Package 'probably'

August 29, 2022

Title Tools for Post-Processing Class Probability Estimates

Version 0.1.0

Description Models can be improved by post-processing class probabilities, by: recalibration, conversion to hard probabilities, assessment of equivocal zones, and other activities. 'probably' contains tools for conducting these operations.

License MIT + file LICENSE

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

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

Depends R (>= 3.4.0)

Imports dplyr (>= 1.0.9), generics (>= 0.1.3), magrittr (>= 2.0.3), rlang (>= 1.0.4), tidyselect (>= 1.1.2), vctrs (>= 0.4.1), yardstick (>= 1.0.0)

Suggests covr, ggplot2, knitr, modeldata, parsnip, rmarkdown, rsample, testthat (>= 3.0.0)

VignetteBuilder knitr

ByteCompile true

Config/Needs/website tidyverse/tidytemplate

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

NeedsCompilation no

Author Max Kuhn [aut, cre], Davis Vaughan [aut], RStudio [cph, fnd]

Maintainer Max Kuhn <max@rstudio.com>

Repository CRAN

Date/Publication 2022-08-29 16:10:02 UTC

R topics documented:

`append_class_pred` *Add a* `class_pred` *column*

Description

This function is similar to `make_class_pred()`, but is useful when you have a large number of class probability columns and want to use `tidyselect` helpers. It appends the new `class_pred` vector as a column on the original data frame.

Usage

```
append_class_pred(
  .data,
  ...,
  levels,
  ordered = FALSE,
  min_prob = 1/length(levels),
  name = ".class_pred"
)
```

Arguments

`.data`	A data frame or tibble.
`...`	One or more unquoted expressions separated by commas to capture the columns of `.data` containing the class probabilities. You can treat variable names like they are positions, so you can use expressions like `x:y` to select ranges of variables or use selector functions to choose which columns. For `make_class_pred`, the columns for all class probabilities should be selected (in the same order as the `levels` object). For `two_class_pred`, a vector of class probabilities should be selected.
`levels`	A character vector of class levels. The length should be the same as the number of selections made through `...`, or length 2 for `make_two_class_pred()`.

ordered	A single logical to determine if the levels should be regarded as ordered (in the order given). This results in a class_pred object that is flagged as ordered.
min_prob	A single numeric value. If any probabilities are less than this value (by row), the row is marked as *equivocal*.
name	A single character value for the name of the appended class_pred column.

Value

.data with an extra class_pred column appended onto it.

Examples

```
# The following two examples are equivalent and demonstrate
# the helper, append_class_pred()

library(dplyr)

species_probs %>%
  mutate(
    .class_pred = make_class_pred(
      .pred_bobcat, .pred_coyote, .pred_gray_fox,
      levels = levels(Species),
      min_prob = .5
    )
  )

lvls <- levels(species_probs$Species)

append_class_pred(
  .data = species_probs,
  contains(".pred_"),
  levels = lvls,
  min_prob = .5
)
```

---

as_class_pred *Coerce to a* class_pred *object*

Description

as_class_pred() provides coercion to class_pred from other existing objects.

Usage

```
as_class_pred(x, which = integer(), equivocal = "[EQ]")
```

Arguments

x	A factor or ordered factor.
which	An integer vector specifying the locations of x to declare as equivocal.
equivocal	A single character specifying the equivocal label used when printing.

Examples

```
x <- factor(c("Yes", "No", "Yes", "Yes"))
as_class_pred(x)
```

---

class_pred *Create a class prediction object*

Description

class_pred() creates a class_pred object from a factor or ordered factor. You can optionally specify values of the factor to be set as *equivocal*.

Usage

```
class_pred(x = factor(), which = integer(), equivocal = "[EQ]")
```

Arguments

x	A factor or ordered factor.
which	An integer vector specifying the locations of x to declare as equivocal.
equivocal	A single character specifying the equivocal label used when printing.

Details

Equivocal values are those that you feel unsure about, and would like to exclude from performance calculations or other metrics.

