Package 'Usmile'

August 8, 2026

Title Threshold-Free Class-Specific Comparison of Binary Classifiers

Version 0.2.0

Description Implements the U-smile methodology for threshold-free, class-specific comparison of probabilistic binary classifiers. The package quantifies prediction improvement and worsening separately for non-events and events using the Brier alteration (BA), relative Brier (RB), improvement proportion (I) coefficients, and relative likelihood ratio (rLR) coefficients, and provides U-smile, prediction improvement-worsening, receiver operating characteristic, and precision-recall plots. The original U-smile framework is described in Kubiak et al. (2024) <doi:10.1371/journal.pone.0303276>, its three-level extension for imbalanced binary classification in Wieckowska et al. (2025) <doi:10.1371/journal.pone.0321661>, and the likelihood-based extension in Wieckowska and Guzik (2026) <doi:10.1038/s41598-026-40545-z>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

LazyDataCompression xz

Depends R (>= 3.5.0)

Imports ggplot2, grid, pROC,

Suggests e1071, naivebayes, nnet, parsnip, randomForest, ranger, testthat (>= 3.0.0), xgboost

Config/testthat/edition 3

URL https://github.com/bbwieckowska/Usmile

BugReports https://github.com/bbwieckowska/Usmile/issues

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Barbara Więckowska [aut, cre]

Maintainer Barbara Więckowska <bbwieckowska@gmail.com>

Repository CRAN

Date/Publication 2026-08-08 12:00:11 UTC

Contents

`CLBplot` *Calibration Plot with Brier Score Comparison*

Description

Creates a calibration plot comparing predicted and empirical event probabilities for a reference model and a new model. The function also calculates Brier score measures for the two models.

Usage

```
CLBplot(data_ref, data_new, title = "Calibration Plot", n_bins = 10)
```

Arguments

`data_ref`	A list returned by `USprep_mdl()` for the reference model. It must contain the elements `y`, `p`, and `n_vars`.
`data_new`	A list returned by `USprep_mdl()` for the new model. It must contain the elements `y`, `p`, and `n_vars`.
`title`	A single character value specifying the plot title. The default is `"Calibration Plot"`.
`n_bins`	A single integer specifying the number of probability intervals used to calculate empirical event probabilities. The default is 10.

Details

Predicted probabilities are divided into equal-width intervals between 0 and 1. Empty intervals are omitted from the calibration curve.

A positive value of `Delta_Brier` indicates that the new model has a lower Brier score than the reference model.

Value

A list containing:

- `plot`: A `ggplot` object showing the calibration curves.
- `results_table`: A data frame containing:
  - `Brier_ref`: Brier score for the reference model.
  - `Brier_new`: Brier score for the new model.
  - `Delta_Brier`: Difference calculated as the reference-model Brier score minus the new-model Brier score.
  - `BSS`: Brier Skill Score calculated relative to the reference model.

Examples

```
data(heart_disease_train)

model_glm_ref <- stats::glm(
  disease ~ age + sex + bp + chol,
  data = heart_disease_train,
  family = stats::binomial()
)

model_glm_new <- stats::glm(
  disease ~ age + sex + bp + chol + cp,
  data = heart_disease_train,
  family = stats::binomial()
)

output_ref <- USprep_mdl(
  model = model_glm_ref,
  dataset = NULL,
  testing = FALSE
)

output_new <- USprep_mdl(
  model = model_glm_new,
  dataset = NULL,
  testing = FALSE
)

calibration_results <- CLBplot(
  data_ref = output_ref,
  data_new = output_new,
  title = "Calibration Plot"
```

```
)

calibration_results$plot
calibration_results$results_table
```

`heart_disease` *Heart Disease Data with Generated Validation Variables*

Description

A processed heart disease dataset derived from data available through the UCI Machine Learning Repository. The dataset combines clinical variables with synthetically generated variables intended for evaluating binary probabilistic classification methods.

A processed clinical dataset (from UCI Machine Learning Repository) with synthetically generated variables for evaluating binary classification methods. Combines real cardiac data with controlled random variables to test model robustness. Contains 661 complete cases (347 healthy, 314 with coronary artery disease).

Usage

```
heart_disease

heart_disease
```

Format

A data frame with 661 rows and 56 columns, including:

disease Coronary artery disease status. A factor coded as `0` for less than 50 percent stenosis and `1` for greater than 50 percent stenosis.

location Data source. A factor with the levels `"cl"` for Cleveland, `"hu"` for Hungary, `"sw"` for Switzerland, and `"va"` for the Veterans Administration data.

age Age in years.

sex Sex coded as `0` for female and `1` for male.

cp Chest pain type. A factor coded as `1` for typical angina, `2` for atypical angina, `3` for non-anginal pain, and `4` for asymptomatic chest pain.

bp Resting systolic blood pressure in mmHg.

chol Serum cholesterol in mg/dL.

glu Indicator of fasting blood glucose greater than 120 mg/dL, coded as `1` for yes and `0` for no.

ecg Resting electrocardiogram result. A factor coded as `0` for normal, `1` for ST-T wave abnormality, and `2` for left ventricular hypertrophy.

hr Maximum heart rate achieved, in beats per minute.

exang Exercise-induced angina, coded as `1` for yes and `0` for no.

stde Exercise-induced ST depression in mm.

rnd_normal Non-stratified random variable generated from a standard normal distribution.

rnd_uniform Non-stratified random variable generated from a uniform distribution on the interval from 0 to 10.

rnd_exp Non-stratified random variable generated from an exponential distribution with rate 1.

rnd_bernoulli Non-stratified Bernoulli random variable with success probability 0.8.

rnd_binomial Non-stratified binomial random variable with size 6 and success probability 0.8.

rnd_poisson Non-stratified Poisson random variable with rate 1.

strat_rnd_normal Class-specific normal random variable generated from a normal distribution with mean 10 and standard deviation 2 for non-events, and mean 12 and standard deviation 2 for events.

strat_rnd_uniform Class-specific uniform random variable generated on the interval from 0 to 6 for non-events and from 2 to 8 for events.

strat_rnd_exp Class-specific exponential random variable generated with rate 0.5 for non-events and rate 1 for events.

strat_rnd_bernoulli Class-specific Bernoulli random variable generated with success probability 0.5 for non-events and 0.2 for events.

