Package ‘radiant.model’

May 13, 2021

Type Package
Title Model Menu for Radiant: Business Analytics using R and Shiny
Version 1.4.0
Date 2021-5-12
Description The Radiant Model menu includes interfaces for linear and logistic regression, naive Bayes, neural networks, classification and regression trees, model evaluation, collaborative filtering, decision analysis, and simulation. The application extends the functionality in 'radiant.data'.

Depends R (>= 3.4.0), radiant.data (>= 1.3.0)
Imports radiant.basics (>= 1.3.0), shiny (>= 1.4.0), nnet (>= 7.3.12), NeuralNetTools (>= 1.5.1), sandwich (>= 2.3.4), car (>= 2.1.3), ggplot2 (>= 2.2.1), data.tree (>= 0.7.4), stringr (>= 1.1.0), lubridate (>= 1.7.2), tidyrr (>= 0.8.2), dplyr (>= 0.8.3), rlang (>= 0.4.0), magrittr (>= 1.5), DiagrammeR (>= 1.0.0), import (>= 1.1.0), psych (>= 1.8.4), e1071 (>= 1.6.8), rpart (>= 4.1.11), ggrepel (>= 0.8), broom (>= 0.7.0), patchwork (>= 1.0.0), ranger (>= 0.11.2), xgboost (>= 0.90.0.2), pdp (>= 0.7.0), yaml
Suggests testthat (>= 2.0.0), pkgdown (>= 1.1.0)

BugReports https://github.com/radiant-rstats/radiant.model/issues/
License AGPL-3 | file LICENSE
LazyData true
Encoding UTF-8
Language en-US
RoxygenNote 7.1.1

Author Vincent Nijs [aut, cre]
Maintainer  Vincent Nijs <radiant@rady.ucsd.edu>
Repository  CRAN
Date/Publication  2021-05-13 21:20:02 UTC

R topics documented:

- auc
- catalog
- confint_robust
- confusion
- crs
- crtree
- cv.crtree
- cv.gbt
- cv.nn
- cv.rforest
- direct_marketing
- dtree
- dtree_parser
- dvd
- evalbin
- evalreg
- find_max
- find_min
- gbt
- houseprices
- ideal
- ketchup
- logistic
- MAE
- minmax
- mnl
- movie_contract
- nb
- nn
- onehot
- plot.confusion
- plot.crs
- plot.crtree
- plot.dtree
- plot.evalbin
- plot.evalreg
- plot.gbt
- plot.logistic
- plot.mnl
- plot.mnl.predict
- plot.model.predict
### R topics documented:

- `plot.nb` ................................................. 44
- `plot.nb.predict` ....................................... 45
- `plot.nn` .................................................. 46
- `plot.regress` ........................................... 47
- `plot.repeater` .......................................... 49
- `plot.rforest` .......................................... 49
- `plot.rforest.predict` ................................... 51
- `plot.simulater` ......................................... 52
- `predict.crtree` ......................................... 53
- `predict.gbt` ............................................ 54
- `predict.logistic` ........................................ 55
- `predict.mnl` ............................................ 56
- `predict.nb` ............................................. 57
- `predict.nn` ............................................. 59
- `predict.regressor` ....................................... 60
- `predict.rforest` ........................................ 61
- `predict_model` .......................................... 62
- `print.crtree.predict` .................................... 63
- `print.gbt.predict` ....................................... 64
- `print.logistic.predict` .................................. 64
- `print.mnl.predict` ....................................... 65
- `print.nb.predict` ........................................ 65
- `print.nn.predict` ........................................ 66
- `print.regress.predict` .................................... 66
- `print.rforest.predict` .................................... 67
- `print_predict_model` ..................................... 67
- `profit` ..................................................... 68
- `radiant.model` .......................................... 68
- `radiant.model-deprecated` ................................ 69
- `radiant.model-viewer` .................................... 69
- `radiant.model-window` .................................... 70
- `ratings` ..................................................... 71
- `regress` ..................................................... 71
- `render.DiagrammeR` ....................................... 72
- `repeater` .................................................... 73
- `rforest` ..................................................... 74
- `rig` ........................................................ 76
- `RMSE` ....................................................... 77
- `Rsq` ......................................................... 77
- `scale_df` ................................................... 78
- `sdw` ........................................................ 78
- `sensitivity` ............................................... 79
- `sensitivity.dtree` ......................................... 79
- `simulater` .................................................. 80
- `sim_cleaner` .............................................. 83
- `sim_cor` .................................................... 83
- `sim_splitter` .............................................. 84
- `sim_summary` .............................................. 84
Area Under the RO Curve (AUC)

**Description**

Area Under the RO Curve (AUC)

**Usage**

```
auc(pred, rvar, lev)
```

**Arguments**

- `pred`: Prediction or predictor
- `rvar`: Response variable
- `lev`: The level in the response variable defined as success

**Details**

See [https://radiant-rstats.github.io/docs/model/evalbin.html](https://radiant-rstats.github.io/docs/model/evalbin.html) for an example in Radiant
Value

AUC statistic

See Also

evalbin to calculate results
summary.evalbin to summarize results
plot.evalbin to plot results

Examples

auc(runif(20000), dvd$buy, "yes")
auc(ifelse(dvd$buy == "yes", 1, 0), dvd$buy, "yes")

---

catalog

Catalog sales for men’s and women’s apparel

data(catalog)

Description

Catalog sales for men’s and women’s apparel

Usage

data(catalog)

Format

A data frame with 200 rows and 5 variables

Details

Description provided in attr(catalog, "description")
confint_robust  Confidence interval for robust estimators

Description
Confidence interval for robust estimators

Usage
confint_robust(object, level = 0.95, dist = "norm", vcov = NULL, ...)

Arguments
- object: A fitted model object
- level: The confidence level required
- dist: Distribution to use ("norm" or "t")
- vcov: Covariance matrix generated by, e.g., sandwich::vcovHC
- ...: Additional argument(s) for methods

Details
Wrapper for confint with robust standard errors. See https://stackoverflow.com/questions/3817182/vcovhc-and-confidence-interval/3820125#3820125

confusion  Confusion matrix

Description
Confusion matrix

Usage
confusion(dataset, pred, rvar, lev = "", cost = 1, margin = 2, train = "All", data_filter = "", envir = parent.frame(), ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataset</td>
<td>Dataset</td>
</tr>
<tr>
<td>pred</td>
<td>Predictions or predictors</td>
</tr>
<tr>
<td>rvar</td>
<td>Response variable</td>
</tr>
<tr>
<td>lev</td>
<td>The level in the response variable defined as success</td>
</tr>
<tr>
<td>cost</td>
<td>Cost for each connection (e.g., email or mailing)</td>
</tr>
<tr>
<td>margin</td>
<td>Margin on each customer purchase</td>
</tr>
<tr>
<td>train</td>
<td>Use data from training (&quot;Training&quot;), test (&quot;Test&quot;), both (&quot;Both&quot;), or all data (&quot;All&quot;) to evaluate model evalbin</td>
</tr>
<tr>
<td>data_filter</td>
<td>Expression entered in, e.g., Data &gt; View to filter the dataset in Radiant. The expression should be a string (e.g., &quot;price &gt; 10000&quot;)</td>
</tr>
<tr>
<td>envir</td>
<td>Environment to extract data from</td>
</tr>
<tr>
<td>...</td>
<td>further arguments passed to or from other methods</td>
</tr>
</tbody>
</table>

Details

Confusion matrix and additional metrics to evaluate binary classification models. See https://radiantr-stats.github.io/docs/model/evalbin.html for an example in Radiant

Value

A list of results

See Also

summary.confusion to summarize results

plot.confusion to plot results

Examples

data.frame(buy = dvd$buy, pred1 = runif(20000), pred2 = ifelse(dvd$buy == "yes", 1, 0)) %>%
  confusion(c("pred1", "pred2"), "buy") %>%
  str()
**Arguments**

- **dataset**: Dataset
- **id**: String with name of the variable containing user ids
- **prod**: String with name of the variable with product ids
- **pred**: Products to predict for
- **rate**: String with name of the variable with product ratings
- **data_filter**: Expression entered in, e.g., Data > View to filter the dataset in Radiant. The expression should be a string (e.g., “training == 1”)
- **envir**: Environment to extract data from

**Details**

See [https://radiant-rstats.github.io/docs/model/crs.html](https://radiant-rstats.github.io/docs/model/crs.html) for an example in Radiant

**Value**

A data.frame with the original data and a new column with predicted ratings

**See Also**

- `summary.crs` to summarize results
- `plot.crs` to plot results if the actual ratings are available

**Examples**

```r
crs(ratings, id = "Users", prod = "Movies", pred = c("M6", "M7", "M8", "M9", "M10"),
    rate = "Ratings", data_filter = "training == 1") %>% str()
```

---

**crtree**

*Classification and regression trees based on the rpart package*

**Description**

Classification and regression trees based on the rpart package

**Usage**

```r
crtree(
    dataset,
    rvar,
    evar,
    type = "",
    lev = "",
    wts = "None",
    minsplit = 2,
```

---
minbucket = round(minsplit/3),
cp = 0.001,
pcp = NA,
nodes = NA,
K = 10,
seed = 1234,
split = "gini",
prior = NA,
adjprob = TRUE,
cost = NA,
margin = NA,
check = "",
data_filter = "",
envir = parent.frame()
)

Arguments

dataset Dataset
rvar The response variable in the model
evar Explanatory variables in the model
type Model type (i.e., "classification" or "regression")
lev The level in the response variable defined as _success_
wts Weights to use in estimation
minsplit The minimum number of observations that must exist in a node in order for a split to be attempted.
minbucket the minimum number of observations in any terminal <leaf> node. If only one of minbucket or minsplit is specified, the code either sets minsplit to minbucket*3 or minbucket to minsplit/3, as appropriate.
cp Minimum proportion of root node deviance required for split (default = 0.001)
pcp Complexity parameter to use for pruning
nodes Maximum size of tree in number of nodes to return
K Number of folds use in cross-validation
seed Random seed used for cross-validation
split Splitting criterion to use (i.e., "gini" or "information")
prior Adjust the initial probability for the selected level (e.g., set to .5 in unbalanced samples)
adjprob Setting a prior will rescale the predicted probabilities. Set adjprob to TRUE to adjust the probabilities back to their original scale after estimation
cost Cost for each treatment (e.g., mailing)
margin Margin associated with a successful treatment (e.g., a purchase)
check Optional estimation parameters (e.g., "standardize")
data_filter Expression entered in, e.g., Data > View to filter the dataset in Radiant. The expression should be a string (e.g., "price > 10000")
envir Environment to extract data from
cv.crtree

Details
See https://radiant-rstats.github.io/docs/model/crtree.html for an example in Radiant

Value
A list with all variables defined in crtree as an object of class tree

See Also
summary.crtree to summarize results
plot.crtree to plot results
predict.crtree for prediction

Examples
crtree(titanic, "survived", c("pclass", "sex"), lev = "Yes") %>% summary()
result <- crtree(titanic, "survived", c("pclass", "sex")) %>% summary()
result <- crtree(diamonds, "price", c("carat", "clarity"), type = "regression") %>% str()

---

cv.crtree

Cross-validation for Classification and Regression Trees

Description
Cross-validation for Classification and Regression Trees

Usage
cv.crtree(
  object,
  K = 5,
  repeats = 1,
  cp,
  pcp = seq(0, 0.01, length.out = 11),
  seed = 1234,
  trace = TRUE,
  fun,
  ...
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Object of type &quot;rpart&quot; or &quot;crtree&quot; to use as a starting point for cross validation</td>
</tr>
<tr>
<td>K</td>
<td>Number of cross validation passes to use</td>
</tr>
<tr>
<td>repeats</td>
<td>Number of times to repeat the K cross-validation steps</td>
</tr>
<tr>
<td>cp</td>
<td>Complexity parameter used when building the (e.g., 0.0001)</td>
</tr>
</tbody>
</table>
cv.gbt

Cross-validation for Gradient Boosted Trees

Description

Cross-validation for Gradient Boosted Trees

pcp Complexity parameter to use for pruning
seed Random seed to use as the starting point
trace Print progress
fun Function to use for model evaluation (e.g., auc for classification or RMSE for regression)
... Additional arguments to be passed to `fun`

Details

See [https://radiant-rstats.github.io/docs/model/cr}\text{tree\textunderscore htm\text{l} for an example in Radiant

Value

A data.frame sorted by the mean, sd, min, and max of the performance metric

See Also

- `cr}\text{tree\textunderscore hte}` to generate an initial model that can be passed to cv.cr}\text{tree\textunderscore hte}
- `Rsq` to calculate an R-squared measure for a regression
- `RMSE` to calculate the Root Mean Squared Error for a regression
- `MAE` to calculate the Mean Absolute Error for a regression
- `auc` to calculate the area under the ROC curve for classification
- `profit` to calculate profits for classification at a cost/margin threshold

Examples

```r
# Not run:
result <- cr}\text{tree\textunderscore hte(dvd, \textquote{buy}, c(\textquote{coupon}, \textquote{purch}, \textquote{last}))
result <- cr}\text{tree\textunderscore hte(result, cp = 0.0001, pcp = seq(0, 0.01, length.out = 11))
result <- cr}\text{tree\textunderscore hte(result, cp = 0.0001, pcp = c(0, 0.001, 0.002), fun = profit, cost = 1, margin = 5)
result <- cr}\text{tree\textunderscore hte(diamonds, \textquote{price}, c(\textquote{carat}, \textquote{color}, \textquote{clarity}), type = \textquote{regression}, cp = 0.001)
result <- cr}\text{tree\textunderscore hte(result, cp = 0.001, pcp = seq(0, 0.01, length.out = 11), fun = MAE)
```

## End(Not run)
Usage

cv.gbt(
  object,
  K = 5,
  repeats = 1,
  params = list(),
  nrounds = 500,
  early_stopping_rounds = 10,
  nthread = 12,
  train = NULL,
  type = "classification",
  trace = TRUE,
  seed = 1234,
  maximize = NULL,
  fun,
  ...
)

