Package: IntegratedLearner (via r-universe)

June 9, 2026

Type Package

Title Integrated Multi-Omics Learning for Survival and Other Outcomes

Version 0.99.0

Description Provides a unified interface for multi-omics prediction using early, late, and intermediate fusion for continuous, binary, multiclass, and survival outcomes. It supports both MultiAssayExperiment and PCL-style inputs, performs input validation and feature/sample harmonization across layers, and provides model fitting, prediction, plotting, and variable-importance utilities.

License MIT + file LICENSE

URL https://github.com/himelmallick/IntegratedLearner

BugReports https://github.com/himelmallick/IntegratedLearner/issues

Encoding UTF-8

Depends R (>= 4.5.0)

Imports methods, stats, utils, nnet, SuperLearner, survival, glmnet, ranger, SummarizedExperiment, MultiAssayExperiment, caret, ROCR, dplyr, stringr, tibble, tidyr

Suggests BART, BiocStyle, CoxBoost, bayesplot, bayesreg, bartMachine, cowplot, ggplot2, gbm, glmnetUtils, mlbench, mboost, nloptr, prediction, quadprog, randomForest, S4Vectors, testthat (>= 3.0.0), timeROC, knitr, rmarkdown, xgboost

SystemRequirements Java (optional; required for SL.BART / bartMachine workflows)

VignetteBuilder knitr

biocViews Software, Classification, Survival, Microbiome

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

Config/pak/sysreqs default-jdk libicu-dev zlib1g-dev

Repository https://biocstaging.r-universe.dev

Date/Publication 2026-06-09 12:03:53 UTC

RemoteUrl https://github.com/BiocStaging/IntegratedLearner

RemoteRef HEAD

RemoteSha 2952c7c8eef230756be15d59837d648871049044

Contents

auc.obj *Title Meta level Objective function: NNLS for gaussian; Rank loss for binary observations*

Description

Title Meta level Objective function: NNLS for gaussian; Rank loss for binary observations

Usage

```
auc.obj(b, X, Y)
```

Arguments

b	Weights vector
X	Design matrix (data frame)
Y	Outcome variable

Value

1 - AUC

Examples

```
X <- data.frame(m1 = c(0.1, 0.4, 0.8, 0.9), m2 = c(0.2, 0.3, 0.7, 0.95))
Y <- c(0, 0, 1, 1)
auc.obj(b = c(0.5, 0.5), X = X, Y = Y)
```

FranzosaE_2019_CuratedMetabolome *Franzosa et al. 2019 training metabolome table*

Description

Metabolomics feature table for the FranzosaE_2019 training cohort.

Usage

```
data(FranzosaE_2019_CuratedMetabolome)
```

Format

A data frame with samples in rows and metabolite features in columns.

FranzosaE_2019_CuratedMetadata

Franzosa et al. 2019 training metadata

Description

Sample-level metadata for the FranzosaE_2019 training cohort.

Usage

```
data(FranzosaE_2019_CuratedMetadata)
```

Format

A data frame with one row per sample and clinical/study covariates in columns.

FranzosaE_2019_CuratedSpeciesProfile

Franzosa et al. 2019 training species profile

Description

Species-level relative abundance table for the FranzosaE_2019 training cohort.

Usage

```
data(FranzosaE_2019_CuratedSpeciesProfile)
```

Format

A data frame with samples in rows and microbial species features in columns.

FranzosaE_2019_Validation_CuratedMetabolome

Franzosa et al. 2019 validation metabolome table

Description

Metabolomics feature table for the FranzosaE_2019 external validation cohort.

Usage

```
data(FranzosaE_2019_Validation_CuratedMetabolome)
```

Format

A data frame with samples in rows and metabolite features in columns.

FranzosaE_2019_Validation_CuratedMetadata

Franzosa et al. 2019 validation metadata

Description

Sample-level metadata for the FranzosaE_2019 external validation cohort.

Usage

```
data(FranzosaE_2019_Validation_CuratedMetadata)
```

Format

A data frame with one row per sample and clinical/study covariates in columns.

FranzosaE_2019_Validation_CuratedSpeciesProfile

Franzosa et al. 2019 validation species profile

Description

Species-level relative abundance table for the FranzosaE_2019 external validation cohort.

Usage

```
data(FranzosaE_2019_Validation_CuratedSpeciesProfile)
```

Format

A data frame with samples in rows and microbial species features in columns.

gene_all *TCGA BRCA gene-level table*

Description

Gene-expression and associated covariate/outcome table for TCGA BRCA examples.

Usage

```
data(gene_all)
```

Format

A data frame with one row per patient and covariates plus gene features in columns.

Source

TCGA-derived example data bundled for package tests/tutorials.

IL_conbin *Integrated machine learning for multi-omics prediction (continuous/binary outcomes)*

Description

Performs integrated machine learning to predict a **binary or continuous** outcome based on two or more omics layers (views). This function implements the core IntegratedLearner engine for **non-survival outcomes**.

Usage

```
IL_conbin(
  feature_table,
  sample_metadata,
  feature_metadata,
  feature_table_valid = NULL,
  sample_metadata_valid = NULL,
  folds = 5,
  seed = 1234,
  base_learner = "SL.BART",
  base_screener = "All",
  run_screening = FALSE,
  screen_pct = NULL,
  filter_method = NULL,
  filter_pct = NULL,
  prevalence_pct = NULL,
```

```
  meta_learner = "SL.nnls.auc",
  run_concat = TRUE,
  run_stacked = TRUE,
  drop_poor_performing_layers = FALSE,
  verbose = FALSE,
  print_learner = TRUE,
  refit.stack = FALSE,
  family = stats::gaussian(),
  ...
)
```

