Package ‘Usmile’

September 28, 2025

Title U-Smile Method

Version 0.0.1

Description Implements the U-smile method for predictive models evaluation.
Computes performance metrics (BA/BR/I/rLR) and generates U-smile plots.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

Imports caret,
earth,
ggplot2,
pROC,
PRROC,
randomForest,
Rcpp (>= 1.0.7)

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

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

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

Contents

`CLBplot` *Calibration Plot with Brier Score Comparison*

Description

Creates a calibration plot comparing predicted vs. empirical probabilities for two models, along with a table of Brier score metrics. The plot shows how well predicted probabilities match observed event rates across probability bins.

Usage

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

Arguments

`data_ref`	List containing reference model output from `USprep_mdl` with elements: • y - vector of observed values (0/1) • p - vector of predicted probabilities • n_vars - number of predictor variables in the model
`data_new`	List containing new model output from `USprep_mdl` (same structure as data_ref)
`title`	Plot title (default: "Calibration Plot")
`n_bins`	Number of bins for calibration calculation (default: 10)

Value

A list containing:

- plot - ggplot object showing calibration curves for both models
- results_table - data frame with Brier score metrics:
  - Brier_ref - Brier score for reference model
  - Brier_new - Brier score for new model
  - Delta_Brier - Improvement in Brier score (reference - new)
  - BSS - Brier Skill Score (relative improvement)

Examples

```
## Not run:
# Load Heart Disease datasets
data(heart_disease)
data(heart_disease_train)
data(heart_disease_test)

# Prepare model outputs
model_glm_ref <- glm(disease ~ age + sex + bp + chol, data = train_data, family = "binomial")
model_glm_new <- glm(disease ~ age + sex + bp + chol + cp, data = train_data, family = "binomial")

train_out_ref <- USprep_mdl(model_glm_ref, dataset = NULL, testing = FALSE)
train_out_new <- USprep_mdl(model_glm_new, dataset = NULL, testing = FALSE)

# Create calibration plot for training data
calibration_results <- CLBplot(
  data_ref = train_out_ref,
  data_new = train_out_new,
  title = "Calibration: Basic vs Extended Cardiac Risk Model"
)

# Display plot and metrics
calibration_results$plot
calibration_results$results_table

# Example with test data and custom bins
test_out_ref <- USprep_mdl(model_glm_ref, dataset = test_data, testing = TRUE)
test_out_new <- USprep_mdl(model_glm_new, dataset = test_data, testing = TRUE)

CLBplot(
  data_ref = test_out_ref,
  data_new = test_out_new,
  n_bins = 5,
  title = "Test Data Calibration (5 bins)"
)

## End(Not run)
```

`heart_disease` *Complete Heart Disease Dataset with Generated Variables for Method Validation*

Description

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
```

Format

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

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

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

Examples

```
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` *Balanced Test Set for the Heart Disease Dataset*

Description

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
```

Format

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

See Also

heart_disease for the full dataset and detailed variable descriptions.

heart_disease_train for the complementary balanced training set.

Examples

```
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 *Complementary Test Set for 10% Imbalanced Training*

Description

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
```

Format

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

See Also

heart_disease_train_imbalanced_10 for the corresponding training set.

---

heart_disease_test_imbalanced_30 *Complementary Test Set for 30% Imbalanced Training*

Description

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
```

Format

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

See Also

`heart_disease_train_imbalanced_30` for the corresponding training set.

