Package ‘tidyrules’

June 30, 2024

Type Package

Title Utilities to Retrieve Rulelists from Model Fits, Filter, Prune, Reorder and Predict on Unseen Data

Version 0.2.7

Maintainer Srikanth Komala Sheshachala <sri.teach@gmail.com>

Depends R (>= 3.6.0),

Imports stringr (>= 1.3.1), magrittr (>= 1.5), purrr (>= 0.3.2),
partykit (>= 1.2.2), rlang (>= 1.1.3), generics (>= 0.1.3),
checkmate (>= 2.3.1), tidytable (>= 0.11.0), data.table (>= 1.14.6), DescTools (>= 0.99.54), MetricsWeighted (>= 1.0.3),
cli (>= 3.6.2), glue (>= 1.7.0), pheatmap (>= 1.0.12), proxy (>= 0.4.27), tibble (>= 3.2.1),

Suggests AmesHousing (>= 0.0.3), dplyr (>= 0.8), C50 (>= 0.1.2),
Cubist (>= 0.2.2), rpart (>= 1.2.2), rpart.plot (>= 3.0.7),
modeldata (>= 0.0.1), testthat (>= 2.0.1), MASS (>= 7.3.50),
milbench (>= 2.1.1), rmarkdown (>= 1.13), palmerpenguins (>= 0.1.1),

Description Provides a framework to work with decision rules. Rules can be extracted from supported models, augmented with (custom) metrics using validation data, manipulated using standard dataframe operations, reordered and pruned based on a metric, predict on unseen (test) data. Utilities include; Creating a rulelist manually, Exporting a rulelist as a SQL case statement and so on. The package offers two classes; rulelist and ruleset based on dataframe.

URL https://github.com/talegari/tidyrules,
https://talegari.github.io/tidyrules/

BugReports https://github.com/talegari/tidyrules/issues

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.1

NeedsCompilation no
Author  Srikanth Komala Sheshachala [aut, cre],
        Amith Kumar Ullur Raghavendra [aut]

Repository  CRAN

Date/Publication  2024-06-29 22:40:06 UTC

Contents

as_rulelist ......................................................... 3
as_rulelist.data.frame ........................................... 3
as_ruleset ............................................................. 4
augment ................................................................. 5
augment.rulelist ..................................................... 5
calculate ............................................................... 8
calculate.rulelist ................................................... 8
convert_rule_flavor .................................................. 11
package_tidyrules ................................................... 12
plot.prune_rulelist .................................................. 12
plot.rulelist ........................................................... 13
predict.rulelist ....................................................... 14
predict.ruleset ....................................................... 15
print.prune_rulelist .................................................. 16
print.rulelist ........................................................ 16
print.ruleset .......................................................... 17
prune ................................................................. 18
prune.rulelist ........................................................ 18
reorder ............................................................... 20
reorder.rulelist ....................................................... 20
rulelist ............................................................... 21
ruleset ............................................................... 22
set_keys .............................................................. 23
set_validation_data ................................................... 24
tidy ................................................................. 25
tidy.C5.0 ............................................................ 25
tidy.constparty ....................................................... 26
tidy.cubist .......................................................... 27
 tidy.rpart ........................................................... 28
to_sql_case .......................................................... 29

Index ................................................................. 31
Description

as_rulelist generic

Usage

as_rulelist(x, ...)

Arguments

x object to be coerced to a rulelist
...

Value

A rulelist

See Also

rulelist, tidy, augment, predict, calculate, prune, reorder

Description

Convert a set of rules in a dataframe to a rulelist

Usage

## S3 method for class 'data.frame'
as_rulelist(x, keys = NULL, model_type = NULL, estimation_type, ...)

Arguments

x dataframe to be coerced to a rulelist
target (character vector, default: NULL) column names which form the key
model_type (string, default: NULL) Name of the model which generated the rules
estimation_type (string) One among: 'regression', 'classification'
...

currently unused
as_ruleset

Description
Returns a ruleset object

Usage
as_ruleset(rulelist)

Arguments
rulelist A rulelist

Value
A ruleset

See Also
rulelist

Examples
model_class_party = partykit::ctree(species ~ .,
data = palmerpenguins::penguins
)

as_ruleset(tidy(model_class_party))

details
Input dataframe should contain these columns: rule_nbr, LHS, RHS. Providing other inputs helps augment better.