Arguments

feature_table	An R data frame containing multi-layer features (in rows) and samples (in columns). Column names of feature_metadata must match the row names of sample_metadata.
sample_metadata	An R data frame of metadata variables (in columns). Must have a column named subjectID describing per-subject unique identifiers. For longitudinal designs, this variable may be non-unique. Additionally, a column named Y must be present which is the continuous or binary outcome of interest. Row names of sample_metadata must match the column names of feature_table.
feature_metadata	An R data frame of feature-specific metadata across views (in columns) and features (in rows). Must have a column named featureID as a unique per-feature identifier and a column named featureType describing the source layers. Row names of feature_metadata must match the row names of feature_table.
feature_table_valid	Feature table from validation set for which prediction is desired. Must have the exact same structure as feature_table. If missing, uses feature_table.
sample_metadata_valid	Sample-specific metadata table from independent validation set when available. Must have the exact same structure as sample_metadata.
folds	How many folds in the V-fold nested cross-validation? Default is 5.
seed	Specify the seed for reproducibility. Default is 1234.
base_learner	Base learner for late fusion and early fusion. Default: SL.BART.
base_screener	Deprecated for this backend; currently ignored and kept only for backward compatibility.
run_screening	Logical; if TRUE, run supervised screening within each training fold (and again on full training data) before fitting base models.
screen_pct	Percentage of features to retain during screening ((0,100]). Applied after any optional filtering.
filter_method	Optional feature-filter method before model fitting. Supported values are 'prevalence' (top detected features by prevalence) and 'variance' (top features by caret-based variance ranking via nearZeroVar + empirical variance). If NULL, defaults to 'prevalence' when filtering is requested.

filter_pct	Optional retention percentage (in (0,100]) for the selected filter_method. Keeps the top filter_pct percent features.
prevalence_pct	Optional retention percentage (in (0,100]) for prevalence-based filtering before model fitting. Deprecated alias of filter_pct with filter_method = 'prevalence'.
meta_learner	Meta-learner for late fusion (stacked generalization). Defaults to SL.nnls.auc.
run_concat	Should early fusion be run? Default is TRUE.
run_stacked	Should stacked model (late fusion) be run? Default is TRUE.
drop_poor_performing_layers	Logical; if TRUE, layers with single-layer performance below the screening threshold are removed from early and late fusion only. The threshold is AUC < 0.5 for binary and R^2 < 0.5 for continuous outcomes. Single-layer outputs are still retained and reported.
verbose	logical; TRUE for printing SuperLearner progress. Default FALSE.
print_learner	logical; Should a detailed summary be printed? Default TRUE.
refit.stack	logical; For late fusion, refit predictions on the entire data are returned if specified. Default FALSE.
family	Allows gaussian() for continuous and binomial() for binary outcomes. Survival outcomes must be handled via IL_survival().
...	Additional arguments (currently unused).

Details

IL_conbin takes a training set (feature_table, sample_metadata, feature_metadata) and, optionally, a corresponding validation set, and returns predicted values based on the validation set. It also performs V-fold nested cross-validation to estimate the prediction accuracy of various fusion algorithms.

Two integration paradigms are supported: early and late. The software includes multiple ML models based on the SuperLearner R package as well as several data exploration capabilities and visualization modules in a unified estimation framework.

Although IL_conbin() is typically called internally by IntegratedLearner() after extracting multi-view feature tables from MultiAssayExperiment objects, advanced users may call IL_conbin() directly when they already have multi-layer feature tables and metadata in matrix/data.frame form.

Value

A list-like IntegratedLearner object containing fitted layer-specific, stacked, and concatenated models, cross-validated performance (AUC or R\302\262), and predictions for training and validation sets.

Author(s)

Himel Mallick, <him4004@med.cornell.edu>

See Also

IntegratedLearner, IL_survival()

Examples

```
is.function(IL_conbin)
if (FALSE) {
  set.seed(1)
  n <- 20
  feature_table <- rbind(
   matrix(rnorm(3 * n), nrow = 3, dimnames = list(paste0("L1_F", 1:3), paste0("S", 1:n))),
   matrix(rnorm(2 * n), nrow = 2, dimnames = list(paste0("L2_F", 1:2), paste0("S", 1:n)))
  )
  sample_metadata <- data.frame(
    subjectID = paste0("ID", 1:n), Y = rnorm(n),
    row.names = colnames(feature_table)
  )
  feature_metadata <- data.frame(
    featureID = rownames(feature_table),
    featureType = c(rep("Layer1", 3), rep("Layer2", 2)),
    row.names = rownames(feature_table)
  )
  fit <- IL_conbin(
    feature_table = feature_table,
    sample_metadata = sample_metadata,
    feature_metadata = feature_metadata,
    folds = 3, base_learner = "SL.mean", run_stacked = FALSE,
    run_concat = FALSE, print_learner = FALSE, family = stats::gaussian()
  )
  names(fit)
}
```

`IL_multiclass` *Integrated machine learning for multi-omics multiclass classification*

Description

Native multiclass backend used by `IntegratedLearner()` when `family = binomial()` and the outcome has more than two classes. This backend preserves the existing API while using multiclass-native probability modeling and multiclass stacking.

Usage

```
IL_multiclass(
  feature_table,
  sample_metadata,
  feature_metadata,
  feature_table_valid = NULL,
  sample_metadata_valid = NULL,
  folds = 5,
  seed = 1234,
  base_learner = "glmnet",
```

```
  base_screener = "All",
  run_screening = FALSE,
  screen_pct = NULL,
  filter_method = NULL,
  filter_pct = NULL,
  prevalence_pct = NULL,
  meta_learner = "glmnet",
  run_concat = TRUE,
  run_stacked = TRUE,
  verbose = FALSE,
  print_learner = TRUE,
  family = stats::binomial(),
  eps = 1e-15,
  ...
)
```

Arguments

feature_table	An R data frame containing multi-layer features (in rows) and samples (in columns). Column names of feature_metadata must match the row names of sample_metadata.
sample_metadata	An R data frame of metadata variables (in columns). Must have a column named subjectID describing per-subject unique identifiers. For longitudinal designs, this variable may be non-unique. Additionally, a column named Y must be present which is the continuous or binary outcome of interest. Row names of sample_metadata must match the column names of feature_table.
feature_metadata	An R data frame of feature-specific metadata across views (in columns) and features (in rows). Must have a column named featureID as a unique per-feature identifier and a column named featureType describing the source layers. Row names of feature_metadata must match the row names of feature_table.
feature_table_valid	Feature table from validation set for which prediction is desired. Must have the exact same structure as feature_table. If missing, uses feature_table.
sample_metadata_valid	Sample-specific metadata table from independent validation set when available. Must have the exact same structure as sample_metadata.
folds	How many folds in the V-fold nested cross-validation? Default is 5.
seed	Specify the seed for reproducibility. Default is 1234.
base_learner	Base learner for late fusion and early fusion. Default: SL.BART.
base_screener	Deprecated for this backend; currently ignored and kept only for backward compatibility.
run_screening	Logical; if TRUE, run supervised screening within each training fold (and again on full training data) before fitting base models.
screen_pct	Percentage of features to retain during screening ((0,100]). Applied after any optional filtering.

