Package ‘fairmetrics’

June 5, 2025

Title Fairness Evaluation Metrics with Confidence Intervals

Version 1.0.2

Description A collection of functions for computing fairness metrics for machine learning and statistical models, including confidence intervals for each metric. The package supports the evaluation of group-level fairness criterion commonly used in fairness research, particularly in healthcare. It is based on the overview of fairness in machine learning written by Gao et al (2024) <doi:10.48550/arXiv.2406.09307>.

Imports stats

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Suggests dplyr, magrittr, corrplot, randomForest, pROC, SpecsVerification, knitr, rmarkdown, testthat, kableExtra, naniar

Config/testthat/edition 3

Depends R (>= 3.5.0)

LazyData true

URL `https://jianhuig.github.io/fairmetrics/`

VignetteBuilder knitr

NeedsCompilation no

Author Jianhui Gao [aut] (ORCID: <`https://orcid.org/0000-0003-0915-1473`>), Benjamin Smith [aut, cre] (ORCID: <`https://orcid.org/0009-0007-2206-0177`>), Benson Chou [aut] (ORCID: <`https://orcid.org/0009-0007-0265-033X`>), Jessica Gronsbell [aut] (ORCID: <`https://orcid.org/0000-0002-5360-5869`>)

Maintainer Benjamin Smith <`benyamin.smith@mail.utoronto.ca`>

Repository CRAN

Date/Publication 2025-06-05 07:50:10 UTC

Contents

`eval_acc_parity` *Examine Accuracy Parity of a Model*

Description

This function assesses *Accuracy Parity*, a fairness criterion that evaluates whether the overall accuracy of a predictive model is consistent across different groups.

Usage

```
eval_acc_parity(
  data,
  outcome,
  group,
  probs,
  cutoff = 0.5,
  confint = TRUE,
  alpha = 0.05,
  bootstraps = 2500,
  digits = 2,
  message = TRUE
)
```

Arguments

`data`	Data frame containing the outcome, predicted outcome, and sensitive attribute
`outcome`	Name of the outcome variable
`group`	Name of the sensitive attribute
`probs`	Predicted probabilities
`cutoff`	Cutoff value for the predicted probabilities
`confint`	Logical indicating whether to calculate confidence intervals
`alpha`	The 1 - significance level for the confidence interval, default is 0.05
`bootstraps`	Number of bootstraps to use for confidence intervals
`digits`	Number of digits to round the results to, default is 2
`message`	Whether to print the results, default is TRUE

Value

A list containing the following elements:

- Accuracy for Group 1
- Accuracy for Group 2
- Difference in accuracy
- Ratio in accuracy If confidence intervals are computed (`confint = TRUE`):
- A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the difference in accuracy
- A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the ratio in accurac

See Also

`eval_cond_acc_equality`

Examples

```
library(fairmetrics)
library(dplyr)
library(magrittr)
library(randomForest)
data("mimic_preprocessed")
set.seed(123)
train_data <- mimic_preprocessed %>%
  dplyr::filter(dplyr::row_number() <= 700)
# Fit a random forest model
rf_model <- randomForest::randomForest(factor(day_28_flg) ~ ., data = train_data, ntree = 1000)
# Test the model on the remaining data
test_data <- mimic_preprocessed %>%
  dplyr::mutate(gender = ifelse(gender_num == 1, "Male", "Female")) %>%
  dplyr::filter(dplyr::row_number() > 700)
```

```
test_data$pred <- predict(rf_model, newdata = test_data, type = "prob")[, 2]

# Fairness evaluation
# We will use sex as the sensitive attribute and day_28_flg as the outcome.
# We choose threshold = 0.41 so that the overall FPR is around 5%.

# Evaluate Accuracy Parity
eval_acc_parity(
  data = test_data,
  outcome = "day_28_flg",
  group = "gender",
  probs = "pred",
  cutoff = 0.41
)
```

---

eval_bs_parity *Examine Brier Score Parity of a Model*

Description

This function evaluates *Brier Score Parity*, a fairness measure that checks whether the Brier score (a measure of the calibration of probabilistic predictions) is similar across different groups. Brier score parity ensures that the model's predicted probabilities are equally well calibrated across subpopulations.