strat_rnd_binomial Class-specific binomial random variable generated with size 7 and success probability 0.6 for non-events, and size 7 and success probability 0.5 for events.

strat_rnd_poisson Class-specific Poisson random variable generated with rate 1 for non-events and 1.6 for events.

hlt_slight_asym Asymmetric class-specific variable generated from a normal distribution with mean 1 and standard deviation 2 for non-events, and mean 0 and standard deviation 1 for events.

ill_slight_asym Asymmetric class-specific variable generated from a normal distribution with mean 0 and standard deviation 1 for non-events, and mean 1 and standard deviation 2 for events.

hlt_high_asym Asymmetric class-specific variable generated from a normal distribution with mean 1 and standard deviation 4 for non-events, and mean 0 and standard deviation 1 for events.

ill_high_asym Asymmetric class-specific variable generated from a normal distribution with mean 0 and standard deviation 1 for non-events, and mean 1 and standard deviation 4 for events.

rnd_normal0_1 Normal random variable constructed to have correlation 0.1 with age.

rnd_normal0_2 Normal random variable constructed to have correlation 0.2 with age.

rnd_normal0_3 Normal random variable constructed to have correlation 0.3 with age.

rnd_normal0_4 Normal random variable constructed to have correlation 0.4 with age.

rnd_normal0_5 Normal random variable constructed to have correlation 0.5 with age.

rnd_normal0_6 Normal random variable constructed to have correlation 0.6 with age.

rnd_normal0_7 Normal random variable constructed to have correlation 0.7 with age.

rnd_normal0_8 Normal random variable constructed to have correlation 0.8 with age.

rnd_normal0_9 Normal random variable constructed to have correlation 0.9 with age.

strat_rnd_normal0_1 Class-specific normal random variable constructed to have correlation 0.1 with age.

strat_rnd_normal0_2 Class-specific normal random variable constructed to have correlation 0.2 with age.

strat_rnd_normal0_3 Class-specific normal random variable constructed to have correlation 0.3 with age.

strat_rnd_normal0_4 Class-specific normal random variable constructed to have correlation 0.4 with age.

strat_rnd_normal0_5 Class-specific normal random variable constructed to have correlation 0.5 with age.

strat_rnd_normal0_6 Class-specific normal random variable constructed to have correlation 0.6 with age.

strat_rnd_normal0_7 Class-specific normal random variable constructed to have correlation 0.7 with age.

strat_rnd_normal0_8 Class-specific normal random variable constructed to have correlation 0.8 with age.

strat_rnd_normal0_9 Class-specific normal random variable constructed to have correlation 0.9 with age.

A data frame with 661 rows, 56 columns, and the following variables:

disease Coronary artery disease status (factor: 0 = <50% stenosis, 1 = >50% stenosis)

location Data source (factor: 'cl' (Cleveland), 'hu' (Hungarian), 'sw' (Switzerland), 'va' (VA))

age Age in years (numeric)

sex Sex (0 = female, 1 = male)

cp Chest pain type (factor: 1 = typical angina, 2 = atypical angina, 3 = non-anginal pain, 4 = asymptomatic)

bp Resting systolic blood pressure (mmHg)

chol Serum cholesterol (mg/dl)

glu Fasting blood sugar >120 mg/dl (1 = yes, 0 = no)

ecg Resting ECG results (factor: 0 = normal, 1 = ST-T abnormality, 2 = LV hypertrophy)

hr Maximum heart rate achieved (bpm)

exang Exercise-induced angina (1 = yes, 0 = no)

stde Exercise-induced ST depression (mm)

rnd_normal Non-stratified random variable: N(0,1)

rnd_uniform Non-stratified random variable: U(0,10)

rnd_exp Non-stratified random variable: Exp(1)

rnd_bernoulli Non-stratified random variable: Bernoulli(0.8)

rnd_binomial Non-stratified random variable: Binomial(6,0.8)

rnd_poisson Non-stratified random variable: Poisson(1)

strat_rnd_normal Stratified random variable: N(10,2) for controls | N(12,2) for cases

strat_rnd_uniform Stratified random variable: U(0,6) | U(2,8)

strat_rnd_exp Stratified random variable: Exp(0.5) | Exp(1)

strat_rnd_bernoulli Stratified random variable: Bern(0.5) | Bern(0.2)

strat_rnd_binomial Stratified random variable: Binom(7,0.6) | Binom(7,0.5)

strat_rnd_poisson Stratified random variable: Pois(1) | Pois(1.6)

hlt_slight_asym Asymmetric stratified variable: N(1,2) for controls | N(0,1) for cases

ill_slight_asym Asymmetric stratified variable: N(0,1) for controls | N(1,2) for cases

hlt_high_asym Asymmetric stratified variable: N(1,4) for controls | N(0,1) for cases

ill_high_asym Asymmetric stratified variable: N(0,1) for controls | N(1,4) for cases

rnd_normal0_1 Age-correlated variable: N(0,1) with r=0.1 to age

rnd_normal0_2 Age-correlated variable: N(0,1) with r=0.2 to age

rnd_normal0_3 Age-correlated variable: N(0,1) with r=0.3 to age

rnd_normal0_4 Age-correlated variable: N(0,1) with r=0.4 to age

rnd_normal0_5 Age-correlated variable: N(0,1) with r=0.5 to age

rnd_normal0_6 Age-correlated variable: N(0,1) with r=0.6 to age

rnd_normal0_7 Age-correlated variable: N(0,1) with r=0.7 to age

rnd_normal0_8 Age-correlated variable: N(0,1) with r=0.8 to age

rnd_normal0_9 Age-correlated variable: N(0,1) with r=0.9 to age

strat_rnd_normal0_1 Stratified age-correlated variable: N(10,2.5)|N(11,2.5) with r=0.1 to age

strat_rnd_normal0_2 Stratified age-correlated variable: N(10,2.5)|N(11,2.5) with r=0.2 to age

strat_rnd_normal0_3 Stratified age-correlated variable: N(10,2.5)|N(11,2.5) with r=0.3 to age

strat_rnd_normal0_4 Stratified age-correlated variable: N(10,2.5)|N(11,2.5) with r=0.4 to age

strat_rnd_normal0_5 Stratified age-correlated variable: N(10,2.5)|N(11,2.5) with r=0.5 to age

strat_rnd_normal0_6 Stratified age-correlated variable: N(10,2.5)|N(11,2.5) with r=0.6 to age

strat_rnd_normal0_7 Stratified age-correlated variable: N(10,2.5)|N(11,2.5) with r=0.7 to age

strat_rnd_normal0_8 Stratified age-correlated variable: N(10,2.5)|N(11,2.5) with r=0.8 to age

strat_rnd_normal0_9 Stratified age-correlated variable: N(10,2.5)|N(11,2.5) with r=0.9 to age

Details

The dataset contains 661 complete observations, including 347 non-events and 314 events.

