Package ‘IntegratedLearner’

November 23, 2023

Title Integrated machine learning for multi-omics prediction and classification

Version 0.0.1

Date 2023-08-02

Authors Himel Mallick (Cornell University) <him4004@med.cornell.edu>, Anupreet Porwal (Google) <porwaa@uw.edu>

Description

IntegratedLearner provides an integrated machine learning framework to 1) consolidate predictions by borrowing information across several longitudinal and cross-sectional omics data layers, 2) decipher the mechanistic role of individual omics features that can potentially lead to new sets of testable hypotheses, and 3) quantify uncertainty of the integration process. Three types of integration paradigms are supported: early, late, and intermediate. 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.

Depends R (>= 3.6)

Imports SuperLearner, tidyverse, caret, mcmcplots, glmnetUtils, ROCR, quadprog, nloptr

Suggests data.table, knitr

License MIT + file LICENSE

LazyData TRUE

RoxygenNote 7.2.3

VignetteBuilder knitr

Encoding UTF-8

R topics documented:

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

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

Description

Performs integrated machine learning to predict a binary or continuous outcome based on two or more omics layers (views). The `IntegratedLearner` function takes a training set (Y, X1, X2,...,Xn) and returns the predicted values based on a validation set. It also performs V-fold nested cross-validation to estimate the prediction accuracy of various fusion algorithms. Three types of integration paradigms are supported: early, late, and intermediate. 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(
  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",
  meta_learner = "SL.nnls.auc",
  run_concat = TRUE,
  run_stacked = TRUE,
  verbose = FALSE,
  print_learner = TRUE,
  refit.stack = FALSE,
  family = gaussian()
)
```

Arguments

`feature_table`	An R data frame containing multiview 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 is expected to have non-unique values. Additionally, a column named `Y` must be present which is the outcome of interest (can be binary or continuous). 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` describing per-feature unique identifiers. Additionally, a column named `featureType` should describe the corresponding 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` for `feature_table_valid`.
`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 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 user manual for all available options. Default is `` `SL.BART` ``.
`base_screener`	Whether to screen variables before fitting base models? `All` means no screening which is the default. Check out the SuperLearner user manual for all available options.
`meta_learner`	Meta-learner for late fusion (stacked generalization). Defaults to `` `SL.nnls.auc` ``. Check out the SuperLearner user manual 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.
`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 is returned if specified. Default is FALSE.
`family`	Currently allows `` `gaussian()` `` for continuous or `` `binomial()` `` for binary outcomes.

Value

A `SuperLearner` object containing the trained model fits.

Author(s)

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

`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

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

Description

Plots the R^2/AUC curves for the training (and test, if provided) set produced by IntegratedLearner object

Usage

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

Arguments

`fit`	fitted "IntegratedLearner" object
`label_size`	(optional) Numerical value indicating the label size. Default is 8.
`label_x`	(optional) Single value or vector of x positions for plot labels, relative to each subplot. Defaults to 0.3 for all labels. (Each label is placed all the way to the left of each plot.)
`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 is available, should the train and test plots be rowwise_plot. Default is TRUE. If FALSE, plots are aligned column-wise.

Value

ggplot2 object

`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(
  fit,
  feature_table_valid = NULL,
  sample_metadata_valid = NULL,
  feature_metadata = NULL
)
```

Arguments

`fit`	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.

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`	object
`newdata`	newdata

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`	object
`newdata`	newdata

Value

Prediction from the meta-learner

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 = F,
  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%. 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.
`...`	Additional arguments (not used)

`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, 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.

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

```
# Load a test dataset.
data(PimaIndiansDiabetes2, package = "mlbench")
data = PimaIndiansDiabetes2

# Omit observations with missing data.
data = na.omit(data)

Y = as.numeric(data$diabetes == "pos")
X = subset(data, select = -diabetes)

set.seed(1, "L'Ecuyer-CMRG")

sl = SuperLearner(Y, X, family = binomial(),
                  SL.library = c("SL.mean", "SL.glm", "SL.glmnet"))
sl
```

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, 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.

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

```
# Load a test dataset.
data(PimaIndiansDiabetes2, package = "mlbench")
data = PimaIndiansDiabetes2

# Omit observations with missing data.
data = na.omit(data)

Y = as.numeric(data$diabetes == "pos")
X = subset(data, select = -diabetes)

set.seed(1, "L'Ecuyer-CMRG")

sl = SuperLearner(Y, X, family = binomial(),
                  SL.library = c("SL.mean", "SL.glm", "SL.glmnet"))
sl
```

`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

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, 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.

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

```
# Load a test dataset.
data(PimaIndiansDiabetes2, package = "mlbench")
data = PimaIndiansDiabetes2

# Omit observations with missing data.
data = na.omit(data)

Y = as.numeric(data$diabetes == "pos")
X = subset(data, select = -diabetes)

set.seed(1, "L'Ecuyer-CMRG")
```

```
sl = SuperLearner(Y, X, family = binomial(),
                  SL.library = c("SL.mean", "SL.glm", "SL.glmnet"))
sl
```

`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`	Y
`X`	X

Value

Estimated meta-learner coefficients and predictions

`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(
  fit,
  feature_table_valid,
  sample_metadata_valid = NULL,
  feature_metadata_valid,
  seed = 1234,
  verbose = FALSE
)
```

Arguments

`fit`	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.
`seed`	Seed for reproducibility. Default is 1234.
`verbose`	Should a summary of fits/ results be printed. Default is FALSE

Value

SL object

Index