Usage

```
eval_bs_parity(
  data,
  outcome,
  group,
  probs,
  confint = TRUE,
  alpha = 0.05,
  bootstraps = 2500,
  digits = 2,
  message = TRUE
)
```

Arguments

data	Data frame containing the outcome, predicted outcome, and sensitive attribute
outcome	Name of the outcome variable
group	Name of the sensitive attribute
probs	Predicted probabilities

`confint`	Logical indicating whether to calculate confidence intervals
`alpha`	The 1 - significance level for the confidence interval, default is 0.05
`bootstraps`	Number of bootstraps to use for confidence intervals
`digits`	Number of digits to round the results to, default is 2
`message`	Whether to print the results, default is TRUE

Value

A list containing the following elements:

- Brier Score for Group 1
- Brier Score for Group 2
- Difference in Brier Score
- Ratio in Brier Score If confidence intervals are computed (`confint = TRUE`):
- A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the difference in Brier Score
- A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the ratio in Brier Score

See Also

`eval_acc_parity`, `eval_cond_acc_equality`, `eval_pos_pred_parity`, `eval_neg_pred_parity`

Examples

```
library(fairmetrics)
library(dplyr)
library(magrittr)
library(randomForest)
data("mimic_preprocessed")
set.seed(123)
train_data <- mimic_preprocessed %>%
  dplyr::filter(dplyr::row_number() <= 700)
# Fit a random forest model
rf_model <- randomForest::randomForest(factor(day_28_flg) ~ ., data = train_data, ntree = 1000)
# Test the model on the remaining data
test_data <- mimic_preprocessed %>%
  dplyr::mutate(gender = ifelse(gender_num == 1, "Male", "Female")) %>%
  dplyr::filter(dplyr::row_number() > 700)

test_data$pred <- predict(rf_model, newdata = test_data, type = "prob")[, 2]

# Fairness evaluation
# We will use sex as the sensitive attribute and day_28_flg as the outcome.

# Evaluate Brier Score Parity
eval_bs_parity(
  data = test_data,
  outcome = "day_28_flg",
  group = "gender",
  probs = "pred"
)
```

---

eval_cond_acc_equality

Examine Conditional Use Accuracy Equality of a Model

Description

This function evaluates *Conditional Use Accuracy Equality*, a fairness criterion that requires predictive performance to be similar across groups when a model makes positive or negative predictions.

Usage

```
eval_cond_acc_equality(
  data,
  outcome,
  group,
  probs,
  cutoff = 0.5,
  confint = TRUE,
  alpha = 0.05,
  bootstraps = 2500,
  digits = 2,
  message = TRUE
)
```

Arguments

`data`	Data frame containing the outcome, predicted outcome, and sensitive attribute
`outcome`	Name of the outcome variable, it must be binary
`group`	Name of the sensitive attribute
`probs`	Name of the predicted outcome variable
`cutoff`	Threshold for the predicted outcome, default is 0.5
`confint`	Whether to compute 95% confidence interval, default is TRUE
`alpha`	The 1 - significance level for the confidence interval, default is 0.05
`bootstraps`	Number of bootstrap samples, default is 2500
`digits`	Number of digits to round the results to, default is 2
`message`	Whether to print the results, default is TRUE

Value

A list containing the following elements:

- PPV_Group1: Positive Predictive Value for the first group
- PPV_Group2: Positive Predictive Value for the second group
- PPV_Diff: Difference in Positive Predictive Value
- NPV_Group1: Negative Predictive Value for the first group
- NPV_Group2: Negative Predictive Value for the second group
- NPV_Diff: Difference in Negative Predictive Value If confidence intervals are computed (`confint = TRUE`):
- PPV_Diff_CI: A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the difference in Positive Predictive Value
- NPV_Diff_CI: A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the difference in Negative Predictive Value

