

# Package ‘radiant.model’

May 15, 2024

**Type** Package

**Title** Model Menu for Radiant: Business Analytics using R and Shiny

**Version** 1.6.6

**Date** 2024-5-14

**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 (>= 4.3.0), radiant.data (>= 1.6.6)

**Imports** radiant.basics (>= 1.6.6), shiny (>= 1.8.1), nnet (>= 7.3.12), NeuralNetTools (>= 1.5.1), sandwich (>= 2.3.4), car (>= 2.1.3), ggplot2 (>= 3.4.2), scales (>= 1.2.1), data.tree (>= 0.7.4), stringr (>= 1.1.0), lubridate (>= 1.7.2), tidyr (>= 0.8.2), dplyr (>= 1.1.2), tidyselect (>= 1.2.0), rlang (>= 0.4.10), magrittr (>= 1.5), DiagrammeR (>= 1.0.9), 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 (>= 1.6.0.1), pdp (>= 0.8.1), vip (>= 0.3.2), stringi, yaml

**Suggests** testthat (>= 2.0.0), pkgdown (>= 1.1.0)

**URL** <https://github.com/radiant-rstats/radiant.model/>,  
<https://radiant-rstats.github.io/radiant.model/>,  
<https://radiant-rstats.github.io/docs/>

**BugReports** <https://github.com/radiant-rstats/radiant.model/issues/>

**License** AGPL-3 | file LICENSE

**LazyData** true

**Encoding** UTF-8

**Language** en-US

**RoxygenNote** 7.3.1

**NeedsCompilation** no

**Author** Vincent Nijs [aut, cre]

**Maintainer** Vincent Nijs <radiant@rady.ucsd.edu>

**Repository** CRAN

**Date/Publication** 2024-05-15 09:10:08 UTC

## R topics documented:

.as_int . . . . .	4
.as_num . . . . .	5
auc . . . . .	5
catalog . . . . .	6
confint_robust . . . . .	7
confusion . . . . .	7
crs . . . . .	9
crtree . . . . .	10
cv.crtree . . . . .	12
cv.gbt . . . . .	13
cv.nn . . . . .	15
cv.rforest . . . . .	16
direct_marketing . . . . .	18
dtree . . . . .	19
dtree_parser . . . . .	20
dvd . . . . .	20
evalbin . . . . .	21
evalreg . . . . .	22
find_max . . . . .	23
find_min . . . . .	24
gbt . . . . .	25
houseprices . . . . .	27
ideal . . . . .	27
kaggle_uplift . . . . .	28
ketchup . . . . .	28
logistic . . . . .	29
MAE . . . . .	30
minmax . . . . .	31
mnl . . . . .	31
movie_contract . . . . .	33
nb . . . . .	33
nn . . . . .	34
onehot . . . . .	36
pdp_plot . . . . .	37
plot.confusion . . . . .	38
plot.crs . . . . .	39
plot.crtree . . . . .	39
plot.dtree . . . . .	41
plot.evalbin . . . . .	42
plot.evalreg . . . . .	43

plot.gbt . . . . .	43
plot.logistic . . . . .	45
plot.mnl . . . . .	46
plot.mnl.predict . . . . .	47
plot.model.predict . . . . .	48
plot.nb . . . . .	49
plot.nb.predict . . . . .	50
plot.nn . . . . .	51
plot.regress . . . . .	52
plot.repeater . . . . .	54
plot.rforest . . . . .	55
plot.rforest.predict . . . . .	56
plot.simulator . . . . .	57
plot.uplift . . . . .	58
predict.crtree . . . . .	59
predict.gbt . . . . .	60
predict.logistic . . . . .	61
predict.mnl . . . . .	62
predict.nb . . . . .	63
predict.nn . . . . .	65
predict.regress . . . . .	66
predict.rforest . . . . .	67
predict_model . . . . .	68
pred_plot . . . . .	69
print.crtree.predict . . . . .	70
print.gbt.predict . . . . .	71
print.logistic.predict . . . . .	71
print.mnl.predict . . . . .	72
print.nb.predict . . . . .	72
print.nn.predict . . . . .	73
print.regress.predict . . . . .	73
print.rforest.predict . . . . .	74
print_predict_model . . . . .	74
profit . . . . .	75
radiant.model . . . . .	75
radiant.model-deprecated . . . . .	76
radiant.model_viewer . . . . .	76
radiant.model_window . . . . .	77
ratings . . . . .	78
regress . . . . .	78
remove_comments . . . . .	80
render.DiagrammeR . . . . .	80
repeater . . . . .	81
rforest . . . . .	82
rig . . . . .	84
RMSE . . . . .	85
Rsq . . . . .	85
scale_df . . . . .	86

sdw . . . . .	86
sensitivity . . . . .	87
sensitivity.dtree . . . . .	87
simulater . . . . .	88
sim_cleaner . . . . .	91
sim_cor . . . . .	91
sim_splitter . . . . .	92
sim_summary . . . . .	92
store.crs . . . . .	93
store.mnl.predict . . . . .	94
store.model . . . . .	94
store.model.predict . . . . .	95
store.nb.predict . . . . .	96
store.rforest.predict . . . . .	97
summary.confusion . . . . .	97
summary.crs . . . . .	98
summary.crtree . . . . .	99
summary.dtree . . . . .	100
summary.evalbin . . . . .	101
summary.evalreg . . . . .	102
summary.gbt . . . . .	102
summary.logistic . . . . .	103
summary.mnl . . . . .	104
summary.nb . . . . .	105
summary.nn . . . . .	106
summary.regress . . . . .	107
summary.repeater . . . . .	108
summary.rforest . . . . .	108
summary.simulater . . . . .	109
summary.uplift . . . . .	110
test_specs . . . . .	111
uplift . . . . .	111
varimp . . . . .	113
varimp_plot . . . . .	114
var_check . . . . .	114
write.coeff . . . . .	115

**Index****116**

---

`.as_int`*Convenience function used in "simulater"*

---

**Description**

Convenience function used in "simulater"

**Usage**

```
.as_int(x, dataset = list())
```

**Arguments**

x                    Character vector to be converted to integer  
dataset             Data list

