Package 'EnsembleBase'

October 12, 2022

Type Package

Title Extensible Package for Parallel, Batch Training of Base Learners for Ensemble Modeling

Version 1.0.2

Date 2016-09-13

Author Alireza S. Mahani, Mansour T.A. Sharabiani

Maintainer Alireza S. Mahani <alireza.s.mahani@gmail.com>

Description Extensible S4 classes and methods for batch training of regression and classification algorithms such as Random Forest, Gradient Boosting Machine, Neural Network, Support Vector Machines, K-Nearest Neighbors, Penalized Regression (L1/L2), and Bayesian Additive Regression Trees. These algorithms constitute a set of 'base learners', which can subsequently be combined together to form ensemble predictions. This package provides cross-validation wrappers to allow for downstream application of ensemble integration techniques, including best-error selection. All base learner estimation objects are retained, allowing for repeated prediction calls without the need for re-training. For large problems, an option is provided to save estimation objects to disk, along with prediction methods that utilize these objects. This allows users to train and predict with large ensembles of base learners without being constrained by system RAM.

License GPL (>= 2)

Depends kknn,methods

Imports gbm,nnet,e1071,randomForest,doParallel,foreach,glmnet,bartMachine

NeedsCompilation no

Repository CRAN

Date/Publication 2016-09-13 22:30:52

R topics documented:

`ALL.Regression.Config-class` *Classes* "KNN.Regression.Config", "NNET.Regression.Config", "RF.Regression.Config", "SVM.Regression.Config", "GBM.Regression.Config", "PENREG.Regression.Config", "BART.Regression.Config"

Description

These base learner configuration objects contain tuning parameters needed for training base learner algorithms. Names are identical to those used in implementation packages. See documentation for those packages for detailed definitions.

Objects from the Class

These objects are typically constructed via calls to `make.configs` and `make.instances`.

Slots

For `KNN.Regression.Config`:

Object of class `"character"`, defining the weighting function applied to neighbors as a function of distance from target point. Options include "rectangular", "epanechnikov", "triweight", and "gaussian".

`kernel` `k`: Object of class `"numeric"`, defining the number of nearest neighbors to include in prediction for each target point. For `NNET.Regression.Config`:

`decay`: Object of class `"numeric"`, defining the weight decay parameter.

`size`: Object of class `"numeric"`, defining the number of hidden-layer neurons.

`maxit`**:** Object of class `"numeric"`, defining the maximum number of iterations in the training. For `RF.Regression.Config`:

`ntree`**:** Object of class `"numeric"`, defining the number of trees in the random forest.

`nodesize`**:** Object of class `"numeric"`, defining the minimum size of terminal nodes.

`mtry.mult`**:** Object of class `"numeric"`, defining the multiplier of the default value for `mtry` parameter in the `randomForest` function call. For `SVM.Regression.Config`:

`cost`**:** Object of class `"numeric"`, defining the cost of constraint violation.

`epsilon`**:** Object of class `"numeric"`, the parameter of insensitive-loss function.

`kernel`**:** Object of class `"character"`, the kernel used in SVM training and prediction. Options include "linear", "polynomial", "radial", and "sigmoid". For `GBM.Regression.Config`:

`n.trees`**:** Object of class `"numeric"`, defining the number of trees to fit.

`interaction.depth`**:** Object of class `"numeric"`, defining th maximum depth of variable interactions.

codeshrinkage: Object of class `"numeric"`, defining the shrinkage parameter applied to each tree in expansion.

`bag.fraction`**:** Object of class `"numeric"`, defining the fraction of training set observations randomly selected to propose the next tree in the expansion. For `PENREG.Regression.Config`:

`alpha`**:** Object of class `"numeric"`, defining the mix of L1 and L2 penalty. Must be between 0.0 and 1.0.

`lambda`**:** Object of class `"numeric"`, defining the shrinkage parameter. Must be non-negative. For `BART.Regression.Config`:

`num_trees`**:** Object of class `"numeric"`, defining the number of trees to be grown in the sum-of-trees model. Must be a positive integer.

`k`**:** Object of class `"numeric"`, controlling the degree of shrinkage and hence conservativeness of the fit. Must be positive.

`q`**:** Object of class `"numeric"`, defining quantile of the prior on the error variance at which the data-based estimate is placed. Higher values of this parameter lead to a more aggressive fit.