See Also

`eval_acc_parity`

Examples

```
library(fairmetrics)
library(dplyr)
library(magrittr)
library(randomForest)
data("mimic_preprocessed")
set.seed(123)
train_data <- mimic_preprocessed %>%
  dplyr::filter(dplyr::row_number() <= 700)
# Fit a random forest model
rf_model <- randomForest::randomForest(factor(day_28_flg) ~ ., data = train_data, ntree = 1000)
# Test the model on the remaining data
test_data <- mimic_preprocessed %>%
  dplyr::mutate(gender = ifelse(gender_num == 1, "Male", "Female")) %>%
  dplyr::filter(dplyr::row_number() > 700)

test_data$pred <- predict(rf_model, newdata = test_data, type = "prob")[, 2]

# Fairness evaluation
# We will use sex as the sensitive attribute and day_28_flg as the outcome.
# We choose threshold = 0.41 so that the overall FPR is around 5%.

# Evaluate Conditional Use Accuracy Equality
eval_cond_acc_equality(
  data = test_data,
  outcome = "day_28_flg",
  group = "gender",
  probs = "pred",
  cutoff = 0.41
)
```

eval_cond_stats_parity

Examine Conditional Statistical Parity of a Model

Description

This function evaluates *conditional statistical parity*, which measures fairness by comparing positive prediction rates across sensitive groups within a defined subgroup of the population. This is useful in scenarios where fairness should be evaluated in a more context-specific way—e.g., within a particular hospital unit or age bracket. Conditional statistical parity is a refinement of standard statistical parity. Instead of comparing prediction rates across groups in the entire dataset, it restricts the comparison to a specified subset of the population, defined by a conditioning variable.

Usage

```
eval_cond_stats_parity(
  data,
  outcome,
  group,
  group2,
  condition,
  probs,
  confint = TRUE,
  cutoff = 0.5,
  bootstraps = 2500,
  alpha = 0.05,
  message = TRUE,
  digits = 2
)
```

Arguments

`data`	Data frame containing the outcome, predicted outcome, and sensitive attribute
`outcome`	Name of the outcome variable, it must be binary
`group`	Name of the sensitive attribute
`group2`	Name of the group to condition on
`condition`	If the conditional group is categorical, the condition supplied must be a character of the levels to condition on. If the conditional group is continuous, the conditions supplied must be a character containing the sign of the condition and the value to threshold the continuous variable (e.g. "<50", ">50", "<=50", ">=50").
`probs`	Name of the predicted outcome variable
`confint`	Whether to compute 95% confidence interval, default is TRUE

`cutoff`	Threshold for the predicted outcome, default is 0.5
`bootstraps`	Number of bootstrap samples, default is 2500
`alpha`	The 1 - significance level for the confidence interval, default is 0.05
`message`	Whether to print the results, default is TRUE
`digits`	Number of digits to round the results to, default is 2

Details

The function supports both categorical and continuous conditioning variables. For continuous variables, you can supply a threshold expression like `"<50"` or `">=75"` to the `condition` parameter.

Value

A list containing the following elements:

- Conditions: The conditions used to calculate the conditional PPR
- PPR_Group1: Positive Prediction Rate for the first group
- PPR_Group2: Positive Prediction Rate for the second group
- PPR_Diff: Difference in Positive Prediction Rate
- PPR_Ratio: Ratio in Positive Prediction Rate If confidence intervals are computed (`confint = TRUE`):
- PPR_Diff_CI: A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the difference in Positive Prediction Rate
- PPR_Ratio_CI: A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the ratio in Positive Prediction Rate

See Also

`eval_stats_parity`

Examples

```
library(fairmetrics)
library(dplyr)
library(magrittr)
library(randomForest)
data("mimic_preprocessed")
set.seed(123)
train_data <- mimic_preprocessed %>%
  dplyr::filter(dplyr::row_number() <= 700)
# Fit a random forest model
rf_model <- randomForest::randomForest(factor(day_28_flg) ~ ., data = train_data, ntree = 1000)
# Test the model on the remaining data
test_data <- mimic_preprocessed %>%
  dplyr::mutate(gender = ifelse(gender_num == 1, "Male", "Female")) %>%
  dplyr::filter(dplyr::row_number() > 700)

test_data$pred <- predict(rf_model, newdata = test_data, type = "prob")[, 2]
```

```
# Fairness evaluation
# We will use sex as the sensitive attribute and day_28_flg as the outcome.
# We choose threshold = 0.41 so that the overall FPR is around 5%.

# Evaluate Conditional Statistical Parity

eval_cond_stats_parity(
  data = test_data,
  outcome = "day_28_flg",
  group = "gender",
  group2 = "service_unit",
  condition = "MICU",
  probs = "pred",
  cutoff = 0.41
)
```