Data processing

The clinical observations were combined from the Cleveland, Hungarian, Switzerland, and Veterans Administration heart disease datasets. Observations containing missing values or zero values for blood pressure or cholesterol were removed. Additional synthetic variables were generated for method validation. The variables `disease`, `cp`, and `ecg` were converted to factors.

Data Processing

- Combined datasets from 4 sources (Cleveland, Hungarian, Swiss, VA)
- Removed cases with missing values or biologically implausible measurements (BP/chol = 0)
- Generated variables using `faux::rnorm_pre()` for correlated variables
- Categorical variables (`disease`, `cp`, `ecg`) converted to factors with reference levels set.

Source

UCI Machine Learning Repository, Heart Disease dataset: https://archive.ics.uci.edu/dataset/45/heart+disease

Clinical data from UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/Heart+Disease

Examples

```
data(heart_disease)

dim(heart_disease)
table(heart_disease$disease)
utils::str(heart_disease[, c("disease", "age", "sex", "chol")])

boxplot(
  strat_rnd_normal ~ disease,
  data = heart_disease,
  xlab = "Disease status",
  ylab = "Class-specific random variable"
)
data(heart_disease)
# Check the structure of the dataset
str(heart_disease)
# Compare distributions of a stratified variable
boxplot(strat_rnd_normal ~ disease, data = heart_disease)
```

---

heart_disease_test *Stratified Test Subset of the Heart Disease Data*

Description

A test subset derived from heart_disease. It is complementary to heart_disease_train, and the two subsets together contain all observations from the complete dataset.

A balanced stratified subset of the complete heart_disease dataset, intended for model testing and validation. Contains 330 complete cases with approximately equal distribution of disease cases. This set is complementary to heart_disease_train and together they form the complete dataset. All variables, both real and generated, are identical to those described in heart_disease.

Usage

```
heart_disease_test

heart_disease_test
```

Format

A data frame with 330 rows and 56 columns.

A data frame with 330 rows and 56 columns. The format and variables are identical to heart_disease.

Details

All variables are described in the documentation for heart_disease.

See Also

heart_disease and heart_disease_train.

heart_disease for the full dataset and detailed variable descriptions.

heart_disease_train for the complementary balanced training set.

Examples

```
data(heart_disease_train)
data(heart_disease_test)

model <- stats::glm(
  disease ~ age + chol,
  data = heart_disease_train,
  family = stats::binomial()
)

probabilities <- stats::predict(
  model,
  newdata = heart_disease_test,
  type = "response"
)

head(probabilities)
data(heart_disease_test)
data(heart_disease_train)
# Train a model on the training set and predict on the test set
model <- glm(disease ~ age + chol, data = heart_disease_train, family = "binomial")
predictions <- predict(model, newdata = heart_disease_test, type = "response")
```

---

heart_disease_test_imbalanced_10

Test Subset Complementary to the 10 Percent Training Subset

Description

The observations from heart_disease that were not included in heart_disease_train_imbalanced_10.

Test set containing the remaining cases after creating the 10% imbalanced training set. Reflects the natural distribution of the original dataset. Intended for validation of models trained on severely imbalanced data.

Usage

```
heart_disease_test_imbalanced_10

heart_disease_test_imbalanced_10
```

Format

A data frame with 330 rows and 56 columns. Variables are described in heart_disease.

A data frame with 330 rows and 56 columns. The format and variables are identical to heart_disease.

Details

The event proportion in this test subset results from the construction of its complementary training subset and should not be assumed to equal 10 percent.

See Also

heart_disease_train_imbalanced_10.

heart_disease_train_imbalanced_10 for the corresponding training set.

Examples

```
data(heart_disease_test_imbalanced_10)

dim(heart_disease_test_imbalanced_10)
prop.table(table(heart_disease_test_imbalanced_10$disease))
```

---

heart_disease_test_imbalanced_30

Test Subset Complementary to the 30 Percent Training Subset

Description

The observations from heart_disease that were not included in heart_disease_train_imbalanced_30.

Test set containing the remaining cases after creating the 30% imbalanced training set. Reflects the natural distribution of the original dataset. Intended for validation of models trained on imbalanced data.

Usage

```
heart_disease_test_imbalanced_30

heart_disease_test_imbalanced_30
```

Format

A data frame with 330 rows and 56 columns. Variables are described in heart_disease.

A data frame with 330 rows and 56 columns. The format and variables are identical to heart_disease.

Details

The event proportion in this test subset results from the construction of its complementary training subset and should not be assumed to equal 30 percent.

See Also

heart_disease_train_imbalanced_30.

heart_disease_train_imbalanced_30 for the corresponding training set.

Examples

```
data(heart_disease_test_imbalanced_30)

dim(heart_disease_test_imbalanced_30)
prop.table(table(heart_disease_test_imbalanced_30$disease))
```

---

heart_disease_test_reduced_10

Test Subset with Approximately 10 Percent Events

Description

A reduced test subset derived by subsampling observations to obtain an event proportion of approximately 10 percent.

A modified test set with controlled class distribution (10% disease cases). Created by subsampling from the original test set to maintain specific prevalence. Useful for evaluating model performance under low prevalence scenarios.

Usage

```
heart_disease_test_reduced_10

heart_disease_test_reduced_10
```

Format

A data frame with 56 columns and a construction-dependent number of rows. Variables are described in heart_disease.

A data frame with variable rows (depending on available cases) and 56 columns. The format and variables are identical to heart_disease.

Details

The exact number of rows depends on the construction procedure and the available numbers of events and non-events.

See Also

heart_disease_test_reduced_30.

heart_disease_test_reduced_30 for 30% prevalence version.

