Package 'caretEnsemble'

December 12, 2019

Encoding UTF-8

Type Package

Title Ensembles of Caret Models

Version 2.0.1

Date 2019-12-11

URL 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, and caretEnsemble() will make a robust linear combination of models using a GLM.

Depends R (>= 3.2.0)

Suggests caTools, testthat, lintr, randomForest, glmnet, rpart, kernlab, nnet, e1071, ipred, pROC, knitr, mlbench, MASS, gbm, klaR, rmarkdown

Imports methods, pbapply, ggplot2, digest, plyr, lattice, gridExtra, data.table, caret

License MIT + file LICENSE

VignetteBuilder knitr

RoxygenNote 6.1.1

NeedsCompilation no

Author Zachary A. Deane-Mayer [aut, cre], Jared E. Knowles [aut]

Maintainer Zachary A. Deane-Mayer <zach.mayer@gmail.com>

Repository CRAN

Date/Publication 2019-12-12 22:10:09 UTC

R topics documented:

`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 *Convenience function for more in-depth diagnostic plots of caretEnsemble objects*

Description

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

Usage

```
autoplot(object, which = c(1:6), mfrow = c(3, 2), xvars = NULL, ...)
```

Arguments

object	a caretEnsemble object
which	an integer index for which of the plots to print. DOES NOTHING.
mfrow	an integer vector of length 2 specifying the number of rows and columns for plots
xvars	a vector of the names of x variables to plot against residuals
...	additional arguments to pass to autoplot

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.

Examples

```
## Not run:
set.seed(42)
models <- caretList(
 iris[1:50,1:2],
 iris[1:50,3],
 trControl=trainControl(method="cv"),
 methodList=c("glm", "rpart"))
ens <- caretEnsemble(models)
autoplot(ens)

## End(Not run)
```

---

bestPreds *Extract the best predictions from a train object*

Description

Extract predictions for the best tune from a model

Usage

```
bestPreds(x)
```

Arguments

x	a train object

---

c.caretList *S3 definition for concatenating caretList*

Description

take N objects of class caretList and concatenat them into a larger object of class caretList for future Ensamble'ing

Usage

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

Arguments

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

Value

a caretList object

Examples

```
## Not run:
 model_list1 <- caretList(Class ~ .,
  data=Sonar, trControl = ctrl1,
  tuneList = list(
                glm=caretModelSpec(method='glm', family='binomial'),
                rpart=caretModelSpec(method='rpart')
                ),
   metric='ROC')

model_list2 <- caretList(Class ~ .,
                         data=Sonar,
                         trControl = ctrl1,
                         tuneList = list(
                           glm=caretModelSpec(method='rpart'),
                           rpart=caretModelSpec(method='rf')
                         ),
                         metric='ROC')

 bigList <- c(model_list1, model_list2)

## End(Not run)
```

---

c.train *S3 definition for concatenating train objects*

Description

take N objects of class train and concatenat into an object of class caretList for future Ensamble'ing

Usage

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

Arguments

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

Value

a caretList object

Examples

```
## Not run:
rpartTrain <- train(Class ~ .,
                    data=Sonar,
                    trControl = ctrl1,
                    method='rpart')

rfTrain <- train(Class ~ .,
                 data=Sonar,
                 trControl = ctrl1,
                 method='rf')

 bigList <- c(model_list1, model_list2)

## End(Not run)
```

---

caretEnsemble *caretEnsemble: Make ensembles of caret models.*

Description

Functions for creating ensembles of caret models: caretList and caretStack

Find a good linear combination of several classification or regression models, using linear regression.

Usage

```
caretEnsemble(all.models, ...)
```

Arguments

all.models	an object of class caretList
...	additional arguments to pass to the optimization function

Details

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

Value

a `caretEnsemble` object

Note

Currently when missing values are present in the training data, weights are calculated using only observations which are complete across all models in the library.The optimizer ignores missing values and calculates the weights with the observations and predictions available for each model separately. If each of the models has a different pattern of missingness in the predictors, then the resulting ensemble weights may be biased and the function issues a message.

Examples

```
## Not run:
set.seed(42)
models <- caretList(iris[1:50,1:2], iris[1:50,3], methodList=c("glm", "lm"))
ens <- caretEnsemble(models)
summary(ens)

## End(Not run)
```

---

caretList *Create a list of several train models from the caret package Build a list of train objects suitable for ensembling using the `caretEnsemble` function.*

Description

Create a list of several train models from the caret package Build a list of train objects suitable for ensembling using the `caretEnsemble` function.

