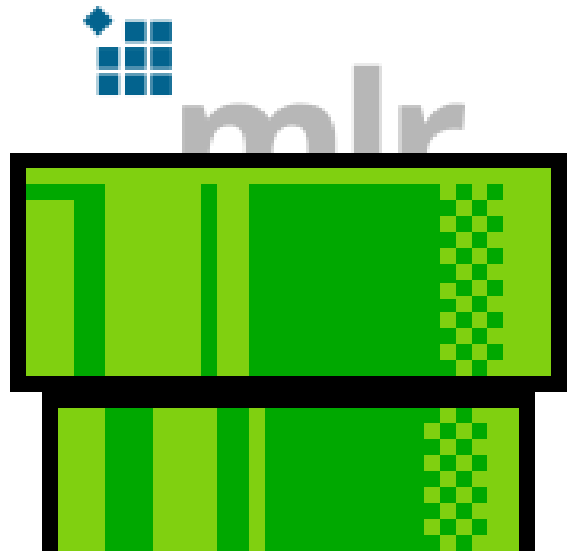
mlr3pipelines: Machine Learning Pipelines as Graphs

https://bit.ly/32jDQOG

Martin Binder, Florian Pfisterer, Michel Lang, **Bernd Bischl**

Computational Statistics, LMU



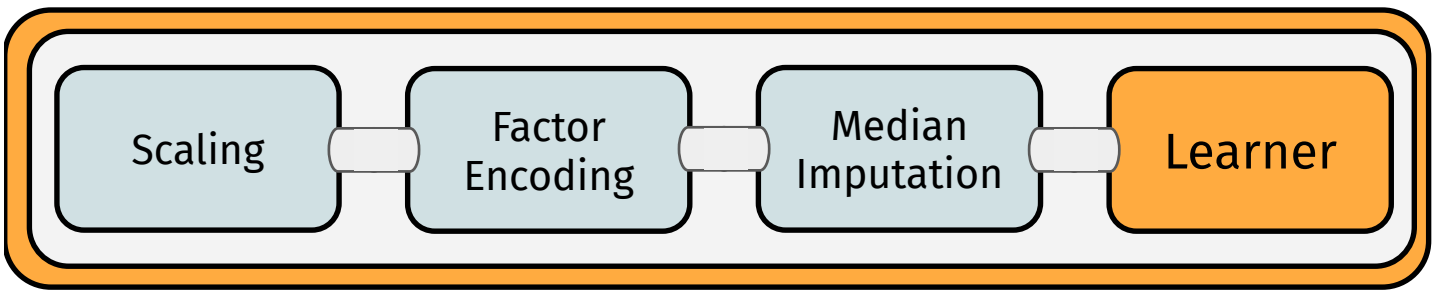
mlr3pipelines

Machine Learning Workflows:

- **Preprocessing**: Feature extraction, feature selection, missing data imputation,...
- **Ensemble methods**: Model averaging, model stacking
- `mlr3`: modular model fitting

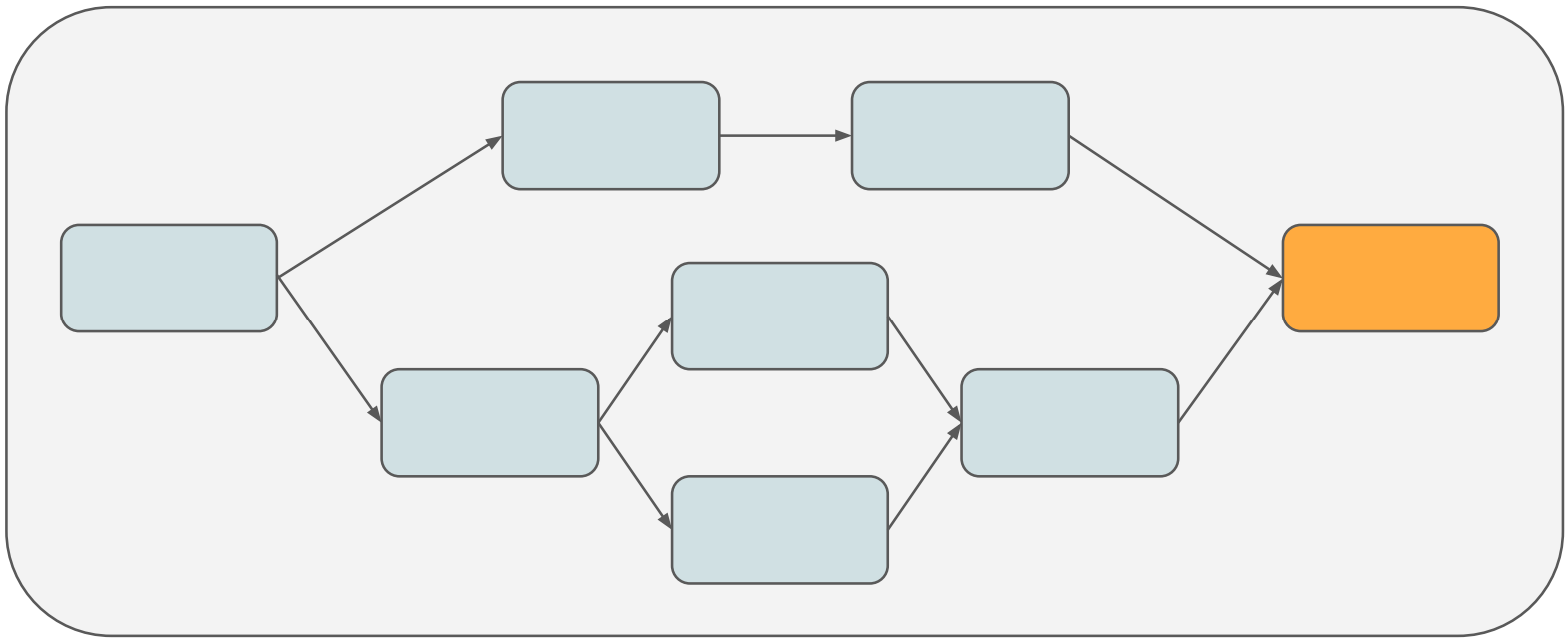
⇒ **`mlr3pipelines`**: modular ML workflows

(replaces `mlr2`'s `mlrCPO` and most "wrappers")



Machine Learning Workflows

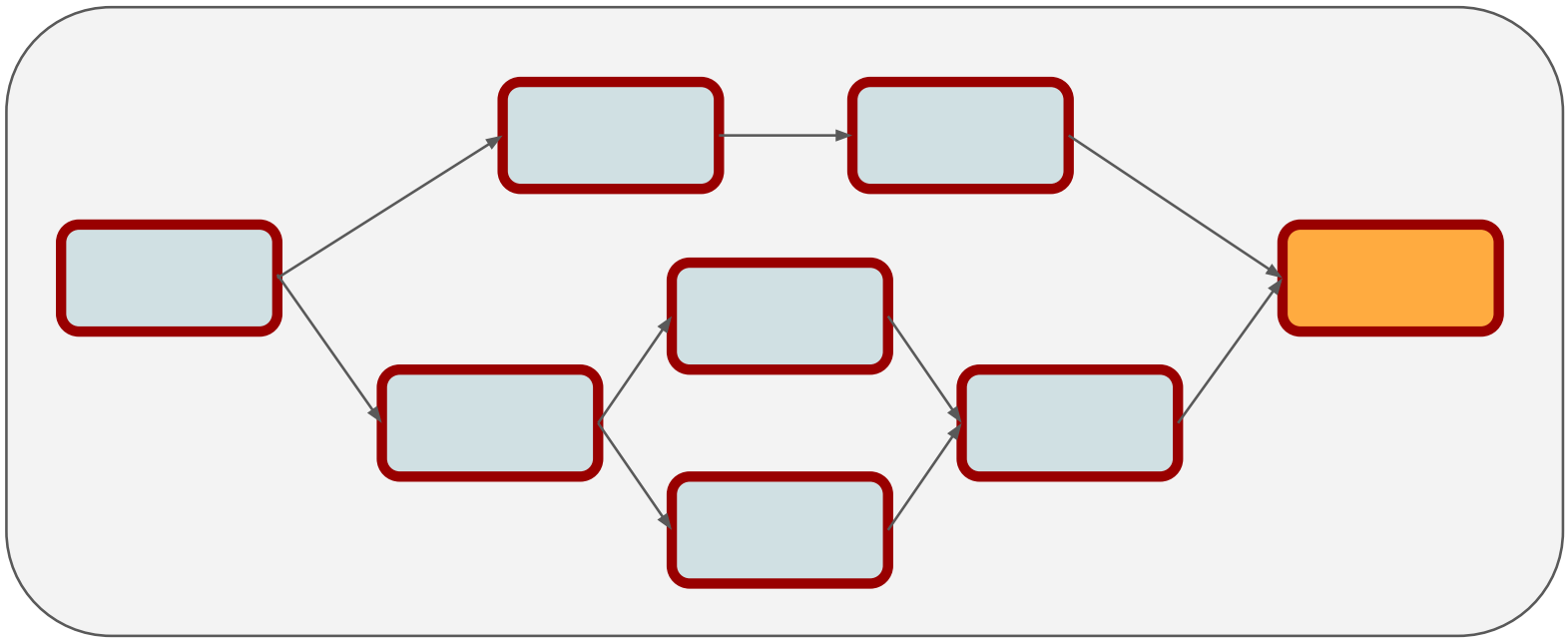
– what do they look like?



Machine Learning Workflows

– what do they look like?

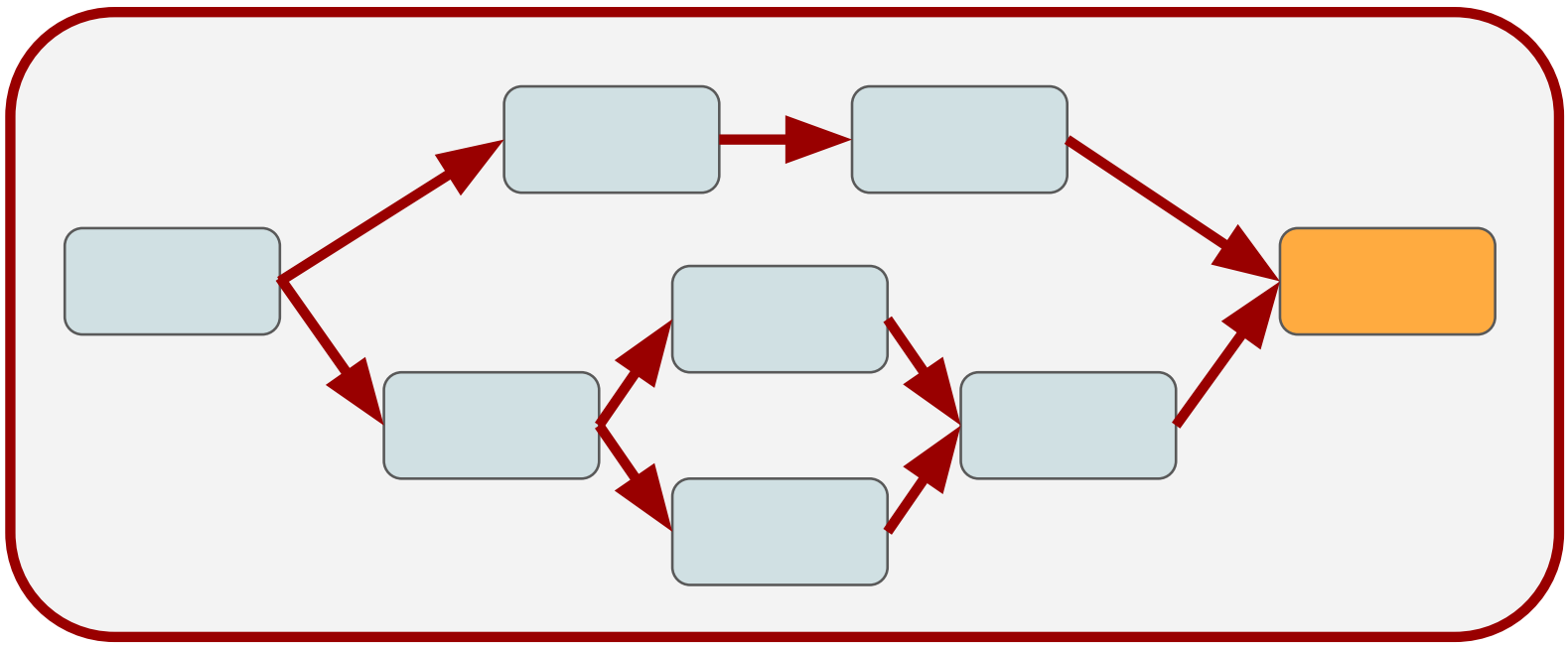
- **Building blocks**: *what* is happening? → `PipeOp`



Machine Learning Workflows

– what do they look like?