**Value**

An integer vector

---

.as\_num                    *Convenience function used in "simulater"*

---

**Description**

Convenience function used in "simulater"

**Usage**

```
.as_num(x, dataset = list())
```

**Arguments**

x                    Character vector to be converted to an numeric value  
dataset             Data list

**Value**

An numeric vector

---

auc                    *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> 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*

---

**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	<i>Confidence interval for robust estimators</i>
----------------	--

---

**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	<i>Confusion matrix</i>
-----------	-------------------------

---

**Description**

Confusion matrix

**Usage**

```
confusion(  
  dataset,  
  pred,  
  rvar,  
  lev = "",  
  cost = 1,  
  margin = 2,  
  scale = 1,  
  train = "All",  
  data_filter = "",  
  arr = "",
```

```

    rows = NULL,
    envir = parent.frame(),
    ...
  )

```

### Arguments

dataset	Dataset
pred	Predictions or predictors
rvar	Response variable
lev	The level in the response variable defined as success
cost	Cost for each connection (e.g., email or mailing)
margin	Margin on each customer purchase
scale	Scaling factor to apply to calculations
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")
arr	Expression to arrange (sort) the data on (e.g., "color, desc(price)")
rows	Rows to select from the specified dataset
envir	Environment to extract data from
...	further arguments passed to or from other methods

### Details

Confusion matrix and additional metrics to evaluate binary classification models. See <https://radiant-rstats.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()

```

---

crs *Collaborative Filtering*

---

**Description**

Collaborative Filtering

**Usage**

```
crs(  
  dataset,  
  id,  
  prod,  
  pred,  
  rate,  
  data_filter = "",  
  arr = "",  
  rows = NULL,  
  envir = parent.frame()  
)
```

**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")
arr	Expression to arrange (sort) the data on (e.g., "color, desc(price)")
rows	Rows to select from the specified dataset
envir	Environment to extract data from

**Details**

See <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

```
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

```
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 = "",
  arr = "",
  rows = NULL,
  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 <code>_success_</code>
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")
arr	Expression to arrange (sort) the data on (e.g., "color, desc(price)")
rows	Rows to select from the specified dataset
envir	Environment to extract data from

## 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**

<code>object</code>	Object of type "rpart" or "crtree" to use as a starting point for cross validation
<code>K</code>	Number of cross validation passes to use
<code>repeats</code>	Number of times to repeat the K cross-validation steps
<code>cp</code>	Complexity parameter used when building the (e.g., 0.0001)
<code>pcp</code>	Complexity parameter to use for pruning
<code>seed</code>	Random seed to use as the starting point
<code>trace</code>	Print progress
<code>fun</code>	Function to use for model evaluation (e.g., auc for classification or RMSE for regression)
<code>...</code>	Additional arguments to be passed to 'fun'

**Details**

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

**Value**

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

**See Also**

[crtree](#) to generate an initial model that can be passed to cv.crtree  
[Rsquared](#) 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**

```
## Not run:
result <- crtree(dvd, "buy", c("coupon", "purch", "last"))
cv.crtree(result, cp = 0.0001, pcp = seq(0, 0.01, length.out = 11))
cv.crtree(result, cp = 0.0001, pcp = c(0, 0.001, 0.002), fun = profit, cost = 1, margin = 5)
result <- crtree(diamonds, "price", c("carat", "color", "clarity"), type = "regression", cp = 0.001)
cv.crtree(result, cp = 0.001, pcp = seq(0, 0.01, length.out = 11), fun = MAE)

## End(Not run)
```

---

cv.gbt

*Cross-validation for Gradient Boosted Trees*

---

**Description**

Cross-validation for Gradient Boosted Trees

**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**

```
## 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)

## End(Not run)
```

---

cv.nn

*Cross-validation for a Neural Network*


---

**Description**

Cross-validation for a Neural Network

**Usage**

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

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

## End(Not run)
```

### 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'

**Details**

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

**Value**

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

**See Also**

[rforest](#) to generate an initial model that can be passed to `cv.rforest`  
[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**

```
## Not run:
result <- rforest(dvd, "buy", c("coupon", "purch", "last"))
cv.rforest(
  result,
  mtry = 1:3, min.node.size = seq(1, 10, 5),
  num.trees = c(100, 200), sample.fraction = 0.632
)
result <- rforest(titanic, "survived", c("pclass", "sex"), max.depth = 1)
cv.rforest(result, mtry = 1:3, min.node.size = seq(1, 10, 5))
cv.rforest(result, mtry = 1:3, num.trees = c(100, 200), fun = profit, cost = 1, margin = 5)
result <- rforest(diamonds, "price", c("carat", "color", "clarity"), type = "regression")
cv.rforest(result, mtry = 1:3, min.node.size = 1)
cv.rforest(result, mtry = 1:3, min.node.size = 1, fun = Rsq)

## End(Not run)
```

---

direct\_marketing

*Direct marketing data*

---

**Description**

Direct marketing data

**Usage**

```
data(direct_marketing)
```

**Format**

A data frame with 1,000 rows and 12 variables

**Details**

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

---

dtree *Create a decision tree*

---

## Description

Create a decision tree

## Usage

```
dtree(y1, opt = "max", base = character(0), envir = parent.frame())
```

## Arguments

y1	A yaml string or a list (e.g., from <code>yaml::yaml.load_file()</code> )
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, the calculated tree, and a data.frame with results (i.e., payoffs, probabilities, etc.)