filter_method	Optional feature-filter method before model fitting. Supported values are 'prevalence' (top detected features by prevalence) and 'variance' (top features by caret-based variance ranking via nearZeroVar + empirical variance). If NULL, defaults to 'prevalence' when filtering is requested.
filter_pct	Optional retention percentage (in (0,100]) for the selected filter_method. Keeps the top filter_pct percent features.
prevalence_pct	Optional retention percentage (in (0,100]) for prevalence-based filtering before model fitting. Deprecated alias of filter_pct with filter_method = 'prevalence'.
meta_learner	Meta-learner for late fusion (stacked generalization). Defaults to SL.nnls.auc.
run_concat	Should early fusion be run? Default is TRUE.
run_stacked	Should stacked model (late fusion) be run? Default is TRUE.
verbose	logical; TRUE for printing SuperLearner progress. Default FALSE.
print_learner	logical; Should a detailed summary be printed? Default TRUE.
family	Allows gaussian() for continuous and binomial() for binary outcomes. Survival outcomes must be handled via IL_survival().
eps	Small positive constant used to stabilize probabilities.
...	Additional arguments (currently unused).

Value

A fitted multiclass IntegratedLearner object.

Examples

```
is.function(IL_multiclass)
if (FALSE) {
  set.seed(1)
  n <- 24
  feature_table <- rbind(
   matrix(rnorm(3 * n), nrow = 3, dimnames = list(paste0("L1_F", 1:3), paste0("S", 1:n))),
   matrix(rnorm(2 * n), nrow = 2, dimnames = list(paste0("L2_F", 1:2), paste0("S", 1:n)))
  )
  y <- rep(c("A", "B", "C"), length.out = n)
  sample_metadata <- data.frame(
    subjectID = paste0("ID", 1:n), Y = y,
    row.names = colnames(feature_table)
  )
  feature_metadata <- data.frame(
    featureID = rownames(feature_table),
    featureType = c(rep("Layer1", 3), rep("Layer2", 2)),
    row.names = rownames(feature_table)
  )
  fit <- IL_multiclass(
    feature_table = feature_table,
    sample_metadata = sample_metadata,
    feature_metadata = feature_metadata,
    folds = 3, run_stacked = FALSE, run_concat = FALSE,
```

```
    print_learner = FALSE
  )
  fit$family
}
```

ILsurv *IntegratedLearner Survival Engine*

Description

BioC-friendly survival backend used by `IntegratedLearner()` for time-to-event outcomes. This preserves the historical `ILsurv()` interface while using model fitting and fusion routines that do not depend on **mlr3proba/mlr3extralearners**.

Usage

```
ILsurv(
  feature_table,
  sample_metadata,
  feature_metadata,
  valid_feature_table = NULL,
  valid_sample_metadata = NULL,
  base_learner = "surv.coxph",
  folds = 5,
  seed = 123,
  run_screening = FALSE,
  screen_pct = NULL,
  drop_poor_performing_layers = FALSE,
  verbose = FALSE,
  do_early_fusion = TRUE,
  weight_method = c("IBS", "COX"),
  t_vec = NULL,
  t_vec_probs = c(0.05, 0.25, 0.5, 0.75, 0.95),
  layer_score = c("sum", "mean", "l2"),
  eps = 1e-12,
  weight_lambda = 0.02,
  weight_penalty = c("l2_to_uniform", "entropy"),
  weight_cap = 1,
  optim_maxit_cox = 4000,
  optim_maxit_ibs = 300,
  ibs_shrink_to_uniform = 0,
  ...
)

IL_survival(
  feature_table,
  sample_metadata,
```

```
  feature_metadata,
  valid_feature_table = NULL,
  valid_sample_metadata = NULL,
  base_learner = "surv.coxph",
  folds = 5,
  seed = 123,
  run_screening = FALSE,
  screen_pct = NULL,
  drop_poor_performing_layers = FALSE,
  verbose = FALSE,
  do_early_fusion = TRUE,
  weight_method = c("IBS", "COX"),
  t_vec = NULL,
  t_vec_probs = c(0.05, 0.25, 0.5, 0.75, 0.95),
  layer_score = c("sum", "mean", "l2"),
  eps = 1e-12,
  weight_lambda = 0.02,
  weight_penalty = c("l2_to_uniform", "entropy"),
  weight_cap = 1,
  optim_maxit_cox = 4000,
  optim_maxit_ibs = 300,
  ibs_shrink_to_uniform = 0,
  ...
)
```

Arguments

`feature_table`	An R data frame containing multi-layer features (in rows) and samples (in columns). Column names of `feature_metadata` must match the row names of `sample_metadata`.
`sample_metadata`	An R data frame of metadata variables (in columns). Must have a column named `subjectID` describing per-subject unique identifiers. For longitudinal designs, this variable may be non-unique. Additionally, a column named `Y` must be present which is the continuous or binary outcome of interest. Row names of `sample_metadata` must match the column names of `feature_table`.
`feature_metadata`	An R data frame of feature-specific metadata across views (in columns) and features (in rows). Must have a column named `featureID` as a unique per-feature identifier and a column named `featureType` describing the source layers. Row names of `feature_metadata` must match the row names of `feature_table`.
`valid_feature_table`	Validation feature table (features x samples).
`valid_sample_metadata`	Validation sample metadata containing `time` and `event`.
`base_learner`	Survival base learner.
`folds`	How many folds in the V-fold nested cross-validation? Default is 5.
`seed`	Specify the seed for reproducibility. Default is 1234.