Value
rulelist object

See Also
rulelist, tidy, augment, predict, calculate, prune, reorder

Examples
rules_df = tidytable::tidytable(rule_nbr = 1:2,
LHS = c("var_1 > 50", "var_2 < 30"),
RHS = c(2, 1)
)

as_rulelist(rules_df, estimation_type = "regression")
augment is re-export of generics::augment from tidyrules package

Description
See augment.rulelist

Usage
augment(x, ...)

Arguments
x A rulelist
... For methods to use

See Also
rulelist, tidy, augment, predict, calculate, prune, reorder

 augment.rulelist Augment a rulelist

Description
augment outputs a rulelist with an additional column named augmented_stats based on summary statistics calculated using attribute validation_data.

Usage
## S3 method for class 'rulelist'
augment(x, ...)

Arguments
x A rulelist
... (expressions) To be send to tidytable::summarise for custom aggregations. See examples.

Details
The dataframe-column augmented_stats will have these columns corresponding to the estimation_type:
- For regression: support, IQR, RMSE
- For classification: support, confidence, lift

along with custom aggregations.
Value

A rulelist with a new dataframe-column named augmented_stats.

See Also

rulelist, tidy, augment, predict, calculate, prune, reorder

Examples

# Examples for augment ------------------------------------------------------
library("magrittr")

# C5 ----
att = modeldata::attrition
set.seed(100)
train_index = sample(c(TRUE, FALSE), nrow(att), replace = TRUE)
model_c5 = C50::C5.0(Attrition ~., data = att[train_index, ], rules = TRUE)
tidy_c5 =
model_c5 %>%
tidy() %>%
set_validation_data(att[!train_index, ], "Attrition")

tidy_c5

augment(tidy_c5) %>%
tidytable::unnest(augmented_stats, names_sep = "__") %>%
tidytable::glimpse()

# augment with custom aggregator
augment(tidy_c5, output_counts = list(table(Attrition))) %>%
tidytable::unnest(augmented_stats, names_sep = "__") %>%
tidytable::glimpse()

# rpart ----
set.seed(100)
train_index = sample(c(TRUE, FALSE), nrow(iris), replace = TRUE)

model_class_rpart = rpart::rpart(Species ~ ., data = iris[train_index, ]
tidy_class_rpart = tidy(model_class_rpart) %>%
set_validation_data(iris[!train_index, ], "Species")
tidy_class_rpart

model_regr_rpart = rpart::rpart(Sepal.Length ~ ., data = iris[train_index, ]
tidy_regr_rpart = tidy(model_regr_rpart) %>%
set_validation_data(iris[!train_index, ], "Sepal.Length")
tidy_regr_rpart

# augment (classification case)
augment(tidy_class_rpart) %>%
tidytable::unnest(augmented_stats, names_sep = "__") %>%
tidytable::glimpse()
# augment (regression case)
augment(tidy_regr_rpart) %>%
  tidytable::unnest(augmented_stats, names_sep = "__") %>%
  tidytable::glimpse()

# party ----
pen = palmerpenguins::penguins
  tidytable::drop_na(bill_length_mm)
set.seed(100)
  train_index = sample(c(TRUE, FALSE), nrow(pen), replace = TRUE)

model_class_party = partykit::ctree(species ~ ., data = pen[train_index, ])
tidy_class_party = tidy(model_class_party)
  set_validation_data(pen[!train_index, ], "species")
tidy_class_party

model_regr_party =
  partykit::ctree(bill_length_mm ~ ., data = pen[train_index, ])
tidy_regr_party = tidy(model_regr_party)
  set_validation_data(pen[!train_index, ], "bill_length_mm")
tidy_regr_party

# augment (classification case)
augment(tidy_class_party) %>%
  tidytable::unnest(augmented_stats, names_sep = "__") %>%
  tidytable::glimpse()

# augment (regression case)
augment(tidy_regr_party) %>%
  tidytable::unnest(augmented_stats, names_sep = "__") %>%
  tidytable::glimpse()

# cubist ----
att = modeldata::attrition
set.seed(100)
  train_index = sample(c(TRUE, FALSE), nrow(att), replace = TRUE)
cols_att = setdiff(colnames(att), c("MonthlyIncome", "Attrition"))

model_cubist = Cubist::cubist(x = att[train_index, cols_att],
  y = att[train_index, "MonthlyIncome"]
)
tidy_cubist = tidy(model_cubist)
  set_validation_data(att[!train_index, ], "MonthlyIncome")
tidy_cubist

augment(tidy_cubist) %>%
  tidytable::unnest(augmented_stats, names_sep = "__") %>%
  tidytable::glimpse()
calculate is re-export of generics::calculate from tidyrules package

Description

See calculate.rulelist

Usage

calculate(x, ...)