`nu`**:** Object of class `"numeric"`, defining degrees of freedom for the inverse chi-squared prior. Must be a positive integer.

Extends

Class `"Regression.Config"`, directly. Class `"BaseLearner.Config"`, by class "Regression.Config", distance 2.

Methods

BaseLearner.Fit `signature(object = "KNN.Regression.Config")`: ...

BaseLearner.Fit `signature(object = "NNET.Regression.Config")`: ...

BaseLearner.Fit `signature(object = "RF.Regression.Config")`: ...

BaseLearner.Fit `signature(object = "SVM.Regression.Config")`: ...

BaseLearner.Fit `signature(object = "GBM.Regression.Config")`: ...

BaseLearner.Fit `signature(object = "PENREG.Regression.Config")`: ...

BaseLearner.Fit `signature(object = "BART.Regression.Config")`: ...

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

See Also

make.configs, make.instances, make.configs.knn.regression, make.configs.nnet.regression, make.configs.rf.regression, make.configs.svm.regression, make.configs.gbm.regression, "Regression.Config", "BaseLearner.Config"

ALL.Regression.FitObj-class *Classes* "KNN.Regression.FitObj", "NNET.Regression.FitObj", "RF.Regression.FitObj", "SVM.Regression.FitObj", "GBM.Regression.FitObj", "PENREG.Regression.FitObj", "BART.Regression.FitObj"

Description

Objects returned by base learner training functions.

Objects from the Class

Objects can be created by calls of the form `new("KNN.Regression.FitObj", ...)`.

Slots

All classes inherit slots `config`, `est`, and `pred` from "Regression.FitObj". Some base learners may have additional slots as described below.

For `KNN.Regression.FitObj`:

`formula`: Object of class "formula", copy of same argument from training call BaseLearner.Fit.

`data`: Object of class "data.frame", copy of same argument from training call BaseLearner.Fit.

For `NNET.Regession.FitObj`:

`y.range`: Object of class "numeric", range of response variable in training data. This is used for scaling of data during prediction so that it falls between 0 and 1 for regression tasks.

`y.min`: Object of class "numeric", minimum of response variable in training data. This is used for scaling of data during prediction so that it falls between 0 and 1 for regression tasks. For `PENREG.Regression.FitObj` and `BART.Regression.FitObj`:

`mm`: A list containing data structures needed for creating the matrix of predictors and the response variable from the formula and data frame.

Extends

Class "Regression.FitObj", directly. Class "BaseLearner.FitObj", by class "Regression.FitObj", distance 2.

Methods

None.

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

See Also

"`BaseLearner.FitObj`", "`Regression.FitObj`"

Examples

```
showClass("KNN.Regression.FitObj")
```

`BaseLearner.Batch.FitObj-class`
Classes `"BaseLearner.Batch.FitObj"` *and* `"Regression.Batch.FitObj"`

Description

Classes for containing base learner batch training output.

Objects from the Class

Class `"BaseLearner.Batch.FitObj"` is virtual; therefore No objects may be created from it. Class `"Regression.Batch.FitObj"` extends `"BaseLearner.Batch.FitObj"` and is the output of function `Regression.Batch.Fit`.

Slots

`fitobj.list`: Object of class `"list"`, containing the `BaseLearner.FitObj` outputs of lower-level `BaseLearner.Fit` function calls.

`config.list`: Object of class `"list"`, containing the list of configuration objects for each base learner fit. This list is typically the output of make.configs function call.

`filemethod`: Object of class `"logical"`, indicating whether file method is used for storing the estimation objects.

`tmpfiles`: Object of class `"OptionalCharacter"`, containing (if applicable) the vector of filepaths used for storing estimation objects, if `filemethod==TRUE`. For `Regression.Batch.FitObj` (in addition to above slots):

`pred`: Object of class `"matrix"`, with each column containing the predictions of one base learner.

`y`: Object of class `"numeric"`, containing the response variable corresponding to the training set.

Methods

No methods defined with class "BaseLearner.Batch.FitObj" in the signature.

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

See Also

`Regression.Batch.Fit`

Examples

```
showClass("BaseLearner.Batch.FitObj")
```

BaseLearner.Config-class

Classes "BaseLearner.Config", "Regression.Config"

Description

Base classes in the configuration class hierarchy.