- **Building blocks**: *what* is happening? → `PipeOp`
- **Structure**: In what *sequence* is it happening? → `Graph`

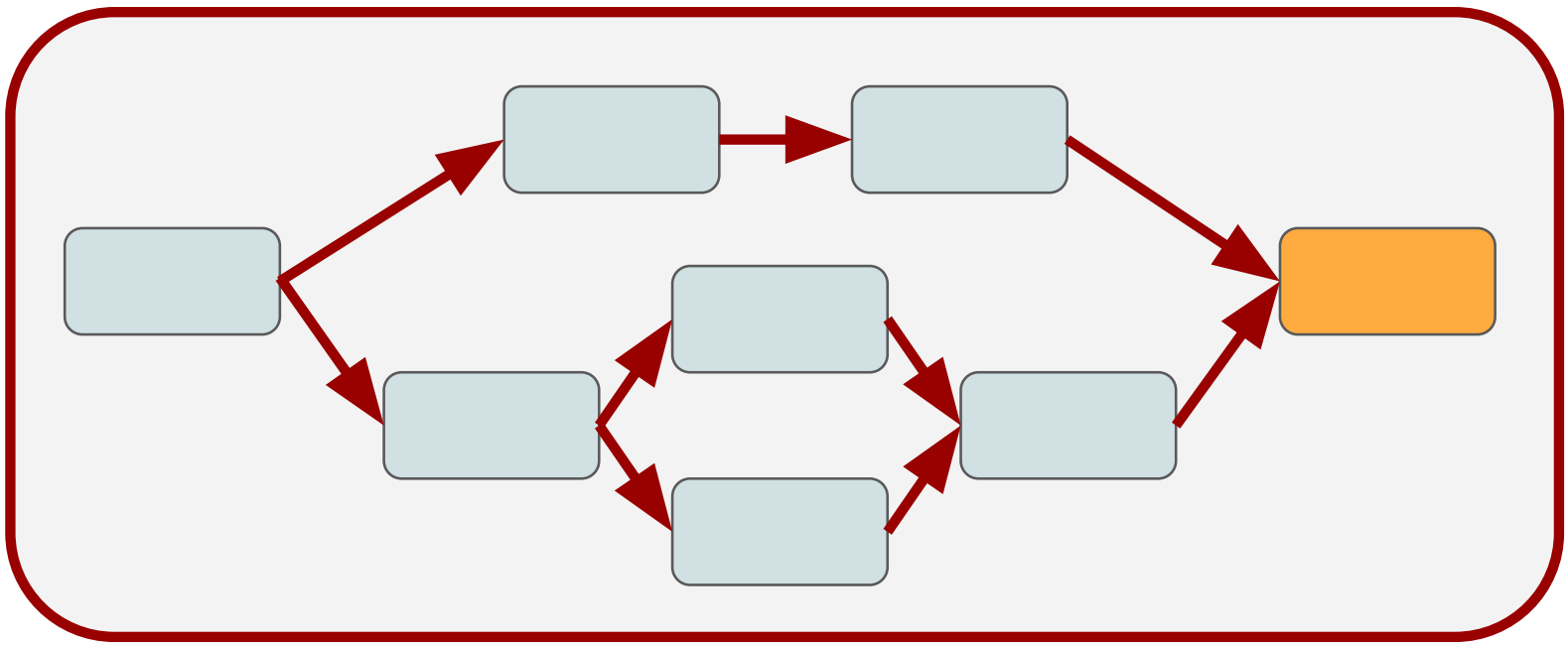


Machine Learning Workflows

– what do they look like?

- **Building blocks**: *what* is happening? → `PipeOp`
- **Structure**: In what *sequence* is it happening? → `Graph`

⇒ `Graph`: `PipeOps` as **nodes** with **edges** (data flow) between them

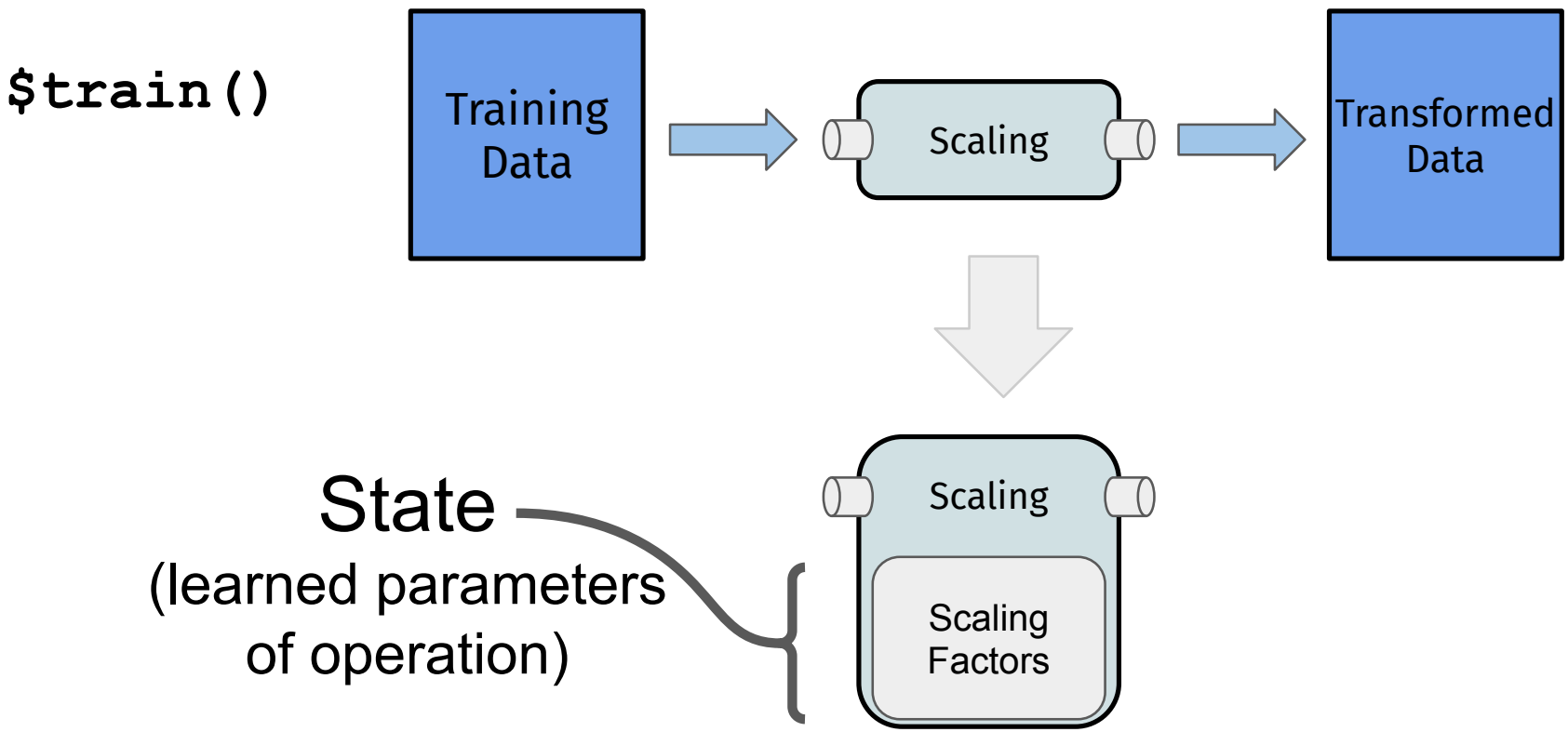


PipeOps

The Building Blocks

PipeOp: Single Unit of Data Operation

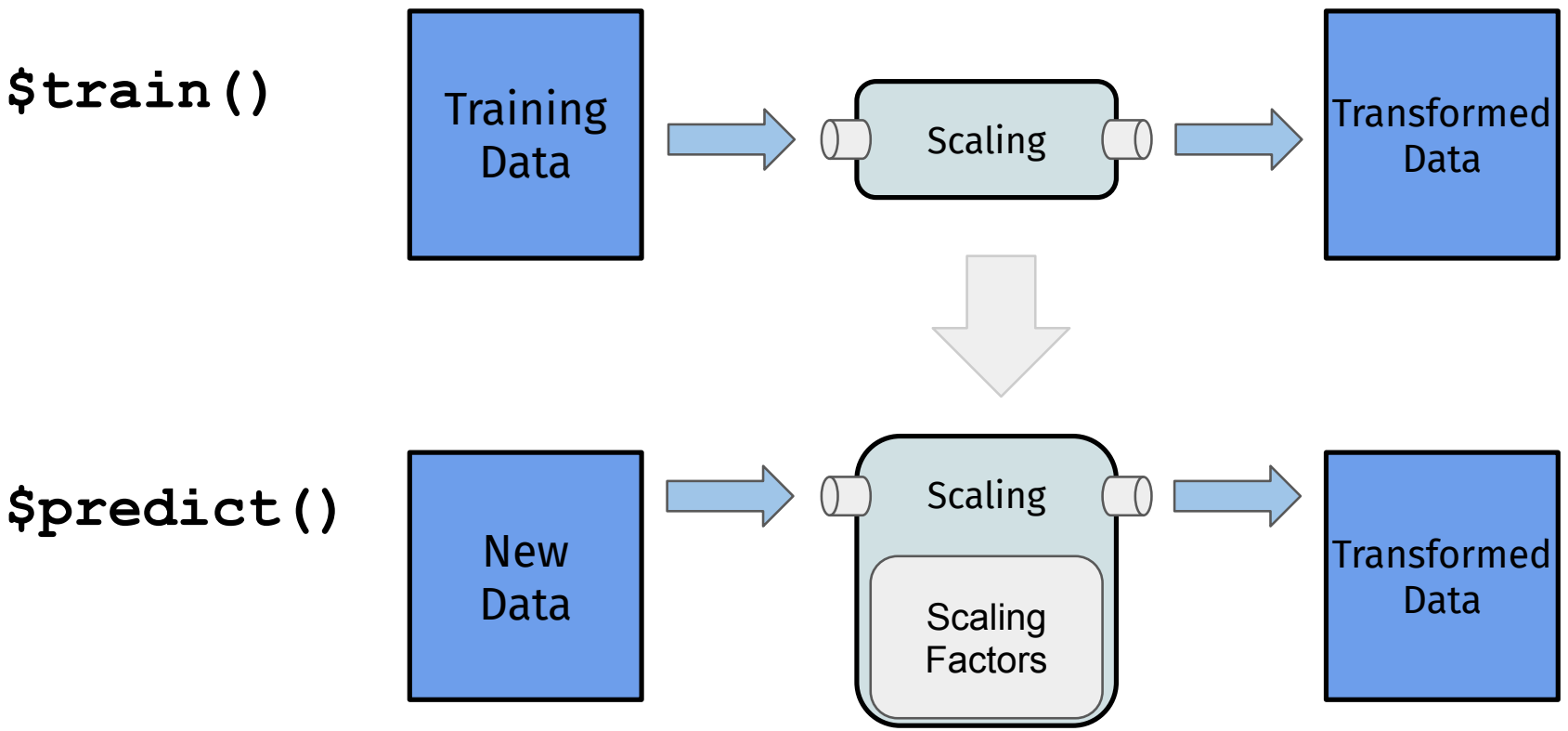
- `$train()`: process data and create `$state`