## 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(movie_contract)$payoff
dtree(movie_contract)$prob
dtree(movie_contract)$solution_df
```

---

dtree_parser	<i>Parse yaml input for dtree to provide (more) useful error messages</i>
--------------	---

---

**Description**

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

**Usage**

```
dtree_parser(y1)
```

**Arguments**

y1                    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	<i>Data on DVD sales</i>
-----	--------------------------

---

**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")`

---

 evalbin

*Evaluate the performance of different (binary) classification models*


---

## Description

Evaluate the performance of different (binary) classification models

## Usage

```
evalbin(
  dataset,
  pred,
  rvar,
  lev = "",
  qnt = 10,
  cost = 1,
  margin = 2,
  scale = 1,
  train = "All",
  data_filter = "",
  arr = "",
  rows = NULL,
  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
scale	Scaling factor to apply to calculations
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")
arr	Expression to arrange (sort) the data on (e.g., "color, desc(price)")
rows	Rows to select from the specified dataset
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> 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**

```
evalreg(
  dataset,
  pred,
  rvar,
  train = "All",
  data_filter = "",
  arr = "",
  rows = NULL,
  envir = parent.frame()
)
```

**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")
arr	Expression to arrange (sort) the data on (e.g., "color, desc(price)")
rows	Rows to select from the specified dataset
envir	Environment to extract data from

### Details

Evaluate different regression models based on predictions. See <https://radiant-rstats.github.io/docs/model/evalreg.html> for an example in Radiant

### 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

*Find maximum value of a vector*

---

### Description

Find maximum value of a vector

### Usage

```
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**

```
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**

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

---

`gbt`*Gradient Boosted Trees using XGBoost*

---

**Description**

Gradient Boosted Trees using XGBoost

**Usage**

```
gbt(  
  dataset,  
  rvar,  
  evar,  
  type = "classification",  
  lev = "",  
  max_depth = 6,  
  learning_rate = 0.3,  
  min_split_loss = 0,  
  min_child_weight = 1,  
  subsample = 1,  
  nrounds = 100,  
  early_stopping_rounds = 10,  
  nthread = 12,  
  wts = "None",  
  seed = NA,  
  data_filter = "",  
  arr = "",  
  rows = NULL,  
  envir = parent.frame(),  
  ...  
)
```

**Arguments**

<code>dataset</code>	Dataset
<code>rvar</code>	The response variable in the model
<code>evar</code>	Explanatory variables in the model
<code>type</code>	Model type (i.e., "classification" or "regression")
<code>lev</code>	Level to use as the first column in prediction output
<code>max_depth</code>	Maximum 'depth' of tree
<code>learning_rate</code>	Learning rate (eta)
<code>min_split_loss</code>	Minimal improvement (gamma)
<code>min_child_weight</code>	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
wt	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")
arr	Expression to arrange (sort) the data on (e.g., "color, desc(price)")
rows	Rows to select from the specified dataset
envir	Environment to extract data from
...	Further arguments to pass to xgboost

### Details

See <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

```
## 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, nthread = 1
) %>% summary()
gbt(
  titanic, "survived", c("pclass", "sex"),
  early_stopping_rounds = 0, nthread = 1
) %>% str()
gbt(
  titanic, "survived", c("pclass", "sex"),
  eval_metric = paste0("error@", 0.5 / 6), nthread = 1
) %>% str()
```

```
gbt(  
  diamonds, "price", c("carat", "clarity"), type = "regression", nthread = 1  
) %>% summary()
```

---

houseprices

*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**

```
data(ideal)
```

**Format**

A data frame with 1,000 rows and 4 variables

**Details**

Description provided in attr(ideal, "description")

---

`kaggle_uplift`*Kaggle uplift*

---

**Description**

Kaggle uplift

**Usage**

```
data(kaggle_uplift)
```

**Format**

A data frame with 1,000 rows and 22 variables

**Details**

Use uplift modeling to quantify the effectiveness of an experimental treatment

---

`ketchup`*Data on ketchup choices*

---

**Description**

Data on ketchup choices

**Usage**

```
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	<i>Logistic regression</i>
----------	----------------------------

---

**Description**

Logistic regression

**Usage**

```
logistic(
  dataset,
  rvar,
  evar,
  lev = "",
  int = "",
  wts = "None",
  check = "",
  form,
  ci_type,
  data_filter = "",
  arr = "",
  rows = NULL,
  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 <code>_success_</code>
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., <code>Data &gt; View</code> to filter the dataset in Radiant. The expression should be a string (e.g., <code>"price &gt; 10000"</code> )
arr	Expression to arrange (sort) the data on (e.g., <code>"color, desc(price)"</code> )
rows	Rows to select from the specified dataset
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**

<code>pred</code>	Prediction (vector)
<code>rvar</code>	Response (vector)

**Value**

Mean Absolute Error

---

minmax	<i>Calculate min and max before standardization</i>
--------	---

---

**Description**

Calculate min and max before standardization

**Usage**

```
minmax(dataset)
```

**Arguments**

dataset          Data frame

**Value**

Data frame min and max attributes

---

mn1	<i>Multinomial logistic regression</i>
-----	--

---

**Description**

Multinomial logistic regression

**Usage**

```
mn1(  
  dataset,  
  rvar,  
  evar,  
  lev = "",  
  int = "",  
  wts = "None",  
  check = "",  
  data_filter = "",  
  arr = "",  
  rows = NULL,  
  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")
arr	Expression to arrange (sort) the data on (e.g., "color, desc(price)")
rows	Rows to select from the specified dataset
envir	Environment to extract data from

## Details

See <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

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

---

movie_contract	<i>Movie contract decision tree</i>
----------------	-------------------------------------

---

**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	<i>Naive Bayes using e1071::naiveBayes</i>
----	--

---

**Description**

Naive Bayes using e1071::naiveBayes

**Usage**

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

### Arguments

dataset	Dataset
rvar	The response variable in the logit (probit) model
evvar	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")
arr	Expression to arrange (sort) the data on (e.g., "color, desc(price)")
rows	Rows to select from the specified dataset
envir	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

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

### Description

Neural Networks using nnet

**Usage**

```
nn(
  dataset,
  rvar,
  evar,
  type = "classification",
  lev = "",
  size = 1,
  decay = 0.5,
  wts = "None",
  seed = NA,
  check = "standardize",
  form,
  data_filter = "",
  arr = "",
  rows = NULL,
  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 <code>_success_</code>
size	Number of units (nodes) in the hidden layer
decay	Parameter decay
wts	Weights to use in estimation
seed	Random seed to use as the starting point
check	Optional estimation parameters ("standardize" is the default)
form	Optional formula to use instead of rvar and evar
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")
arr	Expression to arrange (sort) the data on (e.g., "color, desc(price)")
rows	Rows to select from the specified dataset
envir	Environment to extract data from