run_screening	Logical; if TRUE, run supervised screening within each training fold (and again on full training data) before fitting base models.
screen_pct	Percentage of features to retain during screening ((0,100]). Applied after any optional filtering.
drop_poor_performing_layers	Logical; if TRUE, layers with single-layer performance below the screening threshold are removed from early and late fusion only. The threshold is AUC < 0.5 for binary and $R^2 < 0.5$ for continuous outcomes. Single-layer outputs are still retained and reported.
verbose	logical; TRUE for printing SuperLearner progress. Default FALSE.
do_early_fusion	Logical; run early fusion model.
weight_method	Late-fusion weighting method. Supported: 'IBS', 'COX'.
t_vec	Optional explicit time points for COX-based late-fusion feature construction.
t_vec_probs	Quantiles used to derive t_vec when t_vec is NULL.
layer_score	Layer summary on cumhaz increments: 'sum', 'mean', or 'l2'.
eps	Numerical lower bound for survival probabilities.
weight_lambda	COX-weight optimization regularization strength.
weight_penalty	COX-weight optimization regularizer: 'l2_to_uniform' or 'entropy'.
weight_cap	Optional cap on any one layer weight (COX method).
optim_maxit_cox	Maximum iterations for COX-weight optimization.
optim_maxit_ibs	Maximum iterations for IBS-weight optimization.
ibs_shrink_to_uniform	Optional convex shrinkage of IBS weights toward uniform.
...	Additional base-learner hyperparameters passed to base_learner. You may also pass model_args = list(...) as a named list where each entry is keyed by learner ID.

Value

List with train_out and valid_out in the same nested format as previous survival implementations.

Examples

```
identical(IL_survival, ILsurv)
if (FALSE) {
  set.seed(1)
  n <- 20
  feature_table <- rbind(
   matrix(rnorm(3 * n), nrow = 3, dimnames = list(paste0("L1_F", 1:3), paste0("S", 1:n))),
   matrix(rnorm(2 * n), nrow = 2, dimnames = list(paste0("L2_F", 1:2), paste0("S", 1:n)))
  )
```

```
  sample_metadata <- data.frame(
    subjectID = paste0("ID", 1:n),
    time = rexp(n, rate = 0.1),
    event = rbinom(n, 1, 0.6),
    row.names = colnames(feature_table)
  )
  sample_metadata$Y <- survival::Surv(sample_metadata$time, sample_metadata$event)
  feature_metadata <- data.frame(
    featureID = rownames(feature_table),
    featureType = c(rep("Layer1", 3), rep("Layer2", 2)),
    row.names = rownames(feature_table)
  )
  fit <- ILsurv(
    feature_table = feature_table,
    sample_metadata = sample_metadata,
    feature_metadata = feature_metadata,
    folds = 3
  )
  names(fit)
}
```

`IntegratedLearner` *Integrated machine learning for multi-omics prediction and classification*

Description

Performs integrated machine learning to predict a binary, continuous, or time-to-event outcome based on two or more omics layers (views). The `IntegratedLearner` function takes a training `MultiAssayExperiment` and, optionally, a validation `MultiAssayExperiment`, extracts multi-layer feature tables, and returns predicted values based on the validation set. It also performs V-fold nested cross-validation to estimate the prediction accuracy of various fusion algorithms. Two integration paradigms are supported: early and late. The software includes multiple ML models based on the `SuperLearner` R package as well as several data exploration capabilities and visualization modules in a unified estimation framework.

Usage

```
IntegratedLearner(
  MAE_train = NULL,
  MAE_valid = NULL,
  PCL_train = NULL,
  PCL_valid = NULL,
  experiment = NULL,
  assay.type = NULL,
  outcome_col = "Y",
  subject_id_col = "subjectID",
  na.rm = FALSE,
  folds = 5,
```

```
  seed = 1234,
  base_learner = "SL.BART",
  base_screener = "All",
  run_screening = FALSE,
  screen_pct = NULL,
  filter_method = NULL,
  filter_pct = NULL,
  prevalence_pct = NULL,
  meta_learner = "SL.nnls.auc",
  run_concat = TRUE,
  run_stacked = TRUE,
  drop_poor_performing_layers = FALSE,
  verbose = FALSE,
  print_learner = TRUE,
  refit.stack = FALSE,
  family = stats::gaussian(),
  ...
)
```

Arguments

MAE_train	A MultiAssayExperiment containing the training data. Each experiment corresponds to one view (omics layer), usually stored as a SummarizedExperiment or TreeSummarizedExperiment. The colData of every selected experiment must contain a column named by outcome_col describing the outcome of interest (binary, continuous, or survival-encoded), and a subject identifier column named by subject_id_col.
MAE_valid	Optional MultiAssayExperiment containing an independent validation/test set for which prediction is desired. It must contain the same set of experiments used for training, with identical assay names and identical feature (row) names for each experiment. Additional experiments in mae_test are ignored.
PCL_train	Optional list of per-layer feature matrices (legacy PCL mode for backward compatibility).
PCL_valid	Optional validation list of per-layer feature matrices.
experiment	Optional character or integer vector specifying which experiments (layers) to extract from MAE_train (and MAE_valid, if supplied). If NULL, all experiments in MAE_train are used.
assay.type	Optional character vector of assay names, one per experiment. If NULL, each selected experiment must contain exactly one assay; otherwise, the user must supply assay.type.
outcome_col	Outcome column name in MAE/PCL sample metadata. Defaults to "Y" for backward compatibility.
subject_id_col	Subject identifier column name in MAE/PCL sample metadata. Defaults to "subjectID" for backward compatibility.
na.rm	Logical; if TRUE, features with missing values are dropped after extraction.
folds	How many folds in the V-fold nested cross-validation? Default is 10.

