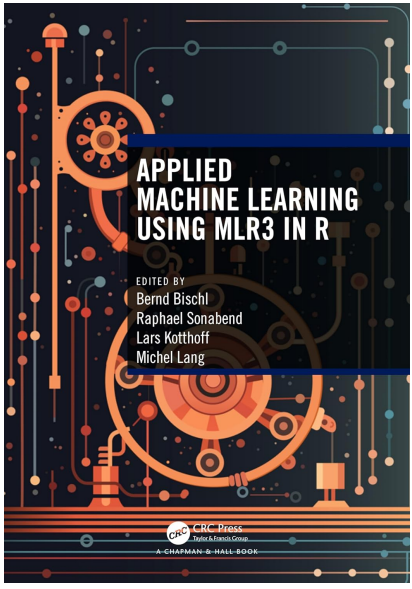
mlr3 pipelines

Lars Kotthoff
Electrical Engineering and Computer Science, School of Computing
University of Wyoming
larsko@uwyo.edu

Open Source Quantitative Finance Conference, 11 April 2025



https://mlr3book.mlr-org.com/

```r
library(mlr3verse)

tasks = tsks(c("breast_cancer", "sonar"))

glrn_rf_tuned = as_learner(ppl("robustify") %>>% auto_tuner(
    tnr("grid_search", resolution = 5),
    lrn("classif.ranger", num.trees = to_tune(200, 500)),
    rsmp("holdout")
))
glrn_rf_tuned$id = "RF"

glrn_stack = as_learner(ppl("robustify") %>>% ppl("stacking",
    lrns(c("classif.rpart", "classif.kknn")),
    lrn("classif.log_reg")
))
glrn_stack$id = "Stack"

learners = c(glrn_rf_tuned, glrn_stack)
bmr = benchmark(benchmark_grid(tasks, learners, rsmp("cv", folds = 3)))

bmr$aggregate(msr("classif.acc"))
```

	nr	task_id	learner_id	resampling_id	iters	classif.acc
	<int>	<char>	<char>	<char>	<int>	<num>
1:	1	breast_cancer	RF	cv	3	0.9677912
2:	2	breast_cancer	Stack	cv	3	0.9164992
3:	3	sonar	RF	cv	3	0.8267081
4:	4	sonar	Stack	cv	3	0.7162871

Pipe Operators

- ▷ PipeOp in mlr3-speak
- ▷ `$train()` and a `$predict()` method
- ▷ `$param_set` field that defines the hyperparameters
- ▷ constructed with the `po()` function

```
library(mlr3verse)
po_pca = po("pca", center = TRUE)

PipeOp: <pca> (not trained)
values: <center=TRUE>
Input channels <name [train type, predict type]>:
  input [Task,Task]
Output channels <name [train type, predict type]>:
  output [Task,Task]
```

Pipe Operators Train

```
po_pca$train(tsks("boston_housing"))

$output
<TaskRegr:boston_housing> (506 x 18): Boston Housing Prices
* Target: cmedv
* Properties: -
* Features (17):
  - dbl (15): PC1, PC10, PC11, PC12, PC13, PC14, PC15, PC2, PC3, PC4,
    PC5, PC6, PC7, PC8, PC9
  - fct (2): chas, town
```

Pipe Operators State

```
po_pca$state
```

```
Standard deviations (1, .., p=15):
 [1] 1.387078e+03 1.080930e+02 7.750918e+01 2.807860e+01 1.630019e+01
 [6] 6.907191e+00 5.276367e+00 3.876985e+00 2.915733e+00 1.762825e+00
[11] 1.083099e+00 5.033412e-01 6.944262e-02 4.966327e-02 3.862021e-02

Rotation (n x k) = (15 x 15):
                  PC1           PC2           PC3          PC4           PC5
age      9.928068e-03  0.0548286996 -7.652884e-03  0.765442047 -6.282490e-01
b       -2.423736e-02 -0.4607717741 -8.866967e-01  0.022865175 -4.174316e-03
crim     3.405922e-03  0.0218600096  5.015232e-03  0.004508408 -3.219726e-02
dis     -7.572546e-04 -0.0046940019  6.983044e-04 -0.045481910 -3.178385e-03
indus    2.864462e-03  0.0268024509 -9.875540e-03  0.093219299  1.615220e-02
lat      9.848547e-06 -0.0003154347  1.218493e-04  0.000247237 -6.237778e-05
...
```