The Building Blocks

PipeOp: Single Unit of Data Operation

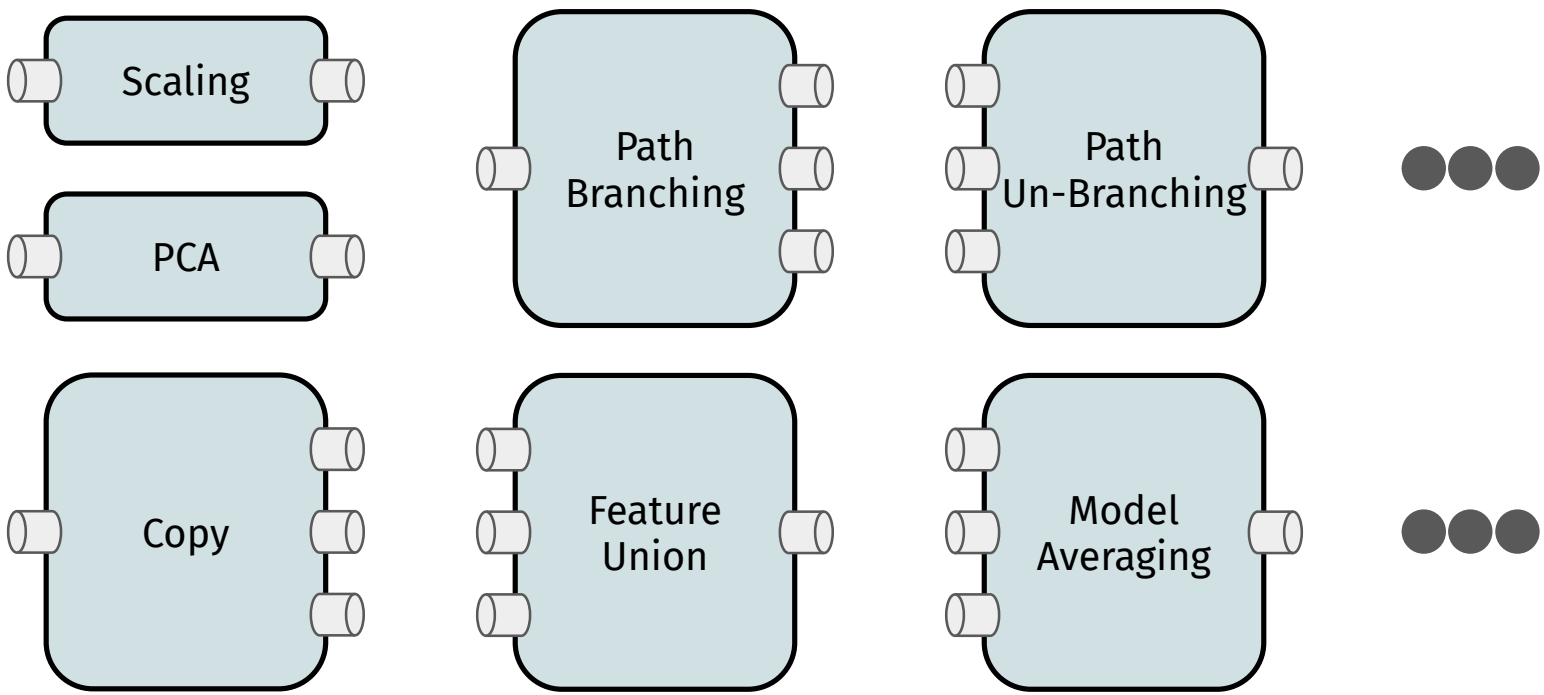
- `$train()`: process data and create `$state`
- `$predict()`: process data depending on the `$state`



The Building Blocks

PipeOp: Single Unit of Data Operation

- `$train()`: process data and create `$state`
- `$predict()`: process data depending on the `$state`
- Multiple inputs or multiple outputs



PipeOps so far and planned

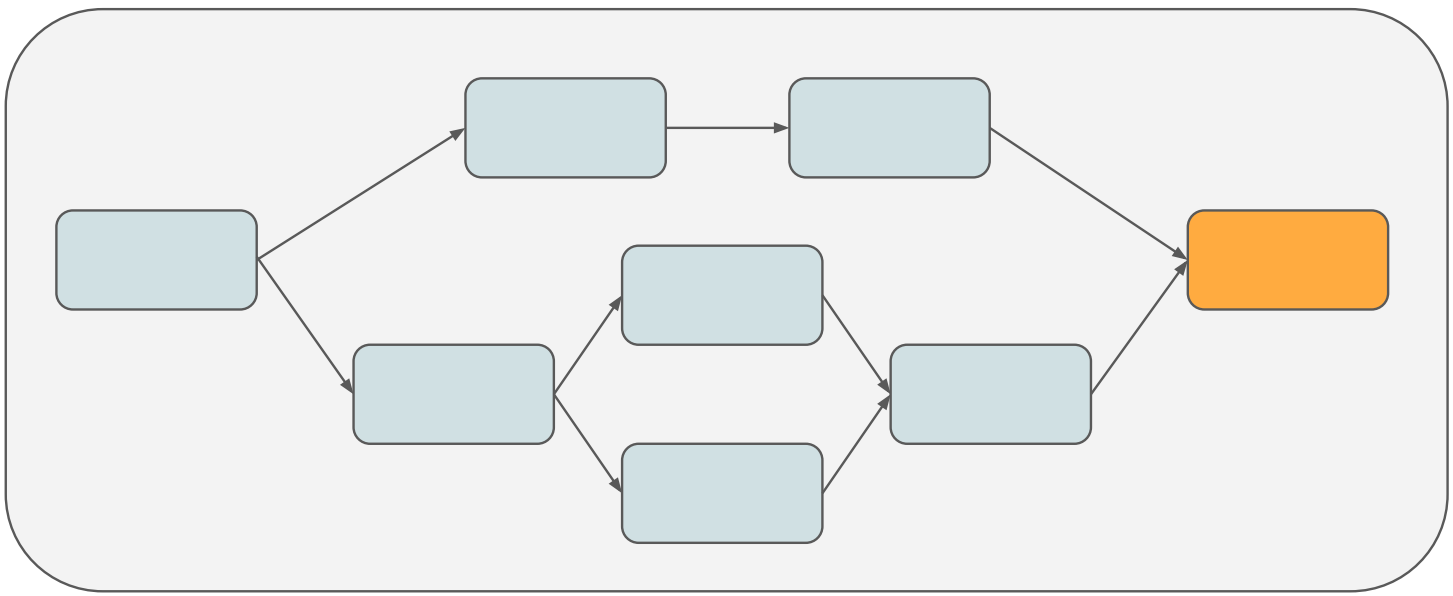
- Simple preprocessing operations: `scale`, `pca`, `apply`, `mutate`
- Missing value imputation: `impute`
- Feature selection and filtering: `select`, `filter`
- Categorical data encoding: `encode`
- Undersampling / subsampling: `balancesample`, `subsample`, `chunk`
- Learners: `learner`, `learner_cv`
- Ensemble methods on Predictions: `majorityvote`, `modelavg`
- Simultaneous and alternative branching: `copy`, `branch`, `unbranch`
- Combination of data: `featureunion`
- Backup prediction: `backuplearner`
- Text processing *(planned)*
- Time series and spatio-temporal data *(planned)*
- Multi-output and ordinal targets *(planned)*
- Outlier detection *(planned)*
- Hurdle models *(planned)*
- ...

mlr

Graph Operations

The Structure

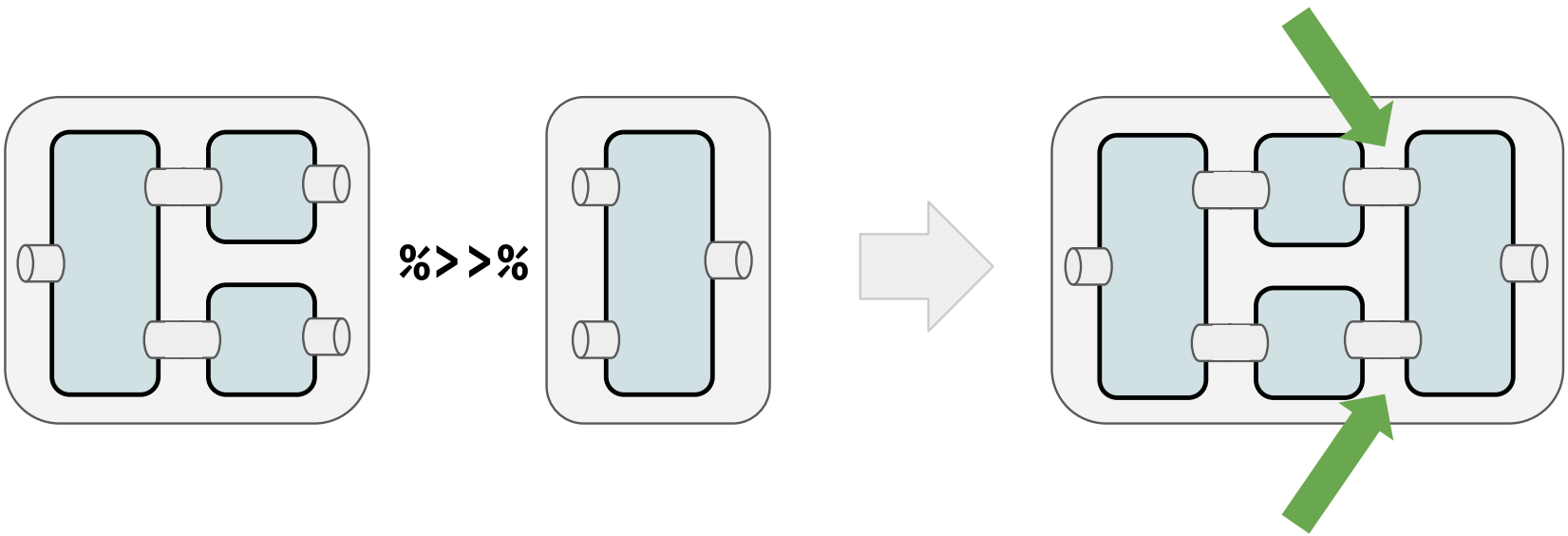
Graph Operations



The Structure

Graph Operations

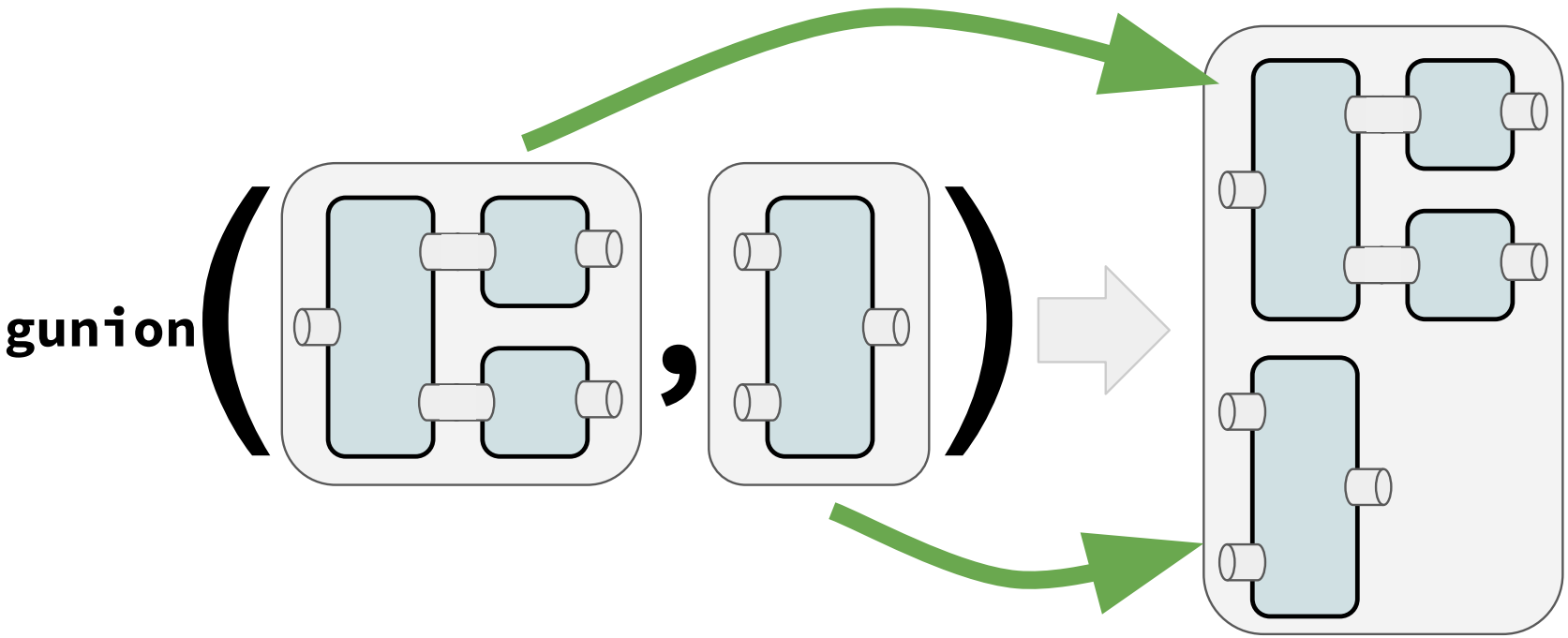
- The %>>%-operator concatenates `Graphs` and `PipeOps`



