Package ‘radiant.model’

July 26, 2019

Type Package

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

Version 1.0.0

Date 2019-7-25

Description The Radiant Model menu includes interfaces for linear and logistic regression, naive Bayes, neural networks, classification and regression trees, model evaluation, collaborative filtering, decision analysis, and simulation. The application extends the functionality in radiant.data.

Depends R (>= 3.4.0), radiant.data (>= 1.0.0)

Imports radiant.basics (>= 0.9.9), shiny (>= 1.1.0), nnet (>= 7.3.12), NeuralNetTools (>= 1.5.1), sandwich (>= 2.3.4), car (>= 2.1.3), ggplot2 (>= 2.2.1), gridExtra (>= 2.0.0), data.tree (>= 0.7.4), stringr (>= 1.1.0), lubridate (>= 1.7.2), tidyr (>= 0.8.2), dplyr (>= 0.8.0), rlang (>= 0.4.0), magrittr (>= 1.5), DiagrammeR (>= 1.0.0), import (>= 1.1.0), psych (>= 1.8.4), e1071 (>= 1.6.8), rpart (>= 4.1.11), 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

RoxygenNote 6.1.1

NeedsCompilation no

Author Vincent Nijs [aut, cre]

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

Repository CRAN

Date/Publication 2019-07-26 04:40:02 UTC
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>auc</td>
<td>2</td>
</tr>
<tr>
<td>catalog</td>
<td>3</td>
</tr>
<tr>
<td>confint_robust</td>
<td>3</td>
</tr>
<tr>
<td>confusion</td>
<td>4</td>
</tr>
<tr>
<td>crs</td>
<td>5</td>
</tr>
<tr>
<td>crtree</td>
<td>6</td>
</tr>
<tr>
<td>cv.crtree</td>
<td>7</td>
</tr>
<tr>
<td>cv.nn</td>
<td>8</td>
</tr>
<tr>
<td>direct_marketing</td>
<td>10</td>
</tr>
<tr>
<td>dtree</td>
<td>10</td>
</tr>
<tr>
<td>dtree_parser</td>
<td>11</td>
</tr>
<tr>
<td>dvd</td>
<td>12</td>
</tr>
<tr>
<td>evalbin</td>
<td>12</td>
</tr>
<tr>
<td>evalreg</td>
<td>13</td>
</tr>
<tr>
<td>find_max</td>
<td>14</td>
</tr>
<tr>
<td>find_min</td>
<td>15</td>
</tr>
<tr>
<td>houseprices</td>
<td>15</td>
</tr>
<tr>
<td>ideal</td>
<td>16</td>
</tr>
<tr>
<td>logistic</td>
<td>16</td>
</tr>
<tr>
<td>MAE</td>
<td>17</td>
</tr>
<tr>
<td>minmax</td>
<td>18</td>
</tr>
<tr>
<td>movie_contract</td>
<td>18</td>
</tr>
<tr>
<td>nb</td>
<td>19</td>
</tr>
<tr>
<td>nn</td>
<td>20</td>
</tr>
<tr>
<td>plot.confusion</td>
<td>21</td>
</tr>
<tr>
<td>plot.crs</td>
<td>22</td>
</tr>
<tr>
<td>plot.crtree</td>
<td>22</td>
</tr>
<tr>
<td>plot.dtree</td>
<td>24</td>
</tr>
<tr>
<td>plot.evalbin</td>
<td>25</td>
</tr>
<tr>
<td>plot.evalreg</td>
<td>26</td>
</tr>
<tr>
<td>plot.logistic</td>
<td>26</td>
</tr>
<tr>
<td>plot.model.predict</td>
<td>28</td>
</tr>
<tr>
<td>plot.nb</td>
<td>29</td>
</tr>
<tr>
<td>plot.nb.predict</td>
<td>30</td>
</tr>
<tr>
<td>plot.nn</td>
<td>31</td>
</tr>
<tr>
<td>plot.regressor</td>
<td>32</td>
</tr>
<tr>
<td>plot.repeater</td>
<td>33</td>
</tr>
<tr>
<td>plot.simulater</td>
<td>34</td>
</tr>
<tr>
<td>predict.crtree</td>
<td>35</td>
</tr>
<tr>
<td>predict.logistic</td>
<td>36</td>
</tr>
<tr>
<td>predict.nb</td>
<td>37</td>
</tr>
<tr>
<td>predict.nn</td>
<td>38</td>
</tr>
<tr>
<td>predict.regress</td>
<td>39</td>
</tr>
<tr>
<td>predict_model</td>
<td>40</td>
</tr>
<tr>
<td>print.crtree.predict</td>
<td>41</td>
</tr>
<tr>
<td>print.logistic.predict</td>
<td>41</td>
</tr>
</tbody>
</table>
R topics documented:

print.nb.predict .......................................................... 42
print.nn.predict ........................................................... 42
print.regress.predict ..................................................... 43
print_predict_model ...................................................... 43
profit ................................................................. 44
radiant.model .......................................................... 44
radiant.model-deprecated .............................................. 45
radiant.model_viewer .................................................... 45
radiant.model_window .................................................... 46
ratings ................................................................. 47
regress ................................................................. 47
render.DiagrammeR ...................................................... 48
repeater ................................................................. 49
RMSE ................................................................. 50
Rsq ................................................................. 50
scale_df ............................................................... 51
sdw ................................................................. 52
sensitivity ............................................................ 52
sensitivity.dtree ......................................................... 53
simulator .............................................................. 54
sim_cleaner ............................................................. 56
sim_cor ............................................................... 56
sim_splitter ............................................................. 57
sim_summary ............................................................ 57
store.crs .............................................................. 58
store.model ........................................................... 59
store.model.predict .................................................... 59
store.nb.predict ......................................................... 60
summary.confusion ...................................................... 61
summary.crs ............................................................ 62
summary.crtree .......................................................... 62
summary.dtree .......................................................... 63
summary.evalbin ........................................................ 64
summary.evalreg ........................................................ 65
summary.logistic ........................................................ 66
summary.nb ............................................................. 67
summary.nn ............................................................. 68
summary.regress ........................................................ 68
summary.repeater ........................................................ 69
summary.simulater ...................................................... 70
test_specs ............................................................. 71
var_check .............................................................. 71
write.coeff ............................................................. 72
Description

Area Under the Curve (AUC)

Usage

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

Arguments

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

Details

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

Value

AUC statistic

See Also

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

Examples

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

| 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  

Confidence interval for robust estimators

**Description**

Confidence interval for robust estimators

**Usage**

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

**Arguments**

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

**Details**

Wrapper for confint with robust standard errors. See [http://stackoverflow.com/a/3820125/1974918](http://stackoverflow.com/a/3820125/1974918)
**confusion**  

**Confusion matrix**

**Description**

Confusion matrix

**Usage**

```r
confusion(dataset, pred, rvar, lev = "", cost = 1, margin = 2, 
train = "All", data_filter = "", ...) 
```

**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
- `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")
- `...`: 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**

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

Description

Collaborative Filtering

Usage

crs(dataset, id, prod, pred, rate, data_filter = "")

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")

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()
Ctree

Classification and regression trees based on the rpart package

Description

Classification and regression trees based on the rpart package

Usage

cTREE(dataset, rvar, evar, type = "", lev = "", wts = "None",
msplit = 2, minbucket = round(msplit/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 = "")

Arguments

dataset       Dataset
rvar           The response variable in the model
evar           Explanatory variables in the model
type           Model type (i.e., "classification" or "regression")
lev            The level in the response variable defined as _success_
wts            Weights to use in estimation
minsplit       The minimum number of observations that must exist in a node in order for a
               split to be attempted.
minbucket      the minimum number of observations in any terminal <leaf> node. If only one of
               msplit or minbucket is specified, the code either sets msplit to minbucket*3
               or minbucket to msplit/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 used 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")
cv.crtree

Details

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

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

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

See https://radiant-rstats.github.io/docs/model/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

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

Value

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

See Also

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

Examples

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

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>yl</code></td>
<td>A yaml string or a list (e.g., from <code>yaml::yaml.load_file()</code>)</td>
</tr>
<tr>
<td><code>opt</code></td>
<td>Find the maximum (&quot;max&quot;) or minimum (&quot;min&quot;) value for each decision node</td>
</tr>
<tr>
<td><code>base</code></td>
<td>List of variable definitions from a base tree used when calling a sub-tree</td>
</tr>
</tbody>
</table>