Usage

```
caretList(..., trControl = NULL, methodList = NULL, tuneList = NULL,
  continue_on_fail = FALSE)
```

Arguments

...	arguments to pass to train. These arguments will determine which train method gets dispatched.
trControl	a trainControl object. We are going to intercept this object check that it has the "index" slot defined, and define the indexes if they are not.
methodList	optional, a character vector of caret models to ensemble. One of methodList or tuneList must be specified.
tuneList	optional, a NAMED list of caretModelSpec objects. This much more flexible than methodList and allows the specificaiton of model-specific parameters (e.g. passing trace=FALSE to nnet)
continue_on_fail,	logical, should a valid caretList be returned that excludes models that fail, default is FALSE

Value

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

Examples

```
## Not run:
myControl <- trainControl(method="cv", number=5)
caretList(
  Sepal.Length ~ Sepal.Width,
  head(iris, 50),
  methodList=c("glm", "lm"),
  trControl=myControl
  )
caretList(
  Sepal.Length ~ Sepal.Width,
  head(iris, 50), methodList=c("lm"),
  tuneList=list(
    nnet=caretModelSpec(method="nnet", trace=FALSE, tuneLength=1)
 ),
  trControl=myControl
  )

## End(Not run)
```

---

caretModelSpec *Generate a specification for fitting a caret model*

Description

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

Usage

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

Arguments

method	the modeling method to pass to caret::train
...	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

Find a good linear combination of several classification or regression models, using either linear regression, elastic net regression, or greedy optimization.

Usage

```
caretStack(all.models, ...)
```

Arguments

all.models	a list of caret models to ensemble.
...	additional arguments to pass to the optimization function

Details

Check the models, and make a matrix of obs and preds

Value

S3 caretStack object

References

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.2859&rep=rep1&type=pdf

Examples

```
## Not run:
library("rpart")
models <- caretList(
  x=iris[1:50,1:2],
  y=iris[1:50,3],
  trControl=trainControl(method="cv"),
  methodList=c("rpart", "glm")
)
caretStack(models, method="glm")

## End(Not run)
```

`check_bestpreds_indexes` *Check row indexes*

Description

Check that the row indexes from a caretList are valid

Usage

```
check_bestpreds_indexes(modelLibrary)
```

Arguments

`modelLibrary`	a list of predictins from caret models

`check_bestpreds_obs` *Check observeds*

Description

Check that a list of observed values from a caretList are valid

Usage

```
check_bestpreds_obs(modelLibrary)
```

Arguments

`modelLibrary`	a list of predictins from caret models

check_bestpreds_preds *Check predictions*

Description

Check that a list of predictions from a caretList are valid

Usage

```
check_bestpreds_preds(modelLibrary)
```

Arguments

modelLibrary	a list of predictins from caret models

check_bestpreds_resamples *Check resamples*

Description

Check that the resamples from a caretList are valid

Usage

```
check_bestpreds_resamples(modelLibrary)
```

Arguments

modelLibrary	a list of predictins from caret models

check_caretList_classes *Checks caretList model classes*

Description

This function checks caretList classes

Usage

```
check_caretList_classes(list_of_models)
```

Arguments

list_of_models	a list of caret models to check

check_caretList_model_types *Checks that caretList models are all of the same type.*

Description

Validate a caretList

Usage

```
check_caretList_model_types(list_of_models)
```

Arguments

list_of_models	a list of caret models to check

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

```
## S3 method for class 'caretStack'
dotplot(x, data = NULL, ...)
```

Arguments

x	An object of class caretStack
data	passed to dotplot
...	passed to dotplot

Examples

```
## Not run:
set.seed(42)
library("rpart")
models <- caretList(
  x=iris[1:100,1:2],
  y=iris[1:100,3],
  trControl=trainControl(method="cv"),
  methodList=c("rpart", "glm")
```

```
)
meta_model <- caretStack(models, method="lm", trControl=trainControl(method="cv"))
dotplot.caretStack(meta_model)

## End(Not run)
```

`extractBestPreds` *Extract the best predictions from a list of train objects*

Description

Extract predictions for the best tune from a list of caret models

Usage

```
extractBestPreds(list_of_models)
```

Arguments

`list_of_models`	an object of class caretList or a list of caret models

`extractCaretTarget` *Extracts the target variable from a set of arguments headed to the caret::train function.*

Description

This function extracts the y variable from a set of arguments headed to a caret::train model. Since there are 2 methods to call caret::train, this function also has 2 methods.

