Package ‘caretEnsemble’
August 17, 2024

Type Package
Title Ensembles of Caret Models
Version 4.0.0
Date 2024-08-17
URL http://zachmayer.github.io/caretEnsemble/,
https://github.com/zachmayer/caretEnsemble
BugReports https://github.com/zachmayer/caretEnsemble/issues

Description Functions for creating ensembles of caret models: caretList()
and caretStack(). caretList() is a convenience function for fitting multiple
caret::train() models to the same dataset. caretStack() will make linear or
non-linear combinations of these models, using a caret::train() model as a
meta-model.

Depends R (>= 4.1.0)
Suggests MASS, caTools, covr, devtools, earth, gbm, glmnet, klaR,
knitr, lintr, mgcv, mlbench, nnet, pkgdown, randomForest,
rmarkdown, rhub, rpart, spelling, testthat, usethis
Imports caret, data.table, ggplot2, lattice, methods, patchwork,
pbapply, rlang
License MIT + file LICENSE
VignetteBuilder knitr
RoxygenNote 7.3.2
LazyData true
Language en-US
Encoding UTF-8

NeedsCompilation no

Author Zachary A. Deane-Mayer [aut, cre, cph],
Jared E. Knowles [ctb],
Antón López [ctb]

Maintainer Zachary A. Deane-Mayer <zach.mayer@gmail.com>
Repository CRAN
Date/Publication 2024-08-17 17:40:02 UTC
as.caretList  

Convert object to caretList object

Description

Converts object into a caretList

Usage

as.caretList(object)

Arguments

object R Object

Value

a caretList object

as.caretList.default  

Convert object to caretList object - For Future Use

Description

Converts object into a caretList - For Future Use

Usage

## Default S3 method:
as.caretList(object)

Arguments

object R object

Value

NA
as.caretList.list  

Convert list to caretList

Description

Converts list to caretList

Usage

## S3 method for class 'list'
as.caretList(object)

Arguments

object  

list of caret models

Value

a caretList object

autoplot.caretStack  

Convenience function for more in-depth diagnostic plots of caretStack objects

Description

This function provides a more robust series of diagnostic plots for a caretEnsemble object.

Usage

## S3 method for class 'caretStack'
autoplot(object, training_data = NULL, xvars = NULL, show_class_id = 2L, ...)

Arguments

object  
a caretStack object

training_data  
The data used to train the ensemble. Required if xvars is not NULL. Must be in the same row order as when the models were trained.

xvars  
a vector of the names of x variables to plot against residuals

show_class_id  
For classification only: which class level to show on the plot

...  
ignored
Value

A grid of diagnostic plots. Top left is the range of the performance metric across each component model along with its standard deviation. Top right is the residuals from the ensembled model plotted against fitted values. Middle left is a bar graph of the weights of the component models. Middle right is the disagreement in the residuals of the component models (unweighted) across the fitted values. Bottom left and bottom right are the plots of the residuals against two random or user specified variables. Note that the ensemble must have been trained with savePredictions = "final", which is required to get residuals from the stack for the plot.

Examples

```r
set.seed(42)
data(models.reg)
ens <- caretStack(models.reg)
autoplot(ens)
```

---

c.caretList

### S3 definition for concatenating caretList

**Description**

take N objects of class caretList and concatenate them into a larger object of class caretList for future ensembling

**Usage**

```r
## S3 method for class 'caretList'
c(...)
```

**Arguments**

... the objects of class caretList or train to bind into a caretList

**Value**

a caretList object

**Examples**

```r
data(iris)
model_list1 <- caretList(Sepal.Width ~ .,
data = iris,
tuneList = list(
  lm = caretModelSpec(method = "lm")
)
)

model_list2 <- caretList(Sepal.Width ~ .,
```
c.train

data = iris, tuneLength = 1L,
tuneList = list(
    rf = caretModelSpec(method = "rf")
)
}

bigList <- c(model_list1, model_list2)

c.train

S3 definition for concatenating train objects

Description

take N objects of class train and concatenate into an object of class caretList for future ensembling