**Details**

See <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**

```
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()
```

---

onehot

*One hot encoding of data.frames*

---

**Description**

One hot encoding of data.frames

**Usage**

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

**Arguments**

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

**Examples**

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

---

pdp\_plot *Create Partial Dependence Plots*

---

### Description

Create Partial Dependence Plots

### Usage

```
pdp_plot(  
  x,  
  plot_list = list(),  
  incl,  
  incl_int,  
  fix = TRUE,  
  hline = TRUE,  
  nr = 20,  
  minq = 0.025,  
  maxq = 0.975  
)
```

### Arguments

x	Return value from a model
plot_list	List used to store plots
incl	Which variables to include in PDP plots
incl_int	Which interactions to investigate in PDP plots
fix	Set the desired limited on yhat or have it calculated automatically. Set to FALSE to have y-axis limits set by ggplot2 for each plot
hline	Add a horizontal line at the average of the target variable. When set to FALSE no line is added. When set to a specific number, the horizontal line will be added at that value
nr	Number of values to use to generate predictions for a numeric explanatory variable
minq	Quantile to use for the minimum value for simulation of numeric variables
maxq	Quantile to use for the maximum value for simulation of numeric variables

---

plot.confusion      *Plot method for the confusion matrix*

---

### Description

Plot method for the confusion matrix

### Usage

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

### Arguments

x	Return value from <a href="#">confusion</a>
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> for an example in Radiant

### See Also

[confusion](#) to generate results  
[summary.confusion](#) to summarize results

### Examples

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

---

plot.crs                      *Plot method for the crs function*

---

### 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                      *Plot method for the crtree function*

---

### Description

Plot method for the crtree function

### Usage

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

```

    incl = NULL,
    incl_int = NULL,
    shiny = FALSE,
    custom = FALSE,
    ...
  )

```

## Arguments

x	Return value from <a href="#">crtree</a>
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
incl	Which variables to include in a coefficient plot or PDP plot
incl_int	Which interactions to investigate in PDP 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 <a href="https://ggplot2.tidyverse.org">https://ggplot2.tidyverse.org</a> for options.
...	further arguments passed to or from other methods

## Details

See <https://radiant-rstats.github.io/docs/model/crtree.html> for an example in Radiant. The standard tree plot used 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

```

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	<i>Plot method for the dtree function</i>
------------	---

---

## Description

Plot method for the dtree function

## Usage

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

## Arguments

x	Return value from <a href="#">dtree</a>
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> 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

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

### Arguments

x	Return value from <a href="#">evalbin</a>
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 <a href="https://ggplot2.tidyverse.org">https://ggplot2.tidyverse.org</a> for options.
...	further arguments passed to or from other methods

### 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 <a href="#">evalreg</a>
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

### Examples

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

---

plot.gbt                              *Plot method for the gbt function*

---

### Description

Plot method for the gbt function

**Usage**

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

**Arguments**

x	Return value from <a href="#">gbt</a>
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)
incl	Which variables to include in a coefficient plot or PDP plot
incl_int	Which interactions to investigate in PDP 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 <a href="https://ggplot2.tidyverse.org">https://ggplot2.tidyverse.org</a> 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, nthread = 1
)
plot(result)
```

---

plot.logistic	<i>Plot method for the logistic function</i>
---------------	--

---

**Description**

Plot method for the logistic function

**Usage**

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

**Arguments**

<code>x</code>	Return value from <code>logistic</code>
<code>plots</code>	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
<code>conf_lev</code>	Confidence level to use for coefficient and odds confidence intervals (.95 is the default)
<code>intercept</code>	Include the intercept in the coefficient plot (TRUE or FALSE). FALSE is the default
<code>incl</code>	Which variables to include in a coefficient plot
<code>excl</code>	Which variables to exclude in a coefficient plot
<code>incl_int</code>	Which interactions to investigate in PDP plots
<code>nrobs</code>	Number of data points to show in scatter plots (-1 for all)
<code>shiny</code>	Did the function call originate inside a shiny app
<code>custom</code>	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 <a href="https://ggplot2.tidyverse.org">https://ggplot2.tidyverse.org</a> for options.
<code>...</code>	further arguments passed to or from other methods

**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**

<code>x</code>	Return value from <code>mnl</code>
<code>plots</code>	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
<code>conf_lev</code>	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 <a href="https://ggplot2.tidyverse.org">https://ggplot2.tidyverse.org</a> 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**

```
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.model.predict      *Plot method for model.predict functions*

---

**Description**

Plot method for model.predict functions

**Usage**

```
## 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.nb

*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 <a href="#">nb</a>
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)
```

---

plot.nb.predict	<i>Plot method for nb.predict function</i>
-----------------	--

---

### 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**

```
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**

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

**Arguments**

x	Return value from <a href="#">nn</a>
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)
incl	Which variables to include in a coefficient plot or PDP plot
incl_int	Which interactions to investigate in PDP 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 <a href="https://ggplot2.tidyverse.org">https://ggplot2.tidyverse.org</a> for options.
...	further arguments passed to or from other methods

### 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

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

---

plot.regress

*Plot method for the regress function*

---

### Description

Plot method for the regress function

### Usage

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

```

    shiny = FALSE,
    custom = FALSE,
    ...
  )

```

## Arguments

x	Return value from <a href="#">regress</a>
plots	Regression plots to produce for the specified regression model. Enter "" to avoid showing any plots (default). "dist" to shows 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 or PDP plot
excl	Which variables to exclude in a coefficient plot
incl_int	Which interactions to investigate in PDP plots
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 <a href="https://ggplot2.tidyverse.org">https://ggplot2.tidyverse.org</a> 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**

```

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

```

**Arguments**

x	Return value from <a href="#">repeater</a>
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 <a href="https://ggplot2.tidyverse.org">https://ggplot2.tidyverse.org</a> 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

### Usage

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

### Arguments

x	Return value from <a href="#">rforest</a>
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)
incl	Which variables to include in PDP or Prediction plots
incl_int	Which interactions to investigate in PDP or Prediction 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 <a href="https://ggplot2.tidyverse.org">https://ggplot2.tidyverse.org</a> for options.
...	further arguments passed to or from other methods

