Package ‘rules’

March 14, 2022

Title Model Wrappers for Rule-Based Models
Version 0.2.0
License MIT + file LICENSE
Depends modeldata, parsnip (>= 0.2.0)
Imports dials, dplyr, generics (>= 0.1.0), purrr, rlang, stringr, tibble, tidyr
Suggests C50, covr, Cubist, knitr, recipes, rmarkdown, spelling, testthat, xrf (>= 0.2.0)
Config/Needs/website tidyr, tidyverse/tidytemplate, recipes, xrf
Encoding UTF-8
Language en-US
RoxygenNote 7.1.2
NeedsCompilation no
Author Max Kuhn [aut, cre] (https://orcid.org/0000-0003-2402-136X), RStudio [cph]
Maintainer Max Kuhn <max@rstudio.com>
Repository CRAN
Date/Publication 2022-03-14 18:10:02 UTC

R topics documented:

  committees ......................................................... 2
  mtry_prop ......................................................... 2
  multi_predict_C5_rules ........................................ 3
  tidy.cubist ....................................................... 4
Index

committees | Parameter functions for Cubist models

Description
Committee-based models enact a boosting-like procedure to produce ensembles. committees parameter is for the number of models in the ensembles while max_rules can be used to limit the number of possible rules.

Usage
committees(range = c(1L, 100L), trans = NULL)
max_rules(range = c(1L, 500L), trans = NULL)

Arguments
range A two-element vector holding the defaults for the smallest and largest possible values, respectively.
trans A trans object from the scales package, such as scales::log10_trans() or scales::reciprocal_trans(). If not provided, the default is used which matches the units used in range. If no transformation, NULL.

Value
A function with classes "quant_param" and "param"

Examples
committees()
committees(4:5)
max_rules()

mtry_prop | Proportion of Randomly Selected Predictors

Description
Proportion of Randomly Selected Predictors

Usage
mtry_prop(range = c(0.1, 1), trans = NULL)
multi_predict._C5_rules

Arguments

range A two-element vector holding the defaults for the smallest and largest possible values, respectively.

trans A trans object from the scales package, such as scales::log10_trans() or scales::reciprocal_trans(). If not provided, the default is used which matches the units used in range. If no transformation, NULL.

Value

A dials with classes "quant_param" and "param". The range element of the object is always converted to a list with elements "lower" and "upper".

---

multi_predict._C5_rules

multi_predict() methods for rule-based models

Description

multi_predict() methods for rule-based models

Usage

## S3 method for class '``.C5_rules``'
multi_predict(object, new_data, type = NULL, trees = NULL, ...)

## S3 method for class '``.cubist``'
multi_predict(object, new_data, type = NULL, neighbors = NULL, ...)

## S3 method for class '``.xrf``'
multi_predict(object, new_data, type = NULL, penalty = NULL, ...)

Arguments

object An object of class model_fit
new_data A rectangular data object, such as a data frame.
type A single character value or NULL. Possible values are class" and "prob".
trees An numeric vector of trees between one and 100.
... Not currently used.
neighbors An numeric vector of neighbors values between zero and nine.
penalty Non-negative penalty values.

details

For C5.0 rule-based models, the model fit may contain less boosting iterations than the number requested. Printing the object will show how many were used due to early stopping. This can be change using an option in C50::C5.0Control(). Beware that the number of iterations requested
tidy.cubist

**Value**

A tibble with one row for each row of `new_data`. Multiple predictions are contained in a list column called `.pred`. That column has the standard `parsnip` prediction column names as well as the column with the tuning parameter values.

---

**tidy.cubist**  
*Turn regression rule models into tidy tibbles*

---

**Description**

Turn regression rule models into tidy tibbles

**Usage**

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

## S3 method for class 'xrf'
tidy(x, penalty = NULL, unit = c("rules", "columns"), ...)
```

**Arguments**

- `x`  
  A Cubist or xrf object.

- `...`  
  Not currently used.

- `penalty`  
  A single numeric value for the lambda penalty value.

- `unit`  
  What data should be returned? For `unit = "rules"`, each row corresponds to a rule. For `unit = "columns"`, each row is a predictor column. The latter can be helpful when determining variable importance.

**Details**

An example:

```r
library(dplyr)

data(ames, package = "modeldata")

ames <-
  ames %>%
  mutate(Sale_Price = log10(ames$Sale_Price),
         Gr_Liv_Area = log10(ames$Gr_Liv_Area))

