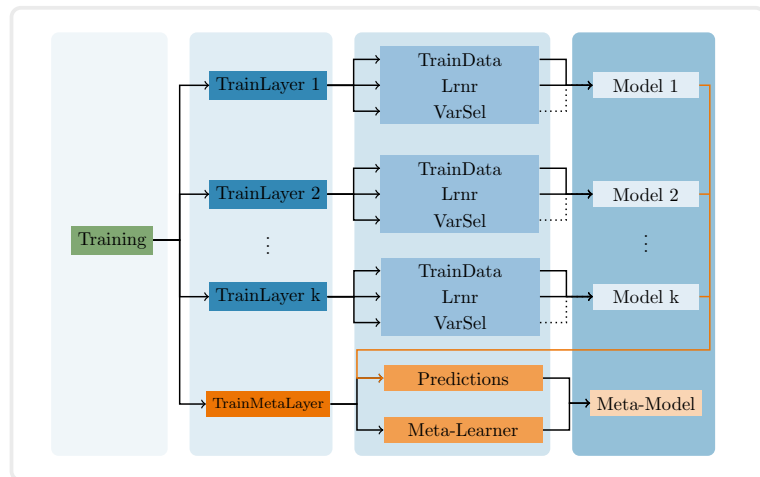


Introduction

The package facilitates integrative modeling by automating the fusion of models trained on different datasets for a common set of individuals. The implementation is built on a set of R6 classes structured as follows:



Class description

Main classes, their utilities, and key tasks are presented.

Class	Utility	Available task
Training	Training layer container	Variable selection, training, predicting and upset plot
Trainlayer	Container of training data, learner and variable selection tool	Variable selection, training and predicting
TrainMetalayer	Meta data container	training and predicting
TrainData	Training datasets storage	–
Learner	Machine learning method storage	training and predicting
Varsel	Storage of variable selection method	variable selection
Testing	Useful for testing; test layer container	upset plot
Testlayer	Test data container	–
TestData	Test dataset	–

How to set up a training flow?

Except for the Training object, all objects are stored within a container object. Layer objects are stored in a training object, while all other objects (TrainData, Lrnr, VarSel, and Model) are stored in a layer object. Models are automatically created after the learning process and cannot be set up by the user. Similarly, only a meta learner can be assigned by the user to the meta layer. Here is an usage illustration. See our GitHub for a complete example.

Illustrative example

```

# Instantiate Training
training <- Training$new(
  id = "tr",
  target = data.frame(), ...)
# Instantiate TrainLayer
tl <- TrainLayer$new(
  id = "tl",
  training = training)
# Instantiate TrainMetalayer
mtl <- TrainMetalayer$new(
  id = "mtl",
  training = tr,
  ...)
# Add TrainData to the layer
td <- TrainMetalayer$new(
  id = "td",
  train_layer = tl, ...)
# Add learner to the layer
# Paramters
lnr_prm <- ParamLrnr$new(
  id = "lnr_prm",
  param_list = list(...),
  ...)
lnr <- Lrnr$new(
  id = "lnr",
  train_layer = tl,
  param = lnr_prm, ...)
# Add variable selection method
# Parameters
varsel_prm <- ParamLrnr$new(
  id = "varsel_prm",
  param_list = list(...), ...)
  
```

```

varsel <- Varsel$new(
  id = "varsel",
  train_layer = tl,
  param = varsel_prm, ...)
  
```

Vizualisation

How individuals overlap across layers?

```
training$upset(order.by = "freq")
```

Variable selection and training

Resampling method is required to create meta layer predictions.

```

# Variable selection
var_sel <- training$varSelection()
# Training
trained <- training$train(
  resampling_method = ...,
  resampling_arg = list(...), ...)
  
```

Prediction

Set up a Testing flow (e.g. testing) similarly to Training and predict:

```

# Variable selection
predictions <- training$predict(
  testing = testing)
  
```

Other useful functions

Additional classical functions like print, summary can be performed on fuseMLR objects. Getter and setter functions are available via the \$ symbol as well. For example: training\$getTrainMetalayer().

Email: cesaire.kuetefouodo@uni-luebeck.de