Examples

```
data(heart_disease_test_reduced_10)

dim(heart_disease_test_reduced_10)
prop.table(table(heart_disease_test_reduced_10$disease))
```

`heart_disease_test_reduced_30`

Test Subset with Approximately 30 Percent Events

Description

A reduced test subset derived by subsampling observations to obtain an event proportion of approximately 30 percent.

A modified test set with controlled class distribution (30% disease cases). Created by subsampling from the original test set to maintain specific prevalence. Useful for evaluating model performance under specific clinical prevalence scenarios.

Usage

```
heart_disease_test_reduced_30

heart_disease_test_reduced_30
```

Format

A data frame with 56 columns and a construction-dependent number of rows. Variables are described in `heart_disease`.

A data frame with variable rows (depending on available cases) and 56 columns. The format and variables are identical to `heart_disease`.

Details

The exact number of rows depends on the construction procedure and the available numbers of events and non-events.

See Also

`heart_disease_test_reduced_10`.

`heart_disease_test_reduced_10` for 10% prevalence version.

Examples

```
data(heart_disease_test_reduced_30)

dim(heart_disease_test_reduced_30)
prop.table(table(heart_disease_test_reduced_30$disease))
```

heart_disease_train *Stratified Training Subset of the Heart Disease Data*

Description

A training subset derived from heart_disease. The subset contains 331 observations and retains approximately similar numbers of non-events and events.

A balanced stratified subset of the complete heart_disease dataset, intended for model training. Contains 331 complete cases with approximately equal distribution of disease cases. All variables, both real and generated, are identical to those described in heart_disease.

Usage

```
heart_disease_train

heart_disease_train
```

Format

A data frame with 331 rows and 56 columns.

A data frame with 331 rows and 56 columns. The format and variables are identical to heart_disease.

Details

All variables are described in the documentation for heart_disease.

See Also

heart_disease and heart_disease_test.

heart_disease for the full dataset and detailed variable descriptions.

heart_disease_test for the complementary balanced test set.

Examples

```
data(heart_disease_train)

dim(heart_disease_train)
table(heart_disease_train$disease)

model <- stats::glm(
  disease ~ age + chol,
  data = heart_disease_train,
  family = stats::binomial()
)

stats::coef(model)
data(heart_disease_train)
```

```
# Train a model on the training set
model <- glm(disease ~ age + chol, data = heart_disease_train, family = "binomial")
summary(model)
```

`heart_disease_train_imbalanced_10`
Training Subset with Approximately 10 Percent Events

Description

A training subset derived from `heart_disease` with severe artificial class imbalance. It contains 331 observations, including approximately 10 percent events and 90 percent non-events.

A training set with severe class imbalance (10% disease cases, 90% healthy controls). Intended for testing classification methods under challenging imbalanced conditions. Contains 331 complete cases (approximately 33 disease, 298 healthy).

Usage

```
heart_disease_train_imbalanced_10

heart_disease_train_imbalanced_10
```

Format

A data frame with 331 rows and 56 columns. Variables are described in `heart_disease`.

A data frame with 331 rows and 56 columns. The format and variables are identical to `heart_disease`.

See Also

`heart_disease_test_imbalanced_10` and `heart_disease_train_imbalanced_30`.

`heart_disease_train_imbalanced_30` for moderate imbalance.

`heart_disease_test_imbalanced_10` for the corresponding test set.

Examples

```
data(heart_disease_train_imbalanced_10)

table(heart_disease_train_imbalanced_10$disease)
prop.table(table(heart_disease_train_imbalanced_10$disease))
data(heart_disease_train_imbalanced_10)
# Check severe class imbalance
prop.table(table(heart_disease_train_imbalanced_10$disease))
```

heart_disease_train_imbalanced_30 *Training Subset with Approximately 30 Percent Events*

Description

A training subset derived from heart_disease with an artificially reduced event proportion. It contains 331 observations, including approximately 30 percent events and 70 percent non-events.

A training set with artificially induced class imbalance (30% disease cases, 70% healthy controls). Intended for testing classification methods under realistic imbalanced conditions. Contains 331 complete cases (approximately 99 disease, 232 healthy).

Usage

```
heart_disease_train_imbalanced_30

heart_disease_train_imbalanced_30
```

Format

A data frame with 331 rows and 56 columns. Variables are described in heart_disease.

A data frame with 331 rows and 56 columns. The format and variables are identical to heart_disease.

Details

The subset is intended for evaluating probabilistic binary classifiers under moderate class imbalance.

See Also

heart_disease_test_imbalanced_30 and heart_disease_train_imbalanced_10.

heart_disease_train_imbalanced_10 for more extreme imbalance.

heart_disease_test_imbalanced_30 for the corresponding test set.

Examples

```
data(heart_disease_train_imbalanced_30)

table(heart_disease_train_imbalanced_30$disease)
prop.table(table(heart_disease_train_imbalanced_30$disease))
data(heart_disease_train_imbalanced_30)
# Check class distribution
table(heart_disease_train_imbalanced_30$disease)
```

PIWplot *Create a Prediction Improvement/Worsening Plot*

Description

Creates a scatter plot comparing predicted probabilities from a reference model and a new model. Observations are classified according to whether the new model improves or worsens prediction for events and non-events.

Creates a scatter plot comparing predicted probabilities from a reference model and a new model. Observations are classified according to outcome class and whether the new-model prediction is better, worse, or unchanged relative to the reference-model prediction.

Usage

```
PIWplot(data_ref, data_new, title = "PIW Plot")

PIWplot(data_ref, data_new, title = "PIW Plot")
```

Arguments

data_ref	A list returned by USprep_mdl() for the reference model. It must contain the elements y, p, and n_vars.
data_new	A list returned by USprep_mdl() for the new model. It must contain the elements y, p, and n_vars.
title	A single character value specifying the plot title. The default is "PIW Plot".

Details

The function does not fit or apply a probability calibrator. If calibrated probabilities are required, supply outputs produced by USprep_mdl() with a previously fitted "usmile_calibrator" object.

Observations with unchanged probabilities are shown in grey and are not assigned to an improvement or worsening subclass.

For non-events, a lower predicted probability from the new model is interpreted as an improvement. For events, a higher predicted probability from the new model is interpreted as an improvement.