seed	Specify the arbitrary seed value for reproducibility. Default is 1234.
base_learner	Base learner for late fusion and early fusion. Check out the SuperLearner package page for all available options. Default is `SL.BART`.
base_screener	Deprecated for IL_conbin and IL_multiclass; kept for backward compatibility and currently ignored in those backends.
run_screening	Logical; if TRUE, run supervised screening on training data within each CV fold (and again on full training data before final fitting), then apply the same selected features to fold-validation or external validation data.
screen_pct	Percentage of features to retain during screening ((0,100]).
filter_method	Optional feature-filter method for non-survival runs. Supported values: 'prevalence' and 'variance'.
filter_pct	Optional retention percentage (in (0,100]) for the selected filter_method. Keeps the top filter_pct percent features.
prevalence_pct	Optional retention percentage (in (0,100]) for prevalence-based feature filtering before model fitting. Deprecated alias of filter_pct with filter_method = 'prevalence'.
meta_learner	Meta-learner for late fusion (stacked generalization). Defaults to `SL.nnls.auc`. Check out the SuperLearner package page for all available options.
run_concat	Should early fusion be run? Default is TRUE. Uses the specified base_learner as the learning algorithm.
run_stacked	Should stacked model (late fusion) be run? Default is TRUE.
drop_poor_performing_layers	Logical; if TRUE, layers with poor single-layer performance are removed from early and late fusion only. The thresholds are AUC < 0.5 for binary, R^2 < 0.5 for continuous, and C-index < 0.5 for survival outcomes. Single-layer outputs are still retained and reported.
verbose	logical; TRUE for SuperLearner printing progress (helpful for debugging). Default is FALSE.
print_learner	logical; Should a detailed summary be printed? Default is TRUE.
refit.stack	logical; For late fusion, post-refit predictions on the entire data are returned if specified. Default is FALSE.
family	Currently allows `gaussian()` for continuous, `binomial()` for binary, or an appropriate Cox/survival family for time-to-event outcomes. Determines whether IL_conbin or IL_survival is used internally.
...	Additional arguments passed to the underlying engines.

Details

Internally, IntegratedLearner() converts the MultiAssayExperiment into tabular multi-view matrices and then calls IL_conbin for continuous/binary outcomes or IL_survival for time-to-event outcomes, depending on the specified family.

Value

A list-like IntegratedLearner object containing the trained model fits (layer-specific, stacked, and/or concatenated models), cross-validated performance estimates, and predicted values for training and, if supplied, validation data.

Author(s)

Himel Mallick, <him4004@med.cornell.edu>

Examples

```
is.function(IntegratedLearner)
if (FALSE) {
  set.seed(1)
  n <- 20
  feature_table <- rbind(
   matrix(rnorm(3 * n), nrow = 3, dimnames = list(paste0("L1_F", 1:3), paste0("S", 1:n))),
   matrix(rnorm(2 * n), nrow = 2, dimnames = list(paste0("L2_F", 1:2), paste0("S", 1:n)))
  )
  sample_metadata <- data.frame(
    subjectID = paste0("ID", 1:n), Y = rnorm(n),
    row.names = colnames(feature_table)
  )
  feature_metadata <- data.frame(
    featureID = rownames(feature_table),
    featureType = c(rep("Layer1", 3), rep("Layer2", 2)),
    row.names = rownames(feature_table)
  )
  pcl <- list(
    feature_table = feature_table,
    sample_metadata = sample_metadata,
    feature_metadata = feature_metadata
  )
  fit <- IntegratedLearner(
    PCL_train = pcl, folds = 3, base_learner = "SL.mean",
    run_stacked = FALSE, run_concat = FALSE, print_learner = FALSE,
    family = stats::gaussian()
  )
  names(fit)
}
```

---

mir_all *TCGA BRCA microRNA-level table*

Description

microRNA and associated covariate/outcome table for TCGA BRCA examples.

Usage

```
data(mir_all)
```

Format

A data frame with one row per patient and covariates plus microRNA features in columns.

Source

TCGA-derived example data bundled for package tests/tutorials.

NLIBD *NLIBD binary-outcome PCL fixture*

Description

External validation cohort example data in PCL format for binary outcome modeling.

Usage

```
data(NLIBD)
```

Format

A list with components:

feature_table data frame of features (rows) by samples (columns).

sample_metadata data frame of sample-level metadata with `Y` and `subjectID`.

feature_metadata data frame of feature-level metadata with `featureID` and `featureType`.

Source

Packaged example data for tutorials and tests.

NNLS *NNLS function to optimize weights of several base learners*

Description

NNLS function to optimize weights of several base learners

Usage

```
NNLS(x, y, wt)
```

Arguments

x	x
y	y
wt	wt

Value

Solution of the quadratic programming problem

Examples

```
x <- cbind(c(0.1, 0.4, 0.6, 0.9), c(0.2, 0.5, 0.7, 0.8))
y <- c(0.15, 0.45, 0.65, 0.85)
fit <- NNLS(x = x, y = y, wt = rep(1, nrow(x)))
fit$solution
```

plot.learner *Plot the summary curves produced by an IntegratedLearner object*

Description

Plots outcome-appropriate performance summaries for the training set and, if available, the validation set produced by an `IntegratedLearner` fit. Depending on the outcome family, this may include ROC curves, R-squared bar plots, multiclass one-vs-rest ROC curves, or survival AUC / Kaplan-Meier panels.

Usage

```
## S3 method for class 'learner'
plot(
  x,
  y = NULL,
  label_size = 8,
  label_x = 0.3,
  vjust = 0.1,
  rowwise_plot = TRUE,
  ...
)
```

Arguments

x	Fitted IntegratedLearner object.
y	Unused (required for S3 signature).
label_size	Optional numeric label size for subplot tags. Default is 8.
label_x	Optional single value or vector of x positions for subplot labels, relative to each subplot. Defaults to 0.3 for all labels.
vjust	Adjusts the vertical position of each label. More positive values move the label further down on the plot canvas. Can be a single value (applied to all labels) or a vector of values (one for each label). Default is 0.1.
rowwise_plot	If both train and test data are available, should the train and test plots be arranged row-wise? Default is TRUE. If FALSE, plots are aligned column-wise.
...	Additional arguments (currently unused).

Value

A list whose $plot entry is a **ggplot2/cowplot** composite object, along with the underlying tabular data used to generate the plot.