`eval_eq_odds` *Examine Equalized Odds of a Predictive Model*

Description

This function evaluates whether a predictive model satisfies the Equalized Odds criterion by comparing both False Negative Rates (FNR) and False Positive Rates (FPR) across two groups defined by a binary sensitive attribute. It reports the rate for each group, their differences, ratios, and bootstrap-based confidence regions. A Bonferroni-corrected union test is used to test whether the model violates the Equalized Odds criterion.

Usage

```
eval_eq_odds(
  data,
  outcome,
  group,
  probs,
  cutoff = 0.5,
  confint = TRUE,
  bootstraps = 2500,
  alpha = 0.05,
  digits = 2,
  message = TRUE
)
```

Arguments

`data`	A data frame containing the true binary outcomes, predicted probabilities, and sensitive group membership.

`outcome`	A string specifying the name of the binary outcome variable in `data`.
`group`	A string specifying the name of the binary sensitive attribute variable (e.g., race, gender) used to define the comparison groups.
`probs`	A string specifying the name of the variable containing predicted probabilities or risk scores.
`cutoff`	A numeric value used to threshold predicted probabilities into binary predictions; defaults to 0.5.
`confint`	Whether to compute 95% confidence interval, default is TRUE.
`bootstraps`	An integer specifying the number of bootstrap resamples for constructing confidence intervals; vdefaults to 2500.
`alpha`	Significance level for the (1 - `alpha`) confidence interval; defaults to 0.05.
`digits`	Number of decimal places to round numeric results; defaults to 2.
`message`	Logical; if TRUE (default), prints a textual summary of the fairness evaluation.

Value

A data frame summarizing group disparities in both FNR and FPR with the following columns:

- `Metric`: The reported metrics ("FNR; FPR").
- `Group1`: Estimated FNR and FPR for the first group.
- `Group2`: Estimated FNR and FPR for the second group.
- `Difference`: Differences in FNR and FPR, computed as Group1 - Group2.
- `95% CR`: Bonferroni-adjusted confidence regions for the differences.
- `Ratio`: Ratios in FNR and FPR, computed as Group1 / Group2.
- `95% CR`: Bonferroni-adjusted confidence regions for the ratios.

Examples

```
library(fairmetrics)
library(dplyr)
library(magrittr)
library(randomForest)
data("mimic_preprocessed")
set.seed(123)
train_data <- mimic_preprocessed %>%
  dplyr::filter(dplyr::row_number() <= 700)
# Fit a random forest model
rf_model <- randomForest::randomForest(factor(day_28_flg) ~ .,
  data = train_data, ntree = 1000
)
# Test the model on the remaining data
test_data <- mimic_preprocessed %>%
  dplyr::mutate(gender = ifelse(gender_num == 1, "Male", "Female")) %>%
  dplyr::filter(dplyr::row_number() > 700)

test_data$pred <- predict(rf_model, newdata = test_data, type = "prob")[, 2]
```

```
# Fairness evaluation
# We will use sex as the sensitive attribute and day_28_flg as the outcome.
# We choose threshold = 0.41 so that the overall FPR is around 5%.

# Evaluate Equalized Odds
eval_eq_odds(
  data = test_data,
  outcome = "day_28_flg",
  group = "gender",
  probs = "pred",
  cutoff = 0.41
)
```

eval_eq_opp *Evaluate Equal Opportunity Compliance of a Predictive Model*

Description

This function evaluates the fairness of a predictive model with respect to the Equal Opportunity criterion, which requires that the False Negative Rate (FNR) be comparable across groups defined by a sensitive attribute. The function quantifies disparities in FNR between two groups and provides both the absolute difference and ratio, along with confidence intervals obtained via bootstrapping.