Usage

## S3 method for class 'train'
c(...)

Arguments

... the objects of class train to bind into a caretList

Value

a caretList object

Examples

data(iris)
model_lm <- caret::train(Sepal.Length ~ .,
    data = iris,
    method = "lm"
)

model_rf <- caret::train(Sepal.Length ~ .,
    data = iris,
    method = "rf",
    tuneLength = 1L
)

model_list <- c(model_lm, model_rf)
caretEnsemble

**caretEnsemble**

Combine several predictive models via weights

**Description**

Find a greedy, positive only linear combination of several `train` objects

Functions for creating ensembles of caret models: `caretList` and `caretStack`

**Usage**

```r
caretEnsemble(all.models, excluded_class_id = 0L, tuneLength = 1L, ...)
```

**Arguments**

- `all.models`: an object of class `caretList`
- `excluded_class_id`: The integer level to exclude from binary classification or multiclass problems. By default no classes are excluded, as the greedy optimizer requires all classes because it cannot use negative coefficients.
- `tuneLength`: The size of the grid to search for tuning the model. Defaults to 1, as the only parameter to optimize is the number of iterations, and the default of 100 works well.
- `...`: additional arguments to pass `caret::train`

**Details**

greedyMSE works well when you want an ensemble that will never be worse than any single model in the dataset. In the worst case scenario, it will select the single best model, if none of them can be ensembled to improve the overall score. It will also never assign any model a negative coefficient, which can help avoid unintuitive cases at prediction time (e.g. if the correlations between predictors breaks down on new data, negative coefficients can lead to bad results).

**Value**

- a `caretEnsemble` object

**Note**

Every model in the "library" must be a separate `train` object. For example, if you wish to combine a random forests with several different values of mtry, you must build a model for each value of mtry. If you use several values of mtry in one `train` model, (e.g. `tuneGrid = expand.grid(.mtry=2:5)`), caret will select the best value of mtry before we get a chance to include it in the ensemble. By default, RMSE is used to ensemble regression models, and AUC is used to ensemble Classification models. This function does not currently support multi-class problems.
Author(s)

Maintainer: Zachary A. Deane-Mayer <zach.mayer@gmail.com> [copyright holder]

Other contributors:

• Jared E. Knowles <jknowles@gmail.com> [contributor]
• Antón López <anton.gomez.lopez@rai.usc.es> [contributor]

See Also

Useful links:

• http://zachmayer.github.io/caretEnsemble/
• https://github.com/zachmayer/caretEnsemble
• Report bugs at https://github.com/zachmayer/caretEnsemble/issues

Examples

set.seed(42)
models <- caretList(iris[1:50, 1:2], iris[1:50, 3], methodList = c("rpart", "rf"))
ens <- caretEnsemble(models)
summary(ens)

caretList

Create a list of several train models from the caret package

Description

Build a list of train objects suitable for ensembling using the caretStack function.

Usage

caretList(
  ..., 
  trControl = NULL, 
  methodList = NULL, 
  tuneList = NULL, 
  metric = NULL, 
  continue_on_fail = FALSE, 
  trim = TRUE
)

Arguments

... arguments to pass to \code{train}. Don’t use the formula interface, its slower and buggier compared to the X, y interface. Use a \code{data.table} for X. Particularly if you have a large dataset and/or many models, using a data.table will avoid unnecessary copies of your data and can save a lot of time and RAM. These arguments will determine which train method gets dispatched.

\code{trControl} a \code{trainControl} object. If NULL, will use defaultControl.

\code{methodList} optional, a character vector of caret models to ensemble. One of methodList or tuneList must be specified.

\code{tuneList} optional, a NAMED list of caretModelSpec objects. This much more flexible than methodList and allows the specification of model-specific parameters (e.g. passing trace=FALSE to \code{nnet})

\code{metric} a string, the metric to optimize for. If NULL, we will choose a good one.

\code{continue_on_fail} logical, should a valid caretList be returned that excludes models that fail, default is FALSE

\code{trim} logical should the train models be trimmed to save memory and speed up stacking

Value

A list of \code{train} objects. If the model fails to build, it is dropped from the list.