Arguments

x          A rulelist
...        See calculate.rulelist

See Also

rulelist, tidy, augment, predict, calculate, prune, reorder

calculate.rulelist  calculate metrics for a rulelist

Description

Computes some metrics (based on estimation_type) in cumulative window function style over the rulelist (in the same order) ignoring the keys.

Usage

## S3 method for class 'rulelist'
calculate(x, metrics_to_exclude = NULL, ...)

Arguments

x          A rulelist
metrics_to_exclude
            (character vector) Names of metrics to exclude
...        Named list of custom metrics. See 'details'.
Details

**Default Metrics:**

These metrics are calculated by default:

- **cumulative_coverage**: For nth rule in the rulelist, number of distinct row_nbrs (of new_data) covered by nth and all preceding rules (in order). In weighted case, we sum the weights corresponding to the distinct row_nbrs.

- **cumulative_overlap**: Up to nth rule in the rulelist, number of distinct row_nbrs (of new_data) already covered by some preceding rule (in order). In weighted case, we sum the weights corresponding to the distinct row_nbrs.

For classification:

- **cumulative_accuracy**: For nth rule in the rulelist, fraction of row_nbrs such that RHS matches the y_name column (of new_data) by nth and all preceding rules (in order). In weighted case, weighted accuracy is computed.

For regression:

- **cumulative_RMSE**: For nth rule in the rulelist, weighted RMSE of all predictions (RHS) predicted by nth rule and all preceding rules.

**Custom metrics:**

Custom metrics to be computed should be passed a named list of function(s) in ... The custom metric function should take these arguments in same order: rulelist, new_data, y_name, weight. The custom metric function should return a numeric vector of same length as the number of rows of rulelist.

**Value**

A dataframe of metrics with a rule_nbr column.

**See Also**

rulelist, tidy, augment, predict, calculate, prune, reorder

**Examples**

```r
library("magrittr")
model_c5 = C50::C5.0(Attrition ~., data = modedata::attrition, rules = TRUE)
tidy_c5 = tidy(model_c5) %>%
  set_validation_data(modedata::attrition, "Attrition") %>%
  set_keys(NULL)

# calculate default metrics (classification)
calculate(tidy_c5)

model_rpart = rpart::rpart(MonthlyIncome ~., data = modedata::attrition)
tidy_rpart =
tidy(model_rpart) %>%
  set_validation_data(modedata::attrition, "MonthlyIncome") %>%
  set_keys(NULL)
```
# calculate default metrics (regression)
calculate(tidy_rpart)

# calculate default metrics with a custom metric
#' custom function to get cumulative MAE
library("tidytable")
get_cumulative_MAE = function(rulelist, new_data, y_name, weight){
  priority_df =
    rulelist %>%
    select(rule_nbr) %>%
    mutate(priority = 1:nrow(rulelist)) %>%
    select(rule_nbr, priority)
  pred_df =
    predict(rulelist, new_data) %>%
    left_join(priority_df, by = "rule_nbr") %>%
    mutate(weight = local(weight)) %>%
    select(rule_nbr, row_nbr, weight, priority)
  new_data2 =
    new_data %>%
    mutate(row_nbr = 1:n()) %>%
    select(all_of(c("row_nbr", y_name)))
  rmse_till_rule = function(rn){
    if (is.character(rulelist$RHS)) {
      inter_df =
        pred_df %>%
        tidytable::filter(priority <= rn) %>%
        left_join(mutate(new_data, row_nbr = 1:n()), by = "row_nbr") %>%
        left_join(select(rulelist, rule_nbr, RHS), by = "rule_nbr") %>%
        nest(by = c("RHS", "rule_nbr", "row_nbr", "priority", "weight")) %>%
        mutate(RHS = purrr::map2_dbl(RHS, data, ~ eval(parse(text = .x), envir = .y))
      ) %>%
      unnest(data)
    } else {
      inter_df =
        pred_df %>%
        tidytable::filter(priority <= rn) %>%
        left_join(new_data2, by = "row_nbr") %>%
        left_join(select(rulelist, rule_nbr, RHS), by = "rule_nbr")
    }
    inter_df %>%
    summarise(rmse = MetricsWeighted::mae(RHS, .data[[y_name]], weight,
### convert_rule_flavor