The Structure

Graph Operations

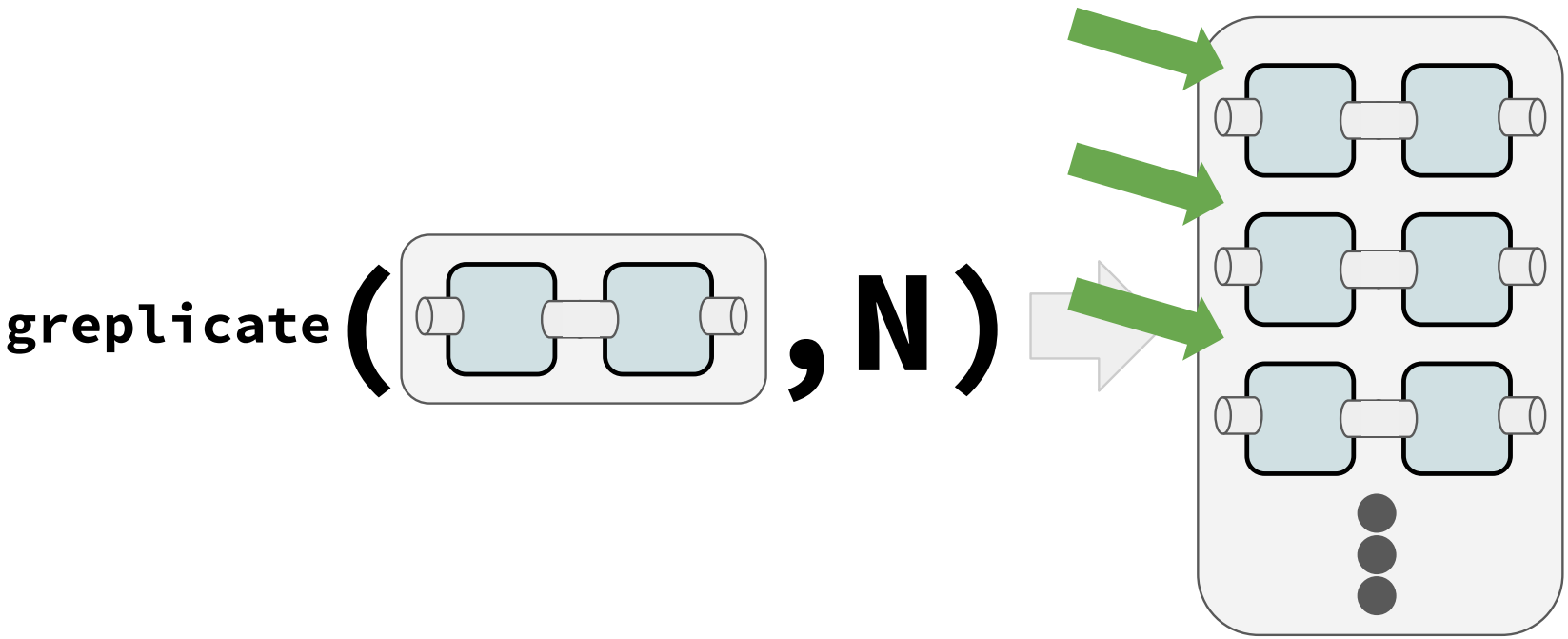
- The `%>>%`-operator concatenates `Graphs` and `PipeOps`
- The `gunion()`-function unites `Graphs` and `PipeOps`



The Structure

Graph Operations

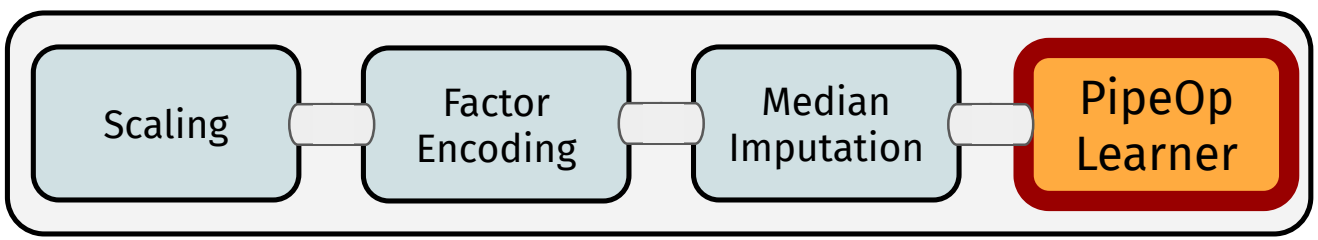
- The `%>>%`-operator concatenates `Graphs` and `PipeOps`
- The `gunion()`-function unites `Graphs` and `PipeOps`
- The `greplicate()`-function unites copies of `Graphs` and `PipeOps`



Learners and Graphs

PipeOpLearner

- **`Learner`** as a **`PipeOp`**
- Fits a model, output is `Prediction`



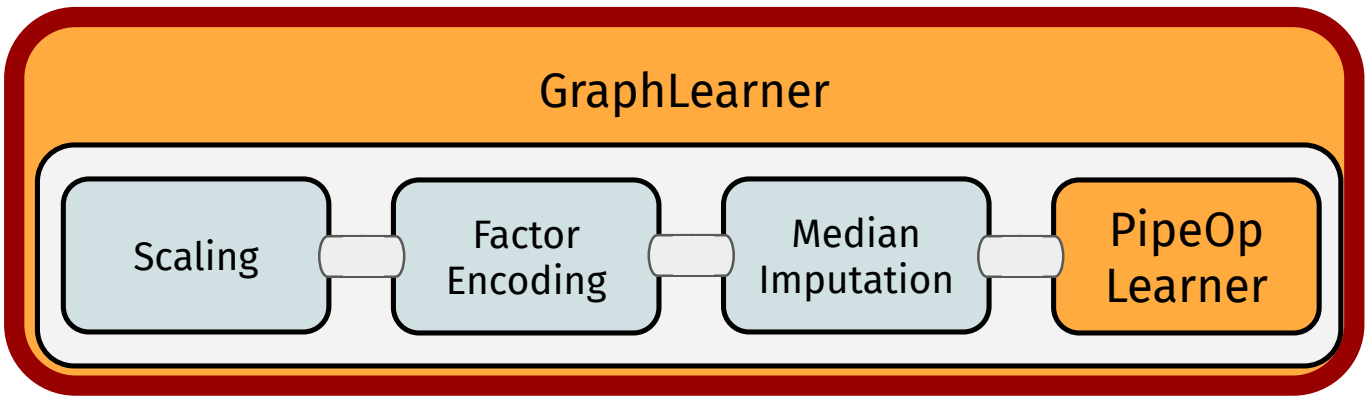
Learners and Graphs

PipeOpLearner

- `Learner` as a `PipeOp`
- Fits a model, output is `Prediction`

GraphLearner

- `Graph` as a `Learner`
- All benefits of `mlr3`: **resampling**, **tuning**, **nested resampling**, ...

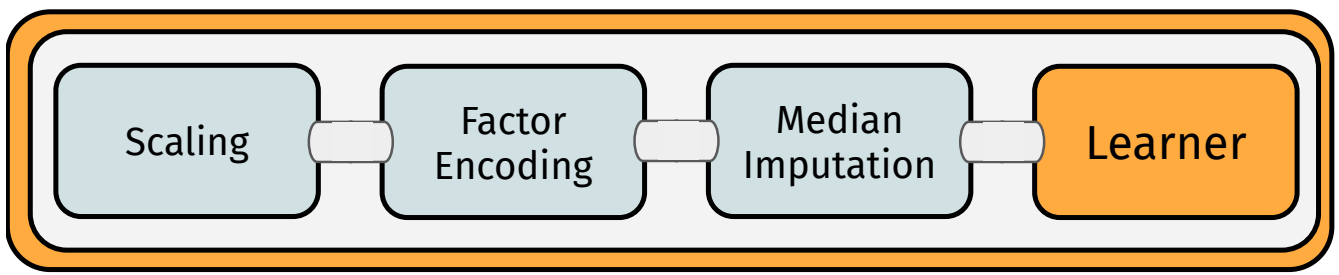


Linear Pipelines

mlr3pipelines in Action

Linear Preprocessing Pipeline

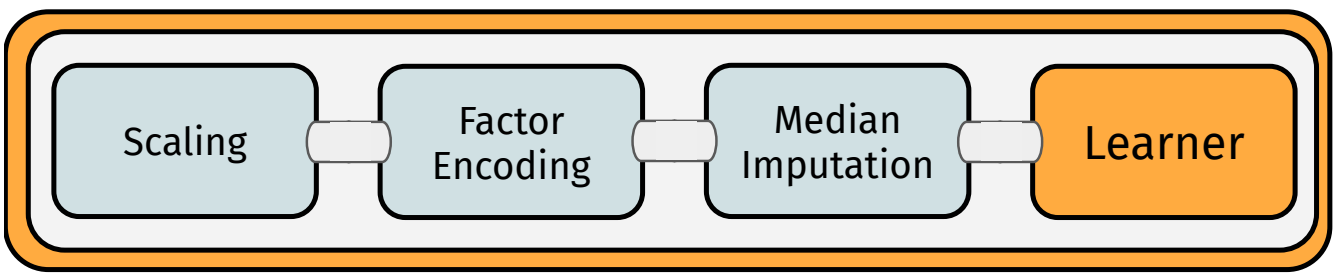
```
graph_pp = mlr_pipeops$get("scale") %>>%
  mlr_pipeops$get("encode") %>>%
  mlr_pipeops$get("impute") %>>%
  mlr_pipeops$get("learner",
    learner = mlr_learners$get("classif.rpart"))
```



mlr3pipelines in Action

Linear Preprocessing Pipeline

```
graph_pp = "scale" %>>% "encode" %>>% "impute" %>>%
  mlr_pipeops$get("learner",
    learner = mlr_learners$get("classif.rpart"))
```

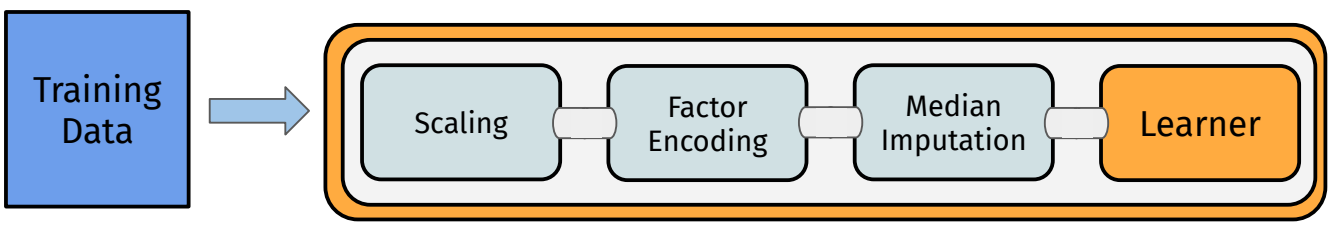


mlr3pipelines in Action

Linear Preprocessing Pipeline

- **`train()`**ing: Data propagates and creates **`$states`**

```
glrn = GraphLearner$new(graph_pp)
glrn$train(task)
```