`heart_disease_test_reduced_10`
Reduced Test Set with 10% Disease Prevalence

Description

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
```

Format

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

See Also

`heart_disease_test_reduced_30` for 30% prevalence version.

`heart_disease_test_reduced_30`
Reduced Test Set with 30% Disease Prevalence

Description

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
```

Format

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

See Also

`heart_disease_test_reduced_10` for 10% prevalence version.

heart_disease_train *Balanced Training Set for the Heart Disease Dataset*

Description

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
```

Format

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

See Also

heart_disease for the full dataset and detailed variable descriptions.

heart_disease_test for the complementary balanced test set.

Examples

```
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 *Imbalanced Training Set (10% Disease Cases)*

Description

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
```

Format

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

See Also

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

`heart_disease_train_imbalanced_30` *Imbalanced Training Set (30% Disease Cases)*

Description

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
```

Format

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

See Also

`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)
# Check class distribution
table(heart_disease_train_imbalanced_30$disease)
```

`PIWplot` *Prediction Improvement/Worsening (PIW) Plot*

Description

Creates a scatter plot comparing predicted probabilities between a reference model and a new model, highlighting where predictions improved or worsened for events and non-events.

Usage

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

Arguments

`data_ref`	List containing reference model output from `USprep_mdl` with elements: • y - vector of observed values (0/1) • p - vector of predicted probabilities • n_vars - number of predictor variables in the model
`data_new`	List containing new model output from `USprep_mdl` (same structure as data_ref)
`title`	Plot title (default: "PIW Plot")

Value

A ggplot object showing:

- Points colored by prediction change direction (improvement/worsening) and event status
- Diagonal reference line (y = x) representing no change between models
- Consistent color scheme with USplot (blue for non-events, red for events)

Examples

```
## Not run:
# Load Heart Disease datasets
data(heart_disease)
data(heart_disease_train)
data(heart_disease_test)

# Prepare model outputs
model_glm_ref <- glm(disease ~ age + sex + bp + chol, data = train_data, family = "binomial")
model_glm_new <- glm(disease ~ age + sex + bp + chol + cp, data = train_data, family = "binomial")

train_out_ref <- USprep_mdl(model_glm_ref, dataset = NULL, testing = FALSE)
train_out_new <- USprep_mdl(model_glm_new, dataset = NULL, testing = FALSE)

# Create PIW plot for training data
PIWplot(data_ref = train_out_ref, data_new = train_out_new,
        title = "Prediction Changes: Adding CP to Cardiac Risk Model")

# Example with test data
test_out_ref <- USprep_mdl(model_glm_ref, dataset = test_data, testing = TRUE)
test_out_new <- USprep_mdl(model_glm_new, dataset = test_data, testing = TRUE)

PIWplot(data_ref = test_out_ref, data_new = test_out_new,
        title = "Test Data Prediction Changes")

## End(Not run)
```

`PRCplot` *Precision-Recall Curve Plot with AUPRC Comparison*

Description

Creates a precision-recall plot comparing two models and computes area under the PR curve (AUPRC) with statistical significance testing via bootstrap. Includes baseline prevalence reference.

Usage

```
PRCplot(
  data_ref,
  data_new,
  title = "Precision-Recall plot",
  n_boot = 1000,
  seed = 123
)
```

Arguments

`data_ref`	List containing reference model output from `USprep_mdl` with elements: • y - vector of observed values (0/1) • p - vector of predicted probabilities • n_vars - number of predictor variables in the model
`data_new`	List containing new model output from `USprep_mdl` (same structure as data_ref)
`title`	Plot title (default: "Precision-Recall plot")
`n_boot`	Number of bootstrap samples for significance testing (default: 1000)
`seed`	Random seed for reproducibility (default: 123)

Value

A list containing:

- plot - ggplot object showing PR curves for both models with baseline prevalence
- results_table - data frame with performance metrics:
  - AUPRC_ref - Area Under PR Curve for reference model
  - AUPRC_new - Area Under PR Curve for new model
  - AUPRC_diff_p_value - Bootstrap p-value for AUPRC difference

Examples

```
## Not run:
# Load Heart Disease datasets
data(heart_disease)
data(heart_disease_train)
data(heart_disease_test)