Convert a R parsable rule to python/sql parsable rule

**Description**

Convert a R parsable rule to python/sql parsable rule

**Usage**

```r
convert_rule_flavor(rule, flavor)
```

**Arguments**

- `rule` (chr vector): R parsable rule(s)
- `flavor` (string): One among: `python`, `sql`

**Value**

(chr vector) of rules

**See Also**

- `rulelist`, `tidy`, `augment`, `predict`, `to_sql_case`

Other Auxiliary Rulelist Utility: `to_sql_case()`
Description

tidyrules package provides a framework to work with decision rules. Rules can be extracted from supported models using tidy, augmented using validation data by augment, manipulated using standard dataframe operations, (modified) rulelists can be used to predict on unseen (test) data. Utilities include: Create a rulelist manually (as_rulelist), Export a rulelist to SQL (to_sql_case) and so on. The package offers two classes; rulelist and ruleset based on dataframe.

Author(s)

Maintainer: Srikanth Komala Sheshachala <sri.teach@gmail.com>

Authors:

• Amith Kumar Ullur Raghavendra <amith54@gmail.com>

See Also

rulelist, tidy, augment, predict

plot.prune_rulelist  Plot method for prune_rulelist class

Description

Plot method for prune_rulelist class

Usage

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

Arguments

x  A 'prune_rulelist' object

...  unused

Value

ggplot2 object (invisibly)
Description

Plots a heatmap with rule_nbr’s on x-side and clusters of row_nbr’s on y-side of a binary matrix with 1 if a rule is applicable for a row.

Usage

```r
## S3 method for class 'rulelist'
plot(x, thres_cluster_rows = 1000, dist_metric = "jaccard", ...)
```

Arguments

- `x` A rulelist
- `thres_cluster_rows` (positive integer) Maximum number of rows beyond which a x-side dendrogram is not computed
- `dist_metric` (string or function, default: "jaccard") Distance metric for y-side (rule_nbr) passed to method argument of Proxy::dist
- `...` Arguments to be passed to pheatmap::pheatmap

Details

Number of clusters is set to min(number of unique rows in the row_nbr X rule_nbr matrix and thres_cluster_rows)

Examples

```r
library("magrittr")
att = modeldata::attrition
tidy_c5 =
  C50::C5.0(Attrition ~., data = att, rules = TRUE) %>%
tidy() %>%
set_validation_data(att, "Attrition") %>%
set_keys(NULL)
plot(tidy_c5)
```
predict.rulelist predict method for a rulelist

Description

Predicts rule_nbr applicable (as per the order in rulelist) for a row_nbr (per key) in new_data

Usage

```r
## S3 method for class 'rulelist'
predict(object, new_data, multiple = FALSE, ...)
```

Arguments

- `object`: A rulelist
- `new_data`: (dataframe)
- `multiple`: (flag, default: FALSE) Whether to output all rule numbers applicable for a row. If FALSE, the first satisfying rule is provided.
- `...`: unused

Details

If a row_nbr is covered more than one rule_nbr per 'keys', then rule_nbr appearing earlier (as in row order of the rulelist) takes precedence.

Output Format:

- When multiple is FALSE (default), output is a dataframe with three or more columns: row_number (int), columns corresponding to 'keys', rule_nbr (int).
- When multiple is TRUE, output is a dataframe with three or more columns: row_number (int), columns corresponding to 'keys', rule_nbr (list column of integers).
- If a row number and 'keys' combination is not covered by any rule, then rule_nbr column has missing value.

Value

A dataframe. See Details.

See Also

rulelist, tidy, augment, predict, calculate, prune, reorder
**predict.ruleset**

**Examples**

```r
model_c5 = C50::C5.0(species ~.,
  data = palmerpenguins::penguins,
  trials = 5,
  rules = TRUE
)
tidy_c5 = tidy(model_c5)
tidy_c5

output_1 = predict(tidy_c5, palmerpenguins::penguins)
output_1 # different rules per 'keys' ('trial_nbr' here)

output_2 = predict(tidy_c5, palmerpenguins::penguins, multiple = TRUE)
output_2 # `rule_nbr` is a list-column of integer vectors
```

---

**predict.ruleset** predict *method for a ruleset*

**Description**

Predicts multiple rule_nbr(s) applicable for a row_nbr (per key) in new_data

**Usage**

```r
## S3 method for class 'ruleset'
predict(object, new_data, ...)
```

**Arguments**

- **object** A ruleset
- **new_data** (dataframe)
- **...** unused

**Value**

A dataframe with three or more columns: row_number (int), columns corresponding to 'keys', rule_nbr (list column of integers). If a row number and 'keys' combination is not covered by any rule, then rule_nbr column has missing value.