Arguments

object Object of type "gbt" or "ranger"
K Number of cross validation passes to use (aka nfold)
repeats Repeated cross validation
params List of parameters (see XGBoost documentation)
nrounds Number of trees to create
early_stopping_rounds Early stopping rule
nthread Number of parallel threads to use. Defaults to 12 if available
train An optional xgb.DMatrix object containing the original training data. Not needed when using Radiant’s gbt function
type Model type ("classification" or "regression")
trace Print progress
seed Random seed to use as the starting point
maximize When a custom function is used, xgb.cv requires the user indicate if the function output should be maximized (TRUE) or minimized (FALSE)
fun Function to use for model evaluation (i.e., auc for classification and RMSE for regression)
... Additional arguments to be passed to 'fun'

Details

See https://radiant-rstats.github.io/docs/model/gbt.html for an example in Radiant

Value

A data.frame sorted by the mean of the performance metric
See Also

gbt to generate an initial model that can be passed to cv.gbt
Rsq to calculate an R-squared measure for a regression
RMSE to calculate the Root Mean Squared Error for a regression
MAE to calculate the Mean Absolute Error for a regression
auc to calculate the area under the ROC curve for classification
profit to calculate profits for classification at a cost/margin threshold

Examples

```r
## Not run:
result <- gbt(dvd, "buy", c("coupon", "purch", "last"))
cv.gbt(result, params = list(max_depth = 1:6))
cv.gbt(result, params = list(max_depth = 1:6), fun = "logloss")
cv.gbt(
  result,
  params = list(learning_rate = seq(0.1, 1.0, 0.1)),
  maximize = TRUE, fun = profit, cost = 1, margin = 5
)
result <- gbt(diamonds, "price", c("carat", "color", "clarity"), type = "regression")
cv.gbt(result, params = list(max_depth = 1:2, min_child_weight = 1:2))
cv.gbt(result, params = list(learning_rate = seq(0.1, 0.5, 0.1)), fun = Rsq, maximize = TRUE)
cv.gbt(result, params = list(learning_rate = seq(0.1, 0.5, 0.1)), fun = MAE, maximize = FALSE)
rig_wrap <- function(preds, dtrain) {
  labels <- xgboost::getinfo(dtrain, "label")
  value <- rig(preds, labels, lev = 1)
  list(metric = "rig", value = value)
}
result <- gbt(titanic, "survived", c("pclass", "sex"), eval_metric = rig_wrap, maximize = TRUE)
cv.gbt(result, params = list(learning_rate = seq(0.1, 0.5, 0.1)))

## End(Not run)
```

### Cross-validation for a Neural Network

---

**Description**

Cross-validation for a Neural Network

**Usage**

```r
cv.nn(
  object,
  K = 5,
  repeats = 1,
)```
cv.nn

```r
decay = seq(0, 1, 0.2),
size = 1:5,
seed = 1234,
trace = TRUE,
fun,
...
)
```

**Arguments**

- `object`: Object of type "nn" or "nnet"
- `K`: Number of cross validation passes to use
- `repeats`: Repeated cross validation
- `decay`: Parameter decay
- `size`: Number of units (nodes) in the hidden layer
- `seed`: Random seed to use as the starting point
- `trace`: Print progress
- `fun`: Function to use for model evaluation (i.e., auc for classification and RMSE for regression)
- `...`: Additional arguments to be passed to 'fun'

**Details**

See [https://radiant-rstats.github.io/docs/model/nn.html](https://radiant-rstats.github.io/docs/model/nn.html) for an example in Radiant

**Value**

A data.frame sorted by the mean of the performance metric

**See Also**

- `nn` to generate an initial model that can be passed to `cv.nn`
- `Rsq` to calculate an R-squared measure for a regression
- `RMSE` to calculate the Root Mean Squared Error for a regression
- `MAE` to calculate the Mean Absolute Error for a regression
- `auc` to calculate the area under the ROC curve for classification
- `profit` to calculate profits for classification at a cost/margin threshold

**Examples**

```r
## Not run:
result <- nn(dvd, "buy", c("coupon", "purch", "last"))
cv.nn(result, decay = seq(0, 1, .5), size = 1:2)
cv.nn(result, decay = seq(0, 1, .5), size = 1:2, fun = profit, cost = 1, margin = 5)
result <- nn(diamonds, "price", c("carat", "color", "clarity"), type = "regression")
cv.nn(result, decay = seq(0, 1, .5), size = 1:2)
```
cv.rforest

cv.nn(result, decay = seq(0, 1, .5), size = 1:2, fun = Rsq)

## End(Not run)

cv.rforest

Cross-validation for a Random Forest

Description

Cross-validation for a Random Forest

Usage

cv.rforest(
  object,
  K = 5,
  repeats = 1,
  mtry = 1:5,
  num.trees = NULL,
  min.node.size = 1,
  sample.fraction = NA,
  trace = TRUE,
  seed = 1234,
  fun,
  ...
)

Arguments

object Object of type "rforest" or "ranger"
K Number of cross validation passes to use
repeats Repeated cross validation
mtry Number of variables to possibly split at in each node. Default is the (rounded down) square root of the number variables
num.trees Number of trees to create
min.node.size Minimal node size
sample.fraction Fraction of observations to sample. Default is 1 for sampling with replacement and 0.632 for sampling without replacement
trace Print progress
seed Random seed to use as the starting point
fun Function to use for model evaluation (i.e., auc for classification and RMSE for regression)
... Additional arguments to be passed to 'fun'
direct_marketing

Direct marketing data

Description

Direct marketing data

Usage

data(direct_marketing)
dtree

Format
A data frame with 1,000 rows and 12 variables

Details
Description provided in attr(direct_marketing, "description")

<table>
<thead>
<tr>
<th>dtree</th>
<th>Create a decision tree</th>
</tr>
</thead>
</table>

Description
Create a decision tree

Usage
dtree(yl, opt = "max", base = character(0), envir = parent.frame())

Arguments
- **yl**: A yaml string or a list (e.g., from yaml::yaml.load_file())
- **opt**: Find the maximum ("max") or minimum ("min") value for each decision node
- **base**: List of variable definitions from a base tree used when calling a sub-tree
- **envir**: Environment to extract data from

Details
See https://radiant-rstats.github.io/docs/model/dtree.html for an example in Radiant

Value
A list with the initial tree and the calculated tree

See Also
- summary.dtree to summarize results
- plot.dtree to plot results
- sensitivity.dtree to plot results

Examples
yaml::as.yaml(movie_contract) %>% cat()
dtree(movie_contract, opt = "max") %>% summary(output = TRUE)
dtree_parser

Parse yaml input for dtree to provide (more) useful error messages

Description

Parse yaml input for dtree to provide (more) useful error messages

Usage

dtree_parser(yl)

Arguments

yl A yaml string

Details

See https://radiant-rstats.github.io/docs/model/dtree.html for an example in Radiant

Value

An updated yaml string or a vector messages to return to the users

See Also

dtree to calculate tree
summary.dtree to summarize results
plot.dtree to plot results

dvd

Data on DVD sales

Description

Data on DVD sales

Usage

data(dvd)

Format

A data frame with 20,000 rows and 4 variables

Details

Binary purchase response to coupon value. Description provided in attr(dvd,"description")
Description
Evaluate the performance of different (binary) classification models

Usage
```r
evalbin(
    dataset,
    pred,
    rvar,
    lev = "",
    qnt = 10,
    cost = 1,
    margin = 2,
    train = "All",
    data_filter = "",
    envir = parent.frame()
)
```

Arguments
- `dataset` Dataset
- `pred` Predictions or predictors
- `rvar` Response variable
- `lev` The level in the response variable defined as success
- `qnt` Number of bins to create
- `cost` Cost for each connection (e.g., email or mailing)
- `margin` Margin on each customer purchase
- `train` Use data from training ("Training"), test ("Test"), both ("Both"), or all data ("All") to evaluate model `evalbin`
- `data_filter` Expression entered in, e.g., Data > View to filter the dataset in Radiant. The expression should be a string (e.g., "price > 10000")
- `envir` Environment to extract data from

Details
Evaluate different (binary) classification models based on predictions. See [https://radiant-rstats.github.io/docs/model/evalbin.html](https://radiant-rstats.github.io/docs/model/evalbin.html) for an example in Radiant

Value
A list of results
See Also

- `summary.evalbin` to summarize results
- `plot.evalbin` to plot results

Examples

data.frame(buy = dvd$buy, pred1 = runif(20000), pred2 = ifelse(dvd$buy == "yes", 1, 0)) %>%
  evalbin(c("pred1", "pred2"), "buy") %>%
  str()

---

**evalreg**

*Evaluate the performance of different regression models*

**Description**

Evaluate the performance of different regression models

**Usage**

```r
evalreg(
  dataset,  # Dataset
  pred,     # Predictions or predictors
  rvar,     # Response variable
  train = "All",  # Use data from training ("Training"), test ("Test"), both ("Both"), or all data ("All") to evaluate model evalreg
  data_filter = "",  # Expression entered in, e.g., Data > View to filter the dataset in Radiant. The expression should be a string (e.g., "training == 1")
  envir = parent.frame()  # Environment to extract data from
)
```

**Arguments**

- `dataset` : Dataset
- `pred` : Predictions or predictors
- `rvar` : Response variable
- `train` : Use data from training ("Training"), test ("Test"), both ("Both"), or all data ("All") to evaluate model evalreg
- `data_filter` : Expression entered in, e.g., Data > View to filter the dataset in Radiant. The expression should be a string (e.g., "training == 1")
- `envir` : Environment to extract data from

**Details**

Evaluate different regression models based on predictions. See [https://radiant-rstats.github.io/docs/model/evalreg.html](https://radiant-rstats.github.io/docs/model/evalreg.html) for an example in Radiant
find_max

Value
A list of results

See Also
summary.evalreg to summarize results
plot.evalreg to plot results

Examples

data.frame(price = diamonds$price, pred1 = rnorm(3000), pred2 = diamonds$price) %>%
evalreg(pred = c("pred1", "pred2"), "price") %>%
str()

find_max (x, y)

Arguments

x Variable to find the maximum for
y Variable to find the value for at the maximum of var

Details
Find the value of y at the maximum value of x

Value
Value of val at the maximum of var

Examples
find_max(1:10, 21:30)
find_min  
*Find minimum value of a vector*

**Description**

Find minimum value of a vector

**Usage**

```r
find_min(x, y)
```

**Arguments**

- `x`  
  Variable to find the minimum for
- `y`  
  Variable to find the value for at the maximum of var

**Details**

Find the value of y at the minimum value of x

**Value**

Value of val at the minimum of var

**Examples**

```r
find_min(1:10, 21:30)
```

---

gbt  
*Gradient Boosted Trees using XGBoost*

**Description**

Gradient Boosted Trees using XGBoost

**Usage**

```r
gbt(
  dataset,
  rvar,
  evar,
  type = "classification",
  lev = "",
  max_depth = 6,
  learning_rate = 0.3,
```

```r
)```
min_split_loss = 0,
min_child_weight = 1,
subsample = 1,
nrounds = 100,
early_stopping_rounds = 10,
nthread = 12,
wts = "None",
seed = NA,
data_filter = "",
envir = parent.frame(),
...
)

Arguments

dataset Dataset
rvar The response variable in the model
evar Explanatory variables in the model
type Model type (i.e., "classification" or "regression")
lev Level to use as the first column in prediction output
max_depth Maximum 'depth' of tree
learning_rate Learning rate (eta)
min_split_loss Minimal improvement (gamma)
min_child_weight Minimum number of instances allowed in each node
subsample Subsample ratio of the training instances (0-1)
nrounds Number of trees to create
early_stopping_rounds Early stopping rule
nthread Number of parallel threads to use. Defaults to 12 if available
wts Weights to use in estimation
seed Random seed to use as the starting point
data_filter Expression entered in, e.g., Data > View to filter the dataset in Radiant. The expression should be a string (e.g., "price > 10000")
envir Environment to extract data from
... Further arguments to pass to xgboost

Details

See [https://radiant-rstats.github.io/docs/model/gbt.html](https://radiant-rstats.github.io/docs/model/gbt.html) for an example in Radiant

Value

A list with all variables defined in gbt as an object of class gbt
See Also

- `summary.gbt` to summarize results
- `plot.gbt` to plot results
- `predict.gbt` for prediction

Examples

```r
## Not run:
gbt(titanic, "survived", c("pclass", "sex"), lev = "Yes") %>% summary()

gbt(titanic, "survived", c("pclass", "sex")) %>% str()

## End(Not run)

gbt(titanic, "survived", c("pclass", "sex"), lev = "Yes", early_stopping_rounds = 0) %>% summary()

gbt(titanic, "survived", c("pclass", "sex"), early_stopping_rounds = 0) %>% str()

gbt(titanic, "survived", c("pclass", "sex"), eval_metric = paste0("error@", 0.5 / 6)) %>% str()

gbt(diamonds, "price", c("carat", "clarity"), type = "regression") %>% str()

rig_wrap <- function(preds, dtrain) {
  labels <- xgboost::getinfo(dtrain, "label")
  value <- rig(preds, labels, lev = 1)
  list(metric = "rig", value = value)
}

gbt(titanic, "survived", c("pclass", "sex"), eval_metric = rig_wrap, maximize = TRUE) %>% str()
```

---

Houseprices

**Description**

Houseprices

**Usage**

data(houseprices)

**Format**

A data frame with 128 home sales and 6 variables

**Details**

Description provided in attr(houseprices, "description")
### ideal

**Ideal data for linear regression**

**Description**

Ideal data for linear regression

**Usage**

```r
data(ideal)
```

**Format**

A data frame with 1,000 rows and 4 variables

**Details**

Description provided in `attr(ideal, "description")`

---

### ketchup

**Data on ketchup choices**

**Description**

Data on ketchup choices

**Usage**

```r
data(ketchup)
```

**Format**

A data frame with 2,798 rows and 14 variables

**Details**

Choice behavior for a sample of 300 individuals in a panel of households in Springfield, Missouri (USA). Description provided in `attr(ketchup,"description")`
logistic **Logistic regression**

**Description**

Logistic regression

**Usage**

```r
logistic(
  dataset,
  rvar,
  evar,
  lev = "",
  int = "",
  wts = "None",
  check = "",
  form,
  ci_type,
  data_filter = "",
  envir = parent.frame()
)
```