# Prepare model outputs
model_glm_ref <- glm(disease ~ age + sex + bp + chol, data = train_data, family = "binomial")
model_glm_new <- glm(disease ~ age + sex + bp + chol + cp, data = train_data, family = "binomial")

train_out_ref <- USprep_mdl(model_glm_ref, dataset = NULL, testing = FALSE)
train_out_new <- USprep_mdl(model_glm_new, dataset = NULL, testing = FALSE)

# Create PR curve plot with default settings
pr_results <- PRCplot(
  data_ref = train_out_ref,
  data_new = train_out_new,
  title = "PR Curve: Basic vs Extended Cardiac Model"
)
```

```
# Display plot and metrics
pr_results$plot
pr_results$results_table

# Example with test data and custom bootstrap samples
test_out_ref <- USprep_mdl(model_glm_ref, dataset = test_data, testing = TRUE)
test_out_new <- USprep_mdl(model_glm_new, dataset = test_data, testing = TRUE)

PRCplot(
  data_ref = test_out_ref,
  data_new = test_out_new,
  n_boot = 500,
  seed = 456,
  title = "Test Data PR Curves (500 bootstraps)"
)

## End(Not run)
```

ROCplot — *Receiver Operating Characteristic (ROC) Curve Plot with AUC Comparison*

Description

Creates an ROC plot comparing two models and computes area under the curve (AUC) with statistical significance testing using DeLong's method. Includes reference line for random classifier performance.

Usage

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

Arguments

`data_ref`	List containing reference model output from `USprep_mdl` with elements: • y - vector of observed values (0/1) • p - vector of predicted probabilities • n_vars - number of predictor variables in the model
`data_new`	List containing new model output from `USprep_mdl` (same structure as data_ref)
`title`	Plot title (default: "ROC plot")
`alternative`	Alternative hypothesis for DeLong's test ("two.sided", "greater", or "less") (default: "two.sided")

Value

A list containing:

- plot - ggplot object showing ROC curves with diagonal reference line
- results_table - data frame with performance metrics:
  - AUC_ref - AUC for reference model

- AUC_p-value_ref - p-value for reference model AUC vs random
- AUC_new - AUC for new model
- AUC_p-value_new - p-value for new model AUC vs random
- AUC_diff_p_value - p-value for AUC difference (DeLong's test)

Examples

```
## Not run:
# Load Heart Disease datasets
data(heart_disease)
data(heart_disease_train)
data(heart_disease_test)

# Prepare model outputs
model_glm_ref <- glm(disease ~ age + sex + bp + chol, data = train_data, family = "binomial")
model_glm_new <- glm(disease ~ age + sex + bp + chol + cp, data = train_data, family = "binomial")

train_out_ref <- USprep_mdl(model_glm_ref, dataset = NULL, testing = FALSE)
train_out_new <- USprep_mdl(model_glm_new, dataset = NULL, testing = FALSE)

# Create ROC plot with default two-sided test
roc_results <- ROCplot(
  data_ref = train_out_ref,
  data_new = train_out_new,
  title = "ROC: Basic vs Extended Cardiac Model"
)

# Display plot and metrics
roc_results$plot
roc_results$results_table

# Example with one-sided test (testing if new model is better)
ROCplot(
  data_ref = train_out_ref,
  data_new = train_out_new,
  alternative = "greater",
  title = "ROC Curve (New > Reference)"
)

# Example with test data
test_out_ref <- USprep_mdl(model_glm_ref, dataset = test_data, testing = TRUE)
test_out_new <- USprep_mdl(model_glm_new, dataset = test_data, testing = TRUE)

ROCplot(
  data_ref = test_out_ref,
  data_new = test_out_new,
  title = "Test Data ROC Curves"
)