Examples

caretList(
  Sepal.Length ~ Sepal.Width,
  head(iris, 50),
  methodList = c("glm", "lm"),
  tuneList = list(
    nnet = caretModelSpec(method = "nnet", trace = FALSE, tuneLength = 1)
  )
)

caretModelSpec \hspace{1em} \textit{Generate a specification for fitting a caret model}

Description

A caret model specification consists of 2 parts: a model (as a string) and the arguments to the train call for fitting that model

Usage

caretModelSpec(method = "rf", ...)

Arguments
method
... other arguments that will eventually be passed to caret::train

Value
a list of lists

Examples
caretModelSpec("rf", tuneLength = 5, preProcess = "ica")

caretStack
Combine several predictive models via stacking

Description
Stack several train models using a train model.

Usage
caretStack(
  all.models,
  new_X = NULL,
  new_y = NULL,
  metric = NULL,
  trControl = NULL,
  excluded_class_id = 1L,
  ...
)

Arguments
all.models a caretList, or an object coercible to a caretList (such as a list of train objects)
new_X Data to predict on for the caretList, prior to training the stack (for transfer learning). If NULL, the stacked predictions will be extracted from the caretList models.
new_y The outcome variable to predict on for the caretList, prior to training the stack (for transfer learning). If NULL, will use the observed levels from the first model in the caret stack. If 0, will include all levels.
metric the metric to use for grid search on the stacking model.
trControl a trainControl object to use for training the ensemble model. If NULL, will use defaultControl.
excluded_class_id The integer level to exclude from binary classification or multiclass problems.
... additional arguments to pass to the stacking model
Details
Uses either transfer learning or stacking to stack models. Assumes that all models were trained on the same number of rows of data, with the same target values. The features, cross-validation strategies, and model types (class vs reg) may vary however. If your stack of models were trained with different number of rows, please provide new_X and new_y so the models can predict on a common set of data for stacking.
If your models were trained on different columns, you should use stacking.
If you have both differing rows and columns in your model set, you are out of luck. You need at least a common set of rows during training (for stacking) or a common set of columns at inference time for transfer learning.

Value
S3 caretStack object

References

Examples
models <- caretList(
  x = iris[1:50, 1:2],
  y = iris[1:50, 3],
  methodList = c("rpart", "glm")
)
caretStack(models, method = "glm")

defaultControl

Construct a default train control for use with caretList

Description
Unlike caret::trainControl, this function defaults to 5 fold CV. CV is good for stacking, as every observation is in the test set exactly once. We use 5 instead of 10 to save compute time, as caretList is for fitting many models. We also construct explicit fold indexes and return the stacked predictions, which are needed for stacking. For classification models we return class probabilities.

Usage
defaultControl(
  target,
  method = "cv",
  number = 5L,
  savePredictions = "final",
)
index = caret::createFolds(target, k = number, list = TRUE, returnTrain = TRUE),
is_class = is.factor(target) || is.character(target),
is_binary = length(unique(target)) == 2L,
...)

Arguments

target the target variable.
method the method to use for trainControl.
number the number of folds to use.
savePredictions the type of predictions to save.
index the fold indexes to use.
is_class logical, is this a classification or regression problem.
is_binary logical, is this binary classification.
... other arguments to pass to trainControl

defaultMetric

Construct a default metric

Description

Caret defaults to RMSE for classification and RMSE for regression. For classification, I would rather use ROC.

Usage

defaultMetric(is_class, is_binary)

Arguments

is_class logical, is this a classification or regression problem.
is_binary logical, is this binary classification.


### dotplot.caretStack

**Comparison dotplot for a caretStack object**

**Description**

This is a function to make a dotplot from a caretStack. It uses dotplot from the caret package on all the models in the ensemble, excluding the final ensemble model. At the moment, this function only works if the ensembling model has the same number of resamples as the component models.

**Usage**

```r
## S3 method for class 'caretStack'
dotplot(x, ...)
```

**Arguments**

- `x` An object of class caretStack
- `...` passed to dotplot

**Examples**

```r
set.seed(42)
models <- caretList(
  x = iris[1:100, 1:2],
  y = iris[1:100, 3],
  methodList = c("rpart", "glm")
)
meta_model <- caretStack(models, method = "lm")
lattice::dotplot(meta_model)
```

---

### extractMetric

**Generic function to extract accuracy metrics from various model objects**

**Description**

A generic function to extract cross-validated accuracy metrics from model objects.