**Arguments**

- `dataset` Dataset
- `rvar` The response variable in the model
- `evar` Explanatory variables in the model
- `lev` The level in the response variable defined as _success_
- `int` Interaction term to include in the model
- `wts` Weights to use in estimation
- `check` Use "standardize" to see standardized coefficient estimates. Use "stepwise-backward" (or "stepwise-forward", or "stepwise-both") to apply step-wise selection of variables in estimation. Add "robust" for robust estimation of standard errors (HC1)
- `form` Optional formula to use instead of rvar, evar, and int
- `ci_type` To use the profile-likelihood (rather than Wald) for confidence intervals use "profile". For datasets with more than 5,000 rows the Wald method will be used, unless "profile" is explicitly set
- `data_filter` Expression entered in, e.g., Data > View to filter the dataset in Radiant. The expression should be a string (e.g., "price > 10000")
- `envir` Environment to extract data from
Details

See https://radiant-rstats.github.io/docs/model/logistic.html for an example in Radiant

Value

A list with all variables defined in logistic as an object of class logistic

See Also

summary.logistic to summarize the results
plot.logistic to plot the results
predict.logistic to generate predictions
plot.model.predict to plot prediction output

Examples

logistic(titanic, "survived", c("pclass", "sex"), lev = "Yes") %>% summary()
logistic(titanic, "survived", c("pclass", "sex")) %>% str()

MAE

Mean Absolute Error

Description

Mean Absolute Error

Usage

MAE(pred, rvar)

Arguments

<table>
<thead>
<tr>
<th>pred</th>
<th>Prediction (vector)</th>
</tr>
</thead>
<tbody>
<tr>
<td>rvar</td>
<td>Response (vector)</td>
</tr>
</tbody>
</table>

Value

Mean Absolute Error
minmax

*Calculate min and max before standardization*

**Description**

Calculate min and max before standardization

**Usage**

```r
minmax(dataset)
```

**Arguments**

- `dataset` : Data frame

**Value**

Data frame min and max attributes

---

**mnl**

*Multinomial logistic regression*

**Description**

Multinomial logistic regression

**Usage**

```r
mnl(
    dataset, rvar, evar, lev = "", int = "", wts = "None", check = "", data_filter = "", envir = parent.frame()
)
```
Arguments

- **dataset**: Dataset
- **rvar**: The response variable in the model
- **evar**: Explanatory variables in the model
- **lev**: The level in the response variable to use as the baseline
- **int**: Interaction term to include in the model
- **wts**: Weights to use in estimation
- **check**: Use "standardize" to see standardized coefficient estimates. Use "stepwise-backward" (or "stepwise-forward", or "stepwise-both") to apply step-wise selection of variables in estimation.
- **data_filter**: Expression entered in, e.g., Data > View to filter the dataset in Radiant. The expression should be a string (e.g., "price > 10000")
- **envir**: Environment to extract data from

Details

See [https://radiant-rstats.github.io/docs/model/mnl.html](https://radiant-rstats.github.io/docs/model/mnl.html) for an example in Radiant

Value

A list with all variables defined in mnl as an object of class mnl

See Also

- `summary.mnl` to summarize the results
- `plot.mnl` to plot the results
- `predict.mnl` to generate predictions
- `plot.model.predict` to plot prediction output

Examples

```r
result <- mnl(
  ketchup,
  rvar = "choice",
  evar = c("price.heinz28", "price.heinz32", "price.heinz41", "price.hunts32"),
  lev = "heinz28"
)
str(result)
```
movie_contract

Movie contract decision tree

Description

Movie contract decision tree

Usage

data(movie_contract)

Format

A nested list for decision and chance nodes, probabilities and payoffs

Details

Use decision analysis to create a decision tree for an actor facing a contract decision

---

nb

Naive Bayes using e1071::naiveBayes

Description

Naive Bayes using e1071::naiveBayes

Usage

nb(dataset, rvar, evar, laplace = 0, data_filter = "", envir = parent.frame())

Arguments

dataset Dataset
rvar The response variable in the logit (probit) model
evar Explanatory variables in the model
laplace Positive double controlling Laplace smoothing. The default (0) disables Laplace smoothing.
data_filter Expression entered in, e.g., Data > View to filter the dataset in Radiant. The expression should be a string (e.g., "price > 10000")
evr Environment to extract data from

Details

See https://radiant-rstats.github.io/docs/model/nb.html for an example in Radiant
Value

A list with all variables defined in nb as an object of class nb

See Also

- `summary.nb` to summarize results
- `plot.nb` to plot results
- `predict.nb` for prediction

Examples

```r
nb(titanic, "survived", c("pclass", "sex", "age")) %>% summary()
nb(titanic, "survived", c("pclass", "sex", "age")) %>% str()
```

---

**neuralnetworks using nnet**

**Description**

Neural Networks using nnet

**Usage**

```r
nn(
  dataset,
  rvar,
  evar,
  type = "classification",
  lev = "",
  size = 1,
  decay = 0.5,
  wts = "None",
  seed = NA,
  check = "standardize",
  form,
  data_filter = "",
  envir = parent.frame()
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataset</td>
<td>Dataset</td>
</tr>
<tr>
<td>rvar</td>
<td>The response variable in the model</td>
</tr>
<tr>
<td>evar</td>
<td>Explanatory variables in the model</td>
</tr>
<tr>
<td>type</td>
<td>Model type (i.e., &quot;classification&quot; or &quot;regression&quot;)</td>
</tr>
</tbody>
</table>
### onehot

**Description**

One hot encoding of data.frames

**Usage**

```r
onehot(dataset, all = FALSE, df = FALSE)
```

---

<table>
<thead>
<tr>
<th>lev</th>
<th>The level in the response variable defined as <em>success</em>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>Number of units (nodes) in the hidden layer</td>
</tr>
<tr>
<td>decay</td>
<td>Parameter decay</td>
</tr>
<tr>
<td>wts</td>
<td>Weights to use in estimation</td>
</tr>
<tr>
<td>seed</td>
<td>Random seed to use as the starting point</td>
</tr>
<tr>
<td>check</td>
<td>Optional estimation parameters (&quot;standardize&quot; is the default)</td>
</tr>
<tr>
<td>form</td>
<td>Optional formula to use instead of rvar and evar</td>
</tr>
<tr>
<td>data_filter</td>
<td>Expression entered in, e.g., Data &gt; View to filter the dataset in Radiant. The expression should be a string (e.g., &quot;price &gt; 10000&quot;)</td>
</tr>
<tr>
<td>envir</td>
<td>Environment to extract data from</td>
</tr>
</tbody>
</table>

**Details**

See [https://radiant-rstats.github.io/docs/model/nn.html](https://radiant-rstats.github.io/docs/model/nn.html) for an example in Radiant

**Value**

A list with all variables defined in nn as an object of class nn

**See Also**

- `summary.nn` to summarize results
- `plot.nn` to plot results
- `predict.nn` for prediction

**Examples**

```r
nn(titanic, "survived", c("pclass", "sex"), lev = "Yes") %>% summary()
nn(titanic, "survived", c("pclass", "sex")) %>% str()
nn(diamonds, "price", c("carat", "clarity"), type = "regression") %>% summary()
```
### Arguments

- **dataset**: Dataset to encode
- **all**: Extract all factor levels (e.g., for tree-based models)
- **df**: Return a data.frame (tibble)

### Examples

```r
head(onehot(diamonds, df = TRUE))
head(onehot(diamonds, all = TRUE, df = TRUE))
```

---

**plot.confusion**  
*Plot method for the confusion matrix*

**Description**

Plot method for the confusion matrix

**Usage**

```r
## S3 method for class 'confusion'
plot(
x,
vars = c("kappa", "index", "ROME", "AUC"),
scale_y = TRUE,
size = 13,
...
)
```

**Arguments**

- **x**: Return value from `confusion`
- **vars**: Measures to plot, i.e., one or more of "TP", "FP", "TN", "FN", "total", "TPR", "TNR", "precision", "accuracy", "kappa", "profit", "index", "ROME", "contact", "AUC"
- **scale_y**: Free scale in faceted plot of the confusion matrix (TRUE or FALSE)
- **size**: Font size used
- **...**: further arguments passed to or from other methods

**Details**

See [https://radiant-rstats.github.io/docs/model/evalbin.html](https://radiant-rstats.github.io/docs/model/evalbin.html) for an example in Radiant
plot.crs

Description
Plot method for the crs function

Usage
## S3 method for class 'crs'
plot(x, ...)

Arguments
x  Return value from crs
... further arguments passed to or from other methods

Details
Plot that compares actual to predicted ratings. See https://radiant-rstats.github.io/docs/model/crs.html for an example in Radiant

See Also
crs to generate results
summary.crs to summarize results
plot.crtree

---

**Description**

Plot method for the crtree function

**Usage**

```r
## S3 method for class 'crtree'
plot(
    x,
    plots = "tree",
    orient = "LR",
    width = "900px",
    labs = TRUE,
    nrobs = Inf,
    dec = 2,
    shiny = FALSE,
    custom = FALSE,
    ...
)
```

**Arguments**

- `x` Return value from `crtree`
- `plots` Plots to produce for the specified rpart tree. "tree" shows a tree diagram. "prune" shows a line graph to evaluate appropriate tree pruning. "imp" shows a variable importance plot
- `orient` Plot orientation for tree: LR for vertical and TD for horizontal
- `width` Plot width in pixels for tree (default is "900px")
- `labs` Use factor labels in plot (TRUE) or revert to default letters used by tree (FALSE)
- `nrobs` Number of data points to show in dashboard scatter plots (-1 for all)
- `dec` Decimal places to round results to
- `shiny` Did the function call originate inside a shiny app
- `custom` Logical (TRUE, FALSE) to indicate if ggplot object (or list of ggplot objects) should be returned. This option can be used to customize plots (e.g., add a title, change x and y labels, etc.). See examples and [https://ggplot2.tidyverse.org](https://ggplot2.tidyverse.org) for options.
- `...` further arguments passed to or from other methods

**Details**

See [https://radiant-rstats.github.io/docs/model/crtree.html](https://radiant-rstats.github.io/docs/model/crtree.html) for an example in Radiant. The standard tree plot used by by the rpart package can be generated by `plot.rpart(result$model)`. See `plot.rpart` for additional details.
See Also

- `crtree` to generate results
- `summary.crtree` to summarize results
- `predict.crtree` for prediction

Examples

```r
result <- crtree(titanic, "survived", c("pclass", "sex"), lev = "Yes")
plot(result)
result <- crtree(diamonds, "price", c("carat", "clarity", "cut"))
plot(result, plots = "prune")
result <- crtree(dvd, "buy", c("coupon", "purch", "last"), cp = .01)
plot(result, plots = "imp")
```

---

**plot.dtree**

*Plot method for the dtree function*

Description

Plot method for the dtree function

Usage

```r
## S3 method for class 'dtree'
plot(
  x,
  symbol = "$",
  dec = 2,
  final = FALSE,
  orient = "LR",
  width = "900px",
  ...
)
```

Arguments

- `x` Return value from `dtree`
- `symbol` Monetary symbol to use ($ is the default)
- `dec` Decimal places to round results to
- `final` If TRUE plot the decision tree solution, else the initial decision tree
- `orient` Plot orientation: LR for vertical and TD for horizontal
- `width` Plot width in pixels (default is "900px")
- `...` further arguments passed to or from other methods
Details

See [https://radiant-rstats.github.io/docs/model/dtree.html](https://radiant-rstats.github.io/docs/model/dtree.html) for an example in Radiant

See Also

dtree to generate the result
summary.dtree to summarize results
sensitivity.dtree to plot results

Examples

dtree(movie_contract, opt = "max") %>% plot()
dtree(movie_contract, opt = "max") %>% plot(final = TRUE, orient = "TD")

---

**plot.evalbin**  
*Plot method for the evalbin function*

Description

Plot method for the evalbin function

Usage

```r
## S3 method for class 'evalbin'
plot(
  x,
  plots = c("lift", "gains"),
  size = 13,
  shiny = FALSE,
  custom = FALSE,
  ...
)
```

Arguments

- `x`  
  Return value from `evalbin`
- `plots`  
  Plots to return
- `size`  
  Font size used
- `shiny`  
  Did the function call originate inside a shiny app
- `custom`  
  Logical (TRUE, FALSE) to indicate if ggplot object (or list of ggplot objects) should be returned. This option can be used to customize plots (e.g., add a title, change x and y labels, etc.). See examples and [https://ggplot2.tidyverse.org](https://ggplot2.tidyverse.org) for options.
- `...`  
  further arguments passed to or from other methods
plot.evalreg

Details

See https://radiant-rstats.github.io/docs/model/evalbin.html for an example in Radiant

See Also

evalbin to generate results

summary.evalbin to summarize results

Examples

data.frame(buy = dvd$buy, pred1 = runif(20000), pred2 = ifelse(dvd$buy == "yes", 1, 0)) %>%
evalbin(c("pred1", "pred2"), "buy") %>%
plot()

plot.evalreg

Plot method for the evalreg function

Description

Plot method for the evalreg function

Usage

## S3 method for class 'evalreg'
plot(x, vars = c("Rsq", "RMSE", "MAE"), ...)

Arguments

x  
Return value from evalreg

vars  
Measures to plot, i.e., one or more of "Rsq", "RMSE", "MAE"

...  
further arguments passed to or from other methods

Details

See https://radiant-rstats.github.io/docs/model/evalreg.html for an example in Radiant

See Also

evalreg to generate results

summary.evalreg to summarize results
plot.gbt

Examples

data.frame(price = diamonds$price, pred1 = rnorm(3000), pred2 = diamonds$price) %>%
evalreg(pred = c("pred1", "pred2"), "price") %>%
plot()

Description

Plot method for the gbt function

Usage

## S3 method for class 'gbt'
plot(x, plots = "", nrobs = Inf, shiny = FALSE, custom = FALSE, ...)

Arguments

x Return value from gbt
plots Plots to produce for the specified Gradient Boosted Tree model. Use "" to avoid
showing any plots (default). Options are ...
nrobs Number of data points to show in scatter plots (-1 for all)
shiny Did the function call originate inside a shiny app
custom Logical (TRUE, FALSE) to indicate if ggplot object (or list of ggplot objects)
should be returned. This option can be used to customize plots (e.g., add a title,
change x and y labels, etc.). See examples and https://ggplot2.tidyverse.org
for options.
...
... further arguments passed to or from other methods

Details

See https://radiant-rstats.github.io/docs/model/gbt.html for an example in Radiant

See Also

gbt to generate results
summary.gbt to summarize results
predict.gbt for prediction

Examples

result <- gbt(titanic, "survived", c("pclass", "sex"), early_stopping_rounds = 0)
plot(result)
### Description

Plot method for the logistic function

### Usage