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

**Value**  
A list with the initial tree and the calculated tree
dtree_parser

See Also

summary.dtree to summarize results
plot.dtree to plot results
sensitivity.dtree to plot results

Examples

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

dtree_parser

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

Description

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

Usage

dtree_parser(yl)

Arguments

yl A yaml string

Details

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

Value

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

See Also

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

**Data on DVD sales**

**Description**

Data on DVD sales

**Usage**

`data(dvd)`

**Format**

A data frame with 20,000 rows and 4 variables

**Details**

Binary purchase response to coupon value. Description provided in `attr(dvd,"description")`

---

### evalbin

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

**Description**

Evaluate the performance of different (binary) classification models

**Usage**

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

**Arguments**

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

**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")

**Details**

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

Value
A list of results

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

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

find_max  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**

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

**Arguments**

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

**Details**

Find the value of `y` at the minimum value of `x`

**Value**

Value of `val` at the minimum of `var`

**Examples**

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

---

**houseprices**

*Houseprices*

**Description**

Houseprices

**Usage**

```r
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")

logistic

Logistic regression

Description

Logistic regression

Usage

logistic(dataset, rvar, evar, lev = "", int = "", wts = "None", check = "", ci_type, data_filter = "")

Arguments

dataset  Dataset
rvar    The response variable in the model
evar   Explanatory variables in the model
lev    The level in the response variable defined as _success_
int   Interaction term to include in the model
wts   Weights to use in estimation
check Use "standardize" to see standardized coefficient estimates. Use "stepwise-backward" (or "stepwise-forward", or "stepwise-both") to apply step-wise selection of variables in estimation. Add "robust" for robust estimation of standard errors (HC1)
To use the profile-likelihood (rather than Wald) for confidence intervals use "profile". For datasets with more than 5,000 rows the Wald method will be used, unless "profile" is explicitly set.

data_filter Expression entered in, e.g., Data > View to filter the dataset in Radiant. The expression should be a string (e.g., "price > 10000")

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

pred Prediction (vector)
rvar Response (vector)

Value

Mean Absolute Error
minmax

*Calculate min and max before standardization*

**Description**

Calculate min and max before standardization

**Usage**

`minmax(dataset)`

**Arguments**

- **dataset**  
  Data frame

**Value**

Data frame min and max attributes

---

movie_contract

*Movie contract decision tree*

**Description**

Movie contract decision tree

**Usage**

`data(movie_contract)`

**Format**

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

**Details**

Use decision analysis to create a decision tree for an actor facing a contract decision
**nb**  
"Naive Bayes using e1071::naiveBayes"

**Description**

Naive Bayes using e1071::naiveBayes

**Usage**

```r
nb(dataset, rvar, evar, laplace = 0, data_filter = "")
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dataset</code></td>
<td>Dataset</td>
</tr>
<tr>
<td><code>rvar</code></td>
<td>The response variable in the logit (probit) model</td>
</tr>
<tr>
<td><code>evar</code></td>
<td>Explanatory variables in the model</td>
</tr>
<tr>
<td><code>laplace</code></td>
<td>Positive double controlling Laplace smoothing. The default (0) disables Laplace smoothing.</td>
</tr>
<tr>
<td><code>data_filter</code></td>
<td>Expression entered in, e.g., Data &gt; View to filter the dataset in Radiant. The expression should be a string (e.g., &quot;price &gt; 10000&quot;)</td>
</tr>
</tbody>
</table>

**Details**

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

**Value**

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

**See Also**

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

**Examples**

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

Neural Networks

Usage

nn(dataset, rvar, evar, type = "classification", lev = "", size = 1, decay = 0.5, wts = "None", seed = NA, check = "standardize", data_filter = "")

Arguments

dataset: Dataset
rvar: The response variable in the model
evar: Explanatory variables in the model
type: Model type (i.e., "classification" or "regression")
lev: The level in the response variable defined as _success_
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)
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")

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()
```

Description

Plot method for the confusion matrix

Usage

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

Arguments

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

Details

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

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