**See Also**

predict.rulelist
Examples

```r
model_c5 = C5::C5.0(species ~.,
    data = palmerpenguins::penguins,
    trials = 5,
    rules = TRUE
  )
tidy_c5_ruleset = as_ruleset(tidy(model_c5))
tidy_c5_ruleset

predict(tidy_c5_ruleset, palmerpenguins::penguins)
```

print.prune_rulelist  
*Print method for prune_rulelist class*

**Description**

Print method for prune_rulelist class

**Usage**

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

**Arguments**

- `x`: A `prune_rulelist` object
- `...`: unused

print.rulelist  
*Print method for rulelist class*

**Description**

Prints rulelist attributes and first few rows.

**Usage**

```r
## S3 method for class 'rulelist'
print(x, banner = TRUE, ...)
```

**Arguments**

- `x`: A rulelist object
- `banner`: (flag, default: `TRUE`) Should the banner be displayed
- `...`: Passed to tidytable::print
print.ruleset

Value

input rulelist (invisibly)

See Also

rulelist, tidy, augment, predict, calculate, prune, reorder

print.ruleset  Print method for ruleset class

Description

Prints the ruleset object

Usage

## S3 method for class 'ruleset'
print(x, banner = TRUE, ...)

Arguments

x  

A rulelist

banner  

(flag, default: TRUE) Should the banner be displayed

...  

Passed to print.rulelist

Value

(invisibly) Returns the ruleset object

See Also

print.rulelist

Examples

model_class_party = partykit::ctree(species ~ .,
                                 data = palmerpenguins::penguins)

as_ruleset(tidy(model_class_party))
Description

See prune.rulelist

Usage

prune(tree, ...)

Arguments

tree A rulelist
...
See prune.rulelist

See Also

rulelist, tidy, augment, predict, calculate, prune, reorder

---

prune.rulelist prune rules of a rulelist

Description

Prune the rulelist by suggesting to keep first 'k' rules based on metrics computed by calculate

Usage

## S3 method for class 'rulelist'
prune(
    tree,
    metrics_to_exclude = NULL,
    stop_expr_string = "relative__cumulative_coverage >= 0.9",
    min_n_rules = 1,
    ...
)

Arguments

tree A rulelist
metrics_to_exclude (character vector or NULL) Names of metrics not to be calculated. See calculate for the list of default metrics.
stop_expr_string (string default: "relative__cumulative_coverage >= 0.9") Parsable condition
min_n_rules (positive integer) Minimum number of rules to keep
...
Named list of custom metrics passed to calculate
Details

1. Metrics are computed using `calculate`. 2. Relative metrics (prepended by 'relative__') are calculated by dividing each metric by its max value. 3. The first rule in rulelist order which meets the 'stop_expr_string' criteria is stored (say 'pos'). Print method suggests to keep rules until pos.

Value

Object of class `prune_ruleslist` with these components: 1. pruned: ruleset keeping only first 'pos' rows. 2. n_pruned_rules: pos. If stop criteria is never met, then pos = nrow(ruleset) 3. n_total_rules: nrow(ruleset), 4. metrics_df: Dataframe with metrics and relative metrics 5. stop_expr_string

See Also

`rulelist`, `tidy`, `augment`, `predict`, `calculate`, `prune`, `reorder`

Examples

```r
library("magrittr")
model_c5 = C50::C5.0(Attrition ~., data = modeldata::attrition, rules = TRUE)
tidy_c5 = tidy(model_c5) %>%
  set_validation_data(modeldata::attrition, "Attrition") %>%
  set_keys(NULL)

#' prune with defaults
prune_obj = prune(tidy_c5)
#' note that all other metrics are visible in the print output
prune_obj
plot(prune_obj)
prune_obj$pruned

#' prune with a different stop_expr_string threshold
prune_obj = prune(tidy_c5,
  stop_expr_string = "relative__cumulative_coverage >= 0.2"
)
prune_obj #' as expected, has smaller then 10 rules as compared to default args
plot(prune_obj)
prune_obj$pruned

#' prune with a different stop_expr_string metric
st = "relative__cumulative_overlap <= 0.7 & relative__cumulative_overlap > 0"
prune_obj = prune(tidy_c5, stop_expr_string = st)
prune_obj #' as expected, has smaller then 10 rules as compared to default args
plot(prune_obj)
prune_obj$pruned
```
reorder

reorder_generic

**Description**

reorder generic for rulelist

**Usage**

```r
reorder(x, ...)
```

**Arguments**

- **x**  
  A rulelist
- **...**  
  See reorder.rulelist

**See Also**

rulelist, tidy, augment, predict, calculate, prune, reorder

---

**reorder.rulelist**  
*Reorder the rules/rows of a rulelist*

**Description**

Implements a greedy strategy to add one rule at a time which maximizes/minimizes a metric.