---

predict.learner *Make predictions using a trained 'IntegratedLearner' model*

Description

This function makes predictions using a trained 'IntegratedLearner' model for new samples for which predictions are to be made

Usage

```
## S3 method for class 'learner'
predict(
  object,
  feature_table_valid = NULL,
  sample_metadata_valid = NULL,
  feature_metadata = NULL,
  outcome_col = NULL,
  subject_id_col = NULL,
  ...
)
```

Arguments

object	Fitted 'IntegratedLearner' object
feature_table_valid	Feature table from validation set. Must have the exact same structure as feature_table.
sample_metadata_valid	OPTIONAL (can provide feature_table_valid and not this): Sample-specific metadata table from independent validation set. If provided, it must have the exact same structure as sample_metadata.
feature_metadata	Matrix containing feature names and their corresponding layers. Must be same as that provided in IntegratedLearner object.
outcome_col	Optional outcome column name in sample_metadata_valid. If NULL, uses the mapping stored in object$column_map (or falls back to "Y").
subject_id_col	Optional subject identifier column name in sample_metadata_valid. If NULL, uses the mapping stored in object$column_map (or falls back to "subjectID").
...	Additional arguments (currently unused)

Value

Predicted values

---

predict.SL.BART *Predict function for SL.BART*

Description

Predict function for SL.BART

Usage

```
## S3 method for class 'SL.BART'
predict(object, newdata, family, X = NULL, Y = NULL, ...)
```

Arguments

object	Fitted SL.BART model object
newdata	Data frame for prediction
family	Family object passed through (unused)
X	Training design matrix (unused)
Y	Training outcome (unused)
...	Additional arguments (unused)

Value

Prediction from the SL.BART

---

predict.SL.nnls.auc *Predict function for SL.nnls.auc*

Description

Predict function for SL.nnls.auc

Usage

```
## S3 method for class 'SL.nnls.auc'
predict(object, newdata, ...)
```

Arguments

object	Fitted SL.nnls.auc model
newdata	Validation layer-level predictions
...	Additional arguments (unused)

Value

Prediction from the meta-learner

pregnancy *Pregnancy continuous-outcome PCL fixture*

Description

Multi-omics pregnancy example data in PCL format for continuous outcome modeling.

Usage

```
data(pregnancy)
```

Format

A list with components:

feature_table data frame of features (rows) by samples (columns).

sample_metadata data frame of sample-level metadata with `Y` and `subjectID`.

feature_metadata data frame of feature-level metadata with `featureID` and `featureType`.

Source

Packaged example data for tutorials and tests.

PRISM *PRISM binary-outcome PCL fixture*

Description

PRISM cohort example data in PCL format for binary outcome modeling.

Usage

```
data(PRISM)
```

Format

A list with components:

feature_table data frame of features (rows) by samples (columns).

sample_metadata data frame of sample-level metadata with `Y` and `subjectID`.

feature_metadata data frame of feature-level metadata with `featureID` and `featureType`.

Source

Packaged example data for tutorials and tests.

SL.BART *Wrapper for bartMachine learner*

Description

Support bayesian additive regression trees via the bartMachine package.

Usage

```
SL.BART(
  Y,
  X,
  newX,
  family,
  obsWeights,
  id,
  num_trees = 50,
  num_burn_in = 250,
  verbose = FALSE,
  alpha = 0.95,
  beta = 2,
  k = 2,
  q = 0.9,
  nu = 3,
  num_iterations_after_burn_in = 1000,
  serialize = TRUE,
  seed = 5678,
  ...
)
```

Arguments

Y	Outcome variable
X	Covariate dataframe
newX	Optional dataframe to predict the outcome
family	'gaussian' for regression, 'binomial' for binary classification
obsWeights	Optional observation-level weights (supported but not tested)
id	Optional id to group observations from the same unit (not used currently).
num_trees	The number of trees to be grown in the sum-of-trees model.
num_burn_in	Number of MCMC samples to be discarded as 'burn-in'.
verbose	Prints information about progress of the algorithm to the screen.
alpha	Base hyperparameter in tree prior for whether a node is nonterminal or not.
beta	Power hyperparameter in tree prior for whether a node is nonterminal or not.

k	For regression, k determines the prior probability that E(Y|X) is contained in the interval (y_{min}, y_{max}), based on a normal distribution. For example, when k=2, the prior probability is 95 percent. For classification, k determines the prior probability that E(Y|X) is between (-3,3). Note that a larger value of k results in more shrinkage and a more conservative fit.
q	Quantile of the prior on the error variance at which the data-based estimate is placed. Note that the larger the value of q, the more aggressive the fit as you are placing more prior weight on values lower than the data-based estimate. Not used for classification.
nu	Degrees of freedom for the inverse chi^2 prior. Not used for classification.
num_iterations_after_burn_in	Number of MCMC samples to draw from the posterior distribution of f(x).
serialize	If TRUE, bartMachine results can be saved to a file, but will require additional RAM.
seed	Seed for reproducibility passed to bartMachine.
...	Additional arguments (not used)

Value

A list with elements `pred` (predictions for `newX`) and `fit` (the fitted model object).

Examples

```
is.function(SL.BART)
if (FALSE) {
  set.seed(1)
  X <- data.frame(x1 = rnorm(20), x2 = rnorm(20))
  Y <- rnorm(20)
  fit <- SL.BART(
    Y = Y, X = X, newX = X[1:3, ],
    family = stats::gaussian(),
    obsWeights = rep(1, nrow(X)), id = seq_len(nrow(X)),
    num_trees = 5, num_burn_in = 5, num_iterations_after_burn_in = 20
  )
  fit$pred
}
```

SL.enet *Elastic net regression, including lasso and ridge with optimized alpha and lambda*

Description

Penalized regression using elastic net. Alpha = 0 corresponds to ridge regression and alpha = 1 corresponds to Lasso.

See `vignette('glmnet_beta', package = 'glmnet')` for a nice tutorial on glmnet.