Observations for which both models return the same probability are labelled as unchanged predictions.

Value

A ggplot object showing new-model probabilities against reference-model probabilities. The diagonal line represents unchanged predictions.

A ggplot object showing new-model probabilities against reference-model probabilities. The diagonal line represents identical predictions from the two models.

See Also

USprep_mdl, USfit_calibrator, USapply_calibrator

Examples

```
data(heart_disease_train)
data(heart_disease_test)

model_ref <- stats::glm(
  disease ~ age + sex + bp + chol,
  data = heart_disease_train,
  family = stats::binomial()
)

model_new <- stats::glm(
  disease ~ age + sex + bp + chol + cp,
  data = heart_disease_train,
  family = stats::binomial()
)

output_ref <- USprep_mdl(
  model = model_ref,
  dataset = heart_disease_test,
  testing = TRUE
)

output_new <- USprep_mdl(
  model = model_new,
  dataset = heart_disease_test,
  testing = TRUE
)

PIWplot(
  data_ref = output_ref,
  data_new = output_new,
  title = "Prediction changes on test data"
)

data(heart_disease_train)

model_glm_ref <- stats::glm(
  disease ~ age + sex + bp + chol,
  data = heart_disease_train,
  family = stats::binomial()
)

model_glm_new <- stats::glm(
  disease ~ age + sex + bp + chol + cp,
  data = heart_disease_train,
  family = stats::binomial()
)
```

```
output_ref <- USprep_mdl(
  model = model_glm_ref,
  dataset = NULL,
  testing = FALSE
)

output_new <- USprep_mdl(
  model = model_glm_new,
  dataset = NULL,
  testing = FALSE
)

PIWplot(
  data_ref = output_ref,
  data_new = output_new,
  title = "Prediction Improvement and Worsening"
)
```

`ROCplot` *Receiver Operating Characteristic Curve with AUC Comparison*

Description

Creates receiver operating characteristic curves for a reference model and a new model. The function calculates the area under each ROC curve and compares the paired curves using DeLong's test.

Usage

```
ROCplot(data_ref, data_new, title = "ROC Plot", alternative = "two.sided")
```

Arguments

`data_ref`	A list returned by `USprep_mdl()` for the reference model. It must contain the elements `y`, `p`, and `n_vars`.
`data_new`	A list returned by `USprep_mdl()` for the new model. It must contain the elements `y`, `p`, and `n_vars`.
`title`	A single character value specifying the plot title. The default is `"ROC Plot"`.
`alternative`	A character value specifying the alternative hypothesis used in DeLong's test. One of `"two.sided"`, `"greater"`, or `"less"`. The default is `"two.sided"`.

Details

The outcome must be coded as 0 for non-events and 1 for events. Predicted probabilities must represent the probability of the event class.

The ROC direction is fixed so that larger predicted probabilities indicate a greater probability of the event.

For a one-sided DeLong test, the interpretation of `alternative` follows the ordering used by `pROC::roc.test()`: the reference-model ROC curve is supplied as the first curve and the new-model ROC curve as the second curve.

Value

A list containing:

- `plot`: A `ggplot` object showing ROC curves for the reference and new models.
- `results_table`: A data frame containing:
  - `AUC_ref`: AUC for the reference model.
  - `AUC_p-value_ref`: Two-sided p-value for testing whether the reference-model AUC differs from 0.5.
  - `AUC_new`: AUC for the new model.
  - `AUC_p-value_new`: Two-sided p-value for testing whether the new-model AUC differs from 0.5.
  - `AUC_diff_p_value`: P-value from the paired DeLong test comparing the two ROC curves.

Examples

```
data(heart_disease_train)

model_glm_ref <- stats::glm(
  disease ~ age + sex + bp + chol,
  data = heart_disease_train,
  family = stats::binomial()
)

model_glm_new <- stats::glm(
  disease ~ age + sex + bp + chol + cp,
  data = heart_disease_train,
  family = stats::binomial()
)

output_ref <- USprep_mdl(
  model = model_glm_ref,
  dataset = NULL,
  testing = FALSE
)

output_new <- USprep_mdl(
  model = model_glm_new,
  dataset = NULL,
  testing = FALSE
)

roc_results <- ROCplot(
  data_ref = output_ref,
  data_new = output_new,
  title = "ROC Curves"
)
```

```
roc_results$plot
roc_results$results_table
```

`USapply_calibrator` *Apply a Fitted Probability Calibrator*

Description

Applies a calibrator fitted by `USfit_calibrator()` to a new vector of predicted event probabilities.

Usage

```
USapply_calibrator(object, p)
```

Arguments

`object`	An object of class `"usmile_calibrator"` returned by `USfit_calibrator()`.
`p`	A numeric vector of predicted probabilities for the event class coded as 1.

Details

This function does not use observed outcomes and does not refit the calibrator. It is therefore suitable for applying a previously fitted calibrator to an independent test dataset.

Value

A numeric vector of calibrated probabilities with the same length as `p`.

See Also

`USfit_calibrator`

Examples

```
y_calibration <- c(0, 0, 0, 1, 0, 1, 1, 1)
p_calibration <- c(0.05, 0.20, 0.35, 0.40, 0.55, 0.60, 0.75, 0.90)

calibrator <- USfit_calibrator(
  y = y_calibration,
  p = p_calibration,
  method = "logistic"
)

p_test <- c(0.10, 0.30, 0.50, 0.70, 0.95)

calibrated_p_test <- USapply_calibrator(
  object = calibrator,
```

```
  p = p_test
)

calibrated_p_test
```

USbind_out *Combine Outputs from Two Models for Comparison*

Description

Combines outputs produced by USprep_mdl() for a reference model and a new model. The function verifies that both outputs describe the same observations and prepares a data frame for U-smile calculations.

Usage

```
USbind_out(model_out_ref, model_out_new)
```

Arguments

model_out_ref	A list returned by USprep_mdl() for the reference model.
model_out_new	A list returned by USprep_mdl() for the new model.

Value

A list containing:

- comparison_df: A data frame containing observed outcomes (y), reference-model probabilities (p_ref), and new-model probabilities (p).
- n_vars_diff: The absolute difference in the numbers of model variables or parameters reported by USprep_mdl().
- ref_vars: Variable names for the reference model.
- new_vars: Variable names for the new model.