## End(Not run)
```

`USbind_out` *Combine Outputs from Two Models for Comparison*

Description

This function combines the outputs from two model evaluations (from USprep_mdl) to prepare them for USMILE analysis. It checks for consistency between models and creates a data frame suitable for comparison.

Usage

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

Arguments

`model_out_ref`	Output from USprep_mdl for the reference model
`model_out_new`	Output from USprep_mdl for the new model to compare

Value

A list containing:

- comparison_df - Data frame with y, reference probabilities (p_ref), and new probabilities (p_new)
- n_vars_diff - Absolute difference in number of variables between models
- ref_vars - List of variables in the reference model
- new_vars - List of variables in the new model

Examples

```
## Not run:
# Load Heart Disease datasets
data(heart_disease)
data(heart_disease_train)
data(heart_disease_test)

### Example 1: Comparing nested logistic regression models
model_glm_ref <- glm(disease ~ age + sex + bp + chol, data = train_data, family = "binomial")
train_out_glm_ref <- USprep_mdl(model_glm_ref, dataset = NULL, testing = FALSE)
model_glm_new <- glm(disease ~ age + sex + bp + chol + cp, data = train_data, family = "binomial")
train_out_glm_new <- USprep_mdl(model_glm_new, dataset = NULL, testing = FALSE)

# Combine outputs for comparison
combined_glm <- USbind_out(train_out_glm_ref, train_out_glm_new)
head(combined_glm$comparison_df)
combined_glm$n_vars_diff  # Shows difference in number of parameters

### Example 2: Comparing logistic regression with Random Forest
model_rf_ref <- randomForest::randomForest(disease ~ age + sex + bp + chol, data = train_data)
train_out_rf_ref <- USprep_mdl(model_rf_ref, dataset = NULL, testing = FALSE)

# Combine with logistic regression output
combined_rf_vs_glm <- USbind_out(train_out_glm_ref, train_out_rf_ref)
head(combined_rf_vs_glm$comparison_df)

## End(Not run)
```

`UScalc_mdl` *Calculate U-smile coefficients for model comparison (based on models)*

Description

Computes USMILE (User-centric Statistical Measures for Interpretable Learning Explanations) coefficients for comparing two predictive models.

Usage

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

Arguments

`ref_model`	Reference model object (glm or randomForest)
`new_model`	New model object to compare (glm or randomForest)
`y_coef`	Type of coefficient to calculate: • "rLR" - Relative Likelihood Ratio • "BA" - Brier Alteration (average absolute change) • "RB" - Relative Brier (relative change)
`dataset`	#' @param dataset Dataset used to prepare data for the models. • If `testing` = `TRUE`, a test dataset must be provided (required). • If `testing` = `FALSE`, the argument can be: – `NULL` if the model stores its own training data (e.g., `glm`, `randomForest`), – the training dataset if the model does not keep training data internally (e.g., `ranger`, `e1071`, `xgboost`, `nnet`, `naive_bayes`, tidymodels). This allows the function to work consistently with both training and test data.
`testing`	Logical indicating whether to use test data (TRUE) or training data (FALSE)

Value

A list containing:

- results - Data frame with detailed comparison metrics
- plot_data - Data for visualization (used by USplot)

Examples

```
## Not run:
# Load Heart Disease datasets
data(heart_disease)
data(heart_disease_train)
data(heart_disease_test)

# Compare two logistic regression models
model_glm_ref <- glm(disease ~ age + sex + bp + chol, data = train_data, family = "binomial")
model_glm_new <- glm(disease ~ age + sex + bp + chol + cp, data = train_data, family = "binomial")
```

```
# Calculate rLR coefficients on training and test data
train_results_rLR <- UScalc_mdl(model_glm_ref, model_glm_new, y_coef = "rLR",
                         dataset = NULL, testing = FALSE)
test_results_rLR <- UScalc_mdl(model_glm_ref, model_glm_new, y_coef = "rLR",
                         dataset = test_data, testing = TRUE)

# Calculate BA coefficients on training data
train_results_BA <- UScalc_mdl(model_glm_ref, model_glm_new, y_coef = "BA",
                        dataset = NULL, testing = FALSE)
# Calculate RB coefficients on training data
train_results_RB <- UScalc_mdl(model_glm_ref, model_glm_new, y_coef = "RB",
                        dataset = NULL, testing = FALSE)