# ---------------------------------------------------------------------------

cb_fit <-
cubist_rules(committees = 10) %>%
```
set_engine("Cubist") %>%
fit(Sale_Price ~ Neighborhood + Longitude + Latitude + Gr_Liv_Area + Central_Air,
data = ames)

cb_res <- tidy(cb_fit)
cb_res

## # A tibble: 157 × 5
## #  committee rule_num rule estimate statistic
## # <int> <int> <chr> <list> <list>
## 1 1 1 ( Central_Air == 'N' ) & ( Gr_Liv_Area <= 3.0326188 ) & ( Neighborhood %in% c( 'Old_Town', 'Ed.' ) ) & ( Gr_Liv_Area <= 3.2284005 ) & ( Neighborhood %in% c( 'North_Ames', 'Des Moines' ) ) & ( Latitude <= 42.009399 ) & ( Neighborhood %in% c( 'College_Creek', 'Ames' ) )

library(recipes)
xrf_reg_mod <-
  rule_fit(trees = 10, penalty = .001) %>%
  set_engine("xrf") %>%
  set_mode("regression")

# Make dummy variables since xgboost will not
ames_rec <-
  recipe(Sale_Price ~ Neighborhood + Longitude + Latitude + Neighborhood %in% c( 'North_Ames', 'Des Moines' ) & ( Latitude <= 42.009399 ) & ( Neighborhood %in% c( 'College_Creek', 'Ames' ) )

## # A tibble: 4 × 2
## term estimate
## <chr> <dbl>
## 1 (Intercept) -408.
## 2 Longitude  1.43
## 3 Latitude  6.6
## 4 Gr_Liv_Area 0.7

cb_res$statistic[[1]]

## # A tibble: 1 × 6
## num_conditions coverage mean min max error
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 2 154 4.94 4.11 5.31 0.0956

# ------------------------------------------------------------------------------
tidy.cubist

Gr_Liv_Area + Central_Air, data = ames) %>%
step_dummy(Neighborhood, Central_Air) %>%
step_zv(all_predictors())

ames_processed <- prep(ames_rec) %>% bake(new_data = NULL)

set.seed(1)
xrf_reg_fit <-
xrf_reg_mod %>%
fit(Sale_Price ~ ., data = ames_processed)

xrf_rule_res <- tidy(xrf_reg_fit)
xrf_rule_res$rule[nrow(xrf_rule_res)] %>% rlang::parse_expr()
## (Gr_Liv_Area < 3.30210185) & (Gr_Liv_Area < 3.38872266) & (Gr_Liv_Area >=
## 2.94571471) & (Gr_Liv_Area >= 3.24870872) & (Latitude < 42.0271072) &
## (Neighborhood_Old_Town >= -9.53674316e-07)

xrf_col_res <- tidy(xrf_reg_fit, unit = "columns")
xrf_col_res
## # A tibble: 149 × 3
## #  rule_id term estimate
## # <chr> <chr> <dbl>
## 1 r0_1 Gr_Liv_Area -1.27e- 2
## 2 r2_4 Gr_Liv_Area -3.70e-10
## 3 r2_2 Gr_Liv_Area  7.59e- 3
## 4 r2_4 Central_Air_Y -3.70e-10
## 5 r3_5 Longitude  1.06e- 1
## 6 r3_6 Longitude  2.65e- 2
## 7 r3_5 Latitude  1.06e- 1
## 8 r3_6 Latitude  2.65e- 2
## 9 r3_5 Longitude  1.06e- 1
## # ... with 139 more rows

Value

The Cubist method has columns committee, rule_num, rule, estimate, and statistics. The
latter two are nested tibbles. estimate contains the parameter estimates for each term in the re-
gression model and statistics has statistics about the data selected by the rules and the model
fit.

The xrf results has columns rule_id, rule, and estimate. The rule_id column has the rule
identifier (e.g., "r0_21") or the feature column name when the column is added directly into the
model. For multiclass models, a class column is included.

In each case, the rule column has a character string with the rule conditions. These can be converted
to an R expression using rlang::parse_expr().
Index

C50::C5.0Control(), 3
committees, 2

max_rules(committees), 2
mtry_prop, 2
multi_predict._C5_rules, 3
multi_predict._cubist
   (multi_predict._C5_rules), 3
multi_predict._xrf
   (multi_predict._C5_rules), 3

rlang::parse_expr(), 6
tidy.cubist, 4
tidy.xrf(tidy.cubist), 4