Usage

```
extractCaretTarget(...)
```

Arguments

`...`	a set of arguments, as in the caret::train function

`extractCaretTarget.default` *Extracts the target variable from a set of arguments headed to the caret::train.default function.*

Description

This function extracts the y variable from a set of arguments headed to a caret::train.default model.

Usage

```
## Default S3 method:
extractCaretTarget(x, y, ...)
```

Arguments

`x`	an object where samples are in rows and features are in columns. This could be a simple matrix, data frame or other type (e.g. sparse matrix). See Details below.
`y`	a numeric or factor vector containing the outcome for each sample.
`...`	ignored

`extractCaretTarget.formula` *Extracts the target variable from a set of arguments headed to the caret::train.formula function.*

Description

This function extracts the y variable from a set of arguments headed to a caret::train.formula model.

Usage

```
## S3 method for class 'formula'
extractCaretTarget(form, data, ...)
```

Arguments

`form`	A formula of the form y ~ x1 + x2 + ...
`data`	Data frame from which variables specified in formula are preferentially to be taken.
`...`	ignored

extractModelName *Extract the method name associated with a single train object*

Description

Extracts the method name associated with a single train object. Note that for standard models (i.e. those already prespecified by caret), the "method" attribute on the train object is used directly while for custom models the "method" attribute within the model$modelInfo attribute is used instead.

Usage

```
extractModelName(x)
```

Arguments

x	a single caret train object

Value

Name associated with model

extractModelTypes *Extracts the model types from a list of train model*

Description

Extracts the model types from a list of train model

Usage

```
extractModelTypes(list_of_models)
```

Arguments

list_of_models	an object of class caretList

`extractModFrame` *Extract a dataframe of all predictors used in a caretEnsemble object.*

Description

This function constructs a dataframe consisting of the outcome and all of the predictors used in any of the models ensembled in a `caretEnsemble` object.

Usage

```
extractModFrame(model)
```

Arguments

`model`	a `caretEnsemble` to extract predictors from

Value

A data.frame combining all of the variables used across all models.

`extractModRes` *Extract the model accuracy metrics of the individual models in an ensemble object.*

Description

Extract the model accuracy metrics of the individual models in an ensemble object.

Usage

```
extractModRes(ensemble)
```

Arguments

`ensemble`	a caretEnsemble to make predictions from.

fortify *Supplement the data fitted to a caret ensemble model with model fit statistics*

Description

This function constructs a dataframe consisting of the outcome, all of the predictors used in any of the models ensembled in a `caretEnsemble` object, and some model fit statistics.

Usage

```
fortify(model, data = NULL, ...)
```

Arguments

`model`	a `caretEnsemble` to extract predictors from
`data`	a data set, defaults to the data used to fit the model
`...`	additional arguments to pass to fortify

Value

The original data with extra columns for fitted values and residuals

getBinaryTargetLevel *Return the configured target binary class level*

Description

For binary classification problems, ensemble stacks and certain performance measures require an awareness of which class in a two-factor outcome is the "target" class. By default, this class will be assumed to be the first level in an outcome factor but that setting can be overridden using `setBinaryTargetLevel(2L)`.

Usage

```
getBinaryTargetLevel()
```

Value

Currently configured binary target level (as integer equal to 1 or 2)

See Also

setBinaryTargetLevel

getMetric *Extract accuracy metrics from a model*

Description

Extract accuracy metrics from a model

Extract accuracy metrics SDs from a model

Extract a model accuracy metric from a `train` object.

Extract the standard deviation from resamples for an accuracy metric from a model object.