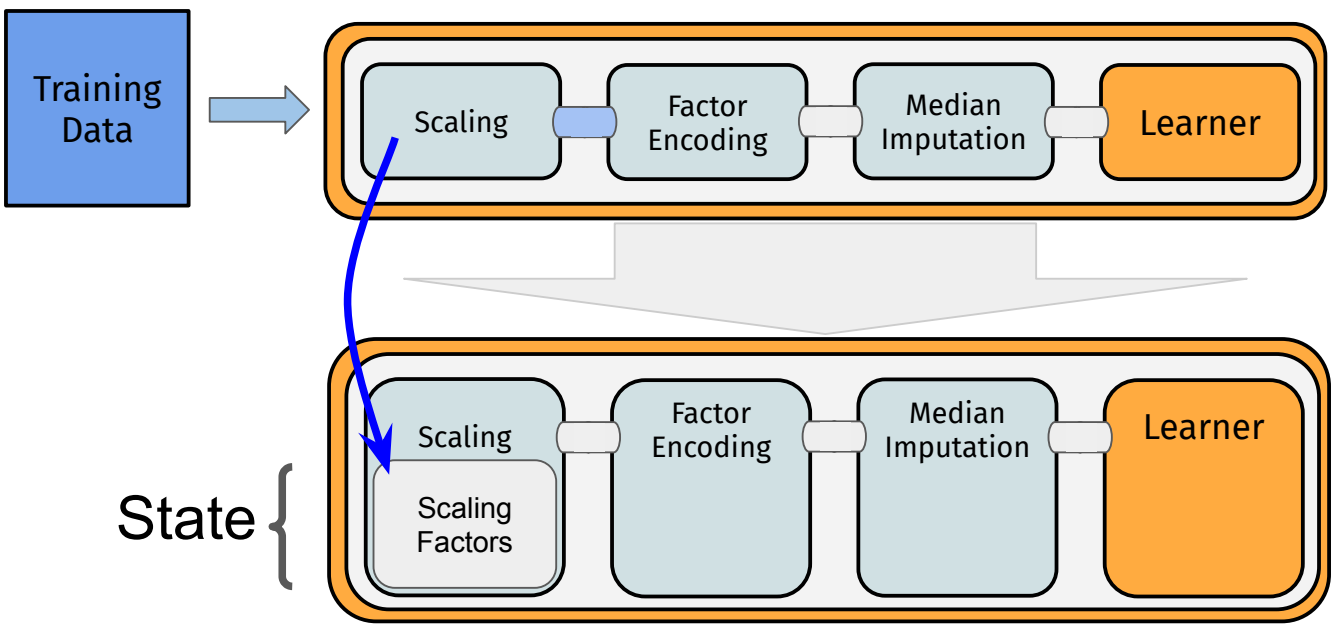


mlr3pipelines in Action

Linear Preprocessing Pipeline

- `train()`ing: Data propagates and creates `$states`

```
glrn = GraphLearner$new(graph_pp)
glrn$train(task)
```

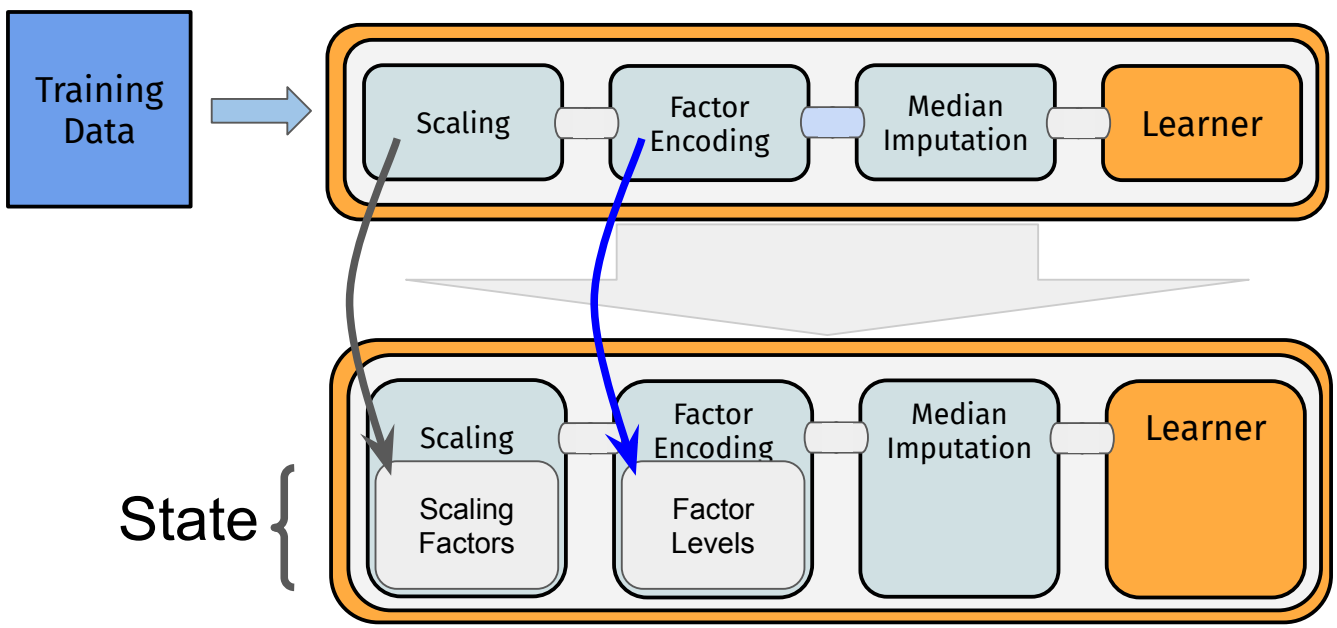


mlr3pipelines in Action

Linear Preprocessing Pipeline

- `train()`ing: Data propagates and creates `$states`

```
glrn = GraphLearner$new(graph_pp)
glrn$train(task)
```

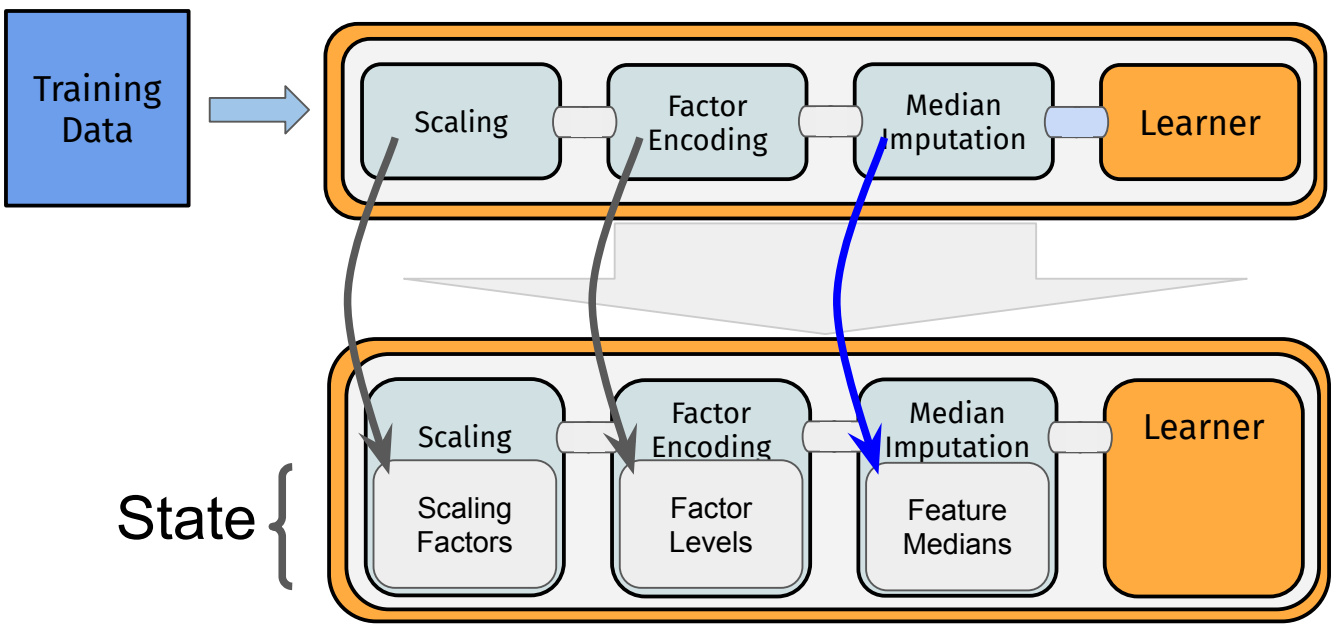


mlr3pipelines in Action

Linear Preprocessing Pipeline

- `train()`ing: Data propagates and creates `$states`

```
glrn = GraphLearner$new(graph_pp)
glrn$train(task)
```

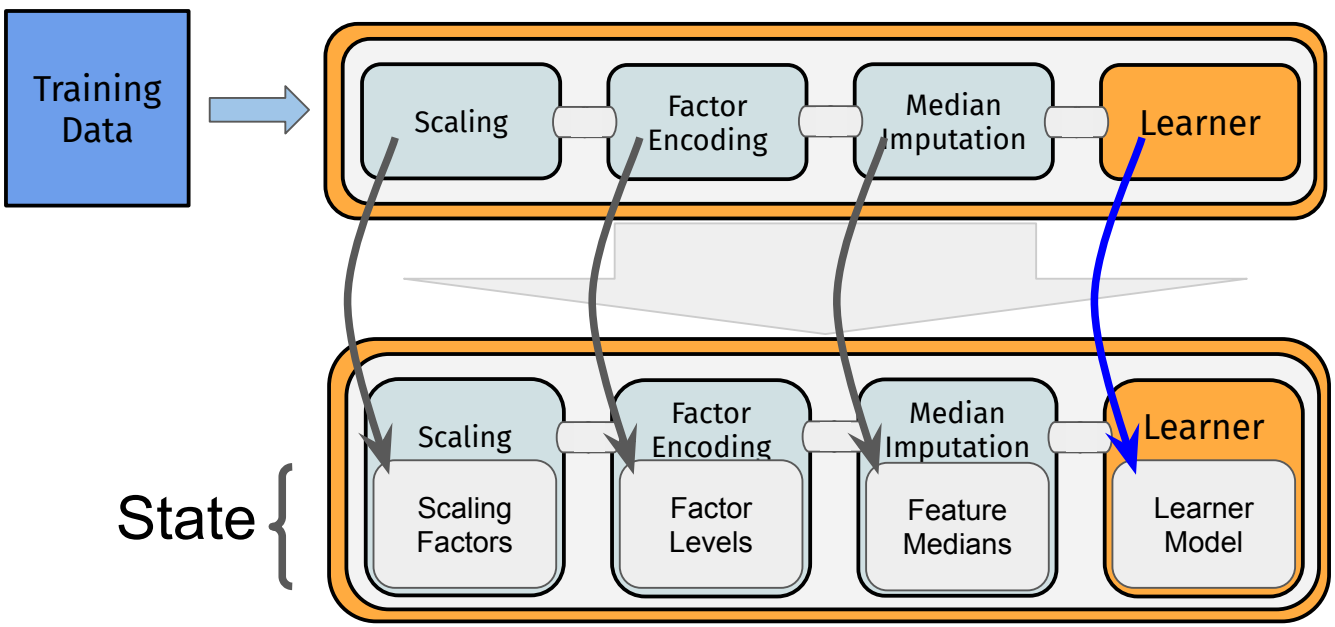


mlr3pipelines in Action

Linear Preprocessing Pipeline

- `train()`ing: Data propagates and creates `$states`

```
glrn = GraphLearner$new(graph_pp)
glrn$train(task)
```