```r
## 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

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

x
Return value from `crtree`

plots
Plots to produce for the specified rpart tree. "tree" shows a tree diagram. "prune" shows a line graph to evaluate appropriate tree pruning. "imp" shows a variable importance plot

orient
Plot orientation for tree: LR for vertical and TD for horizontal

width
Plot width in pixels for tree (default is "900px")

labs
Use factor labels in plot (TRUE) or revert to default letters used by tree (FALSE)

dec
Decimal places to round results to

shiny
Did the function call originate inside a shiny app

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

... further arguments passed to or from other methods

Details

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

See Also

crtree to generate results

summary.crtree to summarize results

predict.crtree for prediction

Examples

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

Plot method for the dtree function

Description
Plot method for the dtree function

Usage

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

Arguments

x

Return value from dtree

symbol
Monetary symbol to use ($ is the default)

dec
Decimal places to round results to

final
If TRUE plot the decision tree solution, else the initial decision tree

orient
Plot orientation: LR for vertical and TD for horizontal

width
Plot width in pixels (default is "900px")

...
further arguments passed to or from other methods

Details

See https://radiant-rstats.github.io/docs/model/dtree.html 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")
Description

Plot method for the evalbin function

Usage

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

Arguments

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

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

Plot method for the evalreg function

Usage

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

Arguments

- **x** Return value from `evalreg`
- **vars** Measures to plot, i.e., one or more of "Rsq", "RMSE", "MAE"
- **...** further arguments passed to or from other methods

Details

See [https://radiant-rstats.github.io/docs/model/evalreg.html](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()
Usage

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

Arguments

- **x**: Return value from `logistic`
- **plots**: Plots to produce for the specified GLM model. Use "" to avoid showing any plots (default). "dist" shows histograms (or frequency bar plots) of all variables in the model. "scatter" shows scatter plots (or box plots for factors) for the response variable with each explanatory variable. "coef" provides a coefficient plot
- **conf_lev**: Confidence level to use for coefficient and odds confidence intervals (.95 is the default)
- **intercept**: Include the intercept in the coefficient plot (TRUE or FALSE). FALSE is the default
- **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 [http://docs.ggplot2.org](http://docs.ggplot2.org) for options.
- **...**: Further arguments passed to or from other methods

Details

See [https://radiant-rstats.github.io/docs/model/logistic.html](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

```r
result <- logistic(titanic, "survived", c("pclass", "sex"), lev = "Yes")
plot(result, plots = "coef")""
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 method for the nb function

Description

Plot method for the nb function

Usage

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

Arguments

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

Details

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

See Also

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

Examples

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

Plot method for nb.predict function

Description

Plot method for nb.predict function

Usage

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

Arguments

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

See Also

predict.nb to generate predictions

Examples

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

Plot method for the nn function

Description

Plot method for the nn function

Usage

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

Arguments

- `x` Return value from `nn`
- `plots` Plots to produce for the specified Neural Network model. Use "" to avoid showing any plots (default). Options are "olden" or "garson" for importance plots, or "net" to depict the network structure
- `size` Font size used
- `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 http://docs.ggplot2.org 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

```r
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

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

Arguments

- `x`  
  Return value from `regress`

- `plots`  
  Regression plots to produce for the specified regression model. Enter "" to avoid showing any plots (default). "dist" to 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. "leverage" to show leverage plots for each explanatory variable

- `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

- `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 [http://docs.ggplot2.org](http://docs.ggplot2.org) for options.

Details

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

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

See Also

repeater to run a repeated simulation
summary.repeater to summarize results from repeated simulation
plot.simulater  

Plot method for the simulater function

Description

Plot method for the simulater function

Usage

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

Arguments

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

Details

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

See Also

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

Examples

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

### Usage

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

### Arguments

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

### Details

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

### See Also

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

### Examples

```r
result <- crtree(titanic, "survived", c("pclass", "sex"), lev = "Yes")
predict(result, pred_cmd = "pclass = levels(pclass)")
result <- crtree(titanic, "survived", "pclass", lev = "Yes")
predict(result, pred_data = titanic) %>% head()
```
predict.logistic  Predict method for the logistic function

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, 
...)

