Applied Machine Learning

Feature Engineering: mlr3pipelines

Learning goals

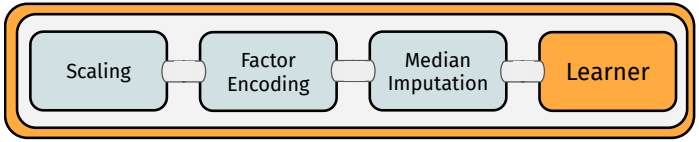
- Introduction to mlr3pipelines
- Building blocks and the graph structure

MLR3PIPELINES

Machine Learning Workflows:

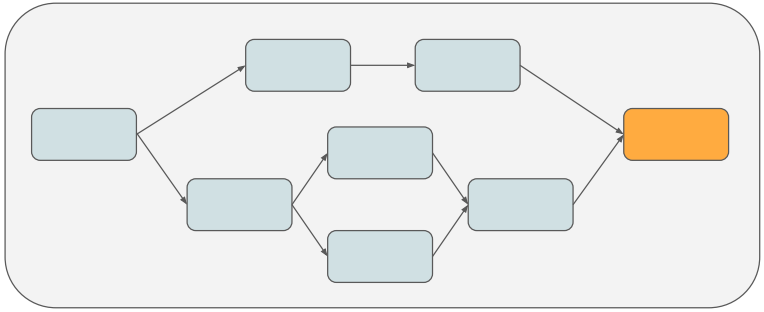
- **Preprocessing**: Feature extraction, feature selection, missing data imputation,...
- **Ensemble methods**: Model averaging, model stacking
- `mlr3`: modular model fitting

$\Rightarrow$ `mlr3pipelines`: modular ML workflows



MACHINE LEARNING WORKFLOWS

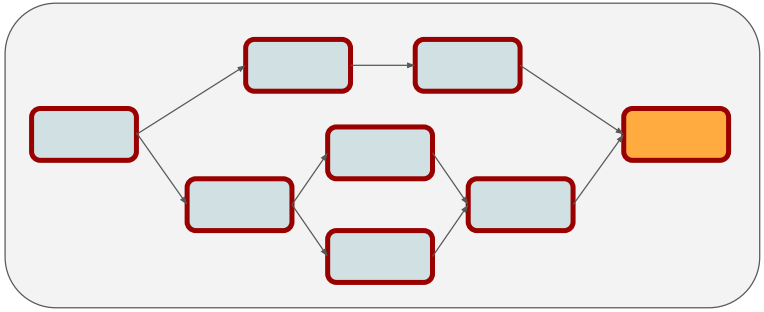
– what do they look like?



MACHINE LEARNING WORKFLOWS

– what do they look like?

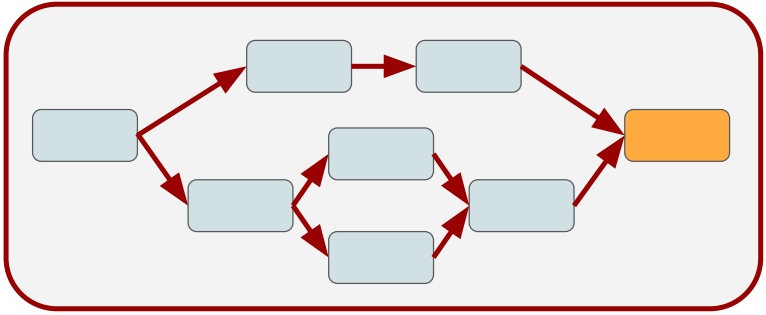
- **Building blocks**: *what* is happening? → `PipeOp`



MACHINE LEARNING WORKFLOWS

– what do they look like?

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

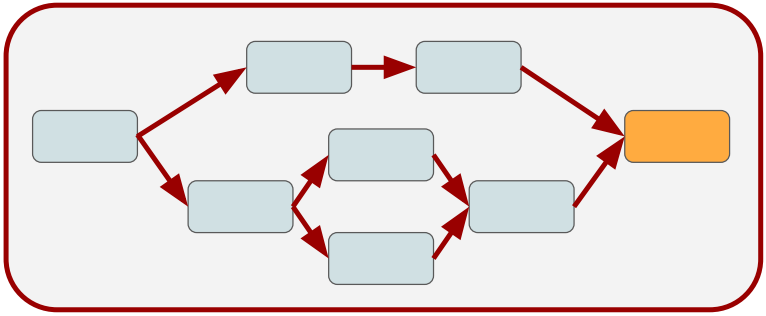


MACHINE LEARNING WORKFLOWS

– what do they look like?

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

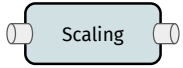
⇒ `Graph`: `PipeOps` as **nodes** with **edges** (data flow) between them