Examples

```
data(heart_disease_train)

model_glm_ref <- stats::glm(
  disease ~ age + sex + bp + chol,
  data = heart_disease_train,
  family = stats::binomial()
)

model_glm_new <- stats::glm(
  disease ~ age + sex + bp + chol + cp,
  data = heart_disease_train,
```

```
  family = stats::binomial()
)

output_ref <- USprep_mdl(
  model_glm_ref,
  dataset = NULL,
  testing = FALSE
)

output_new <- USprep_mdl(
  model_glm_new,
  dataset = NULL,
  testing = FALSE
)

combined <- USbind_out(output_ref, output_new)

head(combined$comparison_df)
combined$n_vars_diff
combined$ref_vars
combined$new_vars
```

`UScalc_mdl` *Calculate U-smile Coefficients from Fitted Models*

Description

Calculates U-smile (Universal Smile Layout for Explanation) coefficients for the comparison of two fitted probabilistic binary classification models.

Usage

```
UScalc_mdl(
  ref_model,
  new_model,
  y_coef,
  dataset = NULL,
  testing = FALSE,
  ref_calibrator = NULL,
  new_calibrator = NULL
)
```

Arguments

`ref_model`	A fitted probabilistic binary classification model used as the reference model.
`new_model`	A fitted probabilistic binary classification model to be compared with `ref_model`.
`y_coef`	A character value specifying the coefficient to calculate. One of:

- "rLR": Relative Likelihood Ratio.
- "BA": Brier Alteration.
- "RB": Relative Brier.

`dataset` An optional data frame used to obtain observed outcomes and generate predictions.

If `testing = TRUE`, an independent test dataset must be supplied.

If `testing = FALSE`, `dataset` may be `NULL` when both fitted models retain sufficient training outcomes and predictions. For model classes that do not retain their training data, the appropriate dataset must be supplied explicitly.

`testing` A logical value indicating whether predictions should be generated for an independent test dataset. The default is `FALSE`.

`ref_calibrator` An optional object of class "usmile_calibrator" returned by `USfit_calibrator()`. It is applied only to the reference-model probabilities. The default is `NULL`.

`new_calibrator` An optional object of class "usmile_calibrator" returned by `USfit_calibrator()`. It is applied only to the new-model probabilities. The default is `NULL`.

Details

The function prepares observed outcomes and predicted probabilities using `USprep_mdl()`, combines the two model outputs using `USbind_out()`, and calculates U-smile coefficients using `UScalc_raw()`.

Previously fitted probability calibrators may be applied independently to the reference-model and new-model predictions.

By default, no calibration is applied. A calibrator is used only when it is supplied explicitly through `ref_calibrator` or `new_calibrator`.

A supplied calibrator is not refitted by this function. For independent model evaluation, each calibrator should be fitted using a separate calibration sample or out-of-fold predictions generated from the training data.

When both models are calibrated, their calibrators should normally be fitted using the same calibration observations and the same calibration strategy. Each model should nevertheless have its own fitted calibrator.

Calibration may alter predicted probabilities, U-smile subgroup assignments, and all coefficient values calculated from those probabilities.

Value

A list containing:

- `results`: A data frame containing detailed U-smile coefficients, sample sizes, and statistical test results.
- `plot_data`: A list containing coefficients and supporting values required by `USplot()`.

See Also

`USprep_mdl`, `USfit_calibrator`, `USapply_calibrator`, `USbind_out`, `UScalc_raw`

Examples

```
data(heart_disease_train)
data(heart_disease_test)

# Comparison without probability calibration
model_glm_ref <- stats::glm(
  disease ~ age + sex + bp + chol,
  data = heart_disease_train,
  family = stats::binomial()
)

model_glm_new <- stats::glm(
  disease ~ age + sex + bp + chol + cp,
  data = heart_disease_train,
  family = stats::binomial()
)

results_train <- UScalc_mdl(
  ref_model = model_glm_ref,
  new_model = model_glm_new,
  y_coef = "rLR",
  dataset = NULL,
  testing = FALSE
)

results_test <- UScalc_mdl(
  ref_model = model_glm_ref,
  new_model = model_glm_new,
  y_coef = "rLR",
  dataset = heart_disease_test,
  testing = TRUE
)

head(results_train$results)
results_test$plot_data$netto_Y_nonev
results_test$plot_data$netto_Y_event

# Comparison after calibration fitted on an independent subset
non_event_rows <- which(
  heart_disease_train$disease == "0"
)

event_rows <- which(
  heart_disease_train$disease == "1"
)

model_rows <- c(
  non_event_rows[seq_len(100)],
  event_rows[seq_len(100)]
)

model_data <- heart_disease_train[
```

```
  model_rows,
  ,
  drop = FALSE
]

calibration_data <- heart_disease_train[
  -model_rows,
  ,
  drop = FALSE
]

model_glm_ref_cal <- stats::glm(
  disease ~ age + sex + bp + chol,
  data = model_data,
  family = stats::binomial()
)

model_glm_new_cal <- stats::glm(
  disease ~ age + sex + bp + chol + cp,
  data = model_data,
  family = stats::binomial()
)

ref_calibration_output <- USprep_mdl(
  model = model_glm_ref_cal,
  dataset = calibration_data,
  testing = TRUE
)

new_calibration_output <- USprep_mdl(
  model = model_glm_new_cal,
  dataset = calibration_data,
  testing = TRUE
)

ref_calibrator <- USfit_calibrator(
  y = ref_calibration_output$y,
  p = ref_calibration_output$p,
  method = "logistic"
)

new_calibrator <- USfit_calibrator(
  y = new_calibration_output$y,
  p = new_calibration_output$p,
  method = "logistic"
)

calibrated_results <- UScalc_mdl(
  ref_model = model_glm_ref_cal,
  new_model = model_glm_new_cal,
  y_coef = "rLR",
  dataset = heart_disease_test,
  testing = TRUE,
```

```
  ref_calibrator = ref_calibrator,
  new_calibrator = new_calibrator
)

head(calibrated_results$results)
calibrated_results$plot_data$netto_Y_nonev
calibrated_results$plot_data$netto_Y_event
```

`UScalc_raw` *Calculate U-smile Coefficients from Raw Predictions*

Description

Calculates U-smile (Universal Smile Layout for Explanation) coefficients directly from observed binary outcomes and predicted probabilities from a reference model and a new model.