# Compare Random Forest to logistic regression
# (both models must be built based on the same variables)
model_fr_ref <- randomForest::randomForest(disease ~ age + sex + bp + chol, data = train_data)
train_results_rf_vs_glm <- UScalc_mdl(model_glm_ref, model_rf_ref, y_coef = "rLR",
                              dataset = NULL, testing = FALSE)

## End(Not run)
```

`UScalc_raw` *Calculate U-smile coefficients for model comparison (based on raw prediction data)*

Description

Computes U-smile (User-centric Statistical Measures for Interpretable Learning Explanations) coefficients directly from the raw prediction data of two models.

Usage

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

Arguments

`raw_data`	Data frame containing raw prediction data with columns: • y - binary outcome values (0/1) • p_ref - predicted probabilities from reference model • p - predicted probabilities from new model
`y_coef`	Type of coefficient to calculate: • "rLR" - Relative Likelihood Ratio • "BA" - Brier Alteration (average absolute change) • "RB" - Relative Brier (relative change)
`n_vars_diff`	Number of variables (degrees of freedom) for statistical tests
`bootstrap_p`	Optional list with bootstrap p-values (from USboot_LRp)

Value

A list containing:

- results - Data frame with detailed comparison metrics
- plot_data - Data for visualization (used by USplot)

Examples

```
## Not run:
# Load Heart Disease datasets
data(heart_disease)
data(heart_disease_train)
data(heart_disease_test)

# Prepare raw predictions from models
model_glm_ref <- glm(disease ~ age + sex + bp + chol, data = train_data, family = "binomial")
train_out_glm_ref <- USprep_mdl(model_glm_ref, dataset = NULL, testing = FALSE)
model_glm_new <- glm(disease ~ age + sex + bp + chol + cp, data = train_data, family = "binomial")
train_out_glm_new <- USprep_mdl(model_glm_new, dataset = NULL, testing = FALSE)

# Combine raw data for comparison
combined_glm <- USbind_out(train_out_glm_ref, train_out_glm_new)
raw_compare <- combined_glm$comparison_df
n_vars_diff <- combined_glm$n_vars_diff  # Shows difference in number of parameters


# Calculate rLR coefficients
results_rLR <- UScalc_raw(raw_compare, y_coef = "rLR", n_vars_diff)

# Calculate BA coefficients
results_BA <- UScalc_raw(raw_compare, y_coef = "BA", n_vars_diff)

# Calculate RB coefficients
results_RB <- UScalc_raw(raw_compare, y_coef = "RB", n_vars_diff)


## End(Not run)
```

USclbr_mdl — *Calibration of Random Forest predictive probabilities*

Description

This function calibrates predictive probabilities using either isotonic regression or logistic regression with cross-validation.

Usage

```
USclbr_mdl(y, p, method = "isotonic", n_folds = 10, random_seed = 123)
```

Arguments

y	Vector of observed binary values (0 or 1)
p	Vector of predicted probabilities (values between 0 and 1)
method	Calibration method ("isotonic" - default or "logistic")
n_folds	Number of folds for cross-validation (default: 10)
random_seed	Random seed for reproducibility (default: 123)

Value

Vector of calibrated probabilities with the same length as the input vector p

Examples

```
## Not run:
# Load Heart Disease datasets
data(heart_disease)
data(heart_disease_train)
data(heart_disease_test)

# Train Random Forest model
model_rf <- randomForest::randomForest(disease ~ age + sex, data = train_data)

# Get raw predictions
preds <- predict(model_rf, type = "prob")[, 2]

# Calibrate predictions using isotonic regression
calibrated_preds <- USclbr_mdl(
  y = as.numeric(train_data$disease) - 1,
  p = preds,
  method = "isotonic"
)