### Details

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

### See Also

[rforest](#) to generate results  
[summary.rforest](#) to summarize results  
[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 <a href="#">simulater</a>
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 <a href="https://ggplot2.tidyverse.org">https://ggplot2.tidyverse.org</a> 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

```
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)
```

---

plot.uplift                      *Plot method for the uplift function*

---

## Description

Plot method for the uplift function

## Usage

```
## S3 method for class 'uplift'
plot(
  x,
  plots = c("inc_uplift", "uplift"),
  size = 13,
  shiny = FALSE,
  custom = FALSE,
  ...
)
```

## Arguments

x	Return value from <a href="#">evalbin</a>
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 <a href="https://ggplot2.tidyverse.org">https://ggplot2.tidyverse.org</a> for options.
...	further arguments passed to or from other methods

## 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()
```

---

predict.crtree      *Predict method for the crtree function*

---

### Description

Predict method for the crtree function

### Usage

```
## 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 <a href="#">crtree</a>
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> for an example in Radiant

### See Also

[crtree](#) to generate the result  
[summary.crtree](#) to summarize results

**Examples**

```
result <- ctree(titanic, "survived", c("pclass", "sex"), lev = "Yes")
predict(result, pred_cmd = "pclass = levels(pclass)")
result <- ctree(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**

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

**Arguments**

object	Return value from <a href="#">gbt</a>
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> for an example in Radiant

**See Also**

[gbt](#) to generate the result  
[summary.gbt](#) to summarize results

**Examples**

```

result <- gbt(
  titanic, "survived", c("pclass", "sex"),
  early_stopping_rounds = 2, nthread = 1
)
predict(result, pred_cmd = "pclass = levels(pclass)")
result <- gbt(diamonds, "price", "carat:color", type = "regression", nthread = 1)
predict(result, pred_cmd = "carat = 1:3")
predict(result, pred_data = diamonds) %>% head()

```

---

predict.logistic	<i>Predict method for the logistic function</i>
------------------	---

---

**Description**

Predict method for the logistic function

**Usage**

```

## 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 <code>logistic</code>
pred_data	Provide the dataframe to generate predictions (e.g., <code>titanic</code> ). The dataset must contain all columns used in the estimation
pred_cmd	Generate predictions using a command. For example, <code>'pclass = levels(pclass)'</code> would produce predictions for the different levels of factor <code>'pclass'</code> . To add another variable, create a vector of prediction strings, (e.g., <code>c('pclass = levels(pclass)', 'age = seq(0,100,20)')</code> )
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

### Details

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

### See Also

[logistic](#) to generate the result  
[summary.logistic](#) to summarize results  
[plot.logistic](#) to plot results  
[plot.model.predict](#) to plot prediction output

### Examples

```
result <- logistic(titanic, "survived", c("pclass", "sex"), lev = "Yes")
predict(result, pred_cmd = "pclass = levels(pclass)")
logistic(titanic, "survived", c("pclass", "sex"), lev = "Yes") %>%
  predict(pred_cmd = "sex = c('male', 'female')")
logistic(titanic, "survived", c("pclass", "sex"), lev = "Yes") %>%
  predict(pred_data = titanic)
```

---

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 <a href="#">mnl</a>
pred_data	Provide the dataframe to generate predictions (e.g., ketchup). 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 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**

```
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))
```

---

predict.nb

*Predict method for the nb function*

---

**Description**

Predict method for the nb function

**Usage**

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

**Arguments**

object	Return value from <code>nb</code>
pred_data	Provide the dataframe to generate predictions (e.g., <code>titanic</code> ). The dataset must contain all columns used in the estimation
pred_cmd	Generate predictions using a command. For example, <code>'pclass = levels(pclass)'</code> would produce predictions for the different levels of factor <code>'pclass'</code> . To add another variable, create a vector of prediction strings, (e.g., <code>c('pclass = levels(pclass)', 'age = seq(0,100,20)')</code> )
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 <code>nb</code> 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> 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	<i>Predict method for the nn function</i>
------------	---

---

## Description

Predict method for the nn function

## Usage

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

## Arguments

object	Return value from <a href="#">nn</a>
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> for an example in Radiant

## See Also

[nn](#) to generate the result

[summary.nn](#) to summarize results

**Examples**

```

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	<i>Predict method for the regress function</i>
-----------------	--

---

**Description**

Predict method for the regress function

**Usage**

```

## 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 <a href="#">regress</a>
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

## Details

See <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

```
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.rforest

*Predict method for the rforest function*

---

## Description

Predict method for the rforest function

## Usage

```
## 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 <a href="#">rforest</a>
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)')
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 set to NULL, 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> for an example in Radiant

### See Also

[rforest](#) to generate the result

[summary.rforest](#) to summarize results

### Examples

```
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()
```

---

predict\_model

*Predict method for model functions*

---

### 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 <a href="#">regress</a>
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> for an example in Radiant

---

pred\_plot

*Prediction Plots*

---

**Description**

Prediction Plots

**Usage**

```

pred_plot(
  x,
  plot_list = list(),
  incl,
  incl_int,
  fix = TRUE,
  hline = TRUE,
  nr = 20,
  minq = 0.025,
  maxq = 0.975
)

```

**Arguments**

<code>x</code>	Return value from a model
<code>plot_list</code>	List used to store plots
<code>incl</code>	Which variables to include in prediction plots
<code>incl_int</code>	Which interactions to investigate in prediction plots
<code>fix</code>	Set the desired limited on $\hat{y}$ or have it calculated automatically. Set to FALSE to have y-axis limits set by ggplot2 for each plot
<code>hline</code>	Add a horizontal line at the average of the target variable. When set to FALSE no line is added. When set to a specific number, the horizontal line will be added at that value
<code>nr</code>	Number of values to use to generate predictions for a numeric explanatory variable
<code>minq</code>	Quantile to use for the minimum value for simulation of numeric variables
<code>maxq</code>	Quantile to use for the maximum value for simulation of numeric variables

**Details**

Faster, but less robust, alternative for PDP plots. Variable values not included in the prediction are set to either the mean or the most common value (level)