Usage

```
UScalc_raw(raw_data, y_coef, n_vars_diff, bootstrap_p = NULL)
```

Arguments

`raw_data`	A data frame containing: • `y`: Observed binary outcomes coded as 0 and 1. • `p_ref`: Predicted probabilities from the reference model. • `p`: Predicted probabilities from the new model.
`y_coef`	A character value specifying the coefficient to calculate. One of: • `"rLR"`: Relative Likelihood Ratio. • `"BA"`: Brier Alteration. • `"RB"`: Relative Brier.
`n_vars_diff`	A single non-negative numeric value representing the difference in the numbers of model variables or parameters. It is used as the number of degrees of freedom in statistical calculations.
`bootstrap_p`	An optional list containing the elements `p_Y_nonev` and `p_Y_event`. Each element must be a single probability between 0 and 1.

Value

A list containing the calculated U-smile coefficients and data required by `USplot()`. The returned structure includes:

- subgroup-specific Y coefficients;
- subgroup-specific I coefficients;
- class-specific and overall net coefficients;
- sample sizes;
- statistical test results;
- data used to construct the U-smile plot.

Examples

```
data(heart_disease_train)

model_glm_ref <- stats::glm(
  disease ~ age + sex + bp + chol,
  data = heart_disease_train,
  family = stats::binomial()
)

model_glm_new <- stats::glm(
  disease ~ age + sex + bp + chol + cp,
  data = heart_disease_train,
  family = stats::binomial()
)

output_ref <- USprep_mdl(
  model = model_glm_ref,
  dataset = NULL,
  testing = FALSE
)

output_new <- USprep_mdl(
  model = model_glm_new,
  dataset = NULL,
  testing = FALSE
)

combined <- USbind_out(output_ref, output_new)

results_rLR <- UScalc_raw(
  raw_data = combined$comparison_df,
  y_coef = "rLR",
  n_vars_diff = combined$n_vars_diff
)

results_BA <- UScalc_raw(
  raw_data = combined$comparison_df,
  y_coef = "BA",
  n_vars_diff = combined$n_vars_diff
)

results_RB <- UScalc_raw(
  raw_data = combined$comparison_df,
  y_coef = "RB",
  n_vars_diff = combined$n_vars_diff
)

results_rLR$netto_Y_nonev
results_rLR$netto_Y_event
```

`USfit_calibrator` *Fit a Probability Calibrator*

Description

Fits a calibration model using observed binary outcomes and predicted event probabilities. The fitted calibrator can subsequently be applied to independent predictions using `USapply_calibrator()`.

Usage

```
USfit_calibrator(
  y,
  p,
  method = c("logistic", "intercept", "isotonic"),
  eps = 1e-06
)
```

Arguments

`y`	A vector of observed binary outcomes coded as 0 and 1.
`p`	A numeric vector of predicted probabilities for the event class coded as 1.
`method`	A character value specifying the calibration method. One of: • `"logistic"`: Logistic recalibration using the logit of the predicted probability as a predictor. • `"intercept"`: Recalibration of the calibration intercept only, with the original logit used as an offset. • `"isotonic"`: Non-decreasing isotonic regression.
`eps`	A single numeric value used to restrict probabilities before applying the logit transformation. It must be greater than 0 and less than 0.5. The default is `1e-6`.

Details

The calibration data should be independent of the data used for final model evaluation. They may come from a separate calibration dataset or from out-of-fold predictions generated within the training data.

Logistic recalibration fits the model:

$$\text{logit}\{\Pr(Y = 1)\} = \alpha + \beta \text{logit}(p).$$

Intercept-only recalibration fixes the coefficient of `logit(p)` at 1 and estimates only the calibration intercept.

Isotonic calibration fits a non-decreasing step function. Predictions below or above the range observed in the calibration data are assigned the first or last fitted isotonic value, respectively.

Value

An object of class "usmile_calibrator" containing the calibration method, fitted calibration model, probability restriction value, and calibration-sample metadata.

See Also

USapply_calibrator

Examples

```
y_calibration <- c(0, 0, 0, 1, 0, 1, 1, 1)
p_calibration <- c(0.05, 0.20, 0.35, 0.40, 0.55, 0.60, 0.75, 0.90)

logistic_calibrator <- USfit_calibrator(
  y = y_calibration,
  p = p_calibration,
  method = "logistic"
)

intercept_calibrator <- USfit_calibrator(
  y = y_calibration,
  p = p_calibration,
  method = "intercept"
)

isotonic_calibrator <- USfit_calibrator(
  y = y_calibration,
  p = p_calibration,
  method = "isotonic"
)

logistic_calibrator
```

USplot *Create a U-smile Plot*

Description

Creates a U-smile (Universal Smile Layout for Explanation) plot for class-specific comparison of two probabilistic binary classifiers.

Usage

```
USplot(
  plot_data = NULL,
  y_coef,
  raw_data = NULL,
  ref_formula = NULL,
```

```
  new_formula = NULL,
  ref_model_type = "glm",
  new_model_type = "glm",
  train_data = NULL,
  test_data = NULL,
  testing = FALSE,
  ref_calibrator = NULL,
  new_calibrator = NULL,
  n_vars_diff = NULL,
  circle_sizes = TRUE,
  y_lim = NULL,
  net = FALSE,
  crit = 0
)
```

Arguments

plot_data	An optional list containing pre-calculated U-smile coefficients. This may be the plot_data element returned by UScalc_mdl() or the object returned directly by UScalc_raw().
y_coef	A character value specifying the coefficient displayed on the vertical axis. One of "rLR", "BA", or "RB".
raw_data	An optional data frame containing the columns y, p_ref, and p. It is used when plot_data is not supplied.
ref_formula	An optional formula used to fit the reference model.
new_formula	An optional formula used to fit the new model.
ref_model_type	A character value specifying the reference-model type. One of "glm" or "randomForest".
new_model_type	A character value specifying the new-model type. One of "glm" or "randomForest".
train_data	An optional data frame used to fit models when formulas are supplied.
test_data	An optional independent test data frame used for model evaluation when testing = TRUE.
testing	A logical value indicating whether fitted models should be evaluated using test_data. The default is FALSE.
ref_calibrator	An optional object of class "usmile_calibrator" returned by USfit_calibrator(). It is applied only to the reference-model probabilities and is not refitted. The default is NULL.
new_calibrator	An optional object of class "usmile_calibrator" returned by USfit_calibrator(). It is applied only to the new-model probabilities and is not refitted. The default is NULL.
n_vars_diff	A single non-negative numeric value specifying the difference in the numbers of model variables or parameters. It is required when raw_data is supplied.
circle_sizes	A logical value indicating whether point sizes should represent subgroup proportions. The default is TRUE.
y_lim	An optional finite numeric vector of length two specifying the vertical-axis limits.

net	A logical value indicating whether class-specific net coefficients should be displayed. The default is FALSE.
crit	A numeric value controlling the connecting-line style. Use 0 for solid grey lines or 2 for class-specific significance-based line types when y_coef = "rLR".