**Usage**

```r
extractMetric(x, ...)
```

**Arguments**

- `x` An object from which to extract metrics. The specific method will be dispatched based on the class of `x`.
- `...` Additional arguments passed to the specific methods.
Value
A data.table

See Also
extractMetric.train, extractMetric.caretList, extractMetric.caretStack

Description
Extract the cross-validated accuracy metrics from each model in a caretList.

Usage
## S3 method for class 'caretList'
extractMetric(x, ...)

Arguments
x a caretList object
... passed to extractMetric.train

Value
A data.table with metrics from each model.

Description
Extract the cross-validated accuracy metrics from the ensemble model and individual models in a caretStack.

Usage
## S3 method for class 'caretStack'
extractMetric(x, ...)

Value
A data.table with metrics from each model.
extractMetric.train

Arguments

x  a caretStack object
    ... passed to extractMetric.train and extractMetric.caretList

Value

A data.table with metrics from the ensemble model and individual models.

greedyMSE

Description

Greedy optimization for MSE

Usage

greedyMSE(X, Y, max_iter = 100L)
Arguments

X  A numeric matrix of features.
Y  A numeric matrix of target values.
max_iter  An integer scalar of the maximum number of iterations.

Value

A list with components:

model_weights  A numeric matrix of model_weights.
RMSE  A numeric scalar of the root mean squared error.
max_iter  An integer scalar of the maximum number of iterations.

greedyMSE_caret  caret interface for greedyMSE

description

caret interface for greedyMSE. greedyMSE works well when you want an ensemble that will never be worse than any single predictor in the dataset. It does not use an intercept and it does not allow for negative coefficients. This makes it highly constrained and in general does not work well on standard classification and regression problems. However, it does work well in the case of: * The predictors are highly correlated with each other * The predictors are highly correlated with the model * You expect or want positive only coefficients In the worse case, this method will select one input and use that, but in many other cases it will return a positive, weighted average of the inputs. Since it never uses negative weights, you never get into a scenario where one model is weighted negative and on new data you get were predictions because a correlation changed. Since this model will always be a positive weighted average of the inputs, it will rarely do worse than the individual models on new data.

Usage

greedyMSE_caret()
permutationImportance  Permutation Importance

Description
Permute each variable in a dataset and use the change in predictions to calculate the importance of each variable. Based on the scikit learn implementation of permutation importance: https://scikit-learn.org/stable/modules/permutation_importance.html. However, we don’t compare to the target by a metric. We JUST look at the change in the model’s predictions, as measured by MAE. (for classification, this is like using a Brier score). We shuffle each variable and recompute the predictions before and after the shuffle. The difference in MAE is the importance of that variable. We normalize by computing the MAE of the shuffled original predictions as an upper bound on the MAE and divide by this value. So a variable that, when shuffled, caused predictions as bad as shuffling the output predictions, we know that variable is 100. Similarly, as with regular permutation importance, a variable that, when shuffled, gives the same MAE as the original model has an importance of 0.

This method cannot yield negative importances. It is merely a measure of how much the models uses the variable, and does not tell you which variables help or hurt generalization. Use the model’s cross-validated metrics to assess generalization.

Usage
permutationImportance(model, newdata, normalize = TRUE)

Arguments
model  A train object from the caret package.
newdata  A data.frame of new data to use to compute importances. Can be the training data.
normalize  A logical indicating whether to normalize the importances to sum to one.

Value
A named numeric vector of variable importances.

plot.caretList  Plot a caretList object

Description
This function plots the performance of each model in a caretList object.

Usage
## S3 method for class 'caretList'
plot(x, metric = NULL, ...)

plot.caretList  Plot a caretList object

Description
This function plots the performance of each model in a caretList object.

Usage
## S3 method for class 'caretList'
plot(x, metric = NULL, ...)

Arguments

- **x**: a caretList object
- **metric**: which metric to plot
- **...**: ignored

Value

A ggplot2 object

---

**plot.caretStack**

*Plot a caretStack object*

**Description**

This function plots the performance of each model in a caretList object.