Objects from the Class

"BaseLearner.Config" is a virtual Class: No objects may be created from it. "Regression.Config" is a base class for configuration classes of specific base learners, such as `SVM.Regression.Config`; therefore, there is typically no need to generate objects from this base class directly.

Extends

Class "`Regression.Config`" extends class "`BaseLearner.Config`", directly.

Methods

No methods defined with class "BaseLearner.Config" or "Regression.Config" in the signature.

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

See Also

`KNN.Regression.Config`, `RF.Regression.Config`, `NNET.Regression.Config`, `GBM.Regression.Config`, `SVM.Regression.Config`

Examples

```
showClass("BaseLearner.Config")
```

`BaseLearner.CV.Batch.FitObj-class` *Classes* `"BaseLearner.CV.Batch.FitObj"` *and* `"Regression.CV.Batch.FitObj"`

Description

Classes for containing base learner batch CV training output.

Objects from the Class

`BaseLearner.CV.Batch.FitObj` is virtual Class; therefore, no objects may be created from it. Class `Regression.CV.Batch.FitObj` is the output of `Regression.CV.Batch.Fit` function.

Slots

`fitobj.list`**:** Object of class `"list"`, contains a list of objects of class `BaseLearner.CV.FitObj`, one per base learner instance.

`instance.list`**:** Object of class `"Instance.List"`, the list of base learner instance passed to the function `Regression.CV.Batch.Fit` that produces the object.

`filemethod`**:** Object of class `"logical"`, the boolean flag indicating whether estimation objects are saved to files or help in memory.

`tmpfiles`**:** Object of class `"OptionalCharacter"`, list of temporary files used for storing estimation objects (if any).

`tmpfiles.index.list`**:** Object of class `"list"`, with elements `start` and `end`, holding the start and end indexes into the tempfiles vector for each of the base learner instances trained.

`tvec`**:** Execution times for each base learner in the batch. Note: Currently implemented for serial execution only. In addition, `Regression.CV.Batch.FitObj` contains the following slots:

`pred`**:** Object of class `"matrix"`, with each column being the training-set prediction for one of base learner instances.

`y`**:** Object of class `"OptionalNumeric"`, holding the response variable values for training set. This slot can be `NULL` for memory efficiency purposes.

Methods

No methods defined with class "BaseLearner.CV.Batch.FitObj" in the signature.

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

See Also

`Regression.CV.Batch.Fit`

`BaseLearner.CV.FitObj-class`
Classes `"BaseLearner.CV.FitObj"` *and* `"Regression.CV.FitObj"`

Description

Classes for containing base learner CV training output.

Objects from the Class

`"BaseLearner.CV.FitObj"` is a virtual class: No objects may be created from it. `"Regression.CV.FitObj"` is the output of `Regression.CV.Fit` function.

Slots

`fitobj.list`**:** Object of class `"list"`, contains a list of objects of class `BaseLearner.Fit`, one per partition fold.

`partition`**:** Object of class `"OptionalInteger"`, representing how data must be split across folds during cross-validation. This is typically the output of `generate.partition` function.

`filemethod`**:** Object of class `"logical"`, determining whether to save individual estimation objects to file or not. In addition, `Regression.CV.FitObj` contains the following slot:

pred Object of class `"OptionalNumeric"`, containing the prediction from the CV fit object for training data. This slot is allowed to take on a "NULL" value to reduce excess memory use by large ensemble models.

Methods

No methods defined with class "BaseLearner.CV.FitObj" in the signature.

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

See Also

Regression.CV.Fit

BaseLearner.Fit-methods

Generic S4 Method for Fitting Base Learners

Description

Each base learner must provide its concrete implementation of this generic method.

Usage

```
BaseLearner.Fit(object, formula, data, tmpfile=NULL, print.level=1, ...)
```

Arguments

`object`	An object of class `BaseLearner.Config` (must be a concrete implementation such as `KNN.Regression.Config`).
`formula`	Formula object expressing response and covariates.
`data`	Data frame containing response and covariates.
`tmpfile`	Filepath to save the estimation object to. If `NULL`, estimation object will not be saved to a file.
`print.level`	Controlling verbosity level during fitting.
`...`	Arguments to be passed to/from other methods.

