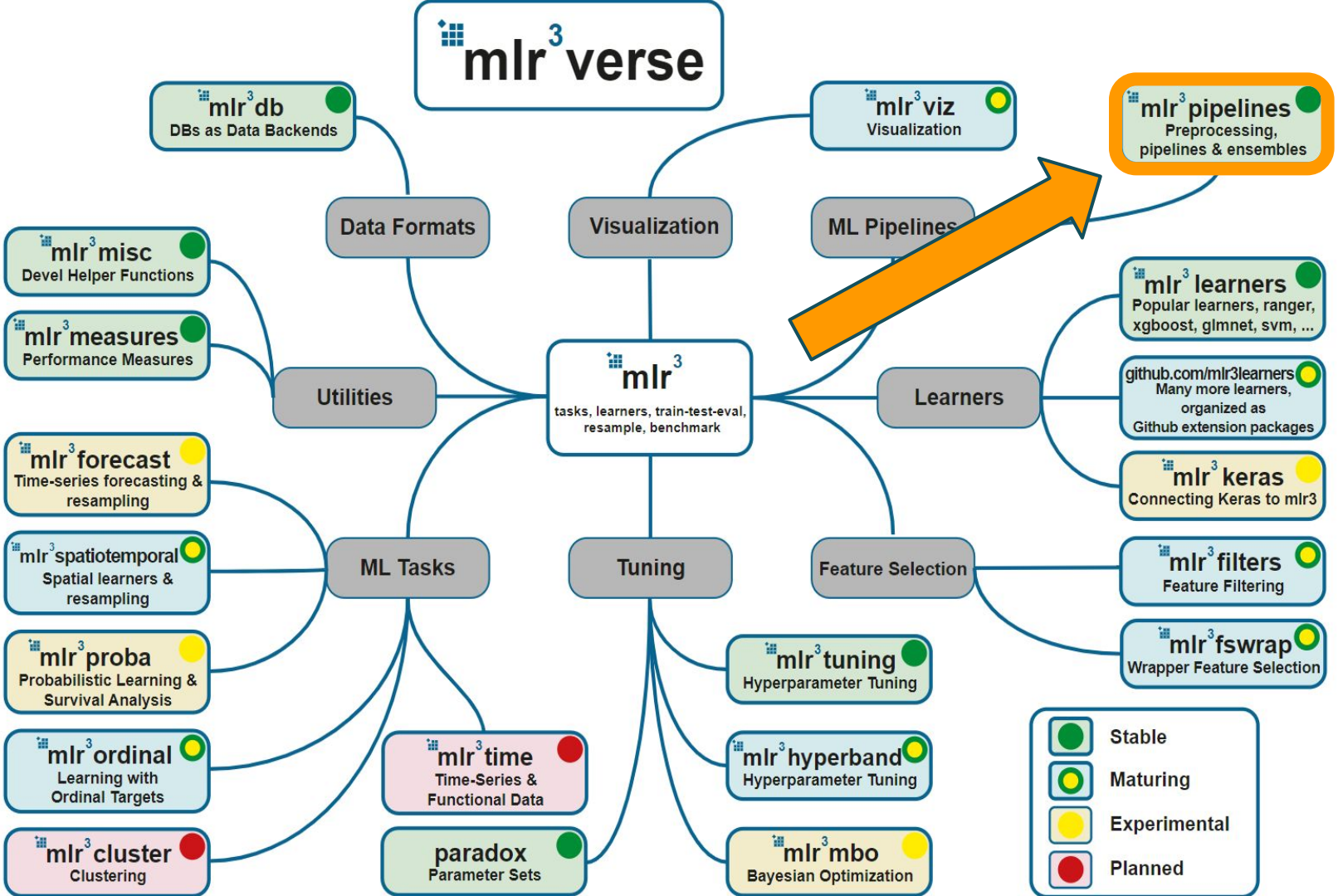
łączenie operacji uczenia maszynowego w potoki

Andrzej Nowikowski, Mateusz Bąkała

czym jest mlr3?

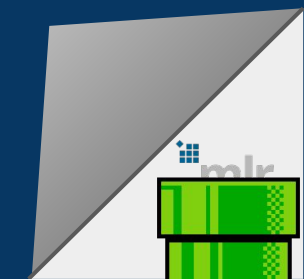
- framework do machine learningu w R
- zbudowany na systemie klas R6
- zawiera tylko kluczowe funkcjonalności (taski, modele, trenowanie, predykcja, resampling)
- rozszerzany przez pod-paczki