Usage

```
eval_eq_opp(
  data,
  outcome,
  group,
  probs,
  cutoff = 0.5,
  confint = TRUE,
  bootstraps = 2500,
  alpha = 0.05,
  digits = 2,
  message = TRUE
)
```

Arguments

`data`	A data frame containing the true binary outcomes, predicted probabilities, and sensitive group membership.
`outcome`	A string specifying the name of the binary outcome variable in `data`.
`group`	A string specifying the name of the sensitive attribute variable (e.g., race, gender).

`probs`	A string specifying the name of the variable containing predicted probabilities or risk scores.
`cutoff`	A numeric value used to threshold predicted probabilities into binary decisions; defaults to 0.5.
`confint`	Whether to compute 95% confidence interval, default is TRUE.
`bootstraps`	An integer specifying the number of bootstrap resamples for constructing confidence intervals; defaults to 2500.
`alpha`	Significance level for constructing the (1 - `alpha`) confidence interval; defaults to 0.05.
`digits`	Integer indicating the number of decimal places to round results to; defaults to 2.
`message`	Logical; if TRUE (default), prints a textual summary of the fairness evaluation.

Value

A data frame summarizing FNR-based group disparity metrics with the following columns:

- `Metric` A label indicating the reported fairness criterion.
- `Group1` Estimated FNR and FPR for the first group.
- `Group2` Estimated FNR and FPR for the second group.
- `Difference` The difference in FNR between the two groups, computed as the FNR of Group1 minus the FNR of Group2.
- `95% Diff CI` The (1 - `alpha`) confidence interval for the FNR difference.
- `Ratio` The ratio of FNRs between Group1 and Group2, computed as FNR for Group1 divided by FNR for Group2.
- `95% Ratio CI` The corresponding confidence interval for the FNR ratio.

Examples

```
library(fairmetrics)
library(dplyr)
library(magrittr)
library(randomForest)
data("mimic_preprocessed")
set.seed(123)
train_data <- mimic_preprocessed %>%
  dplyr::filter(dplyr::row_number() <= 700)
# Fit a random forest model
rf_model <- randomForest::randomForest(factor(day_28_flg) ~ .,
  data =
    train_data, ntree = 1000
)
# Test the model on the remaining data
test_data <- mimic_preprocessed %>%
  dplyr::mutate(gender = ifelse(gender_num == 1, "Male", "Female")) %>%
  dplyr::filter(dplyr::row_number() > 700)
```

```
test_data$pred <- predict(rf_model, newdata = test_data, type = "prob")[, 2]

# Fairness evaluation
# We will use sex as the sensitive attribute and day_28_flg as the outcome.
# We choose threshold = 0.41 so that the overall FPR is around 5%.

# Evaluate Equal Opportunity Compliance
eval_eq_opp(
  data = test_data,
  outcome = "day_28_flg",
  group = "gender",
  probs = "pred",
  cutoff = 0.41
)
```

`eval_neg_class_bal` *Examine Balance for Negative Class of a Model*

Description

This function evaluates *Balance for the Negative Class*, a fairness criterion that checks whether the model assigns similar predicted probabilities across groups among individuals whose true outcome is negative (i.e., \(Y = 0\)).

Usage

```
eval_neg_class_bal(
  data,
  outcome,
  group,
  probs,
  confint = TRUE,
  alpha = 0.05,
  bootstraps = 2500,
  digits = 2,
  message = TRUE
)
```

Arguments

`data`	Data frame containing the outcome, predicted outcome, and sensitive attribute
`outcome`	Name of the outcome variable
`group`	Name of the sensitive attribute
`probs`	Predicted probabilities
`confint`	Logical indicating whether to calculate confidence intervals
`alpha`	The 1 - significance level for the confidence interval, default is 0.05

`bootstraps`	Number of bootstraps to use for confidence intervals
`digits`	Number of digits to round the results to, default is 2
`message`	Whether to print the results, default is TRUE

Value

A list containing the following elements:

- Average predicted probability for Group 1
- Average predicted probability for Group 2
- Difference in average predicted probability
- Ratio in average predicted probability If confidence intervals are computed (`confint = TRUE`):
- A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the difference in average predicted probability
- A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the ratio in average predicted probability