Methods

`signature(object = "GBM.Regression.Config")`

`signature(object = "KNN.Regression.Config")`

`signature(object = "NNET.Regression.Config")`

`signature(object = "RF.Regression.Config")`

`signature(object = "SVM.Regression.Config")`

`signature(object = "PENREG.Regression.Config")`

`signature(object = "BART.Regression.Config")`

BaseLearner.FitObj-class

Classes "BaseLearner.FitObj" *and* "Regression.FitObj"

Description

Base class templates for containing base learner training output.

Objects from the Class

"BaseLearner.FitObj" is a virtual class: No objects may be created from it. "Regression.FitObj" is a base class for objects representing trained models for individual base learners.

Slots

`config`**:** Object of class "BaseLearner.Config"; often one of the derived configuration classes belonging to a particular base learner. For Regression.FitObj, we have the following additional fields:

`est`**:** Object of class "RegressionEstObj", typically containing the low-level list coming out of the training algorithm. If filemethod=TRUE during the fit, this object will be of class "character", containing the filepath to where the estimation object is stored.

`pred`**:** Object of class "OptionalNumeric", fitted values of the model for the training data. It is allowed to be "NULL" in order to reduce memory footrpint during cross-validated ensemble methods.

Methods

No methods defined with these classes in their signature.

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

See Also

"KNN.Regression.FitObj", "RF.Regression.FitObj", "SVM.Regression.FitObj", "GBM.Regression.FitObj", "NNET.Regression.FitObj"

`Instance-class` *Classes* `"Instance"` *and* `"Instance.List"`

Description

A base learner `Instance` is a combination of a base learner configuration and data partition. Instances constitute the major input into the cross-validation-based functions such as `Regression.CV.Batch.Fit`. An `Instance.List` is a collection of instances, along with the underlying definition of data partitions referenced in the instance objects. The function `make.instances` is a convenient function for generating an instance list from all permutations of a given list of base learner configurations and data partitions.

Objects from the Class

Objects can be created by calls of the form `new("Instance", ...)`.

Slots

`Instance` has the following slots:

Object of class `"BaseLearner.Config"` ~~

`config`: **`partid`:** Object of class `"character"` ~~ `Instance.List` has the following slots:

`instances`: Object of class `"list"`, with each element being an object of class `Instance`.

`partitions`: Object of class `"matrix"`, defining data partitions referenced in each instance. This object is typically the output of `generate.partitions`.

Methods

No methods defined with class "Instance" in the signature.

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

See Also

`make.instances`, `generate.partitions`, `Regression.CV.Batch.Fit`

Examples

```
showClass("Instance")
```

make.configs *Helper Functions for Manipulating Base Learner Configurations*

Description

Helper Functions for Manipulating Base Learner Configurations

Usage

```
make.configs(baselearner=c("nnet","rf","svm","gbm","knn","penreg")
  , config.df, type = "regression")
make.configs.knn.regression(df=expand.grid(
  kernel=c("rectangular","epanechnikov","triweight","gaussian")
  , k=c(5,10,20,40)))
make.configs.gbm.regression(df=expand.grid(
  n.trees=c(1000,2000)
  , interaction.depth=c(3,4)
  , shrinkage=c(0.001,0.01,0.1,0.5)
  , bag.fraction=0.5))
make.configs.svm.regression(df=expand.grid(
  cost=c(0.1,0.5,1.0,5.0,10,50,75,100)
  , epsilon=c(0.1,0.25)
  , kernel="radial"))
make.configs.rf.regression(df=expand.grid(
  ntree=c(100,500)
  , mtry.mult=c(1,2)
  , nodesize=c(2,5,25,100)))
make.configs.nnet.regression(df=expand.grid(
  decay=c(1e-4,1e-2,1,100)
  , size=c(5,10,20,40)
  , maxit=2000))
make.configs.penreg.regression(df = expand.grid(
  alpha = 0.0
  , lambda = 10^(-8:+7)))
make.configs.bart.regression(df = rbind(cbind(expand.grid(
  num_trees = c(50, 100), k = c(2,3,4,5)), q = 0.9, nu = 3)
  , cbind(expand.grid(
  num_trees = c(50, 100), k = c(2,3,4,5)), q = 0.75, nu = 10)
  ))
make.instances(baselearner.configs, partitions)
extract.baselearner.name(config, type="regression")
```

Arguments

`baselearner`	Name of base learner algorithm. Currently, seven base learners are included: 1) Neural Network (`nnet` using package `nnet`), 2) Random Forest (`rf` using package `randomForest`), 3) Support Vector Machine (`svm` using package `e1071`), 4)