# Calibrate predictions using logistic regression with 5-fold CV
calibrated_preds_log <- USclbr_mdl(
  y = as.numeric(train_data$disease) - 1,
  p = preds,
  method = "logistic",
  n_folds = 5
)

## End(Not run)
```

USplot *U-Smile plot visualization*

Description

Creates a U-Smile plot showing the performance comparison of models. Models can be of different types (e.g., glm vs randomForest). The function offers multiple ways to generate the plot:

1. From pre-calculated plot data (output from UScalc_mdl or UScalc_raw)
2. From raw data (y, p_ref, p columns)
3. Directly from model specifications (formulas and model types)

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_calibrate = TRUE,
  new_calibrate = TRUE,
  calibration_method = "isotonic",
  n_vars_diff = NULL,
  circle_sizes = TRUE,
  y_lim = NULL,
  net = FALSE,
  crit = 0
)
```

Arguments

`plot_data`	List containing pre-calculated plot data (output from UScalc_mdl or UScalc_raw)
`y_coef`	Type of coefficient being plotted ("rLR", "BA", or "RB")
`raw_data`	Optional raw data frame containing columns: y, p_ref, p (alternative to plot_data)
`ref_formula`	Optional formula for reference model
`new_formula`	Optional formula for new model
`ref_model_type`	Type of reference model ("glm" or "randomForest")
`new_model_type`	Type of new model ("glm" or "randomForest")
`train_data`	Training dataset for model building
`test_data`	Optional test dataset for evaluation
`testing`	Logical indicating whether to evaluate on test data (TRUE) or training data (FALSE)
`ref_calibrate`	Whether to calibrate probabilities for reference model (randomForest only)
`new_calibrate`	Whether to calibrate probabilities for new model (randomForest only)
`calibration_method`	Calibration method ("isotonic" or "logistic")
`n_vars_diff`	Difference in number of variables (degrees of freedom) - required if raw_data is provided
`circle_sizes`	Logical indicating whether to scale point sizes by proportion (default: TRUE)
`y_lim`	Optional vector specifying y-axis limits (default: NULL for automatic)
`net`	Logical indicating whether to show net coefficients (default: FALSE)
`crit`	Criteria for line types (0: solid gray, 2: conditional dotted/solid for rLR)

Details

For model specifications, you can choose to evaluate on training or test data.

