Package 'rules'

June 23, 2022

Title Model Wrappers for Rule-Based Models

Version 1.0.0

Description Bindings for additional models for use with the 'parsnip' package. Models include prediction rule ensembles (Friedman and Popescu, 2008) <doi:10.1214/07-AOAS148>, C5.0 rules (Quinlan, 1992 ISBN: 1558602380), and Cubist (Kuhn and Johnson, 2013) <doi:10.1007/978-1-4614-6849-3>.

License MIT + file LICENSE

URL https://github.com/tidymodels/rules, https://rules.tidymodels.org/

BugReports https://github.com/tidymodels/rules/issues

Depends parsnip (>= 0.2.1.9003), R (>= 3.4)

Imports dials (>= 0.1.1.9001), dplyr, generics (>= 0.1.0), purrr, rlang, stringr, tibble, tidyr

Suggests C50, covr, Cubist, knitr, modeldata, recipes, rmarkdown, spelling, testthat (>= 3.0.0), xrf (>= 0.2.0)

Config/Needs/website tidyr, tidyverse/tidytemplate, recipes, xrf

Config/testthat/edition 3

Encoding UTF-8

Language en-US

RoxygenNote 7.2.0.9000

NeedsCompilation no

Author Emil Hvitfeldt [aut, cre] (<https://orcid.org/0000-0002-0679-1945>), Max Kuhn [aut] (<https://orcid.org/0000-0003-2402-136X>), RStudio [cph, fnd]

Maintainer Emil Hvitfeldt <emilhhvitfeldt@gmail.com>

Repository CRAN

Date/Publication 2022-06-23 13:00:02 UTC

R topics documented:

`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()
```

multi_predict._cubist multi_predict() *methods for rule-based models*

Description

multi_predict() methods for rule-based models

Usage

```
## 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	A model_fit object.
new_data	A rectangular data object, such as a data frame.
type	A single character value or NULL. This argument is ignored in the method for _cubist objects and is handled internally (since type = "numeric" is always used).
neighbors	A numeric vector of neighbors values between zero and nine.
...	Not currently used.
penalty	Non-negative penalty values.