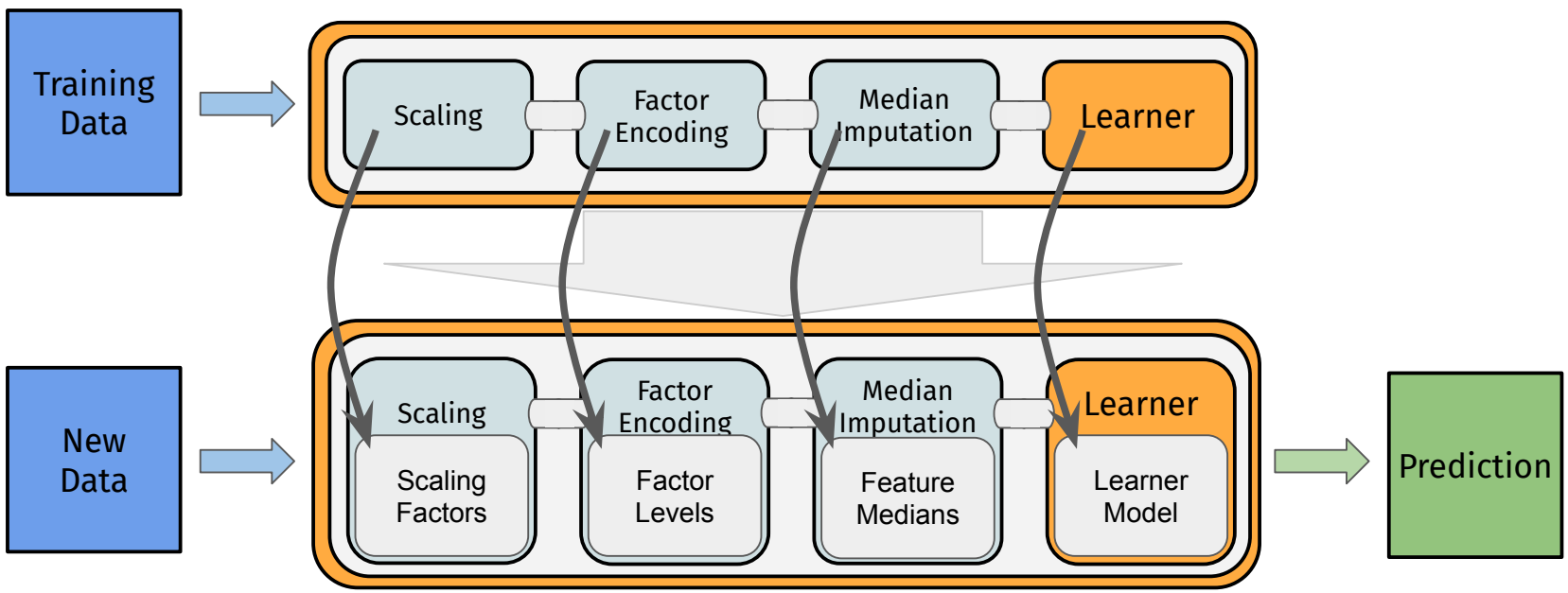


mlr3pipelines in Action

Linear Preprocessing Pipeline

- **`train()`ing**: Data propagates and creates **`$states`**
- **`predict()`tion**: Data propagates, uses **`$states`**

```
glrn$predict(task)
```



mlr3pipelines in Action

Linear Preprocessing PipeLine

- Setting / retrieving parameters: **`$param_set`**

```
graph_pp$pipeops$impute$param_set$values$method_num = "mean"
```

mlr3pipelines in Action

Linear Preprocessing PipeLine

- Setting / retrieving parameters: **`$param_set`**

```
graph_pp$pipeops$impute$param_set$values$method_num = "mean"
```

- Retrieving state: **`$state`** of individual `PipeOps` (*after* `$train()`)

```
graph_pp$pipeops$scale$state %>% head(1)
```

```
## $center
## Petal.Length  Petal.Width Sepal.Length  Sepal.Width
##     3.758000     1.199333     5.843333     3.057333
```

mlr3pipelines in Action

Linear Preprocessing PipeLine

- Setting / retrieving parameters: **`$param_set`**

```
graph_pp$pipeops$impute$param_set$values$method_num = "mean"
```

- Retrieving state: **`$state`** of individual `PipeOps` (*after* `$train()`)

```
graph_pp$pipeops$scale$state %>% head(1)
```

```
## $center
## Petal.Length  Petal.Width Sepal.Length  Sepal.Width
##     3.758000     1.199333     5.843333     3.057333
```

- Retrieving intermediate results: **`$.result`** (set debug option before)

```
graph_pp$pipeops$scale$.result[[1]]$data() %>% head(3)
```

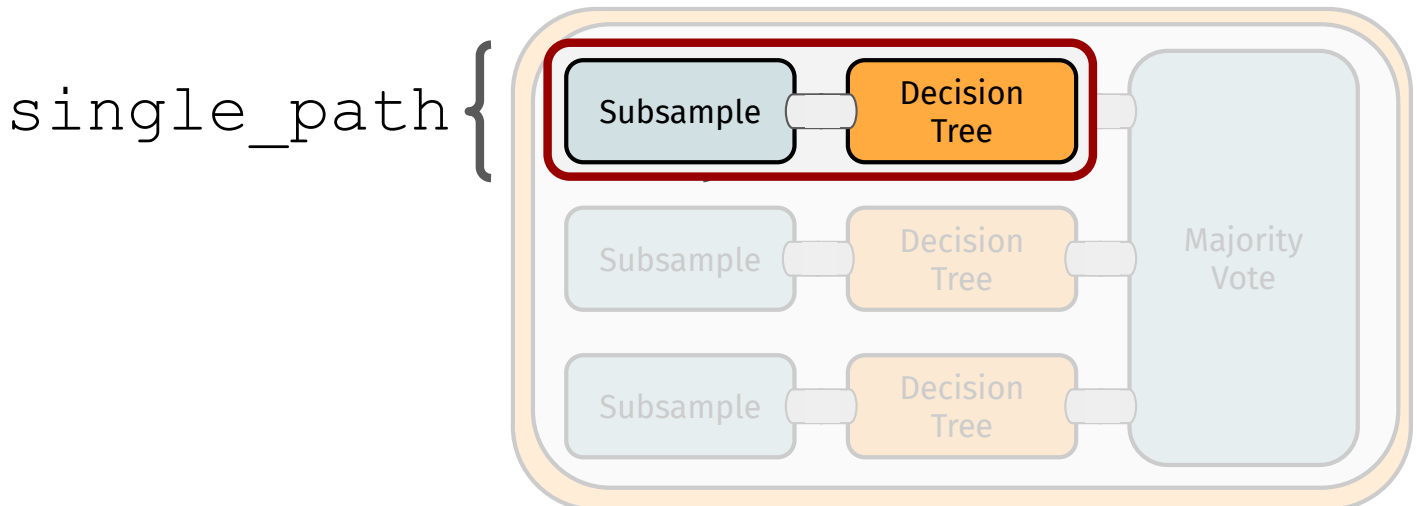
```
##    Species Petal.Length Petal.Width Sepal.Length Sepal.Width
## 1:  setosa    -1.335752   -1.311052   -0.8976739   1.0156020
## 2:  setosa    -1.335752   -1.311052   -1.1392005  -0.1315388
## 3:  setosa    -1.392399   -1.311052   -1.3807271   0.3273175
```

Nonlinear Pipelines

mlr3pipelines in Action

Ensemble Method: Bagging

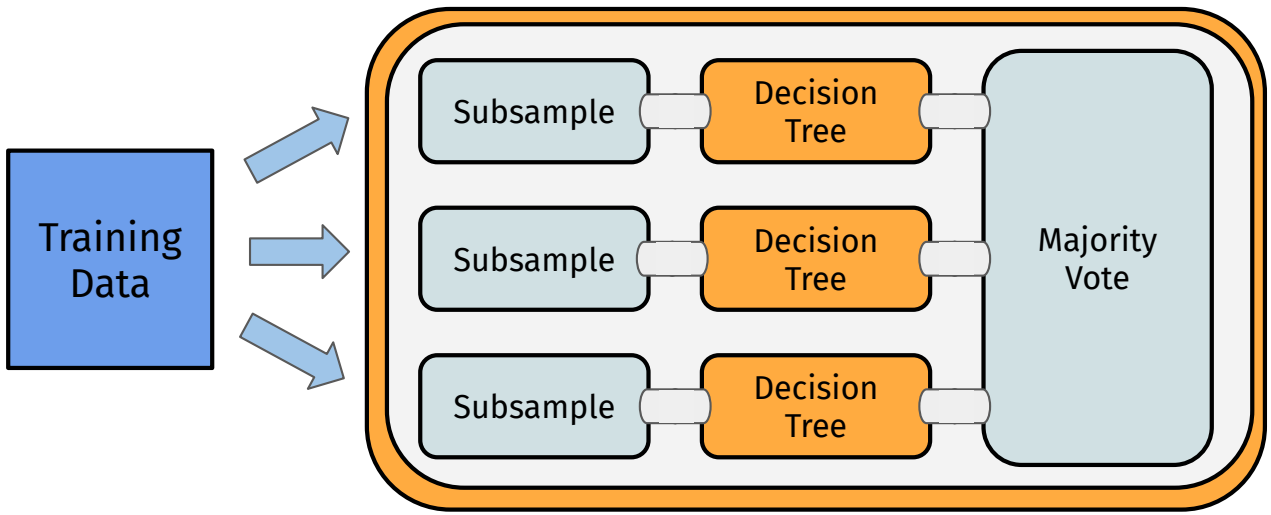
```
single_path = "subsample" %>>%
  mlr_pipeops$get("learner",
    learner = mlr_learners$get("classif.rpart"))
```



mlr3pipelines in Action

Ensemble Method: Bagging

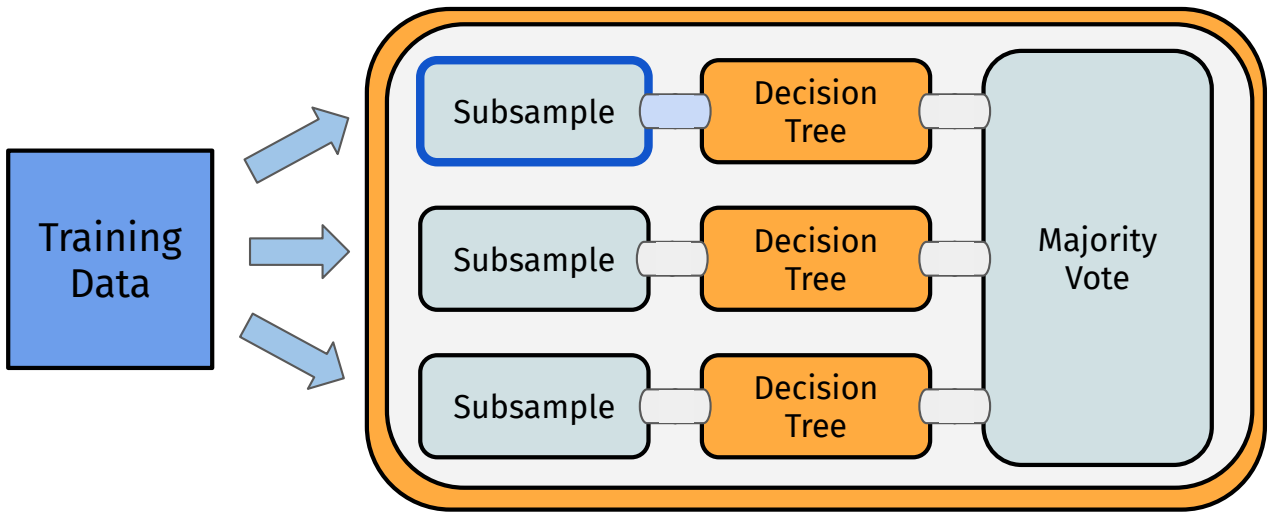
```
single_path = "subsample" %>>%
  mlr_pipeops$get("learner",
    learner = mlr_learners$get("classif.rpart"))

graph_bag = greplicate(single_path, n = 3) %>>%
  mlr_pipeops$get("majorityvote", innum = 3)
```



mlr3pipelines in Action

Ensemble Method: Bagging

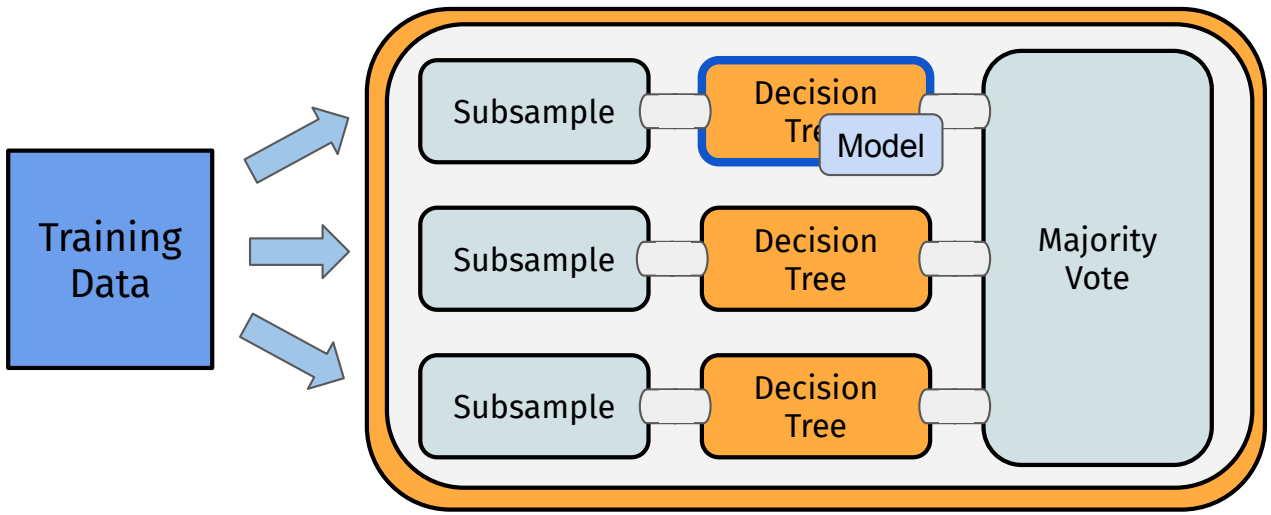
```
single_path = "subsample" %>>%
  mlr_pipeops$get("learner",
    learner = mlr_learners$get("classif.rpart"))

graph_bag = greplicate(single_path, n = 3) %>>%
  mlr_pipeops$get("majorityvote", innum = 3)
```