**Usage**

```r
## S3 method for class 'caretStack'
plot(x, metric = NULL, ...)
```

**Arguments**

- **x**: a caretStack object
- **metric**: which metric to plot. If NULL, will use the default metric used to train the model.
- **...**: ignored

**Value**

A ggplot2 object

---

**predict.caretList**

*Create a matrix of predictions for each of the models in a caretList*

**Description**

Make a matrix of predictions from a list of caret models

**Usage**

```r
## S3 method for class 'caretList'
predict(object, newdata = NULL, verbose = FALSE, excluded_class_id = 1L, ...)
```
predict.caretStack

Arguments

object: an object of class caretList
newdata: New data for predictions. It can be NULL, but this is ill-advised.
verbose: Logical. If FALSE no progress bar is printed if TRUE a progress bar is shown. Default FALSE.
excluded_class_id: Integer. The class id to drop when predicting for multiclass
...

Other arguments to pass to predict.train

Description

Make predictions from a caretStack. This function passes the data to each function in turn to make a matrix of predictions, and then multiplies that matrix by the vector of weights to get a single, combined vector of predictions.

Usage

## S3 method for class 'caretStack'
predict(
  object,
  newdata = NULL,
  se = FALSE,
  level = 0.95,
  excluded_class_id = 0L,
  return_class_only = FALSE,
  verbose = FALSE,
  ...
)

Arguments

object: a caretStack to make predictions from.
newdata: a new dataframe to make predictions on
se: logical, should prediction errors be produced? Default is false.
level: tolerance/confidence level should be returned
excluded_class_id: Which class to exclude from predictions. Note that if the caretStack was trained with an excluded_class_id, that class is ALWAYS excluded from the predictions from the caretList of input models. excluded_class_id for predict.caretStack is for the final ensemble model. So different classes could be excluded from the caretList models and the final ensemble model.
return_class_only

A logical indicating whether to return only the class predictions as a factor. If TRUE, the return will be a factor rather than a data.table. This is a convenience function, and should not be widely used. For example if you have a downstream process that consumes the output of the model, you should have that process consume probabilities for each class. This will make it easier to change prediction probability thresholds if needed in the future.

verbose

A logical indicating whether to print progress

... 

Arguments to pass to predict.train for the ensemble model. Do not specify type here. For classification, type will always be prob, and for regression, type will always be raw.

Details

Prediction weights are defined as variable importance in the stacked caret model. This is not available for all cases such as where the library model predictions are transformed before being passed to the stacking model.

Value

A data.table of predictions

Examples

```r
models <- caretList(
  x = iris[1:100, 1:2],
  y = iris[1:100, 3],
  methodList = c("rpart", "glm")
)
meta_model <- caretStack(models, method = "lm")
RMSE(predict(meta_model, iris[101:150, 1:2]), iris[101:150, 3])
```

Description

Predict method for greedyMSE objects.

Usage

```r
## S3 method for class 'greedyMSE'
predict(object, newdata, return_labels = FALSE, ...)
```
Arguments

object    A greedyMSE object.
newdata   A numeric matrix of new data.
return_labels  A logical scalar of whether to return labels.
...        Additional arguments. Ignored.

Value

A numeric matrix of predictions.

Description

This is a function to print a caretStack.

Usage

## S3 method for class 'caretStack'
print(x, ...)

Arguments

x          An object of class caretStack
...

Examples

models <- caretList(
  x = iris[1:100, 1:2],
  y = iris[1:100, 3],
  methodList = c("rpart", "glm")
)
meta_model <- caretStack(models, method = "lm")
print(meta_model)
print.greedyMSE  
*Print method for greedyMSE*

**Description**

Print method for greedyMSE objects.

**Usage**

```
## S3 method for class 'greedyMSE'
print(x, ...)
```

**Arguments**

- `x` A greedyMSE object.
- `...` Additional arguments. Ignored.

print.summary.caretList

*Print a summary.caretList object*

**Description**

This is a function to print a summary.caretList.

**Usage**

```
## S3 method for class 'summary.caretList'
print(x, ...)
```

**Arguments**

- `x` An object of class summary.caretList
- `...` ignored
print.summary.caretStack

Print a summary.caretStack object

**Description**

This is a function to print a summary.caretStack.