---

`print.crtree.predict` *Print method for predict.crtree*

---

**Description**

Print method for predict.crtree

**Usage**

```

## 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**

```
## 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**

```
## 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

---

`print.mnl.predict`      *Print method for mnl.predict*

---

**Description**

Print method for mnl.predict

**Usage**

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

**Arguments**

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

---

`print.nb.predict`      *Print method for predict.nb*

---

**Description**

Print method for predict.nb

**Usage**

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

**Arguments**

<code>x</code>	Return value from prediction method
<code>...</code>	further arguments passed to or from other methods
<code>n</code>	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` *Print method for predict.rforest*

---

### Description

Print method for predict.rforest

### Usage

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

### Arguments

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

---

`print_predict_model` *Print method for the model prediction*

---

### Description

Print method for the model prediction

### Usage

```
print_predict_model(x, ..., n = 10, header = "")
```

### Arguments

<code>x</code>	Return value from prediction method
<code>...</code>	further arguments passed to or from other methods
<code>n</code>	Number of lines of prediction results to print. Use -1 to print all lines
<code>header</code>	Header line

---

profit	<i>Calculate Profit based on cost:margin ratio</i>
--------	--

---

**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**

```
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	<i>radiant.model</i>
---------------	----------------------

---

**Description**

Launch radiant.model in the default web browser

**Usage**

```
radiant.model(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()

## 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

**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 = "",  
  arr = "",  
  rows = NULL,  
  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")
arr	Expression to arrange (sort) the data on (e.g., "color, desc(price)")
rows	Rows to select from the specified dataset
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()
```

---

remove_comments	<i>Remove comments from formula before it is evaluated</i>
-----------------	--

---

**Description**

Remove comments from formula before it is evaluated

**Usage**

```
remove_comments(x)
```

**Arguments**

x	Input string
---	--------------

**Value**

Cleaned string

---

render.DiagrammeR	<i>Method to render DiagrammeR plots</i>
-------------------	--

---

**Description**

Method to render DiagrammeR plots

**Usage**

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

**Arguments**

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

---

repeater	<i>Repeated simulation</i>
----------	----------------------------

---

**Description**

Repeated simulation

**Usage**

```
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**

```

simdat <- simulator(
  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)

```

---

rforest

*Random Forest using Ranger*


---

**Description**

Random Forest using Ranger

**Usage**

```

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 = "",
    arr = "",
    rows = NULL,
    envir = parent.frame(),
    ...
  )

```

### Arguments

dataset	Dataset
rvar	The response variable in the model
evvar	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")
arr	Expression to arrange (sort) the data on (e.g., "color, desc(price)")
rows	Rows to select from the specified dataset
envir	Environment to extract data from
...	Further arguments to pass to ranger

### Details

See <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**

```
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**

```
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> 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**

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

---

RMSE	<i>Root Mean Squared Error</i>
------	--------------------------------

---

**Description**

Root Mean Squared Error

**Usage**

```
RMSE(pred, rvar)
```

**Arguments**

pred	Prediction (vector)
rvar	Response (vector)

**Value**

Root Mean Squared Error

---

Rsq	<i>R-squared</i>
-----	------------------

---

**Description**

R-squared

**Usage**

```
Rsq(pred, rvar)
```

**Arguments**

pred	Prediction (vector)
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	<i>Method to evaluate sensitivity of an analysis</i>
-------------	--

---

**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	<i>Evaluate sensitivity of the decision tree</i>
-------------------	--

---

**Description**

Evaluate sensitivity of the decision tree

**Usage**

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

### Arguments

object	Return value from <a href="#">dtree</a>
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 <a href="https://ggplot2.tidyverse.org">https://ggplot2.tidyverse.org</a> for options.
...	Additional arguments

### Details

See <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

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

---

simulater

*Simulate data for decision analysis*

---

### Description

Simulate data for decision analysis

**Usage**

```

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
nr	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

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

---

sim_cleaner	<i>Clean input command string</i>
-------------	-----------------------------------

---

**Description**

Clean input command string

**Usage**

```
sim_cleaner(x)
```

**Arguments**

x	Input string
---	--------------

**Value**

Cleaned string

---

sim_cor	<i>Simulate correlated normally distributed data</i>
---------	--

---

**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**

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

---

sim_splitter	<i>Split input command string</i>
--------------	-----------------------------------

---

**Description**

Split input command string

**Usage**

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

**Arguments**

x	Input string
symbol	Symbol used to split the command string

**Value**

Split input command string

---

sim_summary	<i>Print simulation summary</i>
-------------	---------------------------------

---

**Description**

Print simulation summary

**Usage**

```
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**

```
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**

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

**Arguments**

dataset	Dataset
object	Return value from <a href="#">crs</a>
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> for an example in Radiant

---

store.mnl.predict	<i>Store predicted values generated in the mnl function</i>
-------------------	---

---

### Description

Store predicted values generated in the mnl function

### Usage

```
## 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> for an example in Radiant

### Examples

```
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	<i>Store residuals from a model</i>
-------------	-------------------------------------

---

### Description

Store residuals from a model

### Usage

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

**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> for an example in Radiant

**Examples**

```
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**

```
## 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> 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	<i>Store predicted values generated in the nb function</i>
------------------	--

---

**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**

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

**Arguments**

object	Return value from <a href="#">confusion</a>
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> for an example in Radiant

**See Also**

[confusion](#) to generate results  
[plot.confusion](#) to visualize result

**Examples**

```
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**

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

**Arguments**

object	Return value from <a href="#">crs</a>
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

**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 <a href="#">crtree</a>
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 <- ctree(titanic, "survived", c("pclass", "sex"), lev = "Yes")
summary(result)
result <- ctree(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 <a href="#">simulater</a>
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	<i>Summary method for the evalbin function</i>
-----------------	--

---

## Description

Summary method for the evalbin function

## Usage

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

## Arguments

object	Return value from <a href="#">evalbin</a>
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> for an example in Radiant

## See Also

[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") %>%  
  summary()
```

---

summary.evalreg	<i>Summary method for the evalreg function</i>
-----------------	--