Examples

```
x <- factor(c("Yes", "No", "Yes", "Yes"))

# Create a class_pred object from a factor
class_pred(x)

# Say you aren't sure about that 2nd "Yes" value. You could mark it as
# equivocal.
class_pred(x, which = 3)
```

```
# Maybe you want a different equivocal label
class_pred(x, which = 3, equivocal = "eq_value")
```

`is_class_pred` *Test if an object inherits from* `class_pred`

Description

`is_class_pred()` checks if an object is a `class_pred` object.

Usage

```
is_class_pred(x)
```

Arguments

`x`	An object.

Examples

```
x <- class_pred(factor(1:5))

is_class_pred(x)
```

`levels.class_pred` *Extract* `class_pred` *levels*

Description

The levels of a `class_pred` object do *not* include the equivocal value.

Usage

```
## S3 method for class 'class_pred'
levels(x)
```

Arguments

`x`	A `class_pred` object.

Examples

```
x <- class_pred(factor(1:5), which = 1)

# notice that even though `1` is not in the `class_pred` vector, the
# level remains from the original factor
levels(x)
```

`locate-equivocal` *Locate equivocal values*

Description

These functions provide multiple methods of checking for equivocal values, and finding their locations.

Usage

```
is_equivocal(x)

which_equivocal(x)

any_equivocal(x)
```

Arguments

`x`	A `class_pred` object.

Value

`is_equivocal()` returns a logical vector the same length as `x` where `TRUE` means the value is equivocal.

`which_equivocal()` returns an integer vector specifying the locations of the equivocal values.

`any_equivocal()` returns `TRUE` if there are any equivocal values.

Examples

```
x <- class_pred(factor(1:10), which = c(2, 5))

is_equivocal(x)

which_equivocal(x)

any_equivocal(x)
```

make_class_pred *Create a* class_pred *vector from class probabilities*

Description

These functions can be used to convert class probability estimates to class_pred objects with an optional equivocal zone.

Usage

```
make_class_pred(..., levels, ordered = FALSE, min_prob = 1/length(levels))

make_two_class_pred(
  estimate,
  levels,
  threshold = 0.5,
  ordered = FALSE,
  buffer = NULL
)
```

Arguments

...	Numeric vectors corresponding to class probabilities. There should be one for each level in levels, and *it is assumed that the vectors are in the same order as* levels.
levels	A character vector of class levels. The length should be the same as the number of selections made through ..., or length 2 for make_two_class_pred().
ordered	A single logical to determine if the levels should be regarded as ordered (in the order given). This results in a class_pred object that is flagged as ordered.
min_prob	A single numeric value. If any probabilities are less than this value (by row), the row is marked as *equivocal*.
estimate	A single numeric vector corresponding to the class probabilities of the first level in levels.
threshold	A single numeric value for the threshold to call a row to be labeled as the first value of levels.
buffer	A numeric vector of length 1 or 2 for the buffer around threshold that defines the equivocal zone (i.e., threshold - buffer[1] to threshold + buffer[2]). A length 1 vector is recycled to length 2. The default, NULL, is interpreted as no equivocal zone.

Value

A vector of class class_pred.

Examples

```
library(dplyr)

good <- segment_logistic$.pred_good
lvls <- levels(segment_logistic$Class)

# Equivocal zone of .5 +/- .15
make_two_class_pred(good, lvls, buffer = 0.15)

# Equivocal zone of c(.5 - .05, .5 + .15)
make_two_class_pred(good, lvls, buffer = c(0.05, 0.15))

# These functions are useful alongside dplyr::mutate()
segment_logistic %>%
  mutate(
    .class_pred = make_two_class_pred(
      estimate = .pred_good,
      levels = levels(Class),
      buffer = 0.15
    )
  )

# Multi-class example
# Note that we provide class probability columns in the same
# order as the levels
species_probs %>%
  mutate(
    .class_pred = make_class_pred(
      .pred_bobcat, .pred_coyote, .pred_gray_fox,
      levels = levels(Species),
      min_prob = .5
    )
  )
```

---

reportable_rate *Calculate the reportable rate*

Description

The *reportable rate* is defined as the percentage of class predictions that are *not* equivocal.