**Usage**

```r
## S3 method for class 'rulelist'
reorder(x, metric = "cumulative_coverage", minimize = FALSE, init = NULL, ...)
```

**Arguments**

- **x**  
  A rulelist
- **metric**  
  (character vector or named list) Name of metrics or a custom function(s). See calculate. The 'n+1'th metric is used when there is a match at 'nth' level, similar to base::order. If there is a match at final level, row order of the rulelist comes into play.
- **minimize**  
  (logical vector) Whether to minimize. Either TRUE/FALSE or a logical vector of same length as metric
- **init**  
  (positive integer) Initial number of rows after which reordering should begin
- **...**  
  passed to calculate
See Also

rulelist, tidy, augment, predict, calculate, prune, reorder

Examples

```r
library("magrittr")
att = modeldata::attrition
tidy_c5 =
  C50::C5.0(Attrition ~., data = att, rules = TRUE) %>%
  tidy() %>%
  set_validation_data(att, "Attrition") %>%
  set_keys(NULL) %>%
  head(5)
# with defaults
reorder(tidy_c5)
# use 'cumulative_overlap' to break ties (if any)
reorder(tidy_c5, metric = c("cumulative_coverage", "cumulative_overlap"))
# reorder after 2 rules
reorder(tidy_c5, init = 2)
```

### Description

**Structure:**

A rulelist is ordered list of rules stored as a dataframe. Each row, specifies a rule (LHS), expected outcome (RHS) and some other details.

It has these mandatory columns:

- `rule_nbr`: (integer vector) Rule number
- `LHS`: (character vector) A rule is a string that can be parsed using `base::parse()`
- `RHS`: (character vector or a literal)

#### Example:

```r
<table>
<thead>
<tr>
<th>rule_nbr</th>
<th>LHS</th>
<th>RHS</th>
<th>support</th>
<th>confidence</th>
<th>lift</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( island %in% c('Biscoe')) &amp; ( flipper_length_mm &gt; 203 )</td>
<td>Gentoo</td>
<td>122</td>
<td>1.0000000</td>
<td>2.774193</td>
</tr>
<tr>
<td>2</td>
<td>( island %in% c('Biscoe')) &amp; ( flipper_length_mm &lt;= 203 )</td>
<td>Adelie</td>
<td>46</td>
<td>0.9565217</td>
<td>2.164760</td>
</tr>
<tr>
<td>3</td>
<td>( island %in% c('Dream', 'Torgersen')) &amp; ( bill_length_mm &gt; 44.1 )</td>
<td>Chinstrap</td>
<td>65</td>
<td>0.9535978</td>
<td>0.9535978</td>
</tr>
<tr>
<td>4</td>
<td>( island %in% c('Dream', 'Torgersen')) &amp; ( bill_length_mm &lt;= 44.1 )</td>
<td>Adelie</td>
<td>111</td>
<td>0.9459459</td>
<td>2.140825</td>
</tr>
</tbody>
</table>
```

**Create a rulelist:**

A rulelist can be created using `tidy()` on some supported model fits (run: `utils::methods(tidy)`).
It can also be created manually from a existing dataframe using `as_rulelist`. 
Keys and attributes:
Columns identified as 'keys' along with rule_nbr form a unique combination – a group of rules. For example, rule-based C5 model with multiple trials creates rules per each trial_nbr. predict method understands 'keys', thereby provides/predicts a rule number (for each row in new data / test data) within the same trial_nbr.

A rulelist has these mandatory attributes:
• estimation_type: One among regression, classification
• keys: (character vector)Names of the column that forms a key.
• model_type: (string) Name of the model

Set Validation data:
This helps a few methods like augment, calculate, prune, reorder require few additional attributes which can be set using set_validation_data.

Methods for rulelist:
1. Predict: Given a dataframe (possibly without a dependent variable column aka 'test data'), predicts the first rule (as ordered in the rulelist) per 'keys' that is applicable for each row. When multiple = TRUE, returns all rules applicable for a row (per key).
2. Augment: Outputs summary statistics per rule over validation data and returns a rulelist with a new dataframe-column.
3. Calculate: Computes metrics for a rulelist in a cumulative manner such as cumulative_coverage, cumulative_overlap, cumulative_accuracy.
4. Prune: Suggests pruning a rulelist such that some expectation are met (based on metrics). Example: cumulative_coverage of 80% can be met with a first few rules.
5. Reorder: Reorders a rulelist in order to maximize a metric.