```r
## S3 method for class 'logistic'
plot(
x,
plots = "coef",
conf_lev = 0.95,
intercept = FALSE,
incl = NULL,
excl = NULL,
nrobs = -1,
shiny = FALSE,
custom = FALSE,
...
)
```

### Arguments

- **x**
  - Return value from `logistic`

- **plots**
  - Plots to produce for the specified GLM model. Use "" to avoid showing any plots (default). "dist" shows histograms (or frequency bar plots) of all variables in the model. "scatter" shows scatter plots (or box plots for factors) for the response variable with each explanatory variable. "coef" provides a coefficient plot and "influence" shows (potentially) influential observations

- **conf_lev**
  - Confidence level to use for coefficient and odds confidence intervals (.95 is the default)

- **intercept**
  - Include the intercept in the coefficient plot (TRUE or FALSE). FALSE is the default

- **incl**
  - Which variables to include in a coefficient plot

- **excl**
  - Which variables to exclude in a coefficient plot

- **nrobs**
  - Number of data points to show in scatter plots (-1 for all)

- **shiny**
  - Did the function call originate inside a shiny app

- **custom**
  - Logical (TRUE, FALSE) to indicate if ggplot object (or list of ggplot objects) should be returned. This option can be used to customize plots (e.g., add a title, change x and y labels, etc.). See examples and [https://ggplot2.tidyverse.org](https://ggplot2.tidyverse.org) for options.

- **...**
  - Further arguments passed to or from other methods
plot.mnl

Details

See https://radiant-rstats.github.io/docs/model/logistic.html for an example in Radiant

See Also

logistic to generate results
plot.logistic to plot results
predict.logistic to generate predictions
plot.model.predict to plot prediction output

Examples

result <- logistic(titanic, "survived", c("pclass", "sex"), lev = "Yes")
plot(result, plots = "coef")

plot.mnl

Plot method for the mnl function

Description

Plot method for the mnl function

Usage

## S3 method for class 'mnl'
plot(
  x,
  plots = "coef",
  conf_lev = 0.95,
  intercept = FALSE,
  nrobs = -1,
  shiny = FALSE,
  custom = FALSE,
  ...
)

Arguments

x Return value from mnl
plots Plots to produce for the specified MNL model. Use "" to avoid showing any plots (default). "dist" shows histograms (or frequency bar plots) of all variables in the model. "scatter" shows scatter plots (or box plots for factors) for the response variable with each explanatory variable. "coef" provides a coefficient plot
conf_lev Confidence level to use for coefficient and relative risk ratios (RRRs) intervals (.95 is the default)
intercept  Include the intercept in the coefficient plot (TRUE or FALSE). FALSE is the default
nrobs    Number of data points to show in scatter plots (-1 for all)
shiny    Did the function call originate inside a shiny app
custom   Logical (TRUE, FALSE) to indicate if ggplot object (or list of ggplot objects) should be returned. This option can be used to customize plots (e.g., add a title, change x and y labels, etc.). See examples and https://ggplot2.tidyverse.org for options.
...

further arguments passed to or from other methods

Details

See https://radiant-rstats.github.io/docs/model/mnl.html for an example in Radiant

See Also

mnl to generate results
predict.mnl to generate predictions
plot.model.predict to plot prediction output

Examples

result <- mnl(
  ketchup,
  rvar = "choice",
  evar = c("price.heinz28", "price.heinz32", "price.heinz41", "price.hunts32"),
  lev = "heinz28"
)
plot(result, plots = "coef")

plot.mnl.predict  Plot method for mnl.predict function

Description

Plot method for mnl.predict function

Usage

## S3 method for class 'mnl.predict'
plot(x, xvar = "", facet_row = ".", facet_col = ".", color = ".class", ...)
Arguments

- **x**: Return value from predict function `predict.mnl`
- **xvar**: Variable to display along the X-axis of the plot
- **facet_row**: Create vertically arranged subplots for each level of the selected factor variable
- **facet_col**: Create horizontally arranged subplots for each level of the selected factor variable
- **color**: Adds color to a scatter plot to generate a heat map. For a line plot one line is created for each group and each is assigned a different color
- ... further arguments passed to or from other methods

See Also

- `predict.mnl` to generate predictions

Examples

```r
result <- mnl(
  ketchup,
  rvar = "choice",
  evar = c("price.heinz28", "price.heinz32", "price.heinz41", "price.hunts32"),
  lev = "heinz28"
)
pred <- predict(result, pred_cmd = "price.heinz28 = seq(3, 5, 0.1)")
plot(pred, xvar = "price.heinz28")
```

---

**Description**

Plot method for `model.predict` functions

**Usage**

```r
## S3 method for class 'model.predict'
plot(
  x,
  xvar = "",
  facet_row = ". ",
  facet_col = ". ",
  color = "none",
  conf_lev = 0.95,
  ...
)
```
Arguments

x  Return value from predict functions (e.g., predict.regress)
xvar Variable to display along the X-axis of the plot
facet_row Create vertically arranged subplots for each level of the selected factor variable
facet_col Create horizontally arranged subplots for each level of the selected factor variable
color Adds color to a scatter plot to generate a heat map. For a line plot one line is created for each group and each is assigned a different color
conf_lev Confidence level to use for prediction intervals (.95 is the default)
... further arguments passed to or from other methods

See Also

predict.regress to generate predictions
predict.logistic to generate predictions

Examples

regress(diamonds, "price", c("carat", "clarity")) %>%
predict(pred_cmd = "carat = 1:10") %>%
plot(xvar = "carat")
logistic(titanic, "survived", c("pclass", "sex", "age"), lev = "Yes") %>%
predict(pred_cmd = c("pclass = levels(pclass)", "sex = levels(sex)", "age = 0:100")) %>%
plot(xvar = "age"). color = "sex", facet_col = "pclass")

Plot method for the nb function

Description

Plot method for the nb function

Usage

## S3 method for class 'nb'
plot(x, plots = "correlations", lev = "All levels", nrobs = 1000, ...)

Arguments

x  Return value from nb
plots Plots to produce for the specified model. Use "" to avoid showing any plots. Use "vimp" for variable importance or "correlations" to examine conditional independence
lev The level(s) in the response variable used as the basis for plots (defaults to "All levels")
nrobs Number of data points to show in scatter plots (-1 for all)
... further arguments passed to or from other methods

Details
See https://radiant-rstats.github.io/docs/model/nb.html for an example in Radiant

See Also
nb to generate results
summary.nb to summarize results
predict.nb for prediction

Examples
result <- nb(titanic, "survived", c("pclass", "sex"))
plot(result)
result <- nb(titanic, "pclass", c("sex", "age"))
plot(result)

Description
Plot method for nb.predict function

Usage
## S3 method for class 'nb.predict'
plot(x, xvar = "", facet_row = ".", facet_col = ".", color = ".class", ...)

Arguments
x Return value from predict function predict.nb
xvar Variable to display along the X-axis of the plot
facet_row Create vertically arranged subplots for each level of the selected factor variable
facet_col Create horizontally arranged subplots for each level of the selected factor variable
color Adds color to a scatter plot to generate a heat map. For a line plot one line is created for each group and each is assigned a different color
... further arguments passed to or from other methods
See Also

predict.nb to generate predictions

Examples

```r
result <- nb(titanic, "survived", c("pclass", "sex", "age"))
pred <- predict(result,
    pred_cmd = c("pclass = levels(pclass)", "sex = levels(sex)", "age = seq(0, 100, 20)")
)
plot(pred, xvar = "age", facet_col = "sex", facet_row = "pclass")
pred <- predict(result, pred_data = titanic)
plot(pred, xvar = "age", facet_col = "sex")
```

plot.nn  

Plot method for the nn function

Description

Plot method for the nn function

Usage

```r
## S3 method for class 'nn'
plot(
    x,
    plots = "garson",
    size = 12,
    pad_x = 0.9,
    nrobs = -1,
    shiny = FALSE,
    custom = FALSE,
    ...
)
```

Arguments

- `x`  
  Return value from `nn`
- `plots`  
  Plots to produce for the specified Neural Network model. Use "" to avoid showing any plots (default). Options are "olden" or "garson" for importance plots, or "net" to depict the network structure
- `size`  
  Font size used
- `pad_x`  
  Padding for explanatory variable labels in the network plot. Default value is 0.9, smaller numbers (e.g., 0.5) increase the amount of padding
- `nrobs`  
  Number of data points to show in dashboard scatter plots (-1 for all)
Plot method for the regress function

Plot method for the regress function

```r
## S3 method for class 'regress'
plot(
x,
plots = "",
lines = "",
conf_lev = 0.95,
intercept = FALSE,
incl = NULL,
excl = NULL,
nrobs = -1,
shiny = FALSE,
custom = FALSE,
...
)
```

Details

See https://radiant-rstats.github.io/docs/model/nn.html for an example in Radiant

See Also

- `nn` to generate results
- `summary.nn` to summarize results
- `predict.nn` for prediction

Examples

```r
result <- nn(titanic, "survived", c("pclass", "sex"), lev = "Yes")
plot(result, plots = "net")
plot(result, plots = "olden")
```
Arguments

x  Return value from regress
plots  Regression plots to produce for the specified regression model. Enter "" to avoid showing any plots (default). "dist" to show histograms (or frequency bar plots) of all variables in the model. "correlations" for a visual representation of the correlation matrix selected variables. "scatter" to show scatter plots (or box plots for factors) for the response variable with each explanatory variable. "dashboard" for a series of six plots that can be used to evaluate model fit visually. "resid_pred" to plot the explanatory variables against the model residuals. "coef" for a coefficient plot with adjustable confidence intervals and "influence" to show (potentially) influential observations
lines  Optional lines to include in the select plot. "line" to include a line through a scatter plot. "loess" to include a polynomial regression fit line. To include both use c("line", "loess")
conf_lev  Confidence level used to estimate confidence intervals (.95 is the default)
intercept  Include the intercept in the coefficient plot (TRUE, FALSE). FALSE is the default
incl  Which variables to include in a coefficient plot
excl  Which variables to exclude in a coefficient plot
nrobs  Number of data points to show in scatter plots (-1 for all)
shiny  Did the function call originate inside a shiny app
custom  Logical (TRUE, FALSE) to indicate if ggplot object (or list of ggplot objects) should be returned. This option can be used to customize plots (e.g., add a title, change x and y labels, etc.). See examples and https://ggplot2.tidyverse.org for options.
...  further arguments passed to or from other methods

Details

See https://radiant-rstats.github.io/docs/model/regress.html for an example in Radiant

See Also
regress to generate the results
summary.regress to summarize results
predict.regress to generate predictions

Examples
result <- regress(diamonds, "price", c("carat", "clarity"))
plot(result, plots = "coef", conf_lev = .99, intercept = TRUE)
## Not run:
plot(result, plots = "dist")
plot(result, plots = "scatter", lines = c("line", "loess"))
plot(result, plots = "resid_pred", lines = "line")
plot(result, plots = "dashboard", lines = c("line", "loess"))

## End(Not run)

---

**plot.repeater**  
*Plot repeated simulation*

**Description**

Plot repeated simulation

**Usage**

```r
## S3 method for class 'repeater'
plot(x, bins = 20, shiny = FALSE, custom = FALSE, ...)
```

**Arguments**

- `x` Return value from `repeater`
- `bins` Number of bins used for histograms (1 - 50)
- `shiny` Did the function call originate inside a shiny app
- `custom` Logical (TRUE, FALSE) to indicate if ggplot object (or list of ggplot objects) should be returned. This option can be used to customize plots (e.g., add a title, change x and y labels, etc.). See examples and [https://ggplot2.tidyverse.org](https://ggplot2.tidyverse.org) for options.
- `...` further arguments passed to or from other methods

**See Also**

- `repeater` to run a repeated simulation
- `summary.repeater` to summarize results from repeated simulation

---

**plot.rforest**  
*Plot method for the rforest function*

**Description**

Plot method for the rforest function
plot.rforest

Usage

## S3 method for class 'rforest'
plot(
  x,
  plots = "",
  nrobs = Inf,
  qtiles = FALSE,
  shiny = FALSE,
  custom = FALSE,
  ...
)

Arguments

x Return value from \texttt{rforest}
plots Plots to produce for the specified Random Forest model. Use "" to avoid showing any plots (default). Options are ...
nrobs Number of data points to show in dashboard scatter plots (-1 for all)
qtiles Show 25th and 75th percentiles in partial-dependence plots
shiny Did the function call originate inside a shiny app
custom Logical (TRUE, FALSE) to indicate if ggplot object (or list of ggplot objects) should be returned. This option can be used to customize plots (e.g., add a title, change x and y labels, etc.). See examples and \url{https://ggplot2.tidyverse.org} for options.
...

Details

See \url{https://radiant-rstats.github.io/docs/model/rforest.html} for an example in Radiant

See Also

\texttt{rforest} to generate results
\texttt{summary.rforest} to summarize results
\texttt{predict.rforest} for prediction

Examples

result <- rforest(titanic, "survived", c("pclass", "sex"), lev = "Yes")
plot.rforest.predict  Plot method for rforest.predict function

Description

Plot method for rforest.predict function

Usage

## S3 method for class 'rforest.predict'
plot(x, xvar = "", facet_row = ".", facet_col = ".", color = "none", ...)

Arguments

x  Return value from predict function predict.rforest

xvar  Variable to display along the X-axis of the plot

facet_row  Create vertically arranged subplots for each level of the selected factor variable

facet_col  Create horizontally arranged subplots for each level of the selected factor variable

color  Adds color to a scatter plot to generate a heat map. For a line plot one line is created for each group and each is assigned a different color

...  further arguments passed to or from other methods

See Also

predict.mnl to generate predictions

Examples

result <- mnl(
  ketchup,
  rvar = "choice",
  evar = c("price.heinz28", "price.heinz32", "price.heinz41", "price.hunts32"),
  lev = "heinz28"
)
pred <- predict(result, pred_cmd = "price.heinz28 = seq(3, 5, 0.1)"
plot(pred, xvar = "price.heinz28")
plot.simulater  

Plot method for the simulater function

Description

Plot method for the simulater function

Usage

## S3 method for class 'simulater'
plot(x, bins = 20, shiny = FALSE, custom = FALSE, ...)

Arguments

x  
Return value from simulater

bins  
Number of bins used for histograms (1 - 50)

shiny  
Did the function call originate inside a shiny app

custom  
Logical (TRUE, FALSE) to indicate if ggplot object (or list of ggplot objects) should be returned. This option can be used to customize plots (e.g., add a title, change x and y labels, etc.). See examples and https://ggplot2.tidyverse.org for options.

...  
further arguments passed to or from other methods

Details

See https://radiant-rstats.github.io/docs/model/simulater for an example in Radiant

See Also

simulater to generate the result

summary.simulater to summarize results

Examples