---

**Description**

Summary method for the evalreg function

**Usage**

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

**Arguments**

object	Return value from <a href="#">evalreg</a>
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> for an example in Radiant

**See Also**

[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") %>%  
  summary()
```

---

summary.gbt	<i>Summary method for the gbt function</i>
-------------	--

---

**Description**

Summary method for the gbt function

**Usage**

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

**Arguments**

object	Return value from <a href="#">gbt</a>
prn	Print iteration history
...	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  
[plot.gbt](#) to plot results  
[predict.gbt](#) for prediction

**Examples**

```
result <- gbt(
  titanic, "survived", c("pclass", "sex"),
  early_stopping_rounds = 0, nthread = 1
)
summary(result)
```

---

summary.logistic	<i>Summary method for the logistic function</i>
------------------	---

---

**Description**

Summary method for the logistic function

**Usage**

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

**Arguments**

object	Return value from <a href="#">logistic</a>
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

**Details**

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

**See Also**

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

**Examples**

```
result <- logistic(titanic, "survived", "pclass", lev = "Yes")
result <- logistic(titanic, "survived", "pclass", lev = "Yes")
summary(result, test_var = "pclass")
res <- logistic(titanic, "survived", c("pclass", "sex"), int = "pclass:sex", lev = "Yes")
summary(res, sum_check = c("vif", "confint", "odds"))
titanic %>%
  logistic("survived", c("pclass", "sex", "age"), lev = "Yes") %>%
  summary("vif")
```

---

summary.mnl

*Summary method for the mnl function*


---

**Description**

Summary method for the mnl function

**Usage**

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

**Arguments**

object	Return value from <code>mnl</code>
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.nb

*Summary method for the nb function*

---

## Description

Summary method for the nb function

## Usage

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

## Arguments

object	Return value from <a href="#">nb</a>
dec	Decimals
...	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  
[plot.nb](#) to plot results  
[predict.nb](#) for prediction

**Examples**

```
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**

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

**Arguments**

object	Return value from <a href="#">nb</a>
prn	Print list of weights
...	further arguments passed to or from other methods

**Details**

See <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**

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

---

summary.regress	<i>Summary method for the regress function</i>
-----------------	--

---

**Description**

Summary method for the regress function

**Usage**

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

**Arguments**

object	Return value from <a href="#">regress</a>
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> for an example in Radiant

**See Also**

[regress](#) to generate the results  
[plot.regress](#) to plot results  
[predict.regress](#) to generate predictions

**Examples**

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

---

summary.repeater	<i>Summarize repeated simulation</i>
------------------	--------------------------------------

---

### Description

Summarize repeated simulation

### Usage

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

### Arguments

object	Return value from <a href="#">repeater</a>
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	<i>Summary method for the rforest function</i>
-----------------	--

---

### Description

Summary method for the rforest function

### Usage

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

### Arguments

object	Return value from <a href="#">rforest</a>
...	further arguments passed to or from other methods

### Details

See <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)
```

---

summary.simulater      *Summary method for the simulater function*

---

**Description**

Summary method for the simulater function

**Usage**

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

**Arguments**

object	Return value from <a href="#">simulater</a>
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)
```

---

summary.uplift	<i>Summary method for the uplift function</i>
----------------	---

---

### Description

Summary method for the uplift function

### Usage

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

### Arguments

object	Return value from <a href="#">evalbin</a>
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> for an example in Radiant

### See Also

[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") %>%  
  summary()
```

---

test_specs	<i>Add interaction terms to list of test variables if needed</i>
------------	--

---

**Description**

Add interaction terms to list of test variables if needed

**Usage**

```
test_specs(tv, int)
```

**Arguments**

tv	List of variables to use for testing for regress or logistic
int	Interaction terms specified

**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)"))
```

---

uplift	<i>Evaluate uplift for different (binary) classification models</i>
--------	---

---

**Description**

Evaluate uplift for different (binary) classification models

**Usage**

```

uplift(
  dataset,
  pred,
  rvar,
  lev = "",
  tvar,
  tlev = "",
  qnt = 10,
  cost = 1,
  margin = 2,
  scale = 1,
  train = "All",
  data_filter = "",
  arr = "",
  rows = NULL,
  envir = parent.frame()
)

```

**Arguments**

dataset	Dataset
pred	Predictions or predictors
rvar	Response variable
lev	The level in the response variable defined as success
tvar	Treatment variable
tlev	The level in the treatment variable defined as the treatment
qnt	Number of bins to create
cost	Cost for each connection (e.g., email or mailing)
margin	Margin on each customer purchase
scale	Scaling factor to apply to calculations
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")
arr	Expression to arrange (sort) the data on (e.g., "color, desc(price)")
rows	Rows to select from the specified dataset
envir	Environment to extract data from

**Details**

Evaluate uplift for different (binary) classification models based on predictions. See <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()
```

---

varimp	<i>Variable importance using the vip package and permutation importance</i>
--------	---

---

**Description**

Variable importance using the vip package and permutation importance

**Usage**

```
varimp(object, rvar, lev, data = NULL, seed = 1234)
```

**Arguments**

object	Model object created by Radiant
rvar	Label to identify the response or target variable
lev	Reference class for binary classifier (rvar)
data	Data to use for prediction. Will default to the data used to estimate the model
seed	Random seed for reproducibility

---

varimp_plot	<i>Plot permutation importance</i>
-------------	------------------------------------

---

**Description**

Plot permutation importance

**Usage**

```
varimp_plot(object, rvar, lev, data = NULL, seed = 1234)
```

**Arguments**

object	Model object created by Radiant
rvar	Label to identify the response or target variable
lev	Reference class for binary classifier (rvar)
data	Data to use for prediction. Will default to the data used to estimate the model
seed	Random seed for reproducibility

---

var_check	<i>Check if main effects for all interaction effects are included in the model</i>
-----------	--

---

**Description**

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

**Usage**

```
var_check(ev, cn, intv = c())
```

**Arguments**

ev	List of explanatory variables provided to <a href="#">regress</a> or <a href="#">logistic</a>
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**

```
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**

```
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 or return as a data.frame

**Examples**