Manipulating a rulelist:
Rulelists are essentially dataframes. Hence, any dataframe operations which preferably preserve attributes will output a rulelist. as_rulelist and as.data.frame will help in moving back and forth between rulelist and dataframe worlds.

Utilities for a rulelist:
1. as_rulelist: Create a rulelist from a dataframe with some mandatory columns.
2. set_keys: Set or Unset 'keys' of a rulelist.
3. to_sql_case: Outputs a SQL case statement for a rulelist.
4. convert_rule_flavor: Converts R-parsable rule strings to python/SQL parsable rule strings.

See Also
rulelist, tidy, augment, predict, calculate, prune, reorder

ruleset | Ruleset
---|---

Description
ruleset class is a piggyback class that inherits rulelist class for convenience of print and predict methods.
set_keys

Set keys for a rulelist

Description

'keys' are a set of column(s) which identify a group of rules in a rulelist. Methods like predict, augment produce output per key combination.

Usage

set_keys(x, keys, reset = FALSE)

Arguments

x     A rulelist
keys  (character vector or NULL)
reset (flag) Whether to reset the keys to sequential numbers starting with 1 when keys is set to NULL

Details

A new rulelist is returned with attr keys is modified. The input rulelist object is unaltered.

Value

A rulelist object

See Also

rulelist, tidy, augment, predict, calculate, prune, reorder

Other Core Rulelist Utility: set_validation_data()

Examples

model_c5 = C50::C5.0(Attrition ~., data = modeldata::attrition, rules = TRUE)
tidy_c5 = tidy(model_c5)
tidy_c5 # keys are: "trial_nbr"

tidy_c5["rule_nbr"] = 1:nrow(tidy_c5)
new_tidy_c5 = set_keys(tidy_c5, NULL) # remove all keys
new_tidy_c5

new_2_tidy_c5 = set_keys(new_tidy_c5, "trial_nbr") # set "trial_nbr" as key
new_2_tidy_c5

# Note that `tidy_c5` and `new_tidy_c5` are not altered.
tidy_c5
new_tidy_c5
set_validation_data  

Add validation_data to a rulelist

Description

Returns a rulelist with three new attributes set: validation_data, y_name and weight. Methods such as augment, calculate, prune, reorder require this to be set.

Usage

set_validation_data(x, validation_data, y_name, weight = 1)

Arguments

- x: A rulelist
- validation_data: (dataframe) Data to used for computing some metrics. It is expected to contain y_name column.
- y_name: (string) Name of the dependent variable column.
- weight: (non-negative numeric vector, default: 1) Weight per observation/row of validation_data. This is expected to have same length as the number of rows in validation_data. Only exception is when it is a single positive number, which means that all rows have equal weight.

Value

A rulelist with some extra attributes set.

See Also

rulelist, tidy, augment, predict, calculate, prune, reorder

Other Core Rulelist Utility: set_keys()

Examples

```r
att = modeldata::attrition
set.seed(100)
index = sample(c(TRUE, FALSE), nrow(att), replace = TRUE)
model_c5 = C50::C5.0(Attrition ~., data = att[index, ], rules = TRUE)

tidy_c5 = tidy(model_c5)
tidy_c5

tidy_c5_2 = set_validation_data(tidy_c5,
    validation_data = att[!index, ],
    y_name = "Attrition",
    weight = 1 # default
)
```
tidy

Description

`tidy` applied on a supported model fit creates a `rulelist`. See Also section links to documentation of specific methods.

Usage

```r
tidy(x, ...)
```

Arguments

- `x` A supported model object
- `...` For model specific implementations to use

See Also

`rulelist`, `tidy`, `augment`, `predict`, `calculate`, `prune`, `reorder`  
Other Core Tidy Utility: `tidy.C5.0()`, `tidy.cubist()`, `tidy.rpart()`

---

`tidy.C5.0`  
*Get the rulelist from a C5 model*

Description

Each row corresponds to a rule per `trial_nbr`

Usage

```r
## S3 method for class 'C5.0'
tidy(x, ...)
```

Arguments

- `x` `C50::C5.0` model fitted with `rules = TRUE`
- `...` Other arguments (See details)
Details

• The output columns are: rule_nbr, trial_nbr, LHS, RHS, support, confidence, lift.
• Rules per trial_nbr are sorted in this order: desc(confidence), desc(lift), desc(support).