```r
simdat <- simulater(
  const = "cost 3",
  norm = "demand 2000 1000",
  discrete = "price 5 8 .3 .7",
  form = "profit = demand * (price - cost)",
  seed = 1234
)
plot(simdat, bins = 25)
```
predict.crtree

Predict method for the crtree function

Description

Predict method for the crtree function

Usage

```r
## S3 method for class 'crtree'
predict(
  object,
pred_data = NULL,
pred_cmd = "",
conf_lev = 0.95,
se = FALSE,
dec = 3,
envir = parent.frame(),
...
)
```

Arguments

- `object`: Return value from `crtree`
- `pred_data`: Provide the dataframe to generate predictions (e.g., titanic). The dataset must contain all columns used in the estimation
- `pred_cmd`: Generate predictions using a command. For example, `pclass = levels(pclass)` would produce predictions for the different levels of factor `pclass`. To add another variable, create a vector of prediction strings, (e.g., `c('pclass = levels(pclass)', 'age = seq(0,100,20)')`)
- `conf_lev`: Confidence level used to estimate confidence intervals (.95 is the default)
- `se`: Logical that indicates if prediction standard errors should be calculated (default = FALSE)
- `dec`: Number of decimals to show
- `envir`: Environment to extract data from
- `...`: Further arguments passed to or from other methods

Details

See [https://radiant-rstats.github.io/docs/model/crtree.html](https://radiant-rstats.github.io/docs/model/crtree.html) for an example in Radiant

See Also

- `crtree` to generate the result
- `summary.crtree` to summarize results
**Examples**

```r
result <- crtree(titanic, "survived", c("pclass", "sex"), lev = "Yes")
predict(result, pred_cmd = "pclass = levels(pclass)"
result <- crtree(titanic, "survived", "pclass", lev = "Yes")
predict(result, pred_data = titanic) %>% head()
```

**predict.gbt**  
*Predict method for the gbt function*

**Description**

Predict method for the gbt function

**Usage**

```r
## S3 method for class 'gbt'
predict(
  object, 
  pred_data = NULL, 
  pred_cmd = "", 
  dec = 3, 
  envir = parent.frame(), 
  ...
)
```

**Arguments**

- **object**  
  Return value from gbt

- **pred_data**  
  Provide the dataframe to generate predictions (e.g., diamonds). The dataset must contain all columns used in the estimation

- **pred_cmd**  
  Generate predictions using a command. For example, `pclass = levels(pclass)` would produce predictions for the different levels of factor `pclass`. To add another variable, create a vector of prediction strings, (e.g., c(`pclass = levels(pclass)`, `age = seq(0,100,20)`)

- **dec**  
  Number of decimals to show

- **envir**  
  Environment to extract data from

- **...**  
  further arguments passed to or from other methods

**Details**

See [https://radiant-rstats.github.io/docs/model/gbt.html](https://radiant-rstats.github.io/docs/model/gbt.html) for an example in Radiant

**See Also**

- gbt to generate the result

- summary.gbt to summarize results
**predict.logistic**

**Predict method for the logistic function**

### Description

Predict method for the logistic function

### Usage

```r
## S3 method for class 'logistic'
predict(
  object,
  pred_data = NULL,
  pred_cmd = "",
  conf_lev = 0.95,
  se = TRUE,
  interval = "confidence",
  dec = 3,
  envir = parent.frame(),
  ...
)
```

### Arguments

- **object**: Return value from `logistic`
- **pred_data**: Provide the dataframe to generate predictions (e.g., `titanic`). The dataset must contain all columns used in the estimation.
- **pred_cmd**: Generate predictions using a command. For example, ‘`pclass = levels(pclass)`’ would produce predictions for the different levels of factor ‘`pclass`’. To add another variable, create a vector of prediction strings, (e.g., c(’`pclass = levels(pclass)`’, ’`age = seq(0,100,20)`’))
- **conf_lev**: Confidence level used to estimate confidence intervals (.95 is the default)
- **se**: Logical that indicates if prediction standard errors should be calculated (default = FALSE)
- **interval**: Type of interval calculation ("confidence" or "none"). Set to "none" if se is FALSE
- **dec**: Number of decimals to show
- **envir**: Environment to extract data from
- **...**: further arguments passed to or from other methods
predict.mnl

Predict method for the mnl function

Description

Predict method for the mnl function

Usage

## S3 method for class 'mnl'
predict(
  object,
  pred_data = NULL,
  pred_cmd = "",
  pred_names = "",
  dec = 3,
  envir = parent.frame(),
  ...
)

Arguments

object Return value from mnl
pred_data Provide the dataframe to generate predictions (e.g., ketchup). The dataset must contain all columns used in the estimation
predict.nb

**pred_cmd**  Generate predictions using a command. For example, `pclass = levels(pclass)` would produce predictions for the different levels of factor `pclass`. To add another variable, create a vector of prediction strings, (e.g., `c('pclass = levels(pclass)', 'age = seq(0,100,20)')`)

**pred_names**  Names for the predictions to be stored. If one name is provided, only the first column of predictions is stored. If empty, the levels in the response variable of the mnl model will be used

**dec**  Number of decimals to show

**envir**  Environment to extract data from

...  further arguments passed to or from other methods

**Details**

See https://radiant-rstats.github.io/docs/model/mnl.html for an example in Radiant

**See Also**

mnl to generate the result
summary.mnl to summarize results

**Examples**

```r
result <- mnl(
  ketchup,
  rvar = "choice",
  evar = c("price.heinz28", "price.heinz32", "price.heinz41", "price.hunts32"),
  lev = "heinz28"
)
predict(result, pred_cmd = "price.heinz28 = seq(3, 5, 0.1)"
predict(result, pred_data = slice(ketchup, 1:20))
```

---

**Description**

Predict method for the nb function

**Usage**

```r
## S3 method for class 'nb'
predict(
  object,
  pred_data = NULL,
  pred_cmd = "",
  pred_names = "",
```

---

---
dec = 3,
envir = parent.frame(),
...
)

Arguments

object Return value from nb
pred_data Provide the dataframe to generate predictions (e.g., titanic). The dataset must contain all columns used in the estimation
pred_cmd Generate predictions using a command. For example, `pclass = levels(pclass)` would produce predictions for the different levels of factor `pclass`. To add another variable, create a vector of prediction strings, (e.g., c(`pclass = levels(pclass)`, `age = seq(0,100,20)`) pred_names Names for the predictions to be stored. If one name is provided, only the first column of predictions is stored. If empty, the level in the response variable of the nb model will be used
dec Number of decimals to show
envir Environment to extract data from
... further arguments passed to or from other methods

Details

See [https://radiant-rstats.github.io/docs/model/nb.html](https://radiant-rstats.github.io/docs/model/nb.html) for an example in Radiant

See Also

nb to generate the result
summary.nb to summarize results

Examples

result <- nb(titanic, "survived", c("pclass", "sex", "age"))
predict(result, pred_data = titanic)
predict(result, pred_data = titanic, pred_names = c("Yes", "No"))
predict(result, pred_cmd = "pclass = levels(pclass)"
result <- nb(titanic, "pclass", c("survived", "sex", "age"))
predict(result, pred_data = titanic)
predict(result, pred_data = titanic, pred_names = c("1st", "2nd", "3rd"))
predict(result, pred_data = titanic, pred_names = "")
predict.nn

Predict method for the nn function

Description

Predict method for the nn function

Usage

```r
# S3 method for class 'nn'
predict(
  object,
  pred_data = NULL,
  pred_cmd = "",
  dec = 3,
  envir = parent.frame(),
  ...
)
```

Arguments

- `object` Return value from `nn`
- `pred_data` Provide the dataframe to generate predictions (e.g., diamonds). The dataset must contain all columns used in the estimation
- `pred_cmd` Generate predictions using a command. For example, `pclass = levels(pclass)` would produce predictions for the different levels of factor `pclass`. To add another variable, create a vector of prediction strings, (e.g., `c('pclass = levels(pclass)', 'age = seq(0,100,20)')`)
- `dec` Number of decimals to show
- `envir` Environment to extract data from
- `...` further arguments passed to or from other methods

Details

See [https://radiant-rstats.github.io/docs/model/nn.html](https://radiant-rstats.github.io/docs/model/nn.html) for an example in Radiant

See Also

- `nn` to generate the result
- `summary.nn` to summarize results
Examples

```r
result <- nn(titanic, "survived", c("pclass", "sex"), lev = "Yes")
predict(result, pred_cmd = "pclass = levels(pclass)")
result <- nn(diamonds, "price", "carat:color", type = "regression")
predict(result, pred_cmd = "carat = 1:3")
predict(result, pred_data = diamonds) %>% head()
```

---

**predict.regress**

*Predict method for the regress function*

**Description**

Predict method for the regress function

**Usage**

```r
## S3 method for class 'regress'
predict(
  object,
  pred_data = NULL,
  pred_cmd = "",
  conf_lev = 0.95,
  se = TRUE,
  interval = "confidence",
  dec = 3,
  envir = parent.frame(),
  ...
)
```

**Arguments**

- **object**: Return value from `regress`
- **pred_data**: Provide the dataframe to generate predictions (e.g., diamonds). The dataset must contain all columns used in the estimation
- **pred_cmd**: Command used to generate data for prediction
- **conf_lev**: Confidence level used to estimate confidence intervals (.95 is the default)
- **se**: Logical that indicates if prediction standard errors should be calculated (default = FALSE)
- **interval**: Type of interval calculation ("confidence" or "prediction"). Set to "none" if se is FALSE
- **dec**: Number of decimals to show
- **envir**: Environment to extract data from
- **...**: further arguments passed to or from other methods
predict.rforest

Description

Predict method for the rforest function

Usage

```r
## S3 method for class 'rforest'
predict(
  object,
  pred_data = NULL,
  pred_cmd = "",
  pred_names = "",
  OOB = NULL,
  dec = 3,
  envir = parent.frame(),
  ...
)
```

Arguments

- **object**: Return value from `rforest`
- **pred_data**: Provide the dataframe to generate predictions (e.g., diamonds). The dataset must contain all columns used in the estimation

Details

See [https://radiant-rstats.github.io/docs/model/regress.html](https://radiant-rstats.github.io/docs/model/regress.html) for an example in Radiant

See Also

- `regress` to generate the result
- `summary.regress` to summarize results
- `plot.regress` to plot results

Examples

```r
result <- regress(diamonds, "price", c("carat", "clarity"))
predict(result, pred_cmd = "carat = 1:10")
predict(result, pred_cmd = "clarity = levels(clarity)")
result <- regress(diamonds, "price", c("carat", "clarity"), int = "carat:clarity")
predict(result, pred_data = diamonds) %>% head()
```
**predict_model**

**pred_cmd**
Generate predictions using a command. For example, `pclass = levels(pclass)` would produce predictions for the different levels of factor `pclass`. To add another variable, create a vector of prediction strings, (e.g., `c('pclass = levels(pclass)', 'age = seq(0,100,20)')`)

**pred_names**
Names for the predictions to be stored. If one name is provided, only the first column of predictions is stored. If empty, the levels in the response variable of the rforest model will be used

**OOB**
Use Out-Of-Bag predictions (TRUE or FALSE). Relevant when evaluating predictions for the training sample. If missing, datasets will be compared to determine if OOB predictions should be used

**dec**
Number of decimals to show

**envir**
Environment to extract data from

... further arguments passed to or from other methods

**Details**

See [https://radiant-rstats.github.io/docs/model/rforest.html](https://radiant-rstats.github.io/docs/model/rforest.html) for an example in Radiant

**See Also**

- `rforest` to generate the result
- `summary.rforest` to summarize results

**Examples**

```r
result <- rforest(titanic, "survived", c("pclass", "sex"), lev = "Yes")
predict(result, pred_cmd = "pclass = levels(pclass)")
result <- rforest(diamonds, "price", "carat:color", type = "regression")
predict(result, pred_cmd = "carat = 1:3")
predict(result, pred_data = diamonds) %>% head()
```

---

**Description**

Predict method for model functions
Usage

predict_model(
  object,
  pfun,
  mclass,
  pred_data = NULL,
  pred_cmd = "",
  conf_lev = 0.95,
  se = FALSE,
  dec = 3,
  envir = parent.frame(),
  ...
)

Arguments

object Return value from `regress`
pfun Function to use for prediction
mclass Model class to attach
pred_data Dataset to use for prediction
pred_cmd Command used to generate data for prediction (e.g., 'carat = 1:10')
conf_lev Confidence level used to estimate confidence intervals (.95 is the default)
se Logical that indicates if prediction standard errors should be calculated (default = FALSE)
dec Number of decimals to show
envir Environment to extract data from
... further arguments passed to or from other methods

Details

See [https://radiant-rstats.github.io/docs/model/regress.html](https://radiant-rstats.github.io/docs/model/regress.html) for an example in Radiant

print.crTree.predict  Print method for predict.crTree

Description

Print method for `predict.crTree`

Usage