mlr3verse
mlr3db
DBs as Data Backends
mlr3viz
Visualization
mlr3pipelines
Preprocessing, pipelines & ensembles
Data Formats
Visualization
ML Pipelines
mlr3misc
Devel Helper Functions
mlr3measures
Performance Measures
Utilities
mlr3
tasks, learners, train-test-eval, resample, benchmark
Learners
mlr3learners
Popular learners, ranger, xgboost, glmnet, svm, ...
github.com/mlr3learners
Many more learners, organized as Github extension packages
mlr3keras
Connecting Keras to mlr3
mlr3forecast
Time-series forecasting & resampling
mlr3spatiotemporal
Spatial learners & resampling
mlr3proba
Probabilistic Learning & Survival Analysis
mlr3ordinal
Learning with Ordinal Targets
mlr3cluster
Clustering
ML Tasks
Tuning
Feature Selection
mlr3filters
Feature Filtering
mlr3fswrap
Wrapper Feature Selection
mlr3time
Time-Series & Functional Data
paradox
Parameter Sets
mlr3tuning
Hyperparameter Tuning
mlr3hyperband
Hyperparameter Tuning
mlr3mbo
Bayesian Optimization
Stable
Maturing
Experimental
Planned

dlaczego mlr3 nie wystarcza?

- ➔ proste tworzenie tasków i modeli
- ➔ problem przy automatyzacji przetwarzania danych (np. przy preprocessingu)
- ➔ konieczność przebudowy kodu przy zmianie modelu
- ➔ brak wbudowanej agregacji predykcji

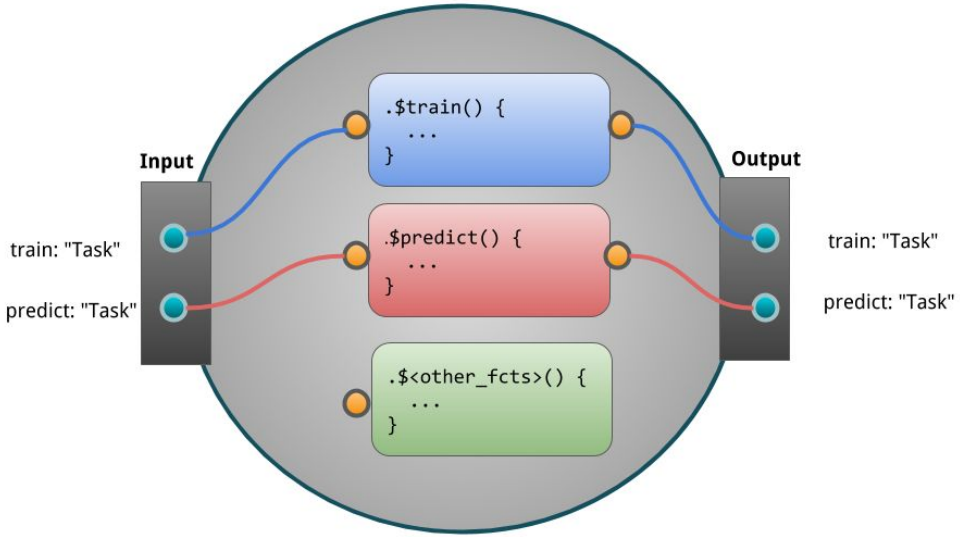


do czego służy mlr3pipelines?

➔ framework do projektowania przepływu danych pomiędzy komponentami z mlr3

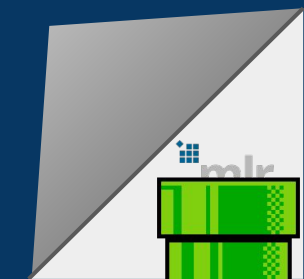
klasa PipeOp

- ➔ reprezentuje przetworzenie danych wejściowych w jakiś konkretny sposób



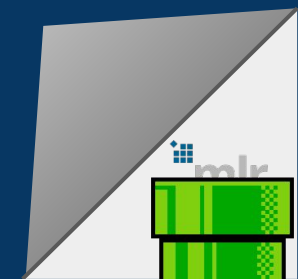
klasa PipeOp (na przykładzie PCA)

```
library(mlr3)
library(mlr3pipelines)

> task = TaskClassif$new("iris", as_data_backend(iris), "Species")
> po = mlr_pipeops$get("pca")
> po$state
NULL
```



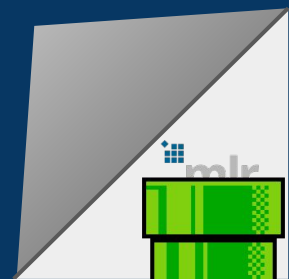
klasa PipeOp (na przykładzie PCA)

```
> po$train(list(task))[[1]]$data()
      Species       PC1         PC2         PC3          PC4
  1:    setosa -2.684126  0.31939725 -0.02791483 -0.002262437
 ---
150: virginica  1.390189 -0.28266094  0.36290965  0.155038628
> po$state
Standard deviations (1, .., p=4):
[1] 2.0562689 0.4926162 0.2796596 0.1543862

Rotation (n x k) = (4 x 4):
...
```