```
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, 6
  - direct\_marketing, 18
  - dvd, 20
  - houseprices, 27
  - ideal, 27
  - kaggle\_uplift, 28
  - ketchup, 28
  - movie\_contract, 33
  - ratings, 78
- .as\_int, 4
- .as\_num, 5
- ann (radiant.model-deprecated), 76
- auc, 5, 13, 14, 16, 18
- catalog, 6
- confint\_robust, 7
- confusion, 7, 38, 98
- copy\_attr, 86
- crs, 9, 39, 93, 98, 99
- crtree, 10, 13, 40, 59, 99
- cv.crtree, 12
- cv.gbt, 13
- cv.nn, 15
- cv.rforest, 16
- direct\_marketing, 18
- dtree, 19, 20, 41, 88, 100
- dtree\_parser, 20
- dvd, 20
- evalbin, 6, 21, 42, 58, 84, 101, 110
- evalreg, 22, 43, 102
- find\_max, 23
- find\_min, 24
- gbt, 14, 25, 44, 60, 103
- houseprices, 27
- ideal, 27
- kaggle\_uplift, 28
- ketchup, 28
- logistic, 29, 45, 46, 61, 62, 103, 104, 114
- MAE, 13, 14, 16, 18, 30
- minmax, 31
- mnl, 31, 46, 47, 63, 104, 105
- movie\_contract, 33
- nb, 33, 49, 50, 64, 105, 106
- nn, 16, 34, 51, 52, 65, 76, 106
- onehot, 36
- pdp\_plot, 37
- plot.confusion, 8, 38, 98
- plot.crs, 9, 39, 99
- plot.crtree, 11, 39, 99
- plot.dtree, 19, 20, 41, 88, 100
- plot.evalbin, 6, 22, 42, 84, 101, 110, 113
- plot.evalreg, 23, 43, 102
- plot.gbt, 26, 43, 103
- plot.logistic, 30, 45, 46, 62, 104
- plot.mnl, 32, 46, 105
- plot.mnl.predict, 47
- plot.model.predict, 30, 32, 46, 47, 48, 62, 104, 105
- plot.nb, 34, 49, 106
- plot.nb.predict, 50
- plot.nn, 36, 51, 106
- plot.regress, 52, 67, 79, 107
- plot.repeater, 54, 81, 108
- plot.rforest, 55, 84, 109
- plot.rforest.predict, 56
- plot.rpart, 40
- plot.simulator, 57, 90, 109
- plot.uplift, 58
- pred\_plot, 69

predict.crtree, [11](#), [40](#), [59](#), [99](#)  
predict.gbt, [26](#), [44](#), [60](#), [103](#)  
predict.logistic, [30](#), [46](#), [49](#), [61](#), [104](#)  
predict.mnl, [32](#), [47](#), [48](#), [56](#), [62](#), [105](#)  
predict.nb, [34](#), [50](#), [51](#), [63](#), [106](#)  
predict.nn, [36](#), [52](#), [65](#), [106](#)  
predict.regress, [49](#), [53](#), [66](#), [79](#), [107](#)  
predict.rforest, [55](#), [67](#), [84](#), [109](#)  
predict\_model, [68](#)  
print.crtree.predict, [70](#)  
print.gbt.predict, [71](#)  
print.logistic.predict, [71](#)  
print.mnl.predict, [72](#)  
print.nb.predict, [72](#)  
print.nn.predict, [73](#)  
print.regress.predict, [73](#)  
print.rforest.predict, [74](#)  
print\_predict\_model, [74](#)  
profit, [13](#), [14](#), [16](#), [18](#), [75](#)

radiant.model, [75](#)  
radiant.model-deprecated, [76](#)  
radiant.model\_viewer, [76](#)  
radiant.model\_window, [77](#)  
ratings, [78](#)  
regress, [53](#), [66](#), [67](#), [69](#), [78](#), [107](#), [114](#)  
remove\_comments, [80](#)  
render.DiagrammeR, [80](#)  
repeater, [54](#), [81](#), [93](#), [108](#)  
rforest, [18](#), [55](#), [67](#), [68](#), [82](#), [108](#), [109](#)  
rig, [84](#)  
RMSE, [13](#), [14](#), [16](#), [18](#), [85](#)  
Rsq, [13](#), [14](#), [16](#), [18](#), [85](#)

scale\_df, [76](#), [86](#)  
sdw, [86](#)  
sensitivity, [87](#)  
sensitivity.dtree, [19](#), [41](#), [87](#), [87](#), [100](#)  
sim\_cleaner, [91](#)  
sim\_cor, [91](#)  
sim\_splitter, [92](#)  
sim\_summary, [92](#)  
simulator, [57](#), [88](#), [93](#), [100](#), [109](#)  
store.crs, [93](#)  
store.mnl.predict, [94](#)  
store.model, [94](#)  
store.model.predict, [95](#)  
store.nb.predict, [96](#)  
store.rforest.predict, [97](#)  
summary.confusion, [8](#), [38](#), [97](#)  
summary.crs, [9](#), [39](#), [98](#)  
summary.crtree, [11](#), [40](#), [59](#), [99](#)  
summary.dtree, [19](#), [20](#), [41](#), [88](#), [100](#)  
summary.evalbin, [6](#), [22](#), [42](#), [58](#), [84](#), [101](#), [113](#)  
summary.evalreg, [23](#), [43](#), [102](#)  
summary.gbt, [26](#), [44](#), [60](#), [102](#)  
summary.logistic, [30](#), [62](#), [103](#)  
summary.mnl, [32](#), [63](#), [104](#)  
summary.nb, [34](#), [50](#), [64](#), [105](#)  
summary.nn, [36](#), [52](#), [65](#), [106](#)  
summary.regress, [53](#), [67](#), [79](#), [107](#)  
summary.repeater, [54](#), [81](#), [108](#)  
summary.rforest, [55](#), [68](#), [84](#), [108](#)  
summary.simulator, [57](#), [90](#), [109](#)  
summary.uplift, [110](#)

test\_specs, [111](#)

uplift, [111](#)

var\_check, [114](#)  
varimp, [113](#)  
varimp\_plot, [114](#)

write.coef, [115](#)