Pipe Operators Predict

```
po_pca$predict(tsks("boston_housing"))[[1]]$data()
```

```
            cmedv    chas        town       PC1        PC2       PC3         PC4
    <num> <fctr>   <fctr>      <num>     <num>      <num>     <num>       <num>
 1:  24.0      0   Nahant   673.7945 -185.6437   32.41160 -6.373605 -6.98056400
 2:  21.6      0 Swampscott 658.8871 -230.9396   57.05639 18.892977 -3.13859100
 3:  34.7      0 Swampscott 657.8002 -230.0266   60.75298  4.717087  8.29586700
 4:  33.4      0 Marblehead 646.6991 -248.5129   68.32003 -6.517082 17.37516500
 5:  36.2      0 Marblehead 645.7392 -248.9604   66.23151  0.186506 11.97645700
---
```

Graphs

- collection of pipeline operators that define individual steps in data science pipelines
- "edges" guide the flow of data
- connect pipeline operators with the %>>% operator

```
po_scale = po("scale")
graph = po_pca %>>% po_scale

Graph with 2 PipeOps:
     ID         State sccssors prdcssors
 <char>        <char>   <char>    <char>
    pca      <prcomp>    scale
  scale <<UNTRAINED>>                pca
```

Graphs Training

```
graph$train(tsk("boston_housing"))

$scale.output
<TaskRegr:boston_housing> (506 x 18): Boston Housing Prices
* Target: cmedv
* Properties: -
* Features (17):
  - dbl (15): PC1, PC10, PC11, PC12, PC13, PC14, PC15, PC2, PC3, PC4,
    PC5, PC6, PC7, PC8, PC9
  - fct (2): chas, town
```

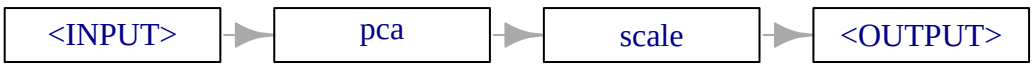
Graphs Predict

```
graph$predict(tsk("boston_housing"))$scale.output$data()
```

```
      cmedv   chas        town        PC1         PC10         PC11         PC12
     <num> <fctr>      <fctr>      <num>        <num>        <num>        <num>
  1:  24.0      0      Nahant 0.4857654 -1.75305179  0.04865207  0.5157509
  2:  21.6      0 Swampscott 0.4750181 -0.25008756 -1.38250786 -0.1003841
  3:  34.7      0 Swampscott 0.4742345 -0.33275540 -1.03983952 -1.1393915
  4:  33.4      0 Marblehead 0.4662313  0.06778422 -1.07052858 -0.7431648
  5:  36.2      0 Marblehead 0.4655392  0.09450762 -1.20760905 -1.2565375
 ---
```

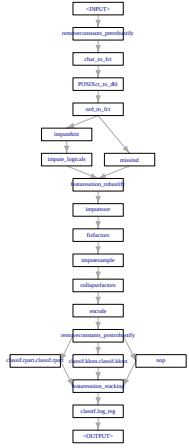
Plotting Graphs

```
graph$plot(horizontal = TRUE)
```



Graphs as Learners

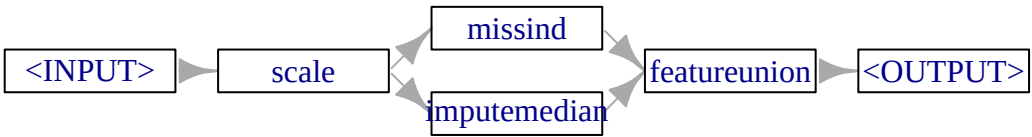
```
glrn_stack = as_learner(ppl("robustify") %>>% ppl("stacking",
    lrns(c("classif.rpart", "classif.kknn")),
    lrn("classif.log_reg")
))
```



Non-Sequential Pipelines

- non-sequential pipelines can perform more complex operations
- `gunion()` can combine multiple pipe operators, graphs, or mixture, into parallel graph

```
graph = po("scale", center = TRUE, scale = FALSE) %>>%
  gunion(list(
    po("missind"),
    po("imputemedian")
  )) %>>%
  po("featureunion")
```



