Package: optbinningR (via r-universe)

October 19, 2026

Type Package

Title Optimal Binning Methods for Predictive Modeling and Analytics

Version 0.2.1

Description Native R tools for optimal binning workflows in predictive modeling. The package provides APIs for binary, multi-class and continuous targets, with multi-variable binning and scorecard workflows. Methods are informed by Navas-Palencia (2020) <doi:10.48550/arXiv.2001.08025> and Navas-Palencia (2021) <doi:10.48550/arXiv.2104.08619>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

Imports stats, utils

Suggests testthat (>= 3.0.0), jsonlite, lintr, covr

Config/testthat/edition 3

URL https://github.com/s-rani1/optbinningR

BugReports https://github.com/s-rani1/optbinningR/issues

Repository https://s-rani1.r-universe.dev

Date/Publication 2026-03-18 03:21:56 UTC

RemoteUrl https://github.com/s-rani1/optbinningr

RemoteRef HEAD

RemoteSha 9d2bc777a630b31e1d3ed66f2813e63518413d0b

Contents

binning_table 2
BinningProcess 3
fit 4
OptimalBinning 6
OptimalBinning2D 7
run_fico_tutorial 8

Index **10**

`binning_table` *Binning Table Handle and Build API*

Description

Instantiate a binning table handle and build Python-style binning tables.

Usage

```
binning_table(object)

build(object, ...)
```

Arguments

`object`	A fitted supported model object (for `binning_table`) or a `BinningTable` object (for `build`).
`...`	Reserved for compatibility.

Value

`binning_table()` returns a `BinningTable` object. `build()` returns a data frame with Python-style columns, row labels and totals row.

Examples

```
x <- c(1, 2, 3, 4, 5, 6)
y <- c(0, 0, 0, 1, 1, 1)

m <- fit(OptimalBinning("x"), x, y, algorithm = "optimal", max_n_bins = 3)
bt <- binning_table(m)
build(bt)
```

`BinningProcess` *Binning Process and Scorecard APIs*

Description

Construct and operate multivariable binning processes and scorecard models, including update and monitoring utilities.

Usage

```
BinningProcess(variable_names, target_dtype = "binary", binning_fit_params = list())

Scorecard(binning_process, pdo = 20, odds = 50, base_points = 600)

update_binning_process(object, x, y, variables = NULL, ...)

predict_proba(object, x)

predict_score(object, x)

scorecard_monitoring(object, x_ref, x_new, y_ref = NULL, y_new = NULL, n_bins = 10L)

counterfactual_scorecard(
  object,
  x_one,
  target,
  direction = "auto",
  max_changes = 2L,
  max_combinations = 50000L
)
```

Arguments

`variable_names`	Character vector of variable names.
`target_dtype`	Target type, e.g. `"binary"` or `"continuous"`.
`binning_fit_params`	Named list of per-variable fit parameter overrides.
`binning_process`	A fitted or unfitted `BinningProcess` object.
`pdo`, `odds`, `base_points`	Score scaling parameters.
`object`	A fitted object.
`x`	Input data frame.
`y`	Target vector.
`variables`	Optional subset of variables to update.

x_ref, x_new	Reference and new/current population predictor data frames.
y_ref, y_new	Optional reference and new/current target vectors.
n_bins	Number of score buckets used for PSI.
x_one	One-row input data frame for counterfactual search.
target	Desired target prediction for counterfactual search.
direction	One of "auto", ">=", or "<=".
max_changes	Maximum number of feature changes in counterfactual search.
max_combinations	Maximum combinations evaluated for counterfactual search.
...	Additional arguments.

Value

Object or summary output depending on function.

Examples

```
set.seed(1)
df <- data.frame(x1 = rnorm(200), x2 = runif(200))
y <- rbinom(200, 1, plogis(df$x1 - df$x2))

bp <- BinningProcess(c("x1", "x2"), target_dtype = "binary")
bp <- fit(bp, df, y, algorithm = "optimal", max_n_bins = 5)

sc <- Scorecard(bp)
sc <- fit(sc, df, y, algorithm = "optimal", max_n_bins = 5)

predict_proba(sc, df[1:5, ])
predict_score(sc, df[1:5, ])
```

---

fit *Fit, Transform, and Inspect Binning Models*

Description

Generic APIs to fit models, transform variables, and inspect fitted binning objects.

Usage

```
fit(object, ...)

fit_transform(
  object,
  x,
  y,
  max_n_bins = 10L,
```

```
  algorithm = "quantile",
  prebinning_method = "cart",
  max_n_prebins = 20L,
  min_bin_size = 0.05,
  min_bin_n_event = 1L,
  min_bin_n_nonevent = 1L,
  monotonic_trend = "none",
  special_codes = NULL,
  solver = "native",
  profile = "standard"
)

transform(object, x, metric = NULL, ...)

partial_fit(object, x, y)

information(object, print_level = 1L)

analysis(object)
```

Arguments

`object`	A model object.
`x`	Predictor vector or data structure accepted by the model.
`y`	Target vector.
`max_n_bins`	Maximum final bins.
`algorithm`	Binning algorithm.
`prebinning_method`	Prebinning method for optimal mode.
`max_n_prebins`	Maximum pre-bins for the `"optimal"` algorithm.
`min_bin_size`	Minimum bin size as count or fraction.
`min_bin_n_event`	Minimum number of events required per bin (native solver).
`min_bin_n_nonevent`	Minimum number of non-events required per bin (native solver).
`monotonic_trend`	Monotonic trend constraint.
`special_codes`	Optional values treated as `Special` bin.
`solver`	Solver backend for `algorithm = "optimal"`.
`profile`	Optimization profile (`"standard"` or `"large_scale"`).
`metric`	Transformation metric. Common values include `"woe"`, `"event_rate"`, and `"bins"`.
`print_level`	Verbosity level for `information()`.
`...`	Additional model-specific arguments.