```r
## S3 method for class 'crTree.predict'
print(x, ..., n = 10)
```
**Arguments**

- `x`: Return value from prediction method
- `...`: further arguments passed to or from other methods
- `n`: Number of lines of prediction results to print. Use -1 to print all lines

---

**print.gbt.predict**  
*Print method for predict.gbt*

**Description**

Print method for predict.gbt

**Usage**

```r
## S3 method for class 'gbt.predict'
print(x, ..., n = 10)
```

**Arguments**

- `x`: Return value from prediction method
- `...`: further arguments passed to or from other methods
- `n`: Number of lines of prediction results to print. Use -1 to print all lines

---

**print.logistic.predict**  
*Print method for logistic.predict*

**Description**

Print method for logistic.predict

**Usage**

```r
## S3 method for class 'logistic.predict'
print(x, ..., n = 10)
```

**Arguments**

- `x`: Return value from prediction method
- `...`: further arguments passed to or from other methods
- `n`: Number of lines of prediction results to print. Use -1 to print all lines
**Description**

Print method for `mnl.predict`

**Usage**

```r
## S3 method for class 'mnl.predict'
print(x, ..., n = 10)
```

**Arguments**

- `x`: Return value from prediction method
- `...`: further arguments passed to or from other methods
- `n`: Number of lines of prediction results to print. Use -1 to print all lines

---

**Description**

Print method for `predict.nb`

**Usage**

```r
## S3 method for class 'nb.predict'
print(x, ..., n = 10)
```

**Arguments**

- `x`: Return value from prediction method
- `...`: further arguments passed to or from other methods
- `n`: Number of lines of prediction results to print. Use -1 to print all lines
print.nn.predict  
Print method for predict.nn

Description

Print method for predict.nn

Usage

## S3 method for class 'nn.predict'
print(x, ..., n = 10)

Arguments

x  
Return value from prediction method

...  
further arguments passed to or from other methods

n  
Number of lines of prediction results to print. Use -1 to print all lines

print.regress.predict  
Print method for predict.regress

Description

Print method for predict.regress

Usage

## S3 method for class 'regress.predict'
print(x, ..., n = 10)

Arguments

x  
Return value from prediction method

...  
further arguments passed to or from other methods

n  
Number of lines of prediction results to print. Use -1 to print all lines
print.rforest.predict  \textit{Print method for predict.rforest}

\section*{Description}

Print method for predict.rforest

\section*{Usage}

\begin{verbatim}
## S3 method for class 'rforest.predict'
print(x, ..., n = 10)
\end{verbatim}

\section*{Arguments}

- \textbf{x} \hspace{1cm} Return value from prediction method
- \textbf{...} \hspace{1cm} further arguments passed to or from other methods
- \textbf{n} \hspace{1cm} Number of lines of prediction results to print. Use -1 to print all lines

print_predict_model \textit{Print method for the model prediction}

\section*{Description}

Print method for the model prediction

\section*{Usage}

\begin{verbatim}
print_predict_model(x, ..., n = 10, header = "")
\end{verbatim}

\section*{Arguments}

- \textbf{x} \hspace{1cm} Return value from prediction method
- \textbf{...} \hspace{1cm} further arguments passed to or from other methods
- \textbf{n} \hspace{1cm} Number of lines of prediction results to print. Use -1 to print all lines
- \textbf{header} \hspace{1cm} Header line
**profit**

*Calculate Profit based on cost:margin ratio*

**Description**

Calculate Profit based on cost:margin ratio

**Usage**

`profit(pred, rvar, lev, cost = 1, margin = 2)`

**Arguments**

- **pred**: Prediction or predictor
- **rvar**: Response variable
- **lev**: The level in the response variable defined as success
- **cost**: Cost per treatment (e.g., mailing costs)
- **margin**: Margin, or benefit, per 'success' (e.g., customer purchase). A cost:margin ratio of 1:2 implies the cost of False Positive are equivalent to the benefits of a True Positive

**Value**

`profit`

**Examples**

```r
profit(runif(20000), dvd$buy, "yes", cost = 1, margin = 2)
profit(ifelse(dvd$buy == "yes", 1, 0), dvd$buy, "yes", cost = 1, margin = 20)
profit(ifelse(dvd$buy == "yes", 1, 0), dvd$buy)
```

---

**radiant.model**

**Description**

Launch radiant.model in the default web browser

**Usage**

`radiant.model(state, ...)`
radiant.model_viewer

Arguments

   state  Path to state file to load
   ...   additional arguments to pass to shiny::runApp (e.g. port = 8080)

Details

   See https://radiant-rstats.github.io/docs/ for documentation and tutorials

Examples

   ## Not run:
   radiant.model()

   ## End(Not run)

radiant.model-deprecated

   Deprecated function(s) in the radiant.model package

Description

   These functions are provided for compatibility with previous versions of radiant. They will eventually be removed.

Usage

   ann(...)  

Arguments

   ...  Parameters to be passed to the updated functions

Details

   ann is now a synonym for nn
   scaledf is now a synonym for scale_df

radiant.model_viewer

   Launch radiant.model in the Rstudio viewer

Description

   Launch radiant.model in the Rstudio viewer
radiant.model_window

Usage

radiant.model_viewer(state, ...)

Arguments

state  Path to state file to load
...    additional arguments to pass to shiny::runApp (e.g. port = 8080)

Details

See https://radiant-rstats.github.io/docs/ for documentation and tutorials

Examples

## Not run:
radiant.model_viewer()

## End(Not run)

---

radiant.model_window  Launch radiant.model in an Rstudio window

Description

Launch radiant.model in an Rstudio window

Usage

radiant.model_window(state, ...)

Arguments

state  Path to state file to load
...    additional arguments to pass to shiny::runApp (e.g. port = 8080)

Details

See https://radiant-rstats.github.io/docs/ for documentation and tutorials

Examples

## Not run:
radiant.model_window()

## End(Not run)
ratings

Movie ratings

Description
Movie ratings

Usage
data(ratings)

Format
A data frame with 110 rows and 4 variables

Details
Use collaborative filtering to create recommendations based on ratings from existing users. Description provided in attr(ratings, "description")

regress
Linear regression using OLS

Description
Linear regression using OLS

Usage
regress(
  dataset, 
  rvar, 
  evar, 
  int = "", 
  check = "", 
  form, 
  data_filter = "", 
  envir = parent.frame() 
)
Arguments

dataset Dataset
rvar The response variable in the regression
evar Explanatory variables in the regression
int Interaction terms to include in the model
check Use "standardize" to see standardized coefficient estimates. Use "stepwise-backward" (or "stepwise-forward", or "stepwise-both") to apply step-wise selection of variables in estimation. Add "robust" for robust estimation of standard errors (HC1)
form Optional formula to use instead of rvar, evar, and int
data_filter Expression entered in, e.g., Data > View to filter the dataset in Radiant. The expression should be a string (e.g., "price > 10000")
envir Environment to extract data from

Details
See https://radiant-rstats.github.io/docs/model/regress.html for an example in Radiant

Value
A list of all variables used in the regress function as an object of class regress

See Also
summary.regress to summarize results
plot.regress to plot results
predict.regress to generate predictions

Examples

regress(diamonds, "price", c("carat", "clarity"), check = "standardize") %>% summary()
regress(diamonds, "price", c("carat", "clarity")) %>% str()

render.DiagrammeR Method to render DiagrammeR plots

Description
Method to render DiagrammeR plots

Usage

## S3 method for class 'DiagrammeR'
render(object, shiny = shiny::getDefaultReactiveDomain(), ...)

----------------------------------------

render.DiagrammeR

Method to render DiagrammeR plots

Description
Method to render DiagrammeR plots

Usage

## S3 method for class 'DiagrammeR'
render(object, shiny = shiny::getDefaultReactiveDomain(), ...)
repeater

Arguments

- **object**: DiagrammeR plot
- **shiny**: Check if function is called from a shiny application
- **...**: Additional arguments

---

**repeater** *Repeated simulation*

---

**Description**

Repeated simulation

**Usage**

```r
repeater(
  dataset,
  nr = 12,
  vars = "",
  grid = "",
  sum_vars = "",
  byvar = ".sim",
  fun = "sum",
  form = "",
  seed = NULL,
  name = "",
  envir = parent.frame()
)
```

**Arguments**

- **dataset**: Return value from the simulator function
- **nr**: Number times to repeat the simulation
- **vars**: Variables to use in repeated simulation
- **grid**: Character vector of expressions to use in grid search for constants
- **sum_vars**: (Numeric) variables to summaries
- **byvar**: Variable(s) to group data by before summarizing
- **fun**: Functions to use for summarizing
- **form**: A character vector with the formula to apply to the summarized data
- **seed**: Seed for the repeated simulation
- **name**: Deprecated argument
- **envir**: Environment to extract data from
See Also

`summary.repeater` to summarize results from repeated simulation
`plot.repeater` to plot results from repeated simulation

Examples

```r
simdat <- simulater(
  const = c("var_cost 5","fixed_cost 1000"),
  norm = "E 0 100;",
  discrete = "price 6 8 .3 .7;",
  form = c(
    "demand = 1000 - 50*price + E",
    "profit = demand*(price-var_cost) - fixed_cost",
    "profit_small = profit < 100"
  ),
  seed = 1234
)

repdat <- repeater(
  simdat,
  nr = 12,
  vars = c("E","price"),
  sum_vars = "profit",
  byvar = ".sim",
  form = "profit_365 = profit_sum < 36500",
  seed = 1234,
)

head(repdat)
summary(repdat)
plot(repdat)
```

---

### Description

Random Forest using Ranger

### Usage

```r
rforest(
  dataset,
  rvar,
  evar,
  type = "classification",
  lev = "",
)```

mtry = NULL,
num.trees = 100,
min.node.size = 1,
sample.fraction = 1,
replace = NULL,
num.threads = 12,
wts = "None",
seed = NA,
data_filter = "",
envir = parent.frame(),
...
)

Arguments

- **dataset**: Dataset
- **rvar**: The response variable in the model
- **evar**: Explanatory variables in the model
- **type**: Model type (i.e., "classification" or "regression")
- **lev**: Level to use as the first column in prediction output
- **mtry**: Number of variables to possibly split at in each node. Default is the (rounded down) square root of the number variables
- **num.trees**: Number of trees to create
- **min.node.size**: Minimal node size
- **sample.fraction**: Fraction of observations to sample. Default is 1 for sampling with replacement and 0.632 for sampling without replacement
- **replace**: Sample with (TRUE) or without (FALSE) replacement. If replace is NULL it will be reset to TRUE if the sample.fraction is equal to 1 and will be set to FALSE otherwise
- **num.threads**: Number of parallel threads to use. Defaults to 12 if available
- **wts**: Case weights to use in estimation
- **seed**: Random seed to use as the starting point
- **data_filter**: Expression entered in, e.g., Data > View to filter the dataset in Radiant. The expression should be a string (e.g., "price > 10000")
- **envir**: Environment to extract data from
- ... Further arguments to pass to ranger

Details

See [https://radiant-rstats.github.io/docs/model/rforest.html](https://radiant-rstats.github.io/docs/model/rforest.html) for an example in Radiant

Value

A list with all variables defined in rforest as an object of class rforest
See Also

- `summary.rforest` to summarize results
- `plot.rforest` to plot results
- `predict.rforest` for prediction

Examples

```r
rforest(titanic, "survived", c("pclass", "sex"), lev = "Yes") %>% summary()
rforest(titanic, "survived", c("pclass", "sex")) %>% str()
rforest(titanic, "survived", c("pclass", "sex"), max.depth = 1)
rforest(diamonds, "price", c("carat", "clarity"), type = "regression") %>% summary()
```

---

### rig

#### Relative Information Gain (RIG)

**Description**

Relative Information Gain (RIG)

**Usage**

```r
rig(pred, rvar, lev, crv = 1e-07, na.rm = TRUE)
```

**Arguments**

- `pred`: Prediction or predictor
- `rvar`: Response variable
- `lev`: The level in the response variable defined as success
- `crv`: Correction value to avoid log(0)
- `na.rm`: Logical that indicates if missing values should be removed (TRUE) or not (FALSE)

**Details**

See [https://radiant-rstats.github.io/docs/model/evalbin.html](https://radiant-rstats.github.io/docs/model/evalbin.html) for an example in Radiant

**Value**

RIG statistic

**See Also**

- `evalbin` to calculate results
- `summary.evalbin` to summarize results
- `plot.evalbin` to plot results
Examples

\begin{verbatim}
rig(runif(20000), dvd$buy, "yes")
rig(ifelse(dvd$buy == "yes", 1, 0), dvd$buy, "yes")
\end{verbatim}

Description

Root Mean Squared Error

Usage

\texttt{RMSE(pred, rvar)}

Arguments

- \texttt{pred}: Prediction (vector)
- \texttt{rvar}: Response (vector)

Value

Root Mean Squared Error

Description

R-squared

Usage

\texttt{Rsq(pred, rvar)}

Arguments

- \texttt{pred}: Prediction (vector)
- \texttt{rvar}: Response (vector)

Value

R-squared
scale_df

Center or standardize variables in a data frame

Description

Center or standardize variables in a data frame

Usage

scale_df(dataset, center = TRUE, scale = TRUE, sf = 2, wts = NULL, calc = TRUE)

Arguments

dataset Data frame
center Center data (TRUE or FALSE)
scale Scale data (TRUE or FALSE)
sf Scaling factor (default is 2)
wts Weights to use (default is NULL for no weights)
calc Calculate mean and sd or use attributes attached to dat

Value

Scaled data frame

See Also

copy_attr to copy attributes from a training to a test dataset

sdw

Standard deviation of weighted sum of variables

Description

Standard deviation of weighted sum of variables

Usage

sdw(...)  