**Usage**

```r
## S3 method for class 'summary.caretStack'
print(x, ...)
```

**Arguments**

- `x` An object of class summary.caretStack
- `...` ignored

---

summary.caretList

Summarize a caretList

**Description**

This function summarizes the performance of each model in a caretList object.

**Usage**

```r
## S3 method for class 'caretList'
summary(object, metric = NULL, ...)
```

**Arguments**

- `object` a caretList object
- `metric` The metric to show. If NULL will use the metric used to train each model
- `...` passed to extractMetric

**Value**

A data.table with metrics from each model.
summary.caretStack  Summarize a caretStack object

Description

This is a function to summarize a caretStack.

Usage

```r
## S3 method for class 'caretStack'
summary(object, ...)
```

Arguments

- `object`: An object of class caretStack
- `...`: ignored

Examples

```r
models <- caretList(
  x = iris[1:100, 1:2],
  y = iris[1:100, 3],
  methodList = c("rpart", "glm")
)
meta_model <- caretStack(models, method = "lm")
summary(meta_model)
```

tuneCheck  Check that the tuning parameters list supplied by the user is valid

Description

This function makes sure the tuning parameters passed by the user are valid and have the proper naming, etc.

Usage

```r
tuneCheck(x)
```

Arguments

- `x`: a list of user-supplied tuning parameters and methods
### varImp.caretStack

**Variable importance for caretStack**

**Description**

This is a function to extract variable importance from a caretStack.

**Usage**

```r
## S3 method for class 'caretStack'
varImp(object, newdata = NULL, normalize = TRUE, ...)
```

**Arguments**

- `object`: An object of class caretStack
- `newdata`: the data to use for computing importance. If NULL, will use the stacked predictions from the models.
- `normalize`: a logical indicating whether to normalize the importances to sum to one.
- `...`: passed to predict.caretList

### varImp.greedyMSE

**variable importance for a greedyMSE model**

**Description**

Variable importance for a greedyMSE model.

**Usage**

```r
## S3 method for class 'greedyMSE'
varImp(object, ...)
```

**Arguments**

- `object`: A greedyMSE object.
- `...`: Additional arguments. Ignored.
**wtd.sd**  
*Calculate a weighted standard deviation*

**Description**

Used to weight deviations among ensembled model predictions

**Usage**

```r
wtd.sd(x, w, na.rm = FALSE)
```

**Arguments**

- `x`  
a numeric vector
- `w`  
a vector of weights equal to length of x
- `na.rm`  
a logical indicating how to handle missing values, default = TRUE

---

**index**  
*Index a caretList*

**Description**

Index a caret list to extract caret models into a new caretList object

**Usage**

```r
## S3 method for class 'caretList'
object[index]
```

**Arguments**

- `object`  
an object of class caretList
- `index`  
selected index
Index

[.caretList, 26

as.caretList, 3
as.caretList.default, 3
as.caretList.list, 4
autoplot.caretStack, 4

c.caretList, 5
c.train, 6
caretEnsemble, 7, 7
caretEnsemble-package (caretEnsemble), 7
caretList, 3–6, 8, 14
caretModelSpec, 9
caretStack, 8, 10, 14, 19

data.table, 9, 14
defaultControl, 11
defaultMetric, 12
dotplot.caretStack, 13

eXtractMetric, 13
eXtractMetric.caretList, 14, 14
eXtractMetric.caretStack, 14, 14
eXtractMetric.train, 14, 15

greedyMSE, 15
greedyMSE_caret, 16

permutationImportance, 17
plot.caretList, 17
plot.caretStack, 18
predict.caretList, 18
predict.caretStack, 19
predict.greedyMSE, 20
predict.train, 19, 20
print.caretStack, 21
print.greedyMSE, 22
print.summary.caretList, 22
print.summary.caretStack, 23

summary.caretList, 23

summary.caretStack, 24

train, 7, 9, 10, 15
trainControl, 9, 12
tuneCheck, 24

varImp.caretStack, 25
varImp.greedyMSE, 25

wtd.sd, 26

27