Usage

```
SL.enet(
  Y,
  X,
  newX,
  family,
  obsWeights,
  id,
  alpha = seq(0, 1, 0.1),
  nfolds = 10,
  nlambda = 100,
  useMin = TRUE,
  loss = "deviance",
  ...
)
```

Arguments

Y	Outcome variable
X	Covariate dataframe
newX	Dataframe to predict the outcome
family	'gaussian' for regression, 'binomial' for binary classification. Untested options: 'multinomial' for multiple classification or 'mgaussian' for multiple response, 'poisson' for non-negative outcome with proportional mean and variance, 'cox'.
obsWeights	Optional observation-level weights
id	Optional id to group observations from the same unit (not used currently).
alpha	Elastic net mixing parameter, range 0 to 1. 0 = ridge regression and 1 = lasso.
nfolds	Number of folds for internal cross-validation to optimize lambda.
nlambda	Number of lambda values to check, recommended to be 100 or more.
useMin	If TRUE use lambda that minimizes risk, otherwise use 1 standard-error rule which chooses a higher penalty with performance within one standard error of the minimum (see Breiman et al. 1984 on CART for background).
loss	Loss function, can be 'deviance', 'mse', or 'mae'. If family = binomial can also be 'auc' or 'class' (misclassification error).
...	Any additional arguments are passed through to cv.glmnet.

Value

A list with elements pred (predictions for newX) and fit (cross-validated glmnet fit metadata).

References

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. Journal of statistical software, 33(1), 1.

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12(1), 55-67.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological), 267-288.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(2), 301-320.

See Also

`predict.SL.glmnet` `cv.glmnet` `glmnet`

Examples

```
set.seed(1)
X <- data.frame(x1 = rnorm(20), x2 = rnorm(20))
Y <- rnorm(20)
fit <- SL.enet(
  Y = Y, X = X, newX = X[1:3, ], family = stats::gaussian(),
  obsWeights = rep(1, nrow(X)), id = seq_len(nrow(X)),
  alpha = c(0, 0.5, 1), nfolds = 3, nlambda = 10
)
head(fit$pred)
```

SL.glmnet2 *Elastic net regression, including lasso and ridge with a fixed alpha*

Description

Penalized regression using elastic net. Alpha = 0 corresponds to ridge regression and alpha = 1 corresponds to Lasso.

See `vignette('glmnet_beta', package = 'glmnet')` for a nice tutorial on glmnet.

Usage

```
SL.glmnet2(
  Y,
  X,
  newX,
  family,
  obsWeights,
  id,
  alpha = 0.5,
  nfolds = 10,
  nlambda = 100,
  useMin = TRUE,
  loss = "deviance",
  ...
)
```

Arguments

Y	Outcome variable
X	Covariate dataframe
newX	Dataframe to predict the outcome
family	'gaussian' for regression, 'binomial' for binary classification. Untested options: 'multinomial' for multiple classification or 'mgaussian' for multiple response, 'poisson' for non-negative outcome with proportional mean and variance, 'cox'.
obsWeights	Optional observation-level weights
id	Optional id to group observations from the same unit (not used currently).
alpha	Elastic net mixing parameter, range 0 to 1. 0 = ridge regression and 1 = lasso.
nfolds	Number of folds for internal cross-validation to optimize lambda.
nlambda	Number of lambda values to check, recommended to be 100 or more.
useMin	If TRUE use lambda that minimizes risk, otherwise use 1 standard-error rule which chooses a higher penalty with performance within one standard error of the minimum (see Breiman et al. 1984 on CART for background).
loss	Loss function, can be 'deviance', 'mse', or 'mae'. If family = binomial can also be 'auc' or 'class' (misclassification error).
...	Any additional arguments are passed through to cv.glmnet.

Value

A list with elements `pred` (predictions for `newX`) and `fit` (cross-validated glmnet fit metadata).

References

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. Journal of statistical software, 33(1), 1.

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12(1), 55-67.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological), 267-288.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(2), 301-320.

See Also

`predict.SL.glmnet` `cv.glmnet` `glmnet`

Examples

```
set.seed(1)
X <- data.frame(x1 = rnorm(20), x2 = rnorm(20))
Y <- rnorm(20)
fit <- SL.glmnet2(
  Y = Y, X = X, newX = X[1:3, ], family = stats::gaussian(),
```

```
  obsWeights = rep(1, nrow(X)), id = seq_len(nrow(X)),
  nfolds = 3, nlambda = 10
)
head(fit$pred)
```

SL.horseshoe *Horseshoe regression*

Description

Horseshoe regression

Usage

```
SL.horseshoe(
  Y,
  X,
  newX,
  family,
  prior = "horseshoe",
  N = 20000L,
  burnin = 1000L,
  thinning = 1L,
  ...
)
```

Arguments

Y	Outcome variable
X	Covariate data frame
newX	Dataframe to predict the outcome
family	'gaussian' for regression, 'binomial' for binary classification. Untested options: 'poisson' for for integer or count data
prior	prior for regression coefficients to use. 'Horseshoe' by default. Untested options: ridge regression (prior='rr' or prior='ridge'), lasso regression (prior='lasso') and horseshoe+ regression (prior='hs+' or prior='horseshoe+')
N	Number of posterior samples to generate.
burnin	Number of burn-in samples.
thinning	Desired level of thinning.
...	other parameters passed to bayesreg function

Value

SL object

Examples

```
is.function(SL.horseshoe)
if (FALSE) {
  set.seed(1)
  X <- data.frame(x1 = rnorm(20), x2 = rnorm(20))
  Y <- rnorm(20)
  fit <- SL.horseshoe(
    Y = Y, X = X, newX = X[1:3, ],
    family = stats::gaussian()
  )
  fit$pred
}
```

SL.LASSO *Elastic net regression, including lasso and ridge with a fixed alpha*

Description

Penalized regression using elastic net. Alpha = 0 corresponds to ridge regression and alpha = 1 corresponds to Lasso.

See `vignette('glmnet_beta', package = 'glmnet')` for a nice tutorial on glmnet.