Arguments

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

```r
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.nb

**Predict method for the nb function**

Description

Predict method for the nb function

Usage

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

Arguments

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

Details

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

See Also

- `nb` to generate the result
- `summary.nb` to summarize results
predict.nn
Predict method for the nn function

Description
Predict method for the nn function

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

Arguments
- **object**: Return value from nn
- **pred_data**: Provide the dataframe to generate predictions (e.g., diamonds). The dataset must contain all columns used in the estimation
- **pred_cmd**: Generate predictions using a command. For example, ‘pclass = levels(pclass)’ would produce predictions for the different levels of factor ‘pclass’. To add another variable, create a vector of prediction strings, (e.g., c(‘pclass = levels(pclass)’, ‘age = seq(0,100,20)’))
- **dec**: Number of decimals to show
- **...**: 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
**predict.regress**

**Examples**

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

---

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

**Description**

Predict method for the regress function

**Usage**

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

**Arguments**

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

**Details**

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

**See Also**

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

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

predict_model

**Predict method for model functions**

**Description**

Predict method for model functions

**Usage**

```r
predict_model(object, pfun, mclass, pred_data = NULL, pred_cmd = ",", conf_lev = 0.95, se = FALSE, dec = 3, ...)
```

**Arguments**

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

**Details**

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

Print method for predict.crtree

Description

Print method for predict.crtree

Usage

## 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.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.nb.predict**  
*Print method for predict.nb*

**Description**

Print method for predict.nb

**Usage**

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

**Arguments**

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

---

**print.nn.predict**  
*Print method for predict.nn*

**Description**

Print method for predict.nn

**Usage**

```r
## 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_predict_model

Print method for the model prediction

Description
Print method for the model prediction

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

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
header Header line
profit

*Calculate Profit based on cost:margin ratio*

**Description**

Calculate Profit based on cost:margin ratio

**Usage**

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

**Arguments**

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

**Value**

`profit`

**Examples**

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

---

radian.t.model

**Description**

radian.t.model

Launch radian.t.model in the default browser

**Usage**

```r
radian.t.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)
### ratings

**Movie ratings**

**Description**

Movie ratings

**Usage**

```r
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**

```r
regress(dataset, rvar, evar, int = "", check = "", data_filter = "")
```

**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)
- `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")
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()
repeater  Repeated simulation

Description
Repeated simulation

Usage
repeater(dataset, nr = 12, vars = "", grid = "", sum_vars = "", byvar = "sim", fun = "sum", form = "", seed = NULL, name = "")

Arguments
dataset  Return value from the simulater 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

See Also
summary.repeater to summarize results from repeated simulation
plot.repeater to plot results from repeated simulation

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

repetdat <- repeater(
  dataset = simdat,
  nr = 12,
  vars = "",
  grid = "",
  sum_vars = "",
  byvar = "sim",
  fun = "sum",
  form = "",
  seed = NULL,
  name = ""
)
simdat,
nr = 12,
vars = c("E","price"),
sum_vars = "profit",
byvar = "sim",
form = "profit_365 = profit < 36500",
seed = 1234,
)

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

---

**RMSE**  
*Root Mean Squared Error*

**Description**

Root Mean Squared Error

**Usage**

RMSE(pred, rvar)

**Arguments**

- **pred**  
  Prediction (vector)
- **rvar**  
  Response (vector)

**Value**

Root Mean Squared Error

---

**Rsq**  
*R-squared*

**Description**

R-squared

**Usage**

Rsq(pred, rvar)
scale_df

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

Method to evaluate sensitivity of an analysis

Description

Method to evaluate sensitivity of an analysis

Usage

sensitivity(object, ...)

Arguments

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

See Also

sensitivity.dtrees to plot results
sensitivity.dtree  Evaluate sensitivity of the decision tree

Description
Evaluate sensitivity of the decision tree

Usage

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

Arguments

- **object**: Return value from `dtree`
- **vars**: Variables to include in the sensitivity analysis
- **decs**: Decisions to include in the sensitivity analysis
- **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 [http://docs.ggplot2.org](http://docs.ggplot2.org) for options.
- **...**: Additional arguments

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

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

Examples

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

**Usage**

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

**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
simulater

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

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

Clean input command string

Description

Clean input command string

Usage

    sim_cleaner(x)