Arguments

... A matched number of weights and stocks

Value

A vector of standard deviation estimates
**sensitivity**  
*Method to evaluate sensitivity of an analysis*

**Description**

Method to evaluate sensitivity of an analysis

**Usage**

`sensitivity(object, ...)`

**Arguments**

- `object` Object of relevant class for which to evaluate sensitivity
- `...` Additional arguments

**See Also**

`sensitivity.dtree` to plot results

---

**sensitivity.dtree**  
*Evaluate sensitivity of the decision tree*

**Description**

Evaluate sensitivity of the decision tree

**Usage**

```r
# S3 method for class 'dtree'
sensitivity(
  object,
  vars = NULL,
  decs = NULL,
  envir = parent.frame(),
  shiny = FALSE,
  custom = FALSE,
  ...
)
```
Simulate data for decision analysis

Arguments

- **object**: Return value from `dtree`
- **vars**: Variables to include in the sensitivity analysis
- **decs**: Decisions to include in the sensitivity analysis
- **envir**: Environment to extract data from
- **shiny**: Did the function call originate inside a shiny app
- **custom**: Logical (TRUE, FALSE) to indicate if ggplot object (or list of ggplot objects) should be returned. This option can be used to customize plots (e.g., add a title, change x and y labels, etc.). See examples and [https://ggplot2.tidyverse.org](https://ggplot2.tidyverse.org) for options.
- **...**: Additional arguments

Details

See [https://radiant-rstats.github.io/docs/model/dtree.html](https://radiant-rstats.github.io/docs/model/dtree.html) for an example in Radiant

See Also

- `dtree` to generate the result
- `plot.dtree` to summarize results
- `summary.dtree` to summarize results

Examples

```r
dtree(movie_contract, opt = "max") %>%
sensitivity(
  vars = "legal fees 0 100000 10000",
  decs = c("Sign with Movie Company", "Sign with TV Network"),
  custom = FALSE
)
```

Description

Simulate data for decision analysis
simulater

Usage

```r
simulater(
const = "", 
lnorm = "", 
norm = "", 
unif = "", 
discrete = "", 
binom = "", 
pois = "", 
sequ = "", 
grid = "", 
data = NULL, 
form = "", 
funcs = "", 
seed = NULL, 
nexact = FALSE, 
ncorr = NULL, 
name = "", 
nr = 1000, 
dataset = NULL, 
envir = parent.frame()
)
```

Arguments

const  A character vector listing the constants to include in the analysis (e.g., `c("cost = 3", "size = 4")`)

lnorm  A character vector listing the log-normally distributed random variables to include in the analysis (e.g., "demand 2000 1000" where the first number is the log-mean and the second is the log-standard deviation)

norm  A character vector listing the normally distributed random variables to include in the analysis (e.g., "demand 2000 1000" where the first number is the mean and the second is the standard deviation)

unif  A character vector listing the uniformly distributed random variables to include in the analysis (e.g., "demand 0 1" where the first number is the minimum value and the second is the maximum value)

discrete  A character vector listing the random variables with a discrete distribution to include in the analysis (e.g., "price 5 8 .3 .7" where the first set of numbers are the values and the second set the probabilities

binom  A character vector listing the random variables with a binomial distribution to include in the analysis (e.g., "crash 100 .01") where the first number is the number of trials and the second is the probability of success)

pois  A character vector listing the random variables with a poisson distribution to include in the analysis (e.g., "demand 10") where the number is the lambda value (i.e., the average number of events or the event rate)
sequ  A character vector listing the start and end for a sequence to include in the analysis (e.g., "trend 1 100 1"). The number of `steps` is determined by the number of simulations.

grid  A character vector listing the start, end, and step for a set of sequences to include in the analysis (e.g., "trend 1 100 1"). The number of rows in the expanded will over ride the number of simulations.

data  Dataset to be used in the calculations.

form  A character vector with the formula to evaluate (e.g., "profit = demand * (price - cost")

funcs  A named list of user defined functions to apply to variables generated as part of the simulation.

seed  Optional seed used in simulation.

nexact  Logical to indicate if normally distributed random variables should be simulated to the exact specified values.

ncorr  A string of correlations used for normally distributed random variables. The number of values should be equal to one or to the number of combinations of variables simulated.

name  Deprecated argument.

rnr  Number of simulations.

dataset  Data list from previous simulation. Used by repeater function.

envir  Environment to extract data from.

Details

See https://radiant-rstats.github.io/docs/model/simulater.html for an example in Radiant.

Value

A data.frame with the simulated data.

See Also

summary.simulater to summarize results.

plot.simulater to plot results.

Examples

```r
simulater(
  const = "cost 3",
  norm = "demand 2000 1000",
  discrete = "price 5 8 3 .7",
  form = "profit = demand * (price - cost)",
  seed = 1234
) %>% str()
```
**sim_cleaner**

*Clean input command string*

**Description**

Clean input command string

**Usage**

`sim_cleaner(x)`

**Arguments**

- `x` Input string

**Value**

Cleaned string

---

**sim_cor**

*Simulate correlated normally distributed data*

**Description**

Simulate correlated normally distributed data

**Usage**

`sim_cor(n, rho, means, sds, exact = FALSE)`

**Arguments**

- `n` The number of values to simulate (i.e., the number of rows in the simulated data)
- `rho` A vector of correlations to apply to the columns of the simulated data. The number of values should be equal to one or to the number of combinations of variables to be simulated
- `means` A vector of means. The number of values should be equal to the number of variables to simulate
- `sds` A vector of standard deviations. The number of values should be equal to the number of variables to simulate
- `exact` A logical that indicates if the inputs should be interpreted as population of sample characteristics
**Value**

A data.frame with the simulated data

**Examples**

```r
sim <- sim_cor(100, .74, c(0, 10), c(1, 5), exact = TRUE)
cor(sim)
sim_summary(sim)
```

---

**sim_splitter**  
*Split input command string*

**Description**

Split input command string

**Usage**

```r
sim_splitter(x, symbol = " ")
```

**Arguments**

- `x`: Input string
- `symbol`: Symbol used to split the command string

**Value**

Split input command string

---

**sim_summary**  
*Print simulation summary*

**Description**

Print simulation summary

**Usage**

```r
sim_summary(dataset, dc = get_class(dataset), fun = "", dec = 4)
```

**Arguments**

- `dataset`: Simulated data
- `dc`: Variable classes
- `fun`: Summary function to apply
- `dec`: Number of decimals to show
See Also

simulater to run a simulation
repeater to run a repeated simulation

Examples

```r
simulater(
  const = "cost 3",
  norm = "demand 2000 1000",
  discrete = "price 5 8 .3 .7",
  form = c("profit = demand * (price - cost)", "profit5K = profit > 5000"),
  seed = 1234
) %>% sim_summary()
```

---

store.crs

Deprecated: Store method for the crs function

Description

 Deprecated: Store method for the crs function

Usage

```r
## S3 method for class 'crs'
store(dataset, object, name, ...)
```

Arguments

- **dataset**: Dataset
- **object**: Return value from `crs`
- **name**: Name to assign to the dataset
- **...**: further arguments passed to or from other methods

Details

Return recommendations See [https://radiant-rstats.github.io/docs/model/crs.html](https://radiant-rstats.github.io/docs/model/crs.html) for an example in Radiant
store.mnl.predict  

Store predicted values generated in the mnl function

Description

Store predicted values generated in the mnl function

Usage

```r
## S3 method for class 'mnl.predict'
store(dataset, object, name = NULL, ...)
```

Arguments

- `dataset`: Dataset to add predictions to
- `object`: Return value from model function
- `name`: Variable name(s) assigned to predicted values. If empty, the levels of the response variable will be used
- `...`: Additional arguments

Details

See [https://radiant-rstats.github.io/docs/model/mnl.html](https://radiant-rstats.github.io/docs/model/mnl.html) for an example in Radiant

Examples

```r
result <- mnl(
  ketchup,
  rvar = "choice",
  evar = c("price.heinz28", "price.heinz32", "price.heinz41", "price.hunts32"),
  lev = "heinz28"
)
pred <- predict(result, pred_data = ketchup)
ketchup <- store(ketchup, pred, name = c("heinz28", "heinz32", "heinz41", "hunts32"))
```

store.model

Store residuals from a model

Description

Store residuals from a model

Usage

```r
## S3 method for class 'model'
store(dataset, object, name = "residuals", ...)
```
**store.model.predict**

**Arguments**

- **dataset**: Dataset to append residuals to
- **object**: Return value from a model function
- **name**: Variable name(s) assigned to model residuals
- **...**: Additional arguments

**Details**

The store method for objects of class "model". Adds model residuals to the dataset while handling missing values and filters. See [https://radiant-rstats.github.io/docs/model/regress.html](https://radiant-rstats.github.io/docs/model/regress.html) for an example in Radiant

**Examples**

```r
regress(diamonds, rvar = "price", evar = c("carat", "cut"), data_filter = "price > 1000") %>%
store(diamonds, ., name = "resid") %>%
head()
```

---

**store.model.predict**  
*Store predicted values generated in model functions*

**Description**

Store predicted values generated in model functions

**Usage**

```r
## S3 method for class 'model.predict'
store(dataset, object, name = "prediction", ...)
```

**Arguments**

- **dataset**: Dataset to add predictions to
- **object**: Return value from model function
- **name**: Variable name(s) assigned to predicted values
- **...**: Additional arguments

**Details**

See [https://radiant-rstats.github.io/docs/model/regress.html](https://radiant-rstats.github.io/docs/model/regress.html) for an example in Radiant
Examples

regress(diamonds, rvar = "price", evar = c("carat", "cut")) %>%
predict(pred_data = diamonds) %>%
store(diamonds, ., name = c("pred", "pred_low", "pred_high")) %>%
head()

---

store.nb.predict  Store predicted values generated in the nb function

Description

Store predicted values generated in the nb function

Usage

## S3 method for class 'nb.predict'
store(dataset, object, name = NULL, ...)

Arguments

dataset  Dataset to add predictions to

object  Return value from model function

name  Variable name(s) assigned to predicted values. If empty, the levels of the
response variable will be used

...  Additional arguments

Details

See https://radiant-rstats.github.io/docs/model/nb.html for an example in Radiant

Examples

result <- nb(titanic, rvar = "survived", evar = c("pclass", "sex", "age"))
pred <- predict(result, pred_data = titanic)
titanic <- store(titanic, pred, name = c("Yes", "No"))
store.rforest.predict  Store predicted values generated in the rforest function

Description

Store predicted values generated in the rforest function

Usage

## S3 method for class 'rforest.predict'
store(dataset, object, name = NULL, ...)

Arguments

dataset  Dataset to add predictions to
object  Return value from model function
name  Variable name(s) assigned to predicted values. If empty, the levels of the response variable will be used
...
Additional arguments

Details

See https://radiant-rstats.github.io/docs/model/rforest.html for an example in Radiant

Examples

result <- rforest(
  ketchup,
  rvar = "choice",
  evar = c("price.heinz28", "price.heinz32", "price.heinz41", "price.hunts32"),
  lev = "heinz28"
)
pred <- predict(result, pred_data = ketchup)
ketchup <- store(ketchup, pred, name = c("heinz28", "heinz32", "heinz41", "hunts32"))

summary.confusion  Summary method for the confusion matrix

Description

Summary method for the confusion matrix
Usage

```r
## S3 method for class 'confusion'
summary(object, dec = 3, ...)
```

Arguments

- `object`: Return value from `confusion`
- `dec`: Number of decimals to show
- `...`: further arguments passed to or from other methods

Details

See [https://radiant-rstats.github.io/docs/model/evalbin.html](https://radiant-rstats.github.io/docs/model/evalbin.html) for an example in Radiant

See Also

- `confusion` to generate results
- `plot.confusion` to visualize result

Examples

```r
data.frame(buy = dvd$buy, pred1 = runif(20000), pred2 = ifelse(dvd$buy == "yes", 1, 0)) %>%
  confusion(c("pred1", "pred2"), "buy") %>%
summary()
```

summary.crs

**Summary method for Collaborative Filter**

Description

Summary method for Collaborative Filter

Usage

```r
## S3 method for class 'crs'
summary(object, n = 36, dec = 2, ...)
```

Arguments

- `object`: Return value from `crs`
- `n`: Number of lines of recommendations to print. Use -1 to print all lines
- `dec`: Number of decimals to show
- `...`: further arguments passed to or from other methods
summary.crtree

Details

See https://radiant-rstats.github.io/docs/model/crs.html for an example in Radiant

See Also

crs to generate the results
plot.crs to plot results if the actual ratings are available

Examples

crs(ratings, id = "Users", prod = "Movies", pred = c("M6", "M7", "M8", "M9", "M10"),
    rate = "Ratings", data_filter = "training == 1") %>% summary()

summary.crtree Summary method for the crtree function

Description

Summary method for the crtree function

Usage

## S3 method for class 'crtree'
summary(object, prn = TRUE, splits = FALSE, cptab = FALSE, modsum = FALSE, ...)

Arguments

object Return value from crtree
prn Print tree in text form
splits Print the tree splitting metrics used
cptab Print the cp table
modsum Print the model summary
... further arguments passed to or from other methods

Details

See https://radiant-rstats.github.io/docs/model/crtree.html for an example in Radiant

See Also

crtree to generate results
plot.crtree to plot results
predict.crtree for prediction
Examples

result <- crtree(titanic, "survived", c("pclass", "sex"), lev = "Yes")
summary(result)
result <- crtree(diamonds, "price", c("carat", "color"), type = "regression")
summary(result)

summary.dtree

Summary method for the dtree function

Description

Summary method for the dtree function

Usage

## S3 method for class 'dtree'
summary(object, input = TRUE, output = FALSE, dec = 2, ...)

Arguments

object Return value from simulater
input Print decision tree input
output Print decision tree output
dec Number of decimals to show
... further arguments passed to or from other methods

Details

See https://radiant-rstats.github.io/docs/model/dtree.html for an example in Radiant

See Also

dtree to generate the results
plot.dtree to plot results
sensitivity.dtree to plot results

Examples

dtree(movie_contract, opt = "max") %>% summary(input = TRUE)
dtree(movie_contract, opt = "max") %>% summary(input = FALSE, output = TRUE)
**summary.evalbin**  
*Summary method for the evalbin function*

**Description**

Summary method for the evalbin function

**Usage**

```r
## S3 method for class 'evalbin'
summary(object, prn = TRUE, dec = 3, ...)
```

**Arguments**

- `object` Return value from `evalbin`
- `prn` Print full table of measures per model and bin
- `dec` Number of decimals to show
- `...` further arguments passed to or from other methods

**Details**

See [https://radiant-rstats.github.io/docs/model/evalbin.html](https://radiant-rstats.github.io/docs/model/evalbin.html) for an example in Radiant

**See Also**

- `evalbin` to summarize results
- `plot.evalbin` to plot results

**Examples**

```r
data.frame(buy = dvd$buy, pred1 = runif(20000), pred2 = ifelse(dvd$buy == "yes", 1, 0)) %>%
evalbin(c("pred1", "pred2"), "buy") %>%
summary()
```
Summary method for the `evalreg` function

### Usage

```r
## S3 method for class 'evalreg'
summary(object, dec = 3, ...)
```

### Arguments

- `object`: Return value from `evalreg`
- `dec`: Number of decimals to show
- `...`: further arguments passed to or from other methods

### Details

See [https://radiant-rstats.github.io/docs/model/evalreg.html](https://radiant-rstats.github.io/docs/model/evalreg.html) for an example in Radiant

### See Also

- `evalreg` to summarize results
- `plot.evalreg` to plot results

### Examples

```r
data.frame(price = diamonds$price, pred1 = rnorm(3000), pred2 = diamonds$price) %>%
evalreg(pred = c("pred1", "pred2"), "price") %>%
summary()
```

Summary method for the `gbt` function

### Usage

```r
## S3 method for class 'gbt'
summary(object, prn = TRUE, ...)
```

### Description

Summary method for the gbt function
Arguments

- **object**: Return value from `gbt`.
- **prn**: Print iteration history.
- **...**: Further arguments passed to or from other methods.