Optional named arguments:

• laplace (flag, default: TRUE) is supported. This computes confidence with laplace correction as documented under 'Rulesets' here: C5 doc.

Value

A rulelist object

See Also

rulelist, tidy, augment, predict, calculate, prune, reorder
Other Core Tidy Utility: tidy(), tidy.cubist(), tidy.rpart()

Examples

model_c5 = C50::C5.0(Attrition ~ ., data = modeldata::attrition, rules = TRUE)
tidy(model_c5)
Details

These types of `party` models are supported: regression (y is numeric), classification (y is factor)

For `party` classification model:

- Output columns are: `rule_nbr`, `LHS`, `RHS`, `support`, `confidence`, `lift`, `terminal_node_id`.
- Rules are sorted in this order: `desc(confidence)`, `desc(lift)`, `desc(support)`.

For `party` regression model:

- Output columns are: `rule_nbr`, `LHS`, `RHS`, `support`, `IQR`, `RMSE`, `terminal_node_id`.
- Rules are sorted in this order: `RMSE`, `desc(support)`.

Value

A `rulelist` object

See Also

`rulelist, tidy, augment, predict, calculate, prune, reorder`

Examples

```r
pen = palmerpenguins::penguins
model_class_party = partykit::ctree(species ~ ., data = pen)
tidy(model_class_party)
model_regr_party = partykit::ctree(bill_length_mm ~ ., data = pen)
tidy(model_regr_party)
```
Details

- The output columns are: `rule_nbr`, `committee`, `LHS`, `RHS`, `support`, `mean`, `min`, `max`, `error`.
- Rules are sorted in this order per committee: `error`, `desc(support)`

Value

A `rulelist` object

See Also

`rulelist`, `tidy`, `augment`, `predict`, `calculate`, `prune`, `reorder`

Other Core Tidy Utility: `tidy()`, `tidy.C5.0()`, `tidy.rpart()`

Examples

```r
att = modeldata::attrition
cols_att = setdiff(colnames(att), c("MonthlyIncome", "Attrition"))
model_cubist = Cubist::cubist(x = att[, cols_att],
                              y = att["MonthlyIncome"]
                           )
tidy(model_cubist)
```

Description

Each row corresponds to a rule

Usage

```r
## S3 method for class 'rpart'
tidy(x, ...)
```

Arguments

- `x` `rpart::rpart` model
- `...` Other arguments (currently unused)
to_sql_case

Details
For rpart rules, one should build the model without ordered factor variable. We recommend you to convert ordered factor to factor or integer class.

For rpart::rpart classification model:
- Output columns are: rule_nbr, LHS, RHS, support, confidence, lift.
- The rules are sorted in this order: desc(confidence), desc(lift), desc(support).

For rpart::rpart regression(anova) model:
- Output columns are: rule_nbr, LHS, RHS, support.
- The rules are sorted in this order: desc(support).

Value
A rulelist object

See Also
rulelist, tidy, augment, predict, calculate, prune, reorder
Other Core Tidy Utility: tidy(), tidy.C5.0(), tidy.cubist()

Examples
model_class_rpart = rpart::rpart(Species ~ ., data = iris)
tidy(model_class_rpart)

model_regr_rpart = rpart::rpart(Sepal.Length ~ ., data = iris)
tidy(model_regr_rpart)

to_sql_case

Extract SQL case statement from a rulelist

Description
Extract SQL case statement from a rulelist

Usage
to_sql_case(rulelist, rhs_column_name = "RHS", output_colname = "output")

Arguments
rulelist A rulelist object

rhs_column_name (string, default: "RHS") Name of the column in the rulelist to be used as RHS (WHEN some_rule THEN rhs) in the sql case statement

output_colname (string, default: "output") Name of the output column created by the SQL statement (used in case ... AS output_column)
Details
As a side-effect, the SQL statement is cat to stdout. The output contains newline character.

Value
(string invisibly) SQL case statement

See Also
rulelist, tidy, augment, predict, convert_rule_flavor
Other Auxiliary Rulelist Utility: convert_rule_flavor()

Examples
model_c5 = C50::C5.0(Attrition ~., data = modeldata::attrition, rules = TRUE)
tidy(model_c5)
to_sql_case(tidy(model_c5))
tidyrules-package (package_tidyrules), 12

tidytable::summarise, 5

to_sql_case, 11, 12, 22, 29