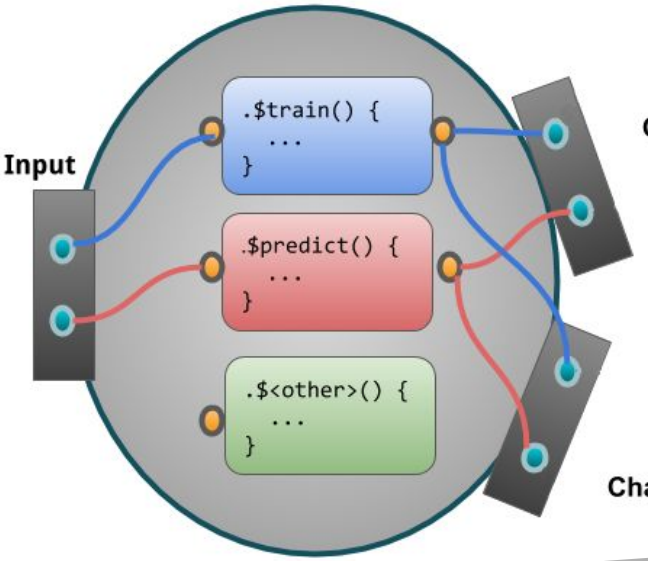
dostępne PipeOpy

```
> mlr_pipeops$keys()
 [1] "boxcox"          "branch"          "chunk"           "classbalancing"  "classifavg"      "classweights"
 [7] "colapply"        "collapsefactors" "copy"            "encode"          "encodeimpact"    "encodelmer"
[13] "featureunion"    "filter"          "fixfactors"      "histbin"         "ica"             "imputehist"
[19] "imputemean"      "imputemedian"    "imputenewlvl"    "imputesample"    "kernelpca"       "learner"
[25] "learner_cv"      "missind"         "modelmatrix"     "mutate"          "nop"             "pca"
[31] "quantilebin"     "regravg"         "removeconstants" "scale"           "scalemaxabs"     "scalerange"
[37] "select"          "smote"           "spatialsign"     "subsample"       "unbranch"        "yeojohnson"
> mlr_pipeops$get("pca", param_vals = list(rank. = 3))
PipeOp: <pca> (not trained)
values: <rank.=3>
```

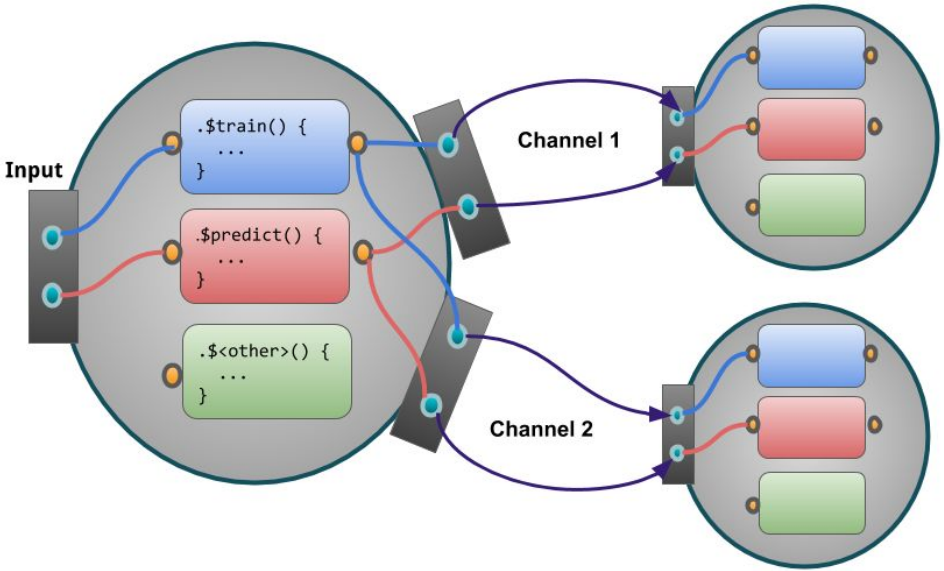


PipeOp – kanały

- ➔ jak w funkcji – liczba parametrów (inputów) zależy od operatora
- ➔ inaczej niż w funkcji – zwraca wiele wyników (w postaci listy)