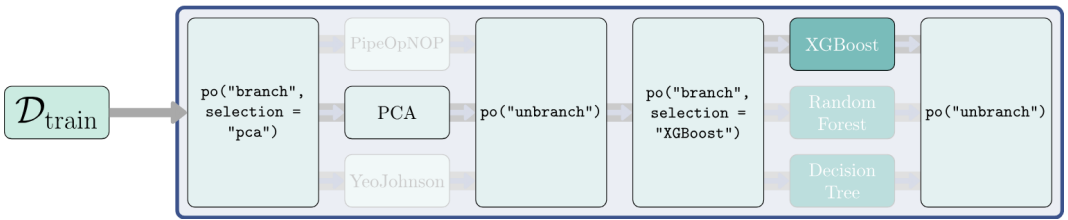
Common Pipeline Patterns

- pipelines useful for more than one problem – `ppl()` operator to easily construct (partial) pipelines from patterns
- `ppl("bagging", graph)` creates a bagging ensemble
- `ppl("branch", graphs)` creates a branch
- `ppl("robustify")` common preprocessing steps
- `ppl("stacking", base_learners, super_learner)` creates a stacking ensemble

```
glrn_stack = as_learner(ppl("robustify") %>>% ppl("stacking",
    lrns(c("classif.rpart", "classif.kknn")),
    lrn("classif.log_reg")
))
```

Branching

- ▷ `po("branch")` creates alternative paths where data flow is determined by selection hyperparameter
- ▷ easy combined model selection and hyperparameter tuning



Preprocessing – Robustify

1. `po("removeconstants")` removes constant values
2. `po("colapply")` encodes character and ordinal features as categorical, date/time features as numeric
3. `po("imputehist")` imputes numeric features by histogram sampling
4. `po("imputesample")` imputes logical features by sampling from the empirical distribution
5. `po("missind")` adds missing data indicators for imputed numeric and logical variables
6. `po("imputeoor")` encodes missing values of categorical features with a new level
7. `po("fixfactors")` ensures that the same levels are present during training and prediction
8. `po("imputesample")` imputes any introduced missing values
9. `po("collapsefactors")` collapses rare factor levels
10. `po("encode")` one-hot encodes categorical features
11. `po("removeconstants")` removes constant values

Hyperparameters

- hyperparameters are collected in `$param_set` for convenient tuning

```
graph = po("scale", center = FALSE, scale = TRUE, id = "scale") %>>%
  po("scale", center = TRUE, scale = FALSE, id = "center") %>>%
  lrn("classif.rpart", cp = 1)
unlist(graph$param_set$values)
```

```
      scale.center        scale.scale       scale.robust      center.center
                 0                  1                  0                  1
      center.scale      center.robust   classif.rpart.cp classif.rpart.xval
                 0                  0                  1                  0
```

Tuning Hyperparameters

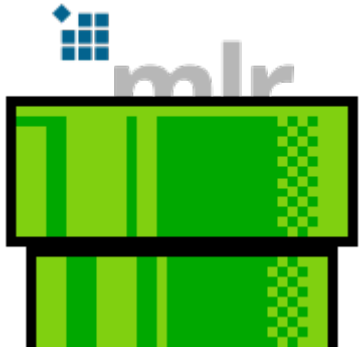
```
graph_learner = as_learner(po("pca") %>>% lrn("classif.kknn"))

lrn_knn = lrn("classif.kknn", k = to_tune(1, 32))
po_pca = po("pca", rank. = to_tune(2, 20))
graph_learner = as_learner(po_pca %>>% lrn_knn)

glrn_tuned = auto_tuner(tnr("mbo"), graph_learner, rsmp("holdout"),
        term_evals = 10)
glrn_untuned = po("pca") %>>% lrn("classif.kknn")
design = benchmark_grid(tsk("sonar"), c(glrn_tuned, glrn_untuned),
        rsmp("cv", folds = 5))
benchmark(design)$aggregate()[, .(learner_id, classif.ce)]
```

```
                learner_id classif.ce
                    <char>      <num>
1: pca.classif.kknn.tuned  0.1681765
2:       pca.classif.kknn  0.2257840
```

https://mlr-org.com/
https://mlr3pipelines.mlr-org.com/
https://mlr3book.mlr-org.com/



Exercises

`https://shorturl.at/9Htr3`
`https://mlr3book.mlr-org.com/`
Exercises for Chapters 7, 8, 9