mlr3pipelines in Action

Ensemble Method: Bagging

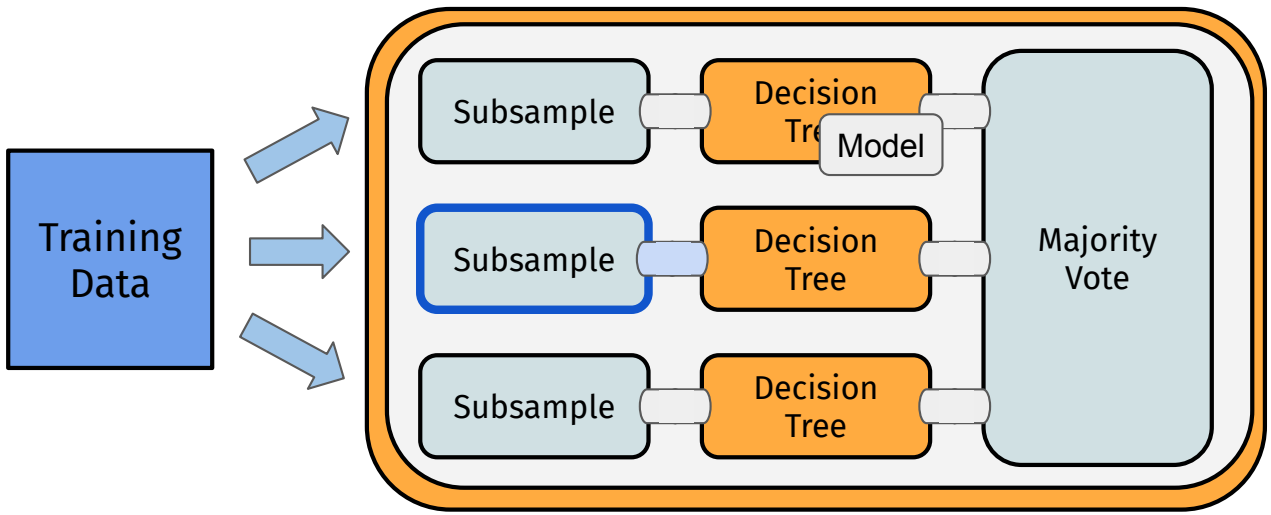
```
single_path = "subsample" %>>%
  mlr_pipeops$get("learner",
    learner = mlr_learners$get("classif.rpart"))

graph_bag = greplicate(single_path, n = 3) %>>%
  mlr_pipeops$get("majorityvote", innum = 3)
```



mlr3pipelines in Action

Ensemble Method: Bagging

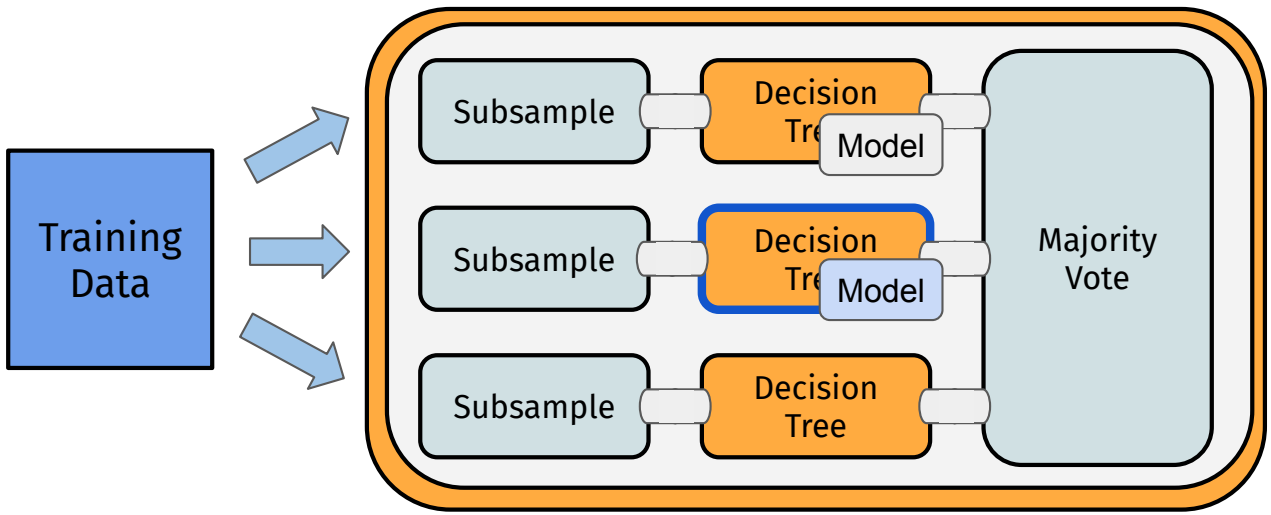
```
single_path = "subsample" %>>%
  mlr_pipeops$get("learner",
    learner = mlr_learners$get("classif.rpart"))

graph_bag = greplicate(single_path, n = 3) %>>%
  mlr_pipeops$get("majorityvote", innum = 3)
```



mlr3pipelines in Action

Ensemble Method: Bagging

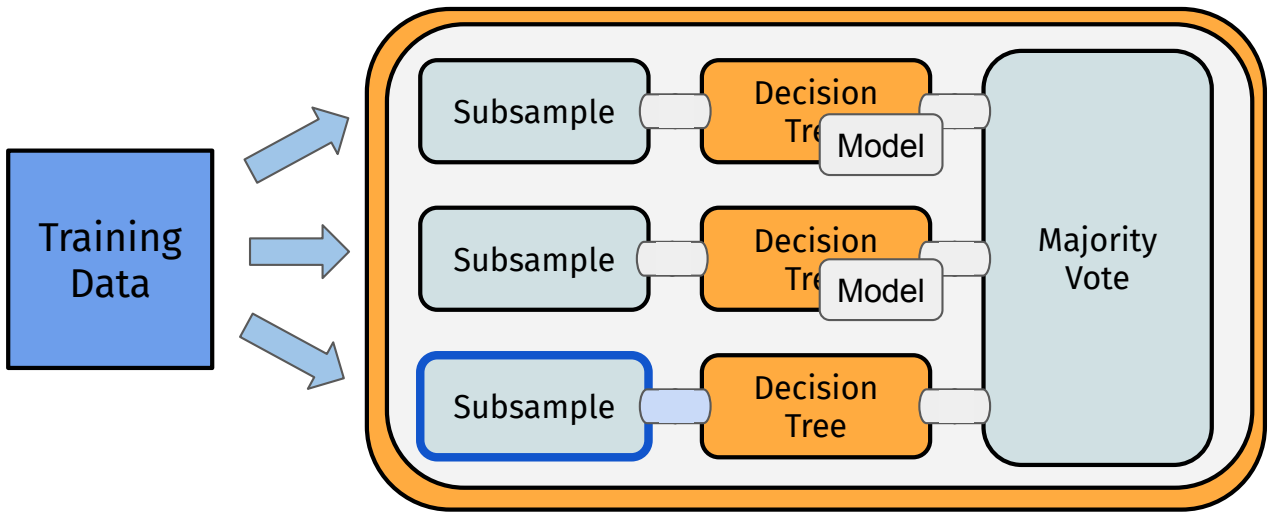
```
single_path = "subsample" %>>%
  mlr_pipeops$get("learner",
    learner = mlr_learners$get("classif.rpart"))

graph_bag = greplicate(single_path, n = 3) %>>%
  mlr_pipeops$get("majorityvote", innum = 3)
```



mlr3pipelines in Action

Ensemble Method: Bagging

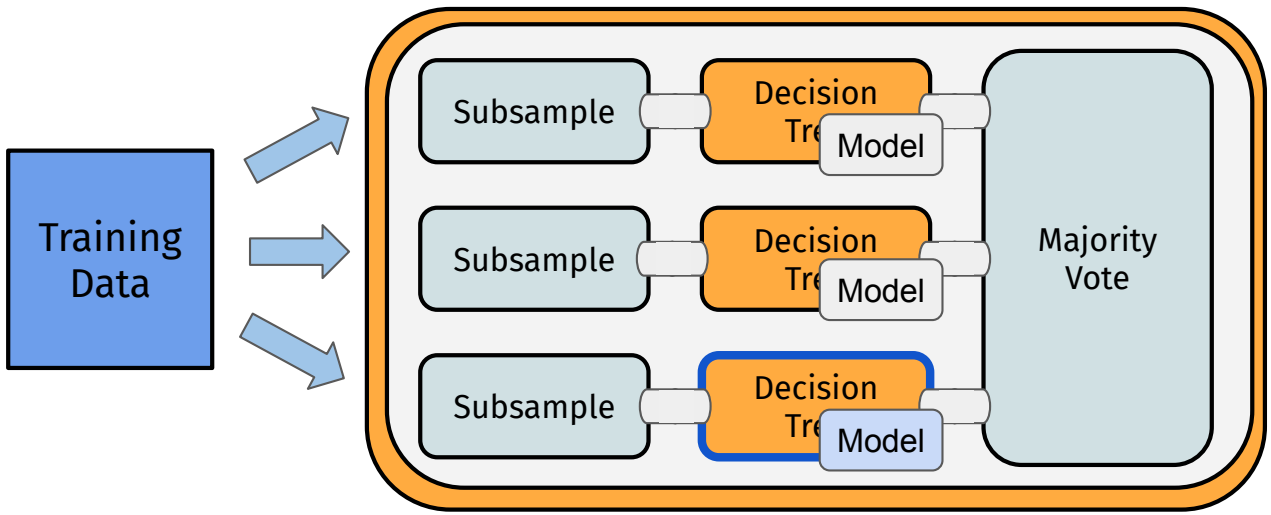
```
single_path = "subsample" %>>%
  mlr_pipeops$get("learner",
    learner = mlr_learners$get("classif.rpart"))

graph_bag = greplicate(single_path, n = 3) %>>%
  mlr_pipeops$get("majorityvote", innum = 3)
```



mlr3pipelines in Action

Ensemble Method: Bagging

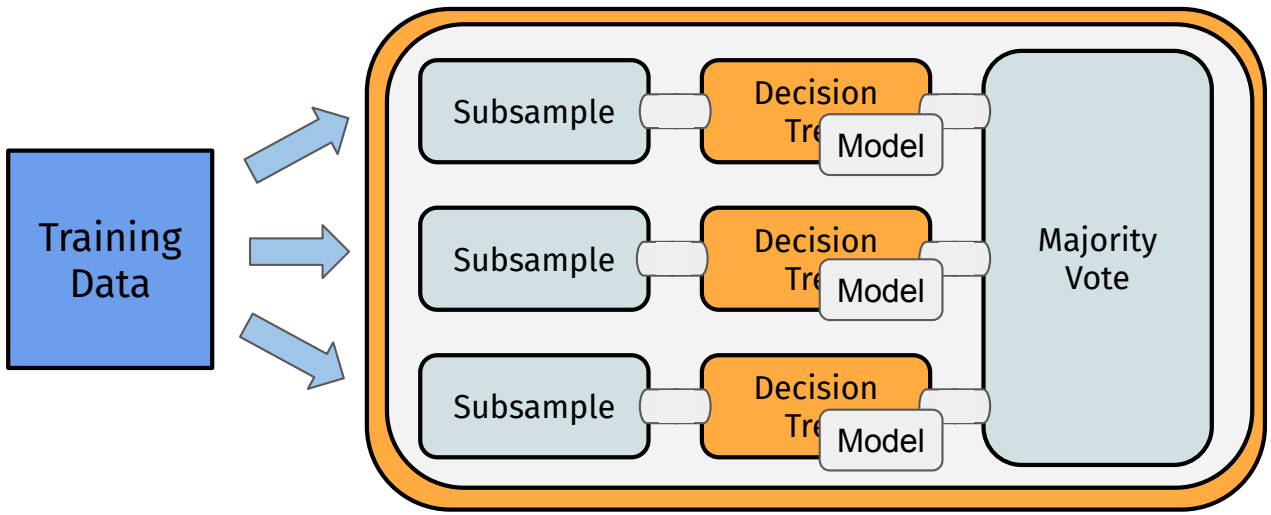
```
single_path = "subsample" %>>%
  mlr_pipeops$get("learner",
    learner = mlr_learners$get("classif.rpart"))

graph_bag = greplicate(single_path, n = 3) %>>%
  mlr_pipeops$get("majorityvote", innum = 3)
```