Usage

```
getMetric(x, metric, ...)

getMetricSD(x, metric, ...)

## S3 method for class 'train'
getMetric(x, metric = NULL, ...)

## S3 method for class 'train'
getMetricSD(x, metric, ...)
```

Arguments

`x`	a train object
`metric`	which metric to get
`...`	passed through

Value

A numeric representing the metric desired metric.

is.caretEnsemble *Check if an object is a caretEnsemble object*

Description

Check if an object is a caretEnsemble object

Usage

```
is.caretEnsemble(object)
```

Arguments

`object`	an R object

is.caretList *Check if an object is a caretList object*

Description

Check if an object is a caretList object

Usage

```
is.caretList(object)
```

Arguments

object	an R object

is.caretStack *Check if an object is a caretStack object*

Description

Check if an object is a caretStack object

Usage

```
is.caretStack(object)
```

Arguments

object	an R object

makePredObsMatrix *Make a prediction matrix from a list of models*

Description

Extract obs from one models, and a matrix of predictions from all other models, a helper function

Usage

```
makePredObsMatrix(list_of_models)
```

Arguments

list_of_models	an object of class caretList

methodCheck *Check that the methods supplied by the user are valid caret methods*

Description

This function uses modelLookup from caret to ensure the list of methods supplied by the user are all models caret can fit.

Usage

```
methodCheck(x)
```

Arguments

x	a list of user-supplied tuning parameters and methods

models.class *caretList of classification models*

Description

Data for the caretEnsemble package

Author(s)

Zachary Deane-Mayer <zach.mayer@gmail.com>

Zachary Deane-Mayer <zach.mayer@gmail.com>

Zachary Deane-Mayer <zach.mayer@gmail.com>

Zachary Deane-Mayer <zach.mayer@gmail.com>

Zachary Deane-Mayer <zach.mayer@gmail.com>

multiResiduals *Calculate the residuals from all component models of a caretEnsemble.*

Description

This function calculates raw residuals for both regression and classification `train` objects within a `caretEnsemble`.

Usage

```
multiResiduals(object, ...)
```

Arguments

`object`	a `caretEnsemble` to make predictions from.
`...`	other arguments to be passed to residuals

Value

A data.frame in the long format with columns for the model method, the observation id, yhat for the fitted values, resid for the residuals, and y for the observed value.

plot.caretEnsemble *Plot Diagnostics for an caretEnsemble Object*

Description

This function makes a short plot of the performance of the component models of a `caretEnsemble` object on the AUC or RMSE metric

Usage

```
## S3 method for class 'caretEnsemble'
plot(x, ...)
```

Arguments

`x`	a `caretEnsemble` object
`...`	additional arguments to pass to plot

Value

A plot

Examples

```
## Not run:
set.seed(42)
models <- caretList(iris[1:50,1:2], iris[1:50,3], methodList=c("glm", "rpart"))
ens <- caretEnsemble(models)
plot(ens)

## End(Not run)
```

`plot.caretStack` *Plot a caretStack object*

Description

This is a function to plot a caretStack.

Usage

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

Arguments

`x`	An object of class caretStack
`...`	passed to plot

Examples

```
## Not run:
library("rpart")
models <- caretList(
  x=iris[1:100,1:2],
  y=iris[1:100,3],
  trControl=trainControl(method="cv"),
  methodList=c("rpart", "glm")
)
meta_model <- caretStack(models, method="rpart", tuneLength=2)
plot(meta_model)

## End(Not run)
```

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

```
## S3 method for class 'caretList'
predict(object, newdata = NULL, ...,
  verbose = FALSE)
```

Arguments

object	an object of class caretList
newdata	New data for predictions. It can be NULL, but this is ill-advised.
...	additional arguments to pass to predict.train. Pass the newdata argument here, DO NOT PASS the "type" argument. Classification models will return probabilities if possible, and regression models will return "raw".
verbose	Logical. If FALSE no progress bar is printed if TRUE a progress bar is shown. Default FALSE.

predict.caretStack *Make predictions from a caretStack*

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, return_weights = FALSE, na.action = na.omit, ...)
```

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
return_weights	a logical indicating whether prediction weights for each model should be returned
na.action	the method for handling missing data passed to predict.train.
...	arguments to pass to predict.train.

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.

Examples

```
## Not run:
library("rpart")
models <- caretList(
  x=iris[1:100,1:2],
  y=iris[1:100,3],
  trControl=trainControl(method="cv"),
  methodList=c("rpart", "glm")
)
meta_model <- caretStack(models, method="lm")
RMSE(predict(meta_model, iris[101:150,1:2]), iris[101:150,3])

## End(Not run)
```

---

print.caretStack *Print a caretStack object*

Description

This is a function to print a caretStack.

Usage

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

Arguments

x	An object of class caretStack
...	ignored

Examples