Value

A ggplot object showing the U-Smile plot with performance metrics

Examples

```
## Not run:
# Load Heart Disease datasets
data(heart_disease)
data(heart_disease_train)
data(heart_disease_test)

# Building models
model_glm_ref <- glm(disease ~ age + sex + bp + chol, data = train_data, family = "binomial")
model_glm_new <- glm(disease ~ age + sex + bp + chol + cp, data = train_data, family = "binomial")

# Example 1: Using pre-calculated plot data
train_results <- UScalc_mdl(model_glm_ref, model_glm_new, y_coef = "rLR")
USplot(plot_data = train_results$plot_data, y_coef = "rLR")

# Example 2a: Using raw data
train_out_glm_ref <- USprep_mdl(model_glm_ref, dataset = NULL, testing = FALSE)
train_out_glm_new <- USprep_mdl(model_glm_new, dataset = NULL, testing = FALSE)

# Combine raw data for comparison
combined_glm <- USbind_out(train_out_glm_ref, train_out_glm_new)
raw_compare <- combined_glm$comparison_df
n_vars_diff <- combined_glm$n_vars_diff  # Shows difference in number of parameters

#' # Calculate rLR coefficients
results_rLR <- UScalc_raw(raw_compare, y_coef = "rLR", n_vars_diff)

USplot(raw_data = raw_data, y_coef = "rLR", n_vars_diff = n_vars_diff)

#' # Example 2b: Using raw data
train_out_glm_ref <- USprep_mdl(model_glm_ref, dataset = NULL, testing = FALSE)
train_out_glm_new <- USprep_mdl(model_glm_new, dataset = NULL, testing = FALSE)
combined_glm <- USbind_out(train_out_glm_ref, train_out_glm_new)
USplot(raw_data=combined_glm$comparison_df, y_coef = "rLR", n_vars_diff = combined_glm$n_vars_diff)

# Example 3: Directly from models and data
# Ex 1: Evaluate on training data
USplot(ref_formula = disease ~ age + sex + bp + chol,
       new_formula = disease ~ age + sex + bp + chol + cp,
       ref_model_type = "glm",
       new_model_type = "glm",
       train_data = train_data,
       testing = FALSE,
       y_coef = "rLR")

# Ex 2: Evaluate on test data
USplot(ref_formula = disease ~ age + sex + bp + chol,
       new_formula = disease ~ age + sex + bp + chol + cp,
       ref_model_type = "glm",
       new_model_type = "glm",
       train_data = train_data,
       test_data = test_data,
       testing = TRUE,
       y_coef = "rLR")

# Ex 3: Evaluate on training data (net coefficients and statistical significance testing)
USplot(ref_formula = disease ~ age + sex + bp + chol,
       new_formula = disease ~ age + sex + bp + chol + cp,
       ref_model_type = "glm",
       new_model_type = "glm",
       train_data = train_data,
       testing = FALSE,
       y_coef = "rLR",
       net = TRUE,
       crit=2)

# Ex 3: With randomForest and calibration on test data
USplot(ref_formula = disease ~ age + sex + bp + chol,
       new_formula = disease ~ age + sex + bp + chol + cp,
       ref_model_type = "randomForest",
       new_model_type = "randomForest",
       train_data = train_data,
       test_data = test_data,
       testing = TRUE,
       y_coef = "rLR",
       calibrate = TRUE)

# Ex 4: Compare GLM with randomForest
USplot(ref_formula = disease ~ age + sex,
       new_formula = disease ~ age + sex,
       ref_model_type = "glm",
       new_model_type = "randomForest",
       train_data = train_data,
       y_coef = "rLR")

# Ex 5: Compare randomForest with calibrated randomForest on test data
USplot(ref_formula = disease ~ age + sex,
       new_formula = disease ~ age + sex,
       ref_model_type = "randomForest",
       new_model_type = "randomForest",
       train_data = train_data,
       test_data = test_data,
       testing = TRUE,
       ref_calibrate = FALSE,
       new_calibrate = TRUE,
       y_coef = "rLR")



## End(Not run)
```

USprep_mdl *Prepare data for model evaluation for various binary classifiers*

Description

This function prepares data for evaluating various binary classification models including Logistic Regression, Random Forest, SVM, XGBoost, Neural Networks, and Naive Bayes. Supports both training and test data with optional probability calibration.

Usage

```
USprep_mdl(
  model,
  dataset = NULL,
  testing = FALSE,
  calibrate = FALSE,
  calibration_method = "isotonic"
)
```

Arguments

`model`	Model object (supported classes: 'glm', 'randomForest', 'ranger', 'svm', 'xgb.Booster', 'nnet', 'naive_bayes', 'model_fit' from tidymodels)
`dataset`	Dataset used to prepare data for the models. • If `testing` = `TRUE`, a test dataset must be provided (required). • If `testing` = `FALSE`, the argument can be: – `NULL` if the model stores its own training data (e.g., `glm`, `randomForest`), – the training dataset if the model does not keep training data internally (e.g., `ranger`, `e1071`, `xgboost`, `nnet`, `naive_bayes`, tidymodels). This allows the function to work consistently with both training and test data.
`testing`	Whether to prepare test data (TRUE) or training data (FALSE)
`calibrate`	Whether to calibrate probabilities (default: FALSE)
`calibration_method`	Calibration method ("isotonic" or "logistic")

Value

A list containing:

- y - vector of observed values (0/1)
- p - vector of predicted probabilities for positive class
- n_vars - number of predictor variables in the model
- var_names - names of predictor variables in the model

Note

This function requires the following packages to be installed for specific model types:

- `randomForest` - for randomForest models
- `ranger` - for ranger models
- `e1071` - for SVM models
- `xgboost` - for XGBoost models

- `nnet` - for Neural Network models
- `parsnip`, `workflows` - for tidymodels models
- `naivebayes` - for Naive Bayes models

The function will load these packages automatically when needed.