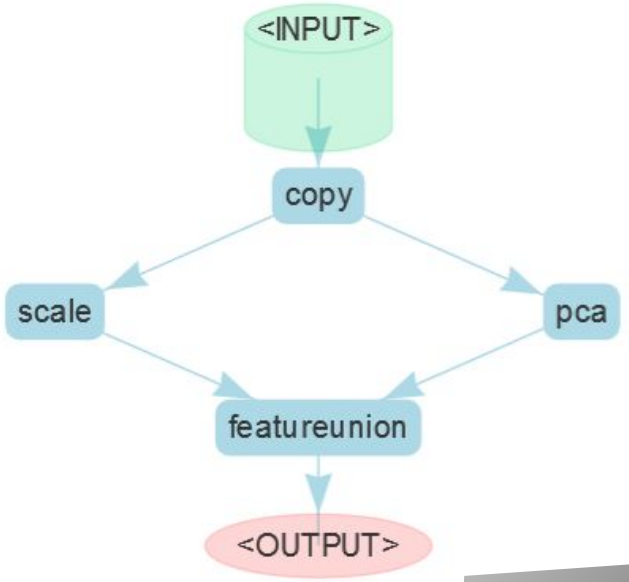
klasa Graph



- graf skierowany
- wierzchołki – PipeOpy
- krawędzie – przepływ danych
- przepływ danych odbywa się przez kanały, nie przez PipeOpy

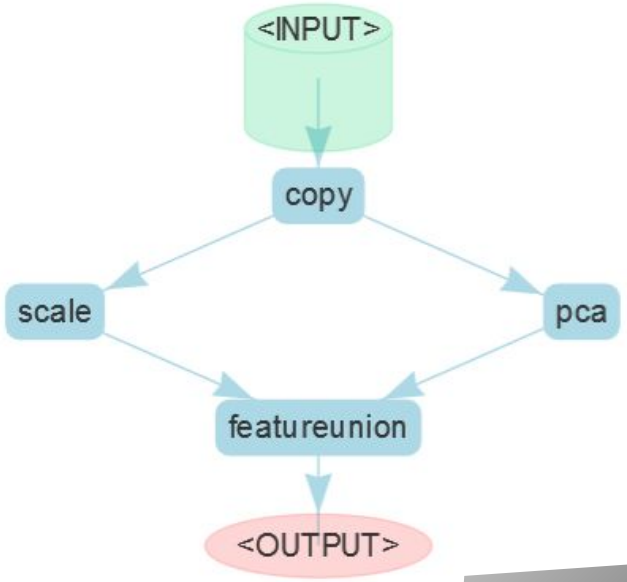
klasa Graph

```
gr = Graph$new()$
  add_pipeop(mlr_pipeops$get("copy", outnum = 2))$
  add_pipeop(mlr_pipeops$get("scale"))$
  add_pipeop(mlr_pipeops$get("pca"))$
  add_pipeop(mlr_pipeops$get("featureunion", innum = 2))

gr$
  add_edge("copy", "scale", src_channel = 1)$
  add_edge("copy", "pca", src_channel = "output2")$
  add_edge("scale", "featureunion", dst_channel = 1)$
  add_edge("pca", "featureunion", dst_channel = 2)
```



klasa Graph - operator %>>%

```
gr = mlr_pipeops$get("copy", outnum = 2) %>>%
  gunion(list(mlr_pipeops$get("scale"),
              mlr_pipeops$get("pca"))) %>>%
  mlr_pipeops$get("featureunion", innum = 2)
```



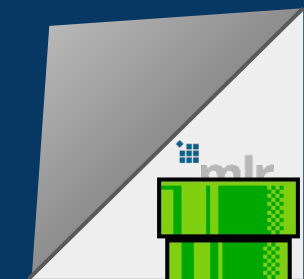
powrót do mlr3

➔ PipeOpLearner

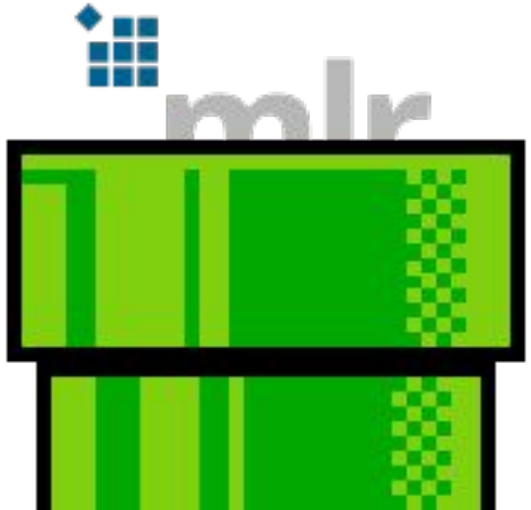
```
> gr = mlr_pipeops$get("pca") %>>% mlr_pipeops$get("learner", mlr_learners$get("classif.rpart"))
> gr$train(task)
$`classif.rpart.output`
NULL
> gr$predict(task)
$`classif.rpart.output`
<PredictionClassif> for 150 observations:
```

➔ GraphOpLearner

```
> lrngrph = GraphLearner$new(gr)
> resample(task, lrngrph, rsmp)
<ResampleResult> of 1 iterations
* Task: iris
* Learner: pca.classif.rpart
```



dziękujemy za uwagę



przykładowy graf (jak na moje do usunięcia)