Details

See [https://radiant-rstats.github.io/docs/model/gbt.html](https://radiant-rstats.github.io/docs/model/gbt.html) for an example in Radiant.

See Also

- `gbt` to generate results.
- `plot.gbt` to plot results.
- `predict.gbt` for prediction.

Examples

```r
result <- gbt(titanic, "survived", c("pclass", "sex"), early_stopping_rounds = 0) %>% str()
summary(result)
```

Description

Summary method for the logistic function.

Usage

```r
## S3 method for class 'logistic'
summary(object, sum_check = "", conf_lev = 0.95, test_var = "", dec = 3, ...)
```

Arguments

- **object**: Return value from `logistic`.
- **sum_check**: Optional output. "vif" to show multicollinearity diagnostics. "confint" to show coefficient confidence interval estimates. "odds" to show odds ratios and confidence interval estimates.
- **conf_lev**: Confidence level to use for coefficient and odds confidence intervals (.95 is the default).
- **test_var**: Variables to evaluate in model comparison (i.e., a competing models Chi-squared test).
- **dec**: Number of decimals to show.
- **...**: Further arguments passed to or from other methods.
Summary method for the mnl function

Description

Summary method for the mnl function

Usage

```r
## S3 method for class 'mnl'
summary(object, sum_check = "", conf_lev = 0.95, test_var = "", dec = 3, ...)```

Arguments

- `object`: Return value from `mnl`
- `sum_check`: Optional output. "confint" to show coefficient confidence interval estimates. "rrr" to show relative risk ratios (RRRs) and confidence interval estimates.
- `conf_lev`: Confidence level to use for coefficient and RRRs confidence intervals (.95 is the default)
- `test_var`: Variables to evaluate in model comparison (i.e., a competing models Chi-squared test)
- `dec`: Number of decimals to show
- `...`: further arguments passed to or from other methods
Details

See https://radiant-rstats.github.io/docs/model/mnl.html for an example in Radiant

See Also

mnl to generate the results
plot.mnl to plot the results
predict.mnl to generate predictions
plot.model.predict to plot prediction output

Examples

result <- mnl(
  ketchup,
  rvar = "choice",
  evar = c("price.heinz28", "price.heinz32", "price.heinz41", "price.hunts32"),
  lev = "heinz28"
)
summary(result)
Summary

See Also

*nb* to generate results

*plot.nb* to plot results

*predict.nb* for prediction

Examples

```r
result <- nb(titanic, "survived", c("pclass", "sex", "age"))
summary(result)
```

**summary.nn**

Summary method for the `nn` function

Description

Summary method for the `nn` function

Usage

```r
## S3 method for class 'nn'
summary(object, prn = TRUE, ...)
```

Arguments

- `object`: Return value from `nn`
- `prn`: Print list of weights
- `...`: further arguments passed to or from other methods

Details

See [https://radiant-rstats.github.io/docs/model/nn.html](https://radiant-rstats.github.io/docs/model/nn.html) for an example in Radiant

See Also

* `nn` to generate results

* `plot.nn` to plot results

* `predict.nn` for prediction

Examples

```r
result <- nn(titanic, "survived", "pclass", lev = "Yes")
summary(result)
```
Summary method for the regress function

**Description**

Summary method for the regress function

**Usage**

```r
## S3 method for class 'regress'
summary(object, sum_check = "", conf_lev = 0.95, test_var = "", dec = 3, ...)
```

**Arguments**

- **object**: Return value from `regress`
- **sum_check**: Optional output. "rmse" to show the root mean squared error and the standard deviation of the residuals. "sumsquares" to show the sum of squares table. "vif" to show multicollinearity diagnostics. "confint" to show coefficient confidence interval estimates.
- **conf_lev**: Confidence level used to estimate confidence intervals (.95 is the default)
- **test_var**: Variables to evaluate in model comparison (i.e., a competing models F-test)
- **dec**: Number of decimals to show
- **...**: Further arguments passed to or from other methods

**Details**

See [https://radiant-rstats.github.io/docs/model/regress.html](https://radiant-rstats.github.io/docs/model/regress.html) for an example in Radiant

**See Also**

- `regress` to generate the results
- `plot.regress` to plot results
- `predict.regress` to generate predictions

**Examples**

```r
class <- regress(diamonds, "price", c("carat", "clarity"))
summary(class, sum_check = c("rmse", "sumsquares", "vif", "confint"), test_var = "clarity")
class2 <- regress(ideal, "y", c("x1", "x2"))
summary(class2, test_var = "x2")
ideal %>% regress("y", "x1:x3") %>% summary()
```
### summary.repeater

**Summarize repeated simulation**

**Description**

Summarize repeated simulation

**Usage**

```r
## S3 method for class 'repeater'
summary(object, dec = 4, ...)
```

**Arguments**

- `object` Return value from `repeater`
- `dec` Number of decimals to show
- `...` further arguments passed to or from other methods

**See Also**

- `repeater` to run a repeated simulation
- `plot.repeater` to plot results from repeated simulation

### summary.rforest

**Summary method for the rforest function**

**Description**

Summary method for the rforest function

**Usage**

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

**Arguments**

- `object` Return value from `rforest`
- `...` further arguments passed to or from other methods

**Details**

See [https://radiant-rstats.github.io/docs/model/rforest.html](https://radiant-rstats.github.io/docs/model/rforest.html) for an example in Radiant
See Also

rforest to generate results
plot.rforest to plot results
predict.rforest for prediction

Examples

result <- rforest(titanic, "survived", "pclass", lev = "Yes")
summary(result)

## S3 method for class 'simulater'
summary(object, dec = 4, ...)

Arguments

object Return value from simulater
dec Number of decimals to show
... further arguments passed to or from other methods

Details

See https://radiant-rstats.github.io/docs/model/simulater.html for an example in Radiant

See Also

simulater to generate the results
plot.simulater to plot results

Examples

simdat <- simulater(norm = "demand 2000 1000", seed = 1234)
summary(simdat)
test_specs

Description
Add interaction terms to list of test variables if needed

Usage
test_specs(tv, int)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tv</td>
<td>List of variables to use for testing for regress or logistic</td>
</tr>
<tr>
<td>int</td>
<td>Interaction terms specified</td>
</tr>
</tbody>
</table>

Details
See https://radiant-rstats.github.io/docs/model/regress.html for an example in Radiant

Value
A vector of variables names to test

Examples
test_specs("a", "a:b")
test_specs("a", c("a:b", "b:c"))
test_specs("a", c("a:b", "b:c", "I(c^2)"))
test_specs(c("a", "b", "c"), c("a:b", "b:c", "I(c^2)"))

var_check

Description
Check if main effects for all interaction effects are included in the model

Usage
var_check(ev, cn, intv = c())
write.coeff

Arguments

ev List of explanatory variables provided to `regress` or `logistic`
cn Column names for all explanatory variables in the dataset
intv Interaction terms specified

Details

If `:` is used to select a range evar is updated. See https://radiant-rstats.github.io/docs/model/regress.html for an example in Radiant

Value

`vars` is a vector of right-hand side variables, possibly with interactions, `iv` is the list of explanatory variables, and `intv` are interaction terms

Examples

```r
var_check("a:d", c("a","b","c","d"))
var_check(c("a", "b"), c("a", "b") , "a:c")
var_check(c("a", "b"), c("a", "b") , "a:c")
var_check(c("a", "b"), c("a", "b") , c("a:c", "I(b^2)"))
```

write.coeff

Write coefficient table for linear and logistic regression

Description

Write coefficient table for linear and logistic regression

Usage

```r
write.coeff(object, file = "", sort = FALSE, intercept = TRUE)
```

Arguments

- `object` A fitted model object of class `regress` or `logistic`
- `file` A character string naming a file. "" indicates output to the console
- `sort` Sort table by variable importance
- `intercept` Include the intercept in the output (TRUE or FALSE). TRUE is the default

Details

Write coefficients and importance scores to csv or return as a data.frame
Examples

```r
regress(
  diamonds, rvar = "price", evar = c("carat", "clarity", "color", "x"),
  int = c("carat:clarity", "clarity:color", "I(x^2)"), check = "standardize"
) %>%
  write.coeff(sort = TRUE) %>%
  format_df(dec = 3)

logistic(titanic, "survived", c("pclass", "sex"), lev = "Yes") %>%
  write.coeff(intercept = FALSE, sort = TRUE) %>%
  format_df(dec = 2)
```
Index

* datasets
  catalog, 5
  direct_marketing, 16
  dvd, 18
  houseprices, 24
  ideal, 25
  ketchup, 25
  movie_contract, 30
  ratings, 71

ann (radiant.model-deprecated), 69
auc, 4, 11, 13, 14, 16

catalog, 5
confint_robust, 6
confusion, 6, 33, 34, 90
copy_attr, 78
crs, 7, 34, 85, 90, 91
crtree, 8, 11, 35, 36, 53, 91
cv.crtree, 10
cv.gbt, 11
cv.nn, 13
cv.rforest, 15
direct_marketing, 16
dtree, 17, 18, 36, 37, 80, 92
dtree_parser, 18
dvd, 18
evalbin, 5, 19, 37, 38, 76, 93
evalreg, 20, 38, 94
find_max, 21
find_min, 22

gbt, 13, 22, 39, 54, 95

houseprices, 24

ideal, 25

ketchup, 25

logistic, 26, 40, 41, 55, 56, 95, 96, 103

MAE, 11, 13, 14, 16, 27
minmax, 28
mnl, 28, 41, 42, 56, 57, 96, 97
movie_contract, 30

nb, 30, 44, 45, 58, 97, 98
nn, 14, 31, 46, 47, 59, 69, 98

onehot, 32

plot.confusion, 7, 33, 90
plot.crs, 8, 34, 91
plot.crtree, 10, 35, 91
plot.dtree, 17, 18, 36, 80, 92
plot.evalbin, 5, 20, 37, 76, 93
plot.evalreg, 21, 38, 94
plot.gbt, 24, 39, 95
plot.logistic, 27, 40, 41, 56, 96
plot.mnl, 29, 41, 97
plot.mnl.predict, 42
plot.model.predict, 27, 29, 41, 42, 43, 56, 96, 97
plot.nb, 31, 44, 98
plot.nb.predict, 45
plot.nn, 32, 46, 98
plot.regress, 47, 61, 72, 99
plot.repeat, 49, 74, 100
plot.rforest, 49, 76, 101
plot.rforest.predict, 51
plot.rpart, 35
plot.simulator, 52, 82, 101
predict.crtree, 10, 36, 53, 91
predict.gbt, 24, 39, 54, 95
predict.logistic, 27, 41, 44, 55, 96
predict.mnl, 29, 42, 43, 51, 56, 97
predict.nb, 31, 45, 46, 57, 98
predict.nn, 32, 47, 59, 98
predict.regress, 44, 48, 60, 72, 99
predict.rforest, 50, 61, 76, 101
predict_model, 62
print.crtree.predict, 63
print.gbt.predict, 64
print.logistic.predict, 64
print.mnl.predict, 65
print.nb.predict, 65
print.nn.predict, 66
print.regress.predict, 66
print.rforest.predict, 67
print_predict_model, 67
profit, 11, 13, 14, 16, 68

radiant.model, 68
radiant.model-deprecated, 69
radiant.model_viewer, 69
radiant.model_window, 70
ratings, 71
regress, 48, 60, 61, 63, 71, 99, 103
render.DiagrammeR, 72
repeater, 49, 73, 85, 100
rforest, 16, 50, 61, 62, 74, 100, 101
rig, 76
RMSE, 11, 13, 14, 16, 77
Rsq, 11, 13, 14, 16, 77

scale_df, 69, 78
sdw, 78
sensitivity, 79
sensitivity.dtree, 17, 37, 79, 79, 92
sim_cleaner, 83
sim_cor, 83
sim_splitter, 84
sim_summary, 84
simulater, 52, 80, 85, 92, 101
store.crs, 85
store.mnl.predict, 86
store.model, 86
store.model.predict, 87
store.nb.predict, 88
store.rforest.predict, 89
summary.confusion, 7, 34, 89
summary.crs, 8, 34, 90
summary.crtree, 10, 36, 53, 91
summary.dtree, 17, 18, 37, 80, 92
summary.evalbin, 5, 20, 38, 76, 93
summary.evalreg, 21, 38, 94
summary.gbt, 24, 39, 54, 94
summary.logistic, 27, 56, 95

summary.mnl, 29, 57, 96
summary.nb, 31, 45, 58, 97
summary.nn, 32, 47, 59, 98
summary.regress, 48, 61, 72, 99
summary.repeater, 49, 74, 100
summary.rforest, 50, 62, 76, 100
summary.simulater, 52, 82, 101
test_specs, 102
var_check, 102
write.coeff, 103