Arguments

    x          Input string

Value

Cleaned string

sim_cor

Simulate correlated normally distributed data

Description

Simulate correlated normally distributed data

Usage

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

Arguments

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

**Value**

A data.frame with the simulated data

**Examples**

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

---

**Description**

Split input command string

**Usage**

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

**Arguments**

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

**Value**

Split input command string

---

**sim_summary**

**Print simulation summary**

**Description**

Print simulation summary

**Usage**

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

**Arguments**

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

simulater to run a simulation
repeater to run a repeated simulation

Examples

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.model

**Store residuals from a model**

**Description**

Store residuals from a model

**Usage**

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

**Examples**

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

---

### store.model.predict

**Store predicted values generated in model functions**

**Description**

Store predicted values generated in model functions

**Usage**

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

- **dataset**
  - Dataset to add predictions to

- **object**
  - Return value from model function

- **name**
  - Variable name(s) assigned to predicted values

- **...**
  - Additional arguments

Details

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

Examples

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

---

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

Description

Store predicted values generated in the nb function

Usage

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

Arguments

- **dataset**
  - Dataset to add predictions two

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

```r
result <- nb(titanic, rvar = "survived", evar = c("pclass", "sex", "age"))
pred <- predict(result, pred_data = titanic)
titanic <- store(titanic, pred, name = c("Yes", "No"))
```

### summary.confusion

**Summary method for the confusion matrix**

**Description**

Summary method for the confusion matrix

**Usage**

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

**Arguments**

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

**Details**

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

**See Also**

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

**Examples**

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

**Summary method for Collaborative Filter**

#### Description

Summary method for Collaborative Filter

#### Usage

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

#### Arguments

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

#### Details

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

#### See Also

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

#### Examples

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

---

### summary.crtree

**Summary method for the crtree function**

#### Description

Summary method for the crtree function
### summary.dtree

**Usage**

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

**Arguments**

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

**Details**

See [https://radiant-rstats.github.io/docs/model/crtree.html](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**

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

---

### summary.dtree

**Summary method for the dtree function**

**Description**

Summary method for the dtree function

**Usage**

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

- object: Return value from `evalbin`
- 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](https://radiant-rstats.github.io/docs/model/dtree.html) for an example in Radiant

See Also

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

Examples

dtree(movie_contract, opt = "max") %>% summary(input = TRUE)
dtree(movie_contract, opt = "max") %>% summary(input = FALSE, output = TRUE)

summary.evalbin

Summary method for the evalbin function

Description

Summary method for the evalbin function

Usage

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

Arguments

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

Summary method for the evalreg function

Usage

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

Arguments

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

Details

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

See Also

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

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

---

**summary.logistic**  
*Summary method for the logistic function*

**Description**

Summary method for the logistic function

**Usage**

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

**Arguments**

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

**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

```r
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.nb**  
*Summary method for the nb function*

**Description**

Summary method for the nb function

**Usage**

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

**Arguments**

- `object`: Return value from `nb`
- `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**

```r
result <- nb(titanic, "survived", c("pclass", "sex", "age"))
summary(result)
```
## S3 method for class 'nn'
summary(object, prn = TRUE, ...)

### Arguments

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

### Details

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

### See Also

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

### Examples

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

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

Arguments

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

Details

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

See Also

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

Examples

```r
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

Summarize repeated simulation

Description

Summarize repeated simulation
### Usage

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

### Arguments

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

### See Also

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

---

### summary.simulater

**Summary method for the simulater function**

### Description

Summary method for the simulater function

### Usage

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

### Arguments

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

### Details

See [https://radiant-rstats.github.io/docs/model/simulater.html](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

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

Add interaction terms to list of test variables if needed

**Description**

Add interaction terms to list of test variables if needed

**Usage**

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

**Value**

A vector of variables names to test

**Examples**

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

---

**var_check**

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

**Description**

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

**Usage**

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

\- **ev** List of explanatory variables provided to `regress` or `logistic` Regression
\- **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](https://radiant-rstats.github.io/docs/model/regress.html) for an example in Radiant

Value

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

Examples

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

---

**write.coeff** Write coefficient table for linear and logistic regression

Description

Write coefficient table for linear and logistic regression

Usage

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

Arguments

\- **object** A fitted model object of class `regress` or `logistic` Regression
\- **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

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