Value

Depends on the method:

- `fit()`: fitted object.
- `fit_transform()`: list with fitted model and transformed values.
- `transform()`: transformed vector.
- `partial_fit()`: updated object.
- `information()`: summary list (invisibly).
- `analysis()`: diagnostics list.

Examples

```
x <- c(1, 2, 3, 4, 5, 6)
y <- c(0, 0, 0, 1, 1, 1)

m <- fit(OptimalBinning("x"), x, y, algorithm = "optimal", max_n_bins = 3)
transform(m, x)
transform(m, x, metric = "bins")
information(m)
analysis(m)
```

`OptimalBinning` *Create Optimal Binning Model Objects*

Description

Constructors for one-dimensional optimal binning model objects for binary, multiclass, and continuous targets.

Usage

```
OptimalBinning(name, dtype = "numerical")

MulticlassOptimalBinning(name, dtype = "numerical")

ContinuousOptimalBinning(name, dtype = "numerical")
```

Arguments

`name`	Feature name.
`dtype`	Feature type. Use `"numerical"` or `"categorical"` where supported.

Value

A model object to be used with `fit()`.

Examples

```
ob <- OptimalBinning("x")
mc <- MulticlassOptimalBinning("x")
ct <- ContinuousOptimalBinning("x")
```

`OptimalBinning2D` *Advanced Binning Model Constructors*

Description

Constructors for 2D, piecewise, sketch/streaming, and uncertainty-aware binning models.

Usage

```
OptimalBinning2D(name_x, name_y, target_dtype = "binary")

OptimalPWBinning(name, target_dtype = "binary", degree = 1L)

OptimalBinningSketch(name, sample_size = 5000L)

OptimalBinningUncertainty(name, uncertainty_strategy = "mean")
```

Arguments

`name_x`, `name_y`	Feature names for two-dimensional binning.
`name`	Feature name.
`target_dtype`	Target type accepted by the specific model.
`degree`	Polynomial degree for piecewise binning model.
`sample_size`	Reservoir sample size for sketch binning.
`uncertainty_strategy`	Imputation strategy for uncertain values: one of `"mean"`, `"median"`, or `"mode"`.

Value

A model object to be used with `fit()` or `partial_fit()`.

Examples

```
ob2 <- OptimalBinning2D("x1", "x2")
pw <- OptimalPWBinning("x")
sk <- OptimalBinningSketch("x")
un <- OptimalBinningUncertainty("x")
```

`run_fico_tutorial` *Tutorial Workflow Helpers*

Description

Run end-to-end workflow helpers inspired by FICO and Telco tutorials.

Usage

```
run_fico_tutorial(
  n_train = 2500L,
  n_update = 1200L,
  train_data = NULL,
  y_train = NULL,
  update_data = NULL,
  y_update = NULL,
  solver = "native",
  profile = "standard",
  prebinning_method = "quantile",
  max_n_bins = 6L,
  max_n_prebins = 20L
)

run_telco_tutorial(
  n = 3000L,
  train_data = NULL,
  y_train = NULL,
  test_data = NULL,
  y_test = NULL,
  solver = "native",
  profile = "standard",
  prebinning_method = "quantile",
  max_n_bins = 6L,
  max_n_prebins = 20L
)
```

Arguments

`n_train`, `n_update`	Sample sizes for synthetic generation when data is not supplied.
`n`	Total sample size used for synthetic Telco generation when data is not supplied.
`train_data`, `update_data`, `test_data`	Optional input data frames.
`y_train`, `y_update`, `y_test`	Optional target vectors.
`solver`	Solver backend selection.

`profile` Optimization profile.

`prebinning_method` Prebinning method.

`max_n_bins`, `max_n_prebins` Binning controls.

Value

A list with fitted artifacts and summary metrics.

Examples

```
res_f <- run_fico_tutorial(n_train = 300, n_update = 150)
res_t <- run_telco_tutorial(n = 600)
```

Index

analysis *(fit)*, 4

binning_table, 2
BinningProcess, 3
build *(binning_table)*, 2

ContinuousOptimalBinning
        (OptimalBinning), 6
counterfactual_scorecard
        (BinningProcess), 3

fit, 4
fit_transform *(fit)*, 4

information *(fit)*, 4

MulticlassOptimalBinning
        (OptimalBinning), 6

OptimalBinning, 6
OptimalBinning2D, 7
OptimalBinningSketch
        (OptimalBinning2D), 7
OptimalBinningUncertainty
        (OptimalBinning2D), 7
OptimalPWBinning *(OptimalBinning2D)*, 7

partial_fit *(fit)*, 4
predict_proba *(BinningProcess)*, 3
predict_score *(BinningProcess)*, 3

run_fico_tutorial, 8
run_telco_tutorial *(run_fico_tutorial)*,
        8

Scorecard *(BinningProcess)*, 3
scorecard_monitoring *(BinningProcess)*, 3

transform *(fit)*, 4

update_binning_process
        (BinningProcess), 3