Examples

```
## Not run:
# Load required packages
library(tidyverse)
library(randomForest)
library(ranger)
library(e1071)
library(xgboost)
library(nnet)
library(parsnip)
library(workflows)
library(naivebayes)

# Load Heart Disease datasets
data(heart_disease)
data(heart_disease_train)
data(heart_disease_test)

# For Logistic Regression model (glm)

# For logistic regression model (reference model with age, sex, bp, chol)
model_glm_ref <- glm(disease ~ age + sex + bp + chol, data = train_data, family = "binomial")
train_out_glm_ref <- USprep_mdl(model_glm_ref, dataset = NULL, testing = FALSE)
test_out_glm_ref <- USprep_mdl(model_glm_ref, dataset = test_data, testing = TRUE)

# For Random Forest model (randomForest package)
model_rf_ref <- randomForest::randomForest(disease ~ age + sex + bp + chol, data = train_data)
train_out_rf_ref <- USprep_mdl(model_rf_ref, dataset = NULL, testing = FALSE)
test_out_rf_ref <- USprep_mdl(model_rf_ref, dataset = test_data, testing = TRUE)

# For Random Forest model (ranger package)
model_ranger <- ranger(disease ~ age + sex + bp + chol, data = train_data,
                      probability = TRUE, keep.inbag = TRUE)
train_result_ranger <- USprep_mdl(model_ranger, dataset = train_data, testing = FALSE)

# For SVM model (e1071 package)
model_svm <- svm(disease ~ age + sex + bp + chol, data = train_data,
                probability = TRUE, kernel = "radial")
train_result_svm <- USprep_mdl(model_svm, dataset = train_data, testing = FALSE)

# For Neural Network model (nnet package)
train_data_scaled <- train_data
numeric_vars <- c("age", "bp", "chol")
train_data_scaled[numeric_vars] <- scale(train_data[numeric_vars])
model_formula <- disease ~ age + sex + bp + chol
model_matrix_train <- model.matrix(model_formula, data = train_data_scaled)[,-1]
model_nnet <- nnet(x = model_matrix_train,
                  y = as.numeric(train_data_scaled$disease) - 1,
                  size = 10, maxit = 1000, linout = FALSE, entropy = TRUE, trace = TRUE)
model_matrix_with_y <- as.data.frame(model_matrix_pred)
```

```
model_matrix_with_y$disease <- as.numeric(train_data_scaled$disease) - 1
train_result_nnet <- USprep_mdl(model_nnet, dataset = model_matrix_with_y, testing = FALSE)

# For Naive Bayes model (naivebayes package)
model_nb <- naive_bayes(disease ~ age + sex + bp + chol, data = train_data)
train_result_nb <- USprep_mdl(model_nb, dataset = train_data, testing = FALSE)

# For tidymodels/parsnip models
model_spec <- logistic_reg() %>% set_engine("glm") %>% set_mode("classification")
model_tidymodels <- model_spec %>% fit(disease ~ age + sex + bp + chol, data = train_data)
train_result_tidy <- USprep_mdl(model_tidymodels, dataset = NULL, testing = FALSE)

# With calibration
model_for_calib <- glm(disease ~ age + sex + bp + chol, data = train_data, family = binomial)
train_result_calibrated <- USprep_mdl(model_for_calib, dataset = NULL, testing = FALSE,
  calibrate = TRUE, calibration_method = "isotonic")

## End(Not run)
```

Index