Details

The plot can be generated from pre-calculated U-smile coefficients, raw prediction data, or model formulas and training data.

Previously fitted probability calibrators may optionally be applied to predictions from the reference and new models.

The function supports three mutually exclusive input workflows:

- supplying previously calculated plot_data;
- supplying raw_data and n_vars_diff;
- supplying model formulas and training data.

By default, no probability calibration is applied. A calibrator is used only when it is supplied explicitly through ref_calibrator or new_calibrator.

A supplied calibrator must be fitted before calling this function, using an independent calibration sample or out-of-fold predictions from the training data. The calibrator is only applied to generated probabilities; it is never refitted inside USplot().

Calibration can change predicted probabilities, subgroup assignments, U-smile coefficients, and displayed point sizes.

The rLR coefficient is displayed as a numeric value rather than as a percentage. For example, an rLR value of 0.25 is displayed as 0.25, not as 25%.

Value

A ggplot object containing the U-smile visualization.

See Also

USfit_calibrator, USapply_calibrator, USprep_mdl, UScalc_mdl, UScalc_raw

Examples

```
data(heart_disease_train)

model_glm_ref <- stats::glm(
  disease ~ age + sex + bp + chol,
  data = heart_disease_train,
  family = stats::binomial()
)

model_glm_new <- stats::glm(
  disease ~ age + sex + bp + chol + cp,
  data = heart_disease_train,
```

```
  family = stats::binomial()
)

results <- UScalc_mdl(
  ref_model = model_glm_ref,
  new_model = model_glm_new,
  y_coef = "rLR"
)

USplot(
  plot_data = results$plot_data,
  y_coef = "rLR",
  net = TRUE,
  crit = 2
)
```

USprep_mdl *Prepare Model Output for U-smile Evaluation*

Description

Extracts observed binary outcomes, predicted event probabilities, and model-variable information from a supported probabilistic binary classification model.

Usage

```
USprep_mdl(model, dataset = NULL, testing = FALSE, calibrator = NULL)
```

Arguments

model	A fitted probabilistic binary classification model. Supported classes include glm, randomForest, ranger, svm, xgb.Booster, nnet, naive_bayes, and model_fit objects from parsnip.
dataset	An optional data frame used to obtain observed outcomes and generate predictions. If testing = TRUE, an independent test dataset must be supplied. If testing = FALSE, dataset may be omitted only when the fitted model retains sufficient training outcomes and predictions.
testing	A logical value indicating whether predictions should be generated for an independent test dataset. The default is FALSE.
calibrator	An optional object of class "usmile_calibrator" returned by USfit_calibrator(). The calibrator is applied to the model probabilities but is not refitted. The default is NULL.

Details

Optionally, a previously fitted probability calibrator can be applied to the extracted probabilities. The calibrator must be fitted independently using `USfit_calibrator()`.

For models returning a probability matrix, the function selects the column named `"1"`. This column is interpreted as the probability of the event class.

The outcome must be binary and coded as 0 and 1, where 1 represents the event.

A supplied calibrator is only applied to already generated probabilities. No calibration parameters are estimated by this function. For independent evaluation, the calibrator should be fitted using a separate calibration dataset or out-of-fold predictions generated from the training data.

Some supported model classes require optional packages. These packages are checked only when the corresponding model class is used.

Value

A list containing:

- `y`: A numeric vector of observed outcomes coded as 0 and 1.
- `p`: A numeric vector of predicted probabilities for the event class coded as 1. If `calibrator` is supplied, these are calibrated probabilities.
- `n_vars`: The number of model variables or parameters extracted from the fitted model.
- `var_names`: Names of the extracted model variables or parameters.

See Also

`USfit_calibrator`, `USapply_calibrator`, `USbind_out`

Examples

```
data(heart_disease_train)
data(heart_disease_test)

# Basic workflow without probability calibration
model_glm <- stats::glm(
  disease ~ age + sex + bp + chol,
  data = heart_disease_train,
  family = stats::binomial()
)

training_output <- USprep_mdl(
  model = model_glm,
  dataset = NULL,
  testing = FALSE
)

test_output <- USprep_mdl(
  model = model_glm,
  dataset = heart_disease_test,
  testing = TRUE
```

```
)

head(training_output$p)
head(test_output$p)
training_output$var_names

# Workflow with an independent calibration subset
non_event_rows <- which(
  heart_disease_train$disease == "0"
)

event_rows <- which(
  heart_disease_train$disease == "1"
)

model_rows <- c(
  non_event_rows[seq_len(100)],
  event_rows[seq_len(100)]
)

model_data <- heart_disease_train[
  model_rows,
  ,
  drop = FALSE
]

calibration_data <- heart_disease_train[
  -model_rows,
  ,
  drop = FALSE
]

model_glm_calibrated <- stats::glm(
  disease ~ age + sex + bp + chol,
  data = model_data,
  family = stats::binomial()
)

calibration_output <- USprep_mdl(
  model = model_glm_calibrated,
  dataset = calibration_data,
  testing = TRUE
)

calibrator <- USfit_calibrator(
  y = calibration_output$y,
  p = calibration_output$p,
  method = "logistic"
)

calibrated_test_output <- USprep_mdl(
  model = model_glm_calibrated,
  dataset = heart_disease_test,
```

```
  testing = TRUE,
  calibrator = calibrator
)

head(calibrated_test_output$p)

range(calibrated_test_output$p)
```

Index