tidy.C5.0 *Turn rule models into tidy tibbles*

Description

Turn rule models into tidy tibbles

Usage

```
## S3 method for class 'C5.0'
tidy(x, trials = x$trials["Actual"], ...)

## S3 method for class 'cubist'
tidy(x, committees = x$committee, ...)

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

Arguments

x	A Cubist, C5.0, or xrf object.
trials	The number of boosting iterations to tidy (defaults to the entire ensemble).
...	Not currently used.
committees	The number of committees to tidy (defaults to the entire ensemble).
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:

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
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' ) . . .  <tibble> <tibble>
##  2         1        2 ( Gr_Liv_Area <= 3.0326. . .  <tibble> <tibble>
##  3         1        3 ( Neighborhood  %in% c(. . .  <tibble> <tibble>
##  4         1        4 ( Neighborhood  %in% c(. . .  <tibble> <tibble>
##  5         1        5 ( Central_Air == 'N' ) . . .  <tibble> <tibble>
##  6         1        6 ( Longitude <= -93.6520. . .  <tibble> <tibble>
##  7         1        7 ( Gr_Liv_Area > 3.22840. . .  <tibble> <tibble>
##  8         1        8 ( Neighborhood  %in% c(. . .  <tibble> <tibble>
##  9         1        9 ( Latitude <= 42.009399. . .  <tibble> <tibble>
## 10         1       10 ( Neighborhood  %in% c(. . .  <tibble> <tibble>
## # . . .  with 147 more rows
```

```
cb_res$estimate[[1]]
```

```
## # 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
```

```
# ------------------------------------------------------------------------------

library(recipes)
```

```
##
## Attaching package: 'recipes'

## The following object is masked from 'package:stats':
##
##     step

## The following object is masked from 'package:devtools':
##
##     check
```

```
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 +
         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)

## New names:
## • `.` -> `....1`
## • `.` -> `....2`
## • `.` -> `....3`
## • `.` -> `....4`
## • `.` -> `....5`
## • `.` -> `....6`
## • `.` -> `....7`
## • `.` -> `....8`
## • `.` -> `....9`
## • `.` -> `....10`
## • `.` -> `....11`
## • `.` -> `....12`
## • `.` -> `....13`
## • `.` -> `....14`
## • `.` -> `....15`
## • `.` -> `....16`
## • `.` -> `....17`
## • `.` -> `....18`
## • `.` -> `....19`
## • `.` -> `....20`
## • `.` -> `....21`
## • `.` -> `....22`
## • `.` -> `....23`
## • `.` -> `....24`
## • `.` -> `....25`
## • `.` -> `....26`
## • `.` -> `....27`
## • `.` -> `....28`
## • `.` -> `....29`
## • `.` -> `....30`
## • `.` -> `....31`
## • `.` -> `....32`
## • `.` -> `....33`
## • `.` -> `....34`
## • `.` -> `....35`
```

```
## • `.` -> `....36`
## • `.` -> `....37`
## • `.` -> `....38`
## • `.` -> `....39`
## • `.` -> `....40`
## • `.` -> `....41`
## • `.` -> `....42`
## • `.` -> `....43`
## • `.` -> `....44`
## • `.` -> `....45`
## • `.` -> `....46`
## • `.` -> `....47`
## • `.` -> `....48`
## • `.` -> `....49`
## • `.` -> `....50`
## • `.` -> `....51`
## • `.` -> `....52`
## • `.` -> `....53`
## • `.` -> `....54`
## • `.` -> `....55`
## • `.` -> `....56`
## • `.` -> `....57`
## • `.` -> `....58`
## • `.` -> `....59`
## • `.` -> `....60`
## • `.` -> `....61`
## • `.` -> `....62`
## • `.` -> `....63`
## • `.` -> `....64`
## • `.` -> `....65`
## • `.` -> `....66`
## • `.` -> `....67`
## • `.` -> `....68`
## • `.` -> `....69`
## • `.` -> `....70`
## • `.` -> `....71`
## • `.` -> `....72`
## • `.` -> `....73`
## • `.` -> `....74`
## • `.` -> `....75`
## • `.` -> `....76`
## • `.` -> `....77`
## • `.` -> `....78`
## • `.` -> `....79`
## • `.` -> `....80`
## • `.` -> `....81`
## • `.` -> `....82`
## • `.` -> `....83`
```

```
## • `.` -> `....84`
## • `.` -> `....85`
## • `.` -> `....86`
## • `.` -> `....87`
## • `.` -> `....88`
## • `.` -> `....89`
## • `.` -> `....90`
## • `.` -> `....91`
## • `.` -> `....92`
## • `.` -> `....93`
## • `.` -> `....94`
## • `.` -> `....95`
## • `.` -> `....96`
## • `.` -> `....97`
## • `.` -> `....98`
## • `.` -> `....99`
## • `.` -> `....100`
## • `.` -> `....101`
## • `.` -> `....102`
## • `.` -> `....103`
## • `.` -> `....104`
## • `.` -> `....105`
## • `.` -> `....106`
## • `.` -> `....107`
## • `.` -> `....108`
## • `.` -> `....109`
## • `.` -> `....110`
## • `.` -> `....111`
## • `.` -> `....112`
## • `.` -> `....113`
## • `.` -> `....114`
## • `.` -> `....115`
## • `.` -> `....116`
## • `.` -> `....117`
## • `.` -> `....118`
```

```
xrf_rule_res <- tidy(xrf_reg_fit)
xrf_rule_res$rule[nrow(xrf_rule_res)] %>% rlang::parse_expr()
```

```
## (Central_Air_Y >= 0.5) & (Gr_Liv_Area < 3.38872266) & (Gr_Liv_Area >=
##     2.94571471) & (Gr_Liv_Area >= 3.24870872) & (Latitude >=
##     42.0271072) & (Neighborhood_Old_Town >= 0.5)
```

```
xrf_col_res <- tidy(xrf_reg_fit, unit = "columns")
xrf_col_res
```

```
## # A tibble: 417 × 3
##    rule_id term          estimate
##    <chr>   <chr>            <dbl>
##  1 r0_1    Gr_Liv_Area    -0.0138
```

```
##  2 r2_3    Gr_Liv_Area    -0.0310
##  3 r2_2    Gr_Liv_Area     0.0127
##  4 r2_3    Central_Air_Y  -0.0310
##  5 r3_5    Longitude       0.0859
##  6 r3_6    Longitude       0.0171
##  7 r3_2    Longitude      -0.0109
##  8 r3_5    Latitude        0.0859
##  9 r3_6    Latitude        0.0171
## 10 r3_5    Longitude       0.0859
## # . . . with 407 more rows
```

Value

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

The C5.0 method has columns `trial`, `rule_num`, `rule`, and `statistics`. The latter two are nested tibbles. `statistic` has statistics about the data selected by the rules.

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