Usage

```
reportable_rate(x)
```

Arguments

x	A class_pred object.

Details

The reportable rate is calculated as `(n_not_equivocal / n)`.

Examples

```
x <- class_pred(factor(1:5), which = c(1, 2))

# 3 / 5
reportable_rate(x)
```

`segment_naive_bayes` *Image segmentation predictions*

Description

Image segmentation predictions

Details

These objects contain test set predictions for the cell segmentation data from Hill, LaPan, Li and Haney (2007). Each data frame are the results from different models (naive Bayes and logistic regression).

Value

`segment_naive_bayes`, `segment_logistic`
a tibble

Source

Hill, LaPan, Li and Haney (2007). Impact of image segmentation on high-content screening data quality for SK-BR-3 cells, *BMC Bioinformatics*, Vol. 8, pg. 340, https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-8-340.

Examples

```
data(segment_naive_bayes)
data(segment_logistic)
```

species_probs *Predictions on animal species*

Description

Predictions on animal species

Details

These data are holdout predictions from resampling for the animal scat data of Reid (2015) based on a C5.0 classification model.

Value

`species_probs` a tibble

Source

Reid, R. E. B. (2015). A morphometric modeling approach to distinguishing among bobcat, coyote and gray fox scats. *Wildlife Biology*, 21(5), 254-262

Examples

```
data(species_probs)
str(species_probs)
```

threshold_perf *Generate performance metrics across probability thresholds*

Description

`threshold_perf()` can take a set of class probability predictions and determine performance characteristics across different values of the probability threshold and any existing groups.

Usage

```
threshold_perf(.data, ...)

## S3 method for class 'data.frame'
threshold_perf(
  .data,
  truth,
  estimate,
  thresholds = NULL,
  na_rm = TRUE,
  event_level = "first",
  ...
)
```

Arguments

.data	A tibble, potentially grouped.
...	Currently unused.
truth	The column identifier for the true two-class results (that is a factor). This should be an unquoted column name.
estimate	The column identifier for the predicted class probabilities (that is a numeric). This should be an unquoted column name.
thresholds	A numeric vector of values for the probability threshold. If unspecified, a series of values between 0.5 and 1.0 are used. **Note**: if this argument is used, it must be named.
na_rm	A single logical: should missing data be removed?
event_level	A single string. Either "first" or "second" to specify which level of truth to consider as the "event".

Details

Note that that the global option yardstick.event_first will be used to determine which level is the event of interest. For more details, see the Relevant level section of yardstick::sens().

The currently calculated metrics are:

- yardstick::j_index()
- yardstick::sens()
- yardstick::spec()
- distance = (1 - sens) ^ 2 + (1 - spec) ^ 2

Value

A tibble with columns: .threshold, .estimator, .metric, .estimate and any existing groups.

Examples

```
library(dplyr)
data("segment_logistic")

# Set the threshold to 0.6
# > 0.6 = good
# < 0.6 = poor
threshold_perf(segment_logistic, Class, .pred_good, thresholds = 0.6)

# Set the threshold to multiple values
thresholds <- seq(0.5, 0.9, by = 0.1)

segment_logistic %>%
  threshold_perf(Class, .pred_good, thresholds)

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

```
# It works with grouped data frames as well
# Let's mock some resampled data
resamples <- 5

mock_resamples <- resamples %>%
  replicate(
    expr = sample_n(segment_logistic, 100, replace = TRUE),
    simplify = FALSE
  ) %>%
  bind_rows(.id = "resample")

resampled_threshold_perf <- mock_resamples %>%
  group_by(resample) %>%
  threshold_perf(Class, .pred_good, thresholds)

resampled_threshold_perf

# Average over the resamples
resampled_threshold_perf %>%
  group_by(.metric, .threshold) %>%
  summarise(.estimate = mean(.estimate))
```

Index