```
## Not run:
library("rpart")
models <- caretList(
  x=iris[1:100,1:2],
  y=iris[1:100,3],
  trControl=trainControl(method="cv"),
  methodList=c("rpart", "glm")
)
meta_model <- caretStack(models, method="lm")
print(meta_model)

## End(Not run)
```

`residuals.caretEnsemble` *Calculate the residuals from a caretEnsemble.*

Description

This function calculates raw residuals for both regression and classification `caretEnsemble` objects.

Usage

```
## S3 method for class 'caretEnsemble'
residuals(object, ...)
```

Arguments

`object`	a `caretEnsemble` to make predictions from.
`...`	other arguments to be passed to residuals

Value

A numeric of the residuals.

`setBinaryTargetLevel` *Set the target binary class level*

Description

For binary classification problems, ensemble stacks and certain performance measures require an awareness of which class in a two-factor outcome is the "target" class. By default, the first level in an outcome factor is used but this value can be overridden using `setBinaryTargetLevel(2L)`

Usage

```
setBinaryTargetLevel(level)
```

Arguments

level	an integer in {1, 2} to be used as target outcome level

See Also

getBinaryTargetLevel

`summary.caretEnsemble` *Summarize the results of caretEnsemble for the user.*

Description

Summarize a caretEnsemble

Usage

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

Arguments

object	a caretEnsemble to make predictions from.
...	optional additional parameters.

Examples

```
## Not run:
set.seed(42)
models <- caretList(iris[1:50,1:2], iris[1:50,3], methodList=c("glm", "lm"))
ens <- caretEnsemble(models)
summary(ens)

## End(Not run)
```

summary.caretStack *Summarize a caretStack object*

Description

This is a function to summarize a caretStack.

Usage

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

Arguments

object	An object of class caretStack
...	ignored

Examples

```
## Not run:
library("rpart")
models <- caretList(
  x=iris[1:100,1:2],
  y=iris[1:100,3],
  trControl=trainControl(method="cv"),
  methodList=c("rpart", "glm")
)
meta_model <- caretStack(models, method="lm")
summary(meta_model)

## End(Not run)
```

trControlCheck *Check that the trainControl object supplied by the user is valid and has defined re-sampling indexes.*

Description

This function checks the user-supplied trainControl object and makes sure it has all the required fields. If the resampling indexes are missing, it adds them to the model. If savePredictions=FALSE or "none", this function sets it to "final".

Usage

```
trControlCheck(x, y)
```

Arguments

x	a trainControl object.
y	the target for the model. Used to determine resampling indexes.

---

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

```
tuneCheck(x)
```

Arguments

x	a list of user-supplied tuning parameters and methods

---

validateBinaryTargetLevel *Validate arguments given as binary target level*

Description

Helper function used to ensure that target binary class levels given by clients can be coerced to an integer and that the resulting integer is in $\{1, 2\}$.

Usage

```
validateBinaryTargetLevel(arg)
```

Arguments

arg	argument to potentially be used as new target level

Value

Binary target level (as integer equal to 1 or 2)

validateCustomModel *Validate a custom caret model info list*

Description

Currently, this only ensures that all model info lists were also assigned a "method" attribute for consistency with usage of non-custom models

Usage

```
validateCustomModel(x)
```

Arguments

x	a model info list (e.g. `getModelInfo("rf",regex=F)\[[1]]`)

Value

validated model info list (i.e. x)

varImp.caretEnsemble *Calculate the variable importance of variables in a caretEnsemble.*

Description

This function wraps the `varImp` function in the `caret` package to provide a weighted estimate of the importance of variables in the ensembled models in a `caretEnsemble` object. Variable importance for each model is calculated and then averaged by the weight of the overall model in the ensembled object.

Usage

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

Arguments

object	a `caretEnsemble` to make predictions from.
...	other arguments to be passed to varImp

Value

A `data.frame` with one row per variable and one column per model in object

wtd.sd *Calculate a weighted standard deviation*

Description

Used to weight deviations among ensembled model preditions

Usage

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

Arguments

x	a vector of numerics
w	a vector of weights equal to length of x
na.rm	a logical indicating how to handle missing values, default = FALSE

[.caretList *Index a caretList*

Description

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

Usage

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

Arguments

object	an object of class caretList
index	selected index

Index