	Gradient Boosting Machine (`gbm` using package `gbm`), 5) K-Nearest-Neighbors (`knn` using package `kknn`), 6) Penalized Regression (`penreg` using package `glmnet`), and Bayesian Additive Regression Trees (`bart`) using package `bartMachine`.
`df,config.df`	Data frame, with columns named after tuning parameters belonging to the base learner, and each row indicating a tuning-parameter combination to include in the configuration list.
`type`	Type of base learner. Currently, only "regression" is supported.
`baselearner.configs`	Base learner configuration list to use in generating instances.
`partitions`	A matrix whose columns define data partitions, usually the output of generate.partitions.
`config`	Base learner configuration object.

Value

The `make.configs` family of functions return a list of objects of various base learner `config` classes, such as `KNN.Regression.Config`. Function `make.instances` returns an object of class `Instance.List`. Function `extract.baselearner.name` returns a character object representing the name of the base learner associated with the passed-in config object. For example, for a `KNN.Regression.Config` object, we get back "KNN". This utility function can be used in printing base learner names based on class of a config object.

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

`OptionalInteger-class` *Class* `"OptionalInteger"`

Description

Utility classes to allow for inclusion of "NULL" an a class instance, for memory efficiency. Each one of these is a class union between the underlying class ("integer", "character" and "numeric") and "NULL".

Objects from the Class

These classes are typically part of more complex classes representing outputs of ensemble fit functions.

Methods

No methods defined with class "OptionalInteger" in the signature.

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

See Also

`Regression.FitObj`, `BaseLearner.CV.FitObj`, `Regression.CV.FitObj`

`Regression.Batch.Fit` *Batch Training, Prediction and Diagnostics of Regression Base Learners*

Description

Batch Training, Prediction and Diagnostics of Regression Base Learners.

Usage

```
Regression.Batch.Fit(config.list, formula, data, ncores = 1
  , filemethod = FALSE, print.level = 1)
## S3 method for class 'Regression.Batch.FitObj'
predict(object, ..., ncores=1)
## S3 method for class 'Regression.Batch.FitObj'
plot(x, errfun=rmse.error, ...)
```

Arguments

`config.list`	List of configuration objects for batch of base learners to be trained.
`formula`	Formula objects expressing response and covariates.
`data`	Data frame containing response and covariates.
`ncores`	Number of cores to use during parallel training.
`filemethod`	Boolean indicator of whether to save estimation objects to disk or not.
`print.level`	Determining level of command-line output verbosity during training.
`object`	Object of class `Regression.Batch.FitObj` to make predictions for.
`...`	Arguments to be passed from/to other functions.
`x`	Object of class `Regression.Batch.FitObj` to plot.
`errfun`	Error function to use for calculating errors plotted.

Value

Function `Regression.Batch.Fit` returns an object of class `Regression.Batch.FitObj`. Function `predict.Regression.Batch.FitObj` returns a matrix of predictions, each column corresponding to one base learner in the trained batch. Function `plot.Regression.Batch.FitObj` creates a plot of base learner errors over the training set, grouped by type of base learner (all configurations within a given base learner using the same symbol).

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

See Also

Regression.Batch.FitObj

Examples

```
data(servo)
myformula <- class~motor+screw+pgain+vgain
myconfigs <- make.configs("knn")
perc.train <- 0.7
index.train <- sample(1:nrow(servo), size = round(perc.train*nrow(servo)))
data.train <- servo[index.train,]
data.predict <- servo[-index.train,]
ret <- Regression.Batch.Fit(myconfigs, myformula, data.train, ncores=2)
newpred <- predict(ret, data.predict)
```

Regression.CV.Batch.Fit

CV Batch Training and Diagnostics of Regression Base Learners

---

Description

CV Batch Training and Diagnostics of Regression Base Learners.