mlr3pipelines in Action

Ensemble Method: Bagging

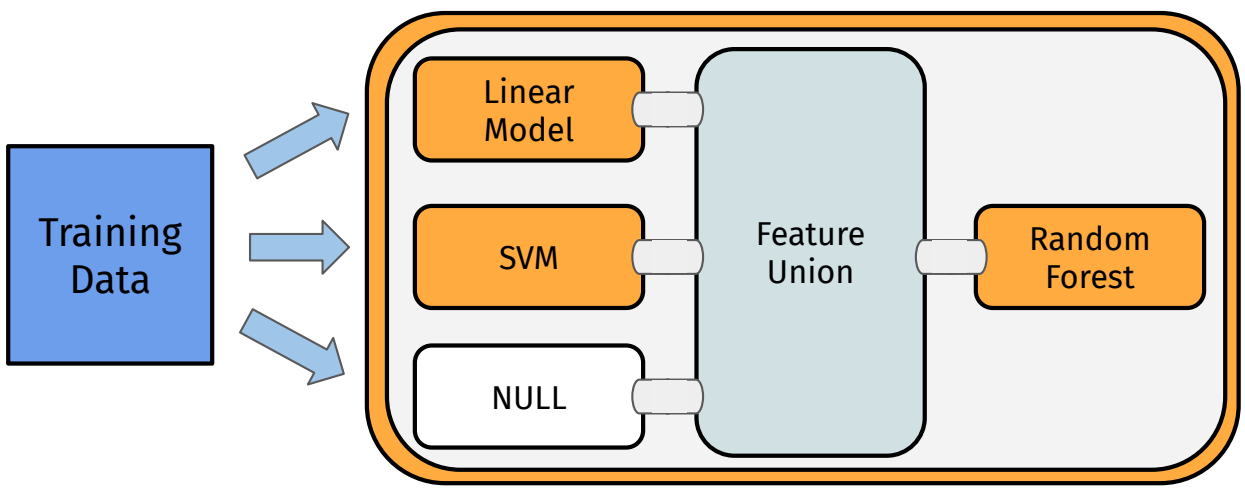
```
single_path = "subsample" %>>%
  mlr_pipeops$get("learner",
    learner = mlr_learners$get("classif.rpart"))

graph_bag = greplicate(single_path, n = 3) %>>%
  mlr_pipeops$get("majorityvote", innum = 3)
```



mlr3pipelines in Action

Ensemble Method: Stacking

```
graph_stack = gunion(list(
      mlr_pipeops$get("learner_cv",
        learner = mlr_learners$get("regr.lm")),
      mlr_pipeops$get("learner_cv",
        learner = mlr_learners$get("regr.svm")),
      "null")) %>>%
  mlr_pipeops$get("featureunion", innum = 3) %>>%
  mlr_pipeops$get("learner",
    learner = mlr_learners$get("regr.ranger"))
```

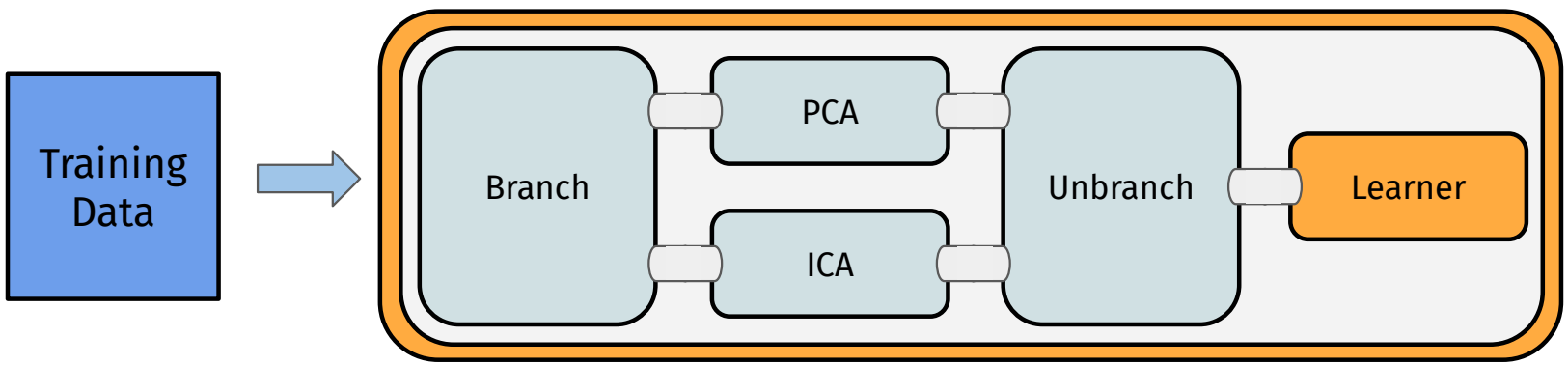


mlr3pipelines in Action

Branching

```
graph_branch = mlr_pipeops$get("branch", c("pca", "ica")) %>>%
  gunion(list("pca", "ica")) %>>%
  mlr_pipeops$get("unbranch", c("pca", "ica")) %>>%
  mlr_pipeops$get("learner",
    learner = mlr_learners$get("classif.kknn"))
```

Execute only one of several alternative paths

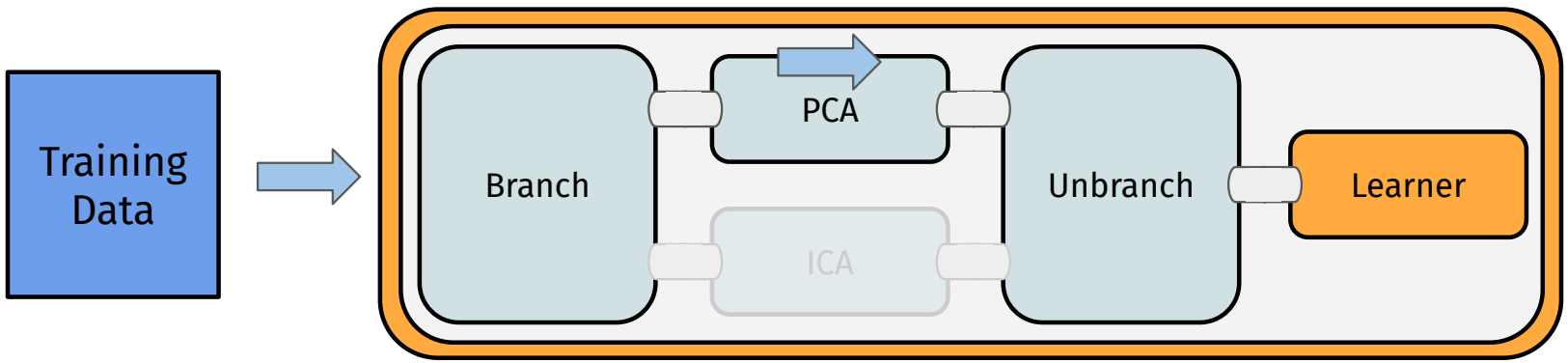


mlr3pipelines in Action

Branching

```
graph_branch = mlr_pipeops$get("branch", c("pca", "ica")) %>>%
  gunion(list("pca", "ica")) %>>%
  mlr_pipeops$get("unbranch", c("pca", "ica")) %>>%
  mlr_pipeops$get("learner",
    learner = mlr_learners$get("classif.kknn"))
```

```
> graph_branch$pipeops$branch$
    param_set$values$selection = "pca"
```

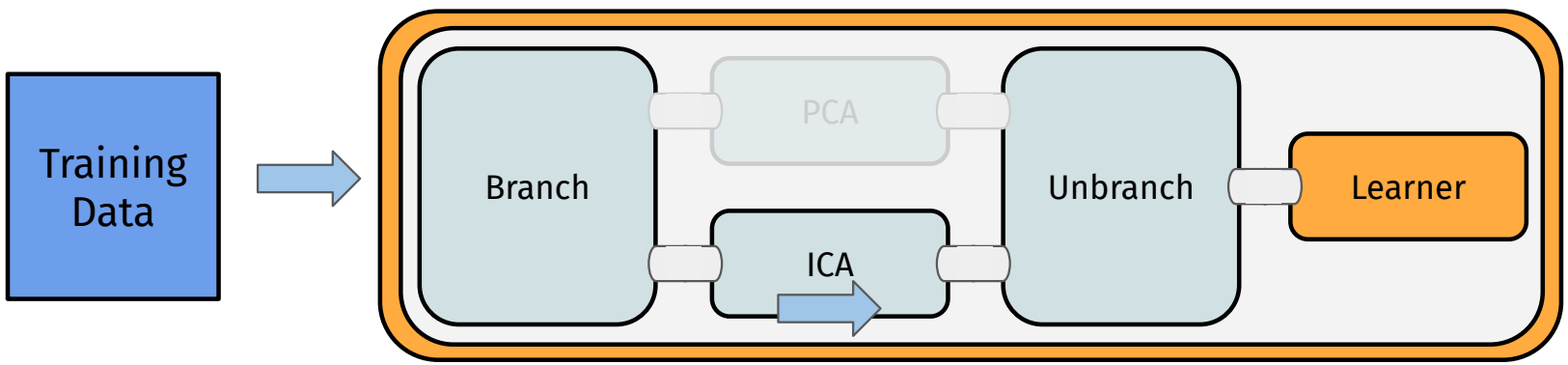


mlr3pipelines in Action

Branching

```
graph_branch = mlr_pipeops$get("branch", c("pca", "ica")) %>>%
  gunion(list("pca", "ica")) %>>%
  mlr_pipeops$get("unbranch", c("pca", "ica")) %>>%
  mlr_pipeops$get("learner",
    learner = mlr_learners$get("classif.kknn"))
```

```
> graph_branch$pipeops$branch$
    param_set$values$selection = "ica"
```



Hyperparameters and Tuning

Hyperparameters and Tuning

- **PipeOps** have *hyperparameters* (using **paradox** pkg)
- `Graphs` have hyperparameters of all components *combined*
- $\Rightarrow$ simultaneous **Tuning** of `Learner` and preprocessing (`mlr3tuning` package)

```
library("paradox") ; library("mlr3tuning")
glrn = "scale" %>>% mlr_pipeops$get("learner",
  mlr_learners$get("classif.rpart"))
ps = ParamSet$new(list(
  ParamLgl$new("scale.scale"),
  ParamInt$new("classif.rpart.minsplit", 1, 20)
))
ff = PerformanceEvaluator$new(task, glrn, "cv", "classif.ce", ps)
tuner = TunerRandomSearch$new(ff, TerminatorEvaluations$new(10))

tuner$tune()

tuner$tune_result()
```

mlr3pipelines

Not Shown Here:

- Many more `PipeOps`: `select`, `apply`, `encode`, ...
- Automatic type-checking when constructing `Graphs`
- Interactive (html + javascript) plots
- Extensible by R6 inheritance of `PipeOp` base class

Upcoming Features

- More `PipeOps`
- Caching of expensive results
- Automatically parallel execution of concurrent operations

Thanks! Questions? Comments? Comment on Github?

`mlr3`: https://github.com/**mlr-org/mlr3**
`mlr3pipelines`: https://github.com/**mlr-org/mlr3pipelines**