Usage

```
SL.LASSO(
  Y,
  X,
  newX,
  family,
  obsWeights,
  id,
  alpha = 1,
  nfolds = 10,
  nlambda = 100,
  useMin = TRUE,
  loss = "deviance",
  ...
)
```

Arguments

Y	Outcome variable
X	Covariate dataframe
newX	Dataframe to predict the outcome
family	'gaussian' for regression, 'binomial' for binary classification. Untested options: 'multinomial' for multiple classification or 'mgaussian' for multiple response, 'poisson' for non-negative outcome with proportional mean and variance, 'cox'.

obsWeights	Optional observation-level weights
id	Optional id to group observations from the same unit (not used currently).
alpha	Elastic net mixing parameter, range 0 to 1. 0 = ridge regression and 1 = lasso.
nfolds	Number of folds for internal cross-validation to optimize lambda.
nlambda	Number of lambda values to check, recommended to be 100 or more.
useMin	If TRUE use lambda that minimizes risk, otherwise use 1 standard-error rule which chooses a higher penalty with performance within one standard error of the minimum (see Breiman et al. 1984 on CART for background).
loss	Loss function, can be 'deviance', 'mse', or 'mae'. If family = binomial can also be 'auc' or 'class' (misclassification error).
...	Any additional arguments are passed through to cv.glmnet.

Value

A list with elements pred (predictions for newX) and fit (cross-validated glmnet fit metadata).

References

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. Journal of statistical software, 33(1), 1.

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12(1), 55-67.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological), 267-288.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(2), 301-320.

See Also

predict.SL.glmnet cv.glmnet glmnet

Examples

```
set.seed(1)
X <- data.frame(x1 = rnorm(20), x2 = rnorm(20))
linpred <- 0.5 * X$x1 - 0.25 * X$x2
Y <- stats::rbinom(nrow(X), 1, stats::plogis(linpred))
fit <- SL.LASSO(
  Y = Y, X = X, newX = X[1:3, ], family = stats::binomial(),
  obsWeights = rep(1, nrow(X)), id = seq_len(nrow(X)),
  nfolds = 3, nlambda = 10
)
head(fit$pred)
```

SL.mxBART *mxBART SuperLearner wrapper*

Description

SuperLearner wrapper for mixed-effects BART using the **mxBART** package. This learner is optional and requires **mxBART** to be installed.

Usage

```
SL.mxBART(
  Y,
  X,
  newX,
  family,
  obsWeights,
  id,
  sparse = FALSE,
  ntree = 50,
  ndpost = 1000,
  nskip = 100,
  keepevery = 10,
  mxps = list(list(prior = 1, df = 3, scale = 1)),
  ...
)
```

Arguments

Y	Outcome variable.
X	Covariate data frame (training).
newX	Covariate data frame (prediction).
family	A family object; typically gaussian() or binomial().
obsWeights	Optional observation weights (currently unused).
id	Optional grouping id for mixed-effects BART.
sparse	Logical; passed to mxBART::mxbart.
ntree	Number of trees.
ndpost	Number of posterior draws.
nskip	Number of burn-in draws.
keepevery	Thinning interval.
mxps	Prior specification list passed to mxBART::mxbart.
...	Additional arguments passed to mxBART::mxbart.

Value

A list with elements `pred` and `fit` (SuperLearner convention).

Examples

```
is.function(SL.mxBART)
if (FALSE) {
  set.seed(1)
  X <- data.frame(x1 = rnorm(20), x2 = rnorm(20))
  Y <- rnorm(20)
  fit <- SL.mxBART(
    Y = Y, X = X, newX = X[1:3, ], family = stats::gaussian(),
    obsWeights = rep(1, nrow(X)), id = rep(1:5, each = 4),
    ntree = 10, ndpost = 20, nskip = 10, keepevery = 2
  )
  fit$pred
}
```

SL.nnls.auc *Combined SuperLearner function for both NNLS/AUC maximization*

Description

Combined SuperLearner function for both NNLS/AUC maximization

Usage

```
SL.nnls.auc(Y, X, newX, family, obsWeights, bounds = c(0, Inf), ...)
```

Arguments

`Y`	Outcome matrix from metalearner
`X`	Layer-level predictions used to train the metalearner
`newX`	Layer-level predictions for validation data
`family`	Family object
`obsWeights`	Observation weights
`bounds`	Lower/upper bounds for weights (binomial case)
`...`	Additional arguments passed through

Value

Estimated meta-learner coefficients and predictions

Examples

```
set.seed(1)
X <- data.frame(m1 = runif(20), m2 = runif(20))
Y <- rnorm(20)
fit <- SL.nnls.auc(
  Y = Y, X = X, newX = X[1:4, ],
  family = stats::gaussian(), obsWeights = rep(1, nrow(X))
)
head(fit$pred)
```

`update.learner` *Update IntegratedLearner fit object based on layers available in the test set*

Description

Allow update of IntegratedLearner if only a subset of omics layers are available in test set. If all layers and features match, it calls predict.learner()

Usage

```
## S3 method for class 'learner'
update(
  object,
  feature_table_valid,
  sample_metadata_valid = NULL,
  feature_metadata_valid,
  outcome_col = NULL,
  subject_id_col = NULL,
  seed = 1234,
  verbose = FALSE,
  ...
)
```

Arguments

`object` fitted 'IntegratedLearner' object

`feature_table_valid`
Feature table from validation set. It should be a data frame with features in rows and samples in columns. Feature names should be a subset of training data feature names.

`sample_metadata_valid`
OPTIONAL (can provide feature_table_valid and not this): Sample-specific metadata table from independent validation set. If provided, it must have the exact same structure as sample_metadata. Default is NULL.

`feature_metadata_valid`
Matrix containing feature names and their corresponding layers. Must be subset of feature_metadata provided in IntegratedLearner object.

outcome_col	Optional outcome column name in sample_metadata_valid. If NULL, uses object$column_map$outcome_col (or "Y").
subject_id_col	Optional subject ID column name in sample_metadata_valid. If NULL, uses object$column_map$subject_id_col (or "subjectID").
seed	Seed for reproducibility. Default is 1234.
verbose	Should a summary of fits/ results be printed. Default is FALSE
...	Additional arguments (unused)

Value

SL object

Examples

```
is.function(getS3method("update", "learner"))
if (FALSE) {
  # Build a fit with IntegratedLearner() first, then update with reduced layers.
  update(
    object = fit,
    feature_table_valid = feature_table_valid,
    sample_metadata_valid = sample_metadata_valid,
    feature_metadata_valid = feature_metadata_valid
  )
}
```

Index