Usage

```
Regression.CV.Batch.Fit(instance.list, formula, data
  , ncores = 1, filemethod = FALSE, print.level = 1
  , preschedule = TRUE
  , schedule.method = c("random", "as.is", "task.length")
  , task.length)
## S3 method for class 'Regression.CV.Batch.FitObj'
predict(object, ..., ncores=1
  , preschedule = TRUE)
## S3 method for class 'Regression.CV.Batch.FitObj'
plot(x, errfun=rmse.error, ylim.adj = NULL, ...)
```

Arguments

`instance.list`	An object of class Instance.List, containing all combinations of base learner configurations and data partitions to perform CV batch training.
`formula`	Formula object expressing response variable and covariates.
`data`	Data frame expressing response variable and covariates.
`ncores`	Number of cores in parallel training.
`filemethod`	Boolean flag, indicating whether to save estimation objects to file or not.
`print.level`	Verbosity level.

preschedule	Boolean flag, indicating whether parallel jobs must be scheduled statically (TRUE) or dynamically (FALSE).
schedule.method	Method used for scheduling tasks across threads. In as.is, tasks are distributed in round-robin fashion. In random, tasks are randomly shuffled before round-robin distribution. In task.length, estimated task execution times are used to allocate them to threads to maximize load balance.
task.length	Estimation task execution times, to be used for loading balancing during parallel execution.
object	Output of Regression.CV.Batch.Fit, object of class Regression.CV.Batch.FitObj.
...	Arguments passed from/to other functions.
x	Object of class Regression.CV.Batch.FitObj, to creates a plot from.
errfun	Error function used in generating plot.
ylim.adj	Optional numeric argument to use for adjusting the range of y-axis.

Value

Function Regression.CV.Batch.Fit produces an object of class Regression.CV.Batch.FitObj. The predict method produces a matrix, whose columns each represent training-set predictions from one of the batch of base learners (in CV fashion).

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

See Also

Regression.CV.Batch.FitObj

Examples

```
data(servo)
myformula <- class~motor+screw+pgain+vgain

perc.train <- 0.7
index.train <- sample(1:nrow(servo)
  , size = round(perc.train*nrow(servo)))
data.train <- servo[index.train,]
data.predict <- servo[-index.train,]

parts <- generate.partitions(1, nrow(data.train))
myconfigs <- make.configs("knn"
  , config.df = expand.grid(kernel = "rectangular", k = c(5, 10)))
instances <- make.instances(myconfigs, parts)

ret <- Regression.CV.Batch.Fit(instances, myformula, data.train)
newpred <- predict(ret, data.predict)
```

Regression.CV.Fit *Cross-Validated Training and Prediction of Regression Base Learners*

Description

This function trains the base learner indicated in the configuration object in a cross-validation scheme using the `partition` argument. The cross-validated predictions are assembled and returned in the `pred` slot of the `Regression.CV.FitObj` object. Individual trained base learners are also assembled and returned in the return object, and used in the `predict` method.

Usage

```
Regression.CV.Fit(regression.config, formula, data
  , partition, tmpfiles = NULL, print.level = 1)
## S3 method for class 'Regression.CV.FitObj'
predict(object, newdata=NULL, ...)
```

Arguments

`regression.config`	An object of class `Regression.Config` (must be a concrete implementation of the base class, such as `KNN.Regression.Config`).
`formula`	Formula object expressing response and covariates.
`data`	Data frame containing response and covariates.
`partition`	Data partition, typically the output of `generate.partition` function.
`tmpfiles`	List of temporary files to save the `est` field of the output `Regression.FitObj`.
`print.level`	Integer setting verbosity level of command-line output during training.
`object`	An object of class `Regression.FitObj`.
`newdata`	Data frame containing new observations.
`...`	Arguments passed to/from other methods.

Value

Function `Regression.CV.Fit` returns an object of class `Regression.CV.FitObj`. Function `predict.Regression.CV.FitO`
returns a numeric vector of length `nrow(newdata)`.

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

See Also

`Regression.CV.FitObj`

Examples

```
data(servo)
myformula <- class~motor+screw+pgain+vgain
myconfig <- make.configs("knn", config.df=data.frame(kernel="rectangular", k=10))
perc.train <- 0.7
index.train <- sample(1:nrow(servo), size = round(perc.train*nrow(servo)))
data.train <- servo[index.train,]
data.predict <- servo[-index.train,]
mypartition <- generate.partition(nrow(data.train),nfold=3)
ret <- Regression.CV.Fit(myconfig[[1]], myformula, data.train, mypartition)
newpred <- predict(ret, data.predict)
```

`Regression.Integrator.Config-class` *Classes* `"Regression.Integrator.Config"`, `"Regression.Select.Config"`, `"Regression.Integrator.FitObj"`, `"Regression.Select.FitObj"`

Description

Virtual base classes to contain configuration and fit objects for integrator operations.