See Also

`eval_neg_class_bal`

Examples

```
library(fairmetrics)
library(dplyr)
library(magrittr)
library(randomForest)
data("mimic_preprocessed")
set.seed(123)
train_data <- mimic_preprocessed %>%
  dplyr::filter(dplyr::row_number() <= 700)
# Fit a random forest model
rf_model <- randomForest::randomForest(factor(day_28_flg) ~ ., data = train_data, ntree = 1000)
# Test the model on the remaining data
test_data <- mimic_preprocessed %>%
  dplyr::mutate(gender = ifelse(gender_num == 1, "Male", "Female")) %>%
  dplyr::filter(dplyr::row_number() > 700)

test_data$pred <- predict(rf_model, newdata = test_data, type = "prob")[, 2]

# Fairness evaluation
# We will use sex as the sensitive attribute and day_28_flg as the outcome.

# Evaluate Balance for Negative Class
eval_neg_class_bal(
  data = test_data,
  outcome = "day_28_flg",
  group = "gender",
  probs = "pred"
)
```

`eval_neg_pred_parity` *Examine Negative Predictive Parity of a Model*

Description

This function evaluates *negative predictive predictive parity*, a key fairness criterion that compares the *Negative Predictive Value (NPV)* between groups defined by a sensitive attribute. In other words, it assesses whether, among individuals predicted to be negative, the probability of being truly negative is equal across subgroups.

Usage

```
eval_neg_pred_parity(
  data,
  outcome,
  group,
  probs,
  cutoff = 0.5,
  confint = TRUE,
  bootstraps = 2500,
  alpha = 0.05,
  digits = 2,
  message = TRUE
)
```

Arguments

`data`	Data frame containing the outcome, predicted outcome, and sensitive attribute
`outcome`	Name of the outcome variable, it must be binary
`group`	Name of the sensitive attribute
`probs`	Name of the predicted outcome variable
`cutoff`	Threshold for the predicted outcome, default is 0.5
`confint`	Whether to compute 95% confidence interval, default is TRUE
`bootstraps`	Number of bootstrap samples, default is 2500
`alpha`	The 1 - significance level for the confidence interval, default is 0.05
`digits`	Number of digits to round the results to, default is 2
`message`	Whether to print the results, default is TRUE

Value

A list containing the following elements:

- NPV_Group1: Negative Predictive Value for the first group
- NPV_Group2: Negative Predictive Value for the second group

- NPV_Diff: Difference in Negative Predictive Value
- NPV_Ratio: Ratio in Negative Predictive Value If confidence intervals are computed (`confint = TRUE`):
- NPV_Diff_CI: A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the difference in Negative Predictive Value
- NPV_Ratio_CI: A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the ratio in Negative Predictive Value

See Also

`eval_pos_pred_parity`

Examples

```
library(fairmetrics)
library(dplyr)
library(magrittr)
library(randomForest)
data("mimic_preprocessed")
set.seed(123)
train_data <- mimic_preprocessed %>%
  dplyr::filter(dplyr::row_number() <= 700)
# Fit a random forest model
rf_model <- randomForest::randomForest(factor(day_28_flg) ~ ., data = train_data, ntree = 1000)
# Test the model on the remaining data
test_data <- mimic_preprocessed %>%
  dplyr::mutate(gender = ifelse(gender_num == 1, "Male", "Female")) %>%
  dplyr::filter(dplyr::row_number() > 700)

test_data$pred <- predict(rf_model, newdata = test_data, type = "prob")[, 2]

# Fairness evaluation
# We will use sex as the sensitive attribute and day_28_flg as the outcome.
# We choose threshold = 0.41 so that the overall FPR is around 5%.

# Evaluate Negative Predictive Parity
eval_neg_pred_parity(
  data = test_data,
  outcome = "day_28_flg",
  group = "gender",
  probs = "pred",
  cutoff = 0.41
)
```

`eval_pos_class_bal` *Examine Balance for the Positive Class of a Model*

Description

This function evaluates *Balance for the Positive Class*, a fairness criterion that checks whether the model assigns similar predicted probabilities across groups among individuals whose true outcome is positive (i.e., \(Y = 1\)).