THE BUILDING BLOCKS

PipeOp: Single Unit of Data Operation

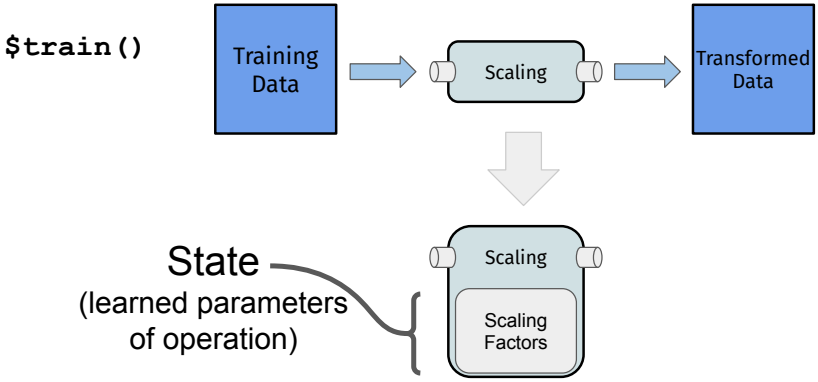
- `pip = po("scale")` to construct



THE BUILDING BLOCKS

PipeOp: Single Unit of Data Operation

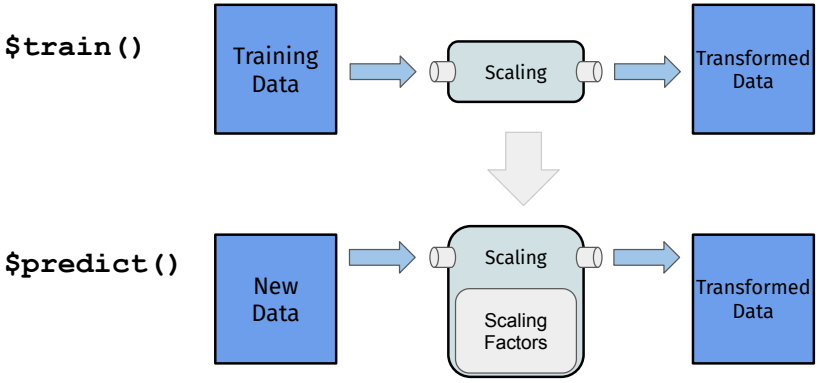
- `pip = po("scale")` to construct
- `pip$train()`: process data and create `pip$state`



THE BUILDING BLOCKS

PipeOp: Single Unit of Data Operation

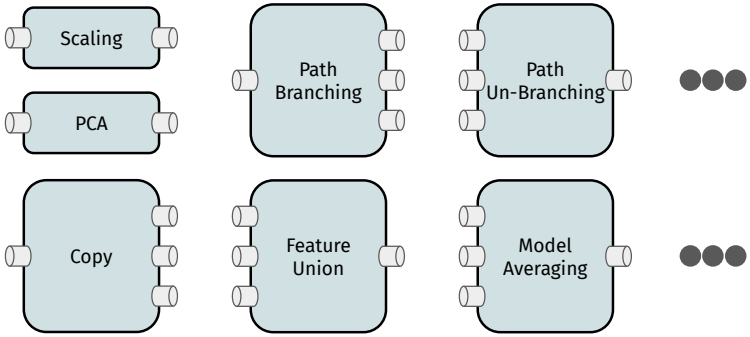
- `pip = po("scale")` to construct
- `pip$train()`: process data and create `pip$state`
- `pip$predict()`: process data depending on the `pip$state`



THE BUILDING BLOCKS

PipeOp: Single Unit of Data Operation

- `pip = po("scale")` to construct
- `pip$train()`: process data and create `pip$state`
- `pip$predict()`: process data depending on the `pip$state`
- Multiple inputs or multiple outputs



THE BUILDING BLOCKS

```
po = po("scale")
trained = po$train(list(task))
trained$output$head(3)
```

```
#>    Species Petal.Length Petal.Width Sepal.Length Sepal.Width
#>     <fctr>        <num>       <num>        <num>       <num>
#> 1:  setosa         -1.3        -1.3         -0.9        1.02
#> 2:  setosa         -1.3        -1.3         -1.1       -0.13
#> 3:  setosa         -1.4        -1.3         -1.4        0.33
```

```
head(po$state, 2)
```

```
#> $center
#> Petal.Length  Petal.Width Sepal.Length  Sepal.Width
#>          3.8          1.2          5.8          3.1
#>
#> $scale
#> Petal.Length  Petal.Width Sepal.Length  Sepal.Width
#>         1.77         0.76         0.83         0.44
```

THE BUILDING BLOCKS

```
po = po("scale")
trained = po$train(list(task))
trained$output$head(3)
```

```
#>    Species Petal.Length Petal.Width Sepal.Length Sepal.Width
#>     <fctr>        <num>       <num>        <num>       <num>
#> 1:  setosa         -1.3        -1.3         -0.9        1.02
#> 2:  setosa         -1.3        -1.3         -1.1       -0.13
#> 3:  setosa         -1.4        -1.3         -1.4        0.33
```

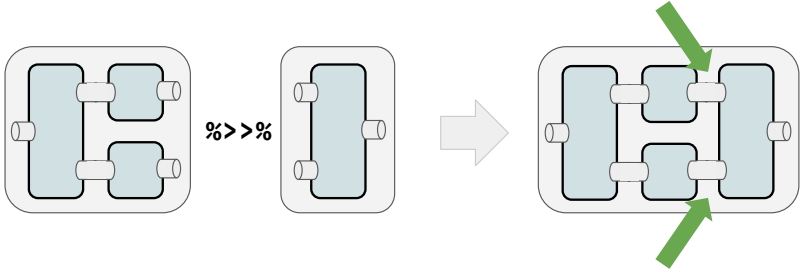
```
smalltask = task$clone()
smalltask = smalltask$filter(1:3)
pred = po$predict(list(smalltask))
pred$output$data()
```

```
#>    Species Petal.Length Petal.Width Sepal.Length Sepal.Width
#>     <fctr>        <num>       <num>        <num>       <num>
#> 1:  setosa         -1.3        -1.3         -0.9        1.02
#> 2:  setosa         -1.3        -1.3         -1.1       -0.13
#> 3:  setosa         -1.4        -1.3         -1.4        0.33
```

THE STRUCTURE

Graph Operations

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



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**, ...