Objects from the Class

All virtual classes; therefore, no objects may be created from them.

Slots

For config classes:

`errfun`: Object of class `"function"` ~~ For FitObj classes:

`config`: Object of class `"Regression.Integrator.Config"` or `"Regression.Select.Config"` for the Integrator and Select classes.

`est`: Object of class `ANY`, containing estimation objects for concrete extensions of the virtual classes.

`pred`: Object of class `"numeric"`, containing the prediction of integrator operations.

Methods

No methods defined with class "Regression.Integrator.Config" in the signature.

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

See Also

`Regression.Integrator.Fit`, `Regression.Select.Fit`

Regression.Integrator.Fit-methods
Generic Integrator Methods in Package **EnsembleBase**

Description

Generic methods that can be extended and used in constructing integrator algorithms by other packages.

Usage

```
Regression.Integrator.Fit(object, X, y, print.level=1)
Regression.Select.Fit(object, X, y, print.level=1)
```

Arguments

`object`	An object typically containing all configuration parameters of a particular integrator algorithm or operations.
`X`	Matrix of covariates.
`y`	Vector of response variables.
`print.level`	Verbosity level.

RegressionEstObj-class
Class `"RegressionEstObj"`

Description

Union of (converted) S3 classes for individual base learners as defined in their corresponding packages. The special class "character" has been added to allow for returning filepaths when saving estimation objects to disk.

Objects from the Class

Objects from this class are typically returned as part of `FitObj` family of classes.

Methods

No methods defined with class "RegressionEstObj" in the signature.

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

See Also

`BaseLearner.FitObj`

`RegressionSelectPred-class` *Class* `"RegressionSelectPred"`

Description

Union of classes `"NULL"`, `"numeric"` and `"matrix"` to hold prediction output of `Select` operations based on generic function `Regression.Select.Fit`. Class `NULL` is included to allow methods to save memory by not returning the prediction, espeically when a highe-level wrapper takes responsibility for holding a global copy of all prediction results. The "numeric" and "matrix" classes allow for a single predictor or multiple predictors to be produced by a `Select` operation.

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

No methods defined with class "RegressionSelectPred" in the signature.

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

See Also

`Regression.Select.Fit`

`servo` *Servo Data Set*

Description

A small regression data set taken from UCI Machine Learning Repository. Response variable is "class".

Usage

```
data("servo")
```

Format

The format is: chr "servo"

Source

Bache, K. & Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Examples

```
data(servo)
lm(class~motor+screw+pgain+vgain, servo)
```

`Utility Functions` *Utility Functions in EnsembleBase Package*

Description

Collection of utility functions for generating random partitions in datasets (for cross-validated operations), extracting regression response variable from dataset, loading an object from memory and assigning it to an arbitrary symbol, and error definitions.

Usage

```
generate.partition(ntot, nfold = 5)
generate.partitions(npart=1, ntot, nfold=5, ids=1:npart)
regression.extract.response(formula, data)
load.object(file)
rmse.error(a,b)
```

Arguments

`ntot`	Total number of observations in the data set to be partitioned.
`nfold`	Number of folds in the data partition.
`npart`	Number of random partitions to generate.
`ids`	Column names for the resulting partition matrix, used as partition ID.
`formula`	Formula object to use for extracting response variable from data set.
`data`	Data frame containing response variable as defined in `formula`.
`file`	Filepath from which to read an `R` object into memory (saved using the `save` function).
`a,b`	Vectors of equal length, used to calculate their RMSE distance.

Value

Function `generate.partition` returns an integer vector of length `ntot`, with entries - nearly - equally split in the range `1:nfold`. Function `generate.partitions` returns a matrix of size `ntot` x `npart`, with each column being a `partition` alike to the output of `generate.partition`. The columns are named `ids`. Function `regression.extract.response` returns a vector of length `nrow(data)`, containing the numeric response variable for regression problems. Function `load.object` returns the saved object, but only works if only a single `R` object was saved to the file. Function `rmse.error` returns a single numeric value representing root-mean-squared-error distance between vectors `a` and `b`.

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

`validate-methods` ~~ *Methods for Function* `validate` *in Package* **EnsembleBase** ~~

Description

~~ Methods for function `validate` in package **EnsembleBase** ~~

Methods:

`signature(object = "Regression.Batch.FitObj")`

`signature(object = "Regression.CV.Batch.FitObj")`

Index