Usage

```
eval_pos_class_bal(
  data,
  outcome,
  group,
  probs,
  confint = TRUE,
  alpha = 0.05,
  bootstraps = 2500,
  digits = 2,
  message = TRUE
)
```

Arguments

`data`	Data frame containing the outcome, predicted outcome, and sensitive attribute
`outcome`	Name of the outcome variable
`group`	Name of the sensitive attribute
`probs`	Predicted probabilities
`confint`	Logical indicating whether to calculate confidence intervals
`alpha`	The 1 - significance level for the confidence interval, default is 0.05
`bootstraps`	Number of bootstraps to use for confidence intervals
`digits`	Number of digits to round the results to, default is 2
`message`	Whether to print the results, default is TRUE

Value

A list containing the following elements:

- Average predicted probability for Group 1
- Average predicted probability for Group 2
- Difference in average predicted probability
- Ratio in average predicted probability If confidence intervals are computed (`confint = TRUE`):
- A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the difference in average predicted probability
- A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the ratio in average predicted probability

See Also

`eval_neg_class_bal`

Examples

```
library(fairmetrics)
library(dplyr)
library(magrittr)
library(randomForest)
data("mimic_preprocessed")
set.seed(123)
train_data <- mimic_preprocessed %>%
  dplyr::filter(dplyr::row_number() <= 700)
# Fit a random forest model
rf_model <- randomForest::randomForest(factor(day_28_flg) ~ ., data = train_data, ntree = 1000)
# Test the model on the remaining data
test_data <- mimic_preprocessed %>%
  dplyr::mutate(gender = ifelse(gender_num == 1, "Male", "Female")) %>%
  dplyr::filter(dplyr::row_number() > 700)

test_data$pred <- predict(rf_model, newdata = test_data, type = "prob")[, 2]

# Fairness evaluation
# We will use sex as the sensitive attribute and day_28_flg as the outcome.

# Evaluate Balance for Positive Class
eval_pos_class_bal(
  data = test_data,
  outcome = "day_28_flg",
  group = "gender",
  probs = "pred"
)
```

`eval_pos_pred_parity` *Examine Positive Predictive Parity of a Model*

Description

This function evaluates *positive predictive predictive parity*, a key fairness criterion that compares the *Positive Predictive Value (PPV)* between groups defined by a sensitive attribute. In other words, it assesses whether, among individuals predicted to be positive, the probability of being truly positive is equal across subgroups.

Usage

```
eval_pos_pred_parity(
  data,
  outcome,
  group,
  probs,
  cutoff = 0.5,
  confint = TRUE,
  bootstraps = 2500,
  alpha = 0.05,
  digits = 2,
  message = TRUE
)
```

Arguments

`data`	Data frame containing the outcome, predicted outcome, and sensitive attribute
`outcome`	Name of the outcome variable, it must be binary
`group`	Name of the sensitive attribute
`probs`	Name of the predicted outcome variable
`cutoff`	Threshold for the predicted outcome, default is 0.5
`confint`	Whether to compute 95% confidence interval, default is TRUE
`bootstraps`	Number of bootstrap samples, default is 2500
`alpha`	The 1 - significance level for the confidence interval, default is 0.05
`digits`	Number of digits to round the results to, default is 2
`message`	Whether to print the results, default is TRUE

Value

A list containing the following elements:

- PPV_Group1: Positive Predictive Value for the first group
- PPV_Group2: Positive Predictive Value for the second group
- PPV_Diff: Difference in Positive Predictive Value
- PPV_Ratio: Ratio in Positive Predictive Value If confidence intervals are computed (`confint = TRUE`):
- PPV_Diff_CI: A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the difference in Positive Predictive Value
- PPV_Ratio_CI: A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the ratio in Positive Predictive Value

See Also

`eval_neg_pred_parity`

Examples

```
library(fairmetrics)
library(dplyr)
library(magrittr)
library(randomForest)
data("mimic_preprocessed")
set.seed(123)
train_data <- mimic_preprocessed %>%
  dplyr::filter(dplyr::row_number() <= 700)
# Fit a random forest model
rf_model <- randomForest::randomForest(factor(day_28_flg) ~ ., data = train_data, ntree = 1000)
# Test the model on the remaining data
test_data <- mimic_preprocessed %>%
  dplyr::mutate(gender = ifelse(gender_num == 1, "Male", "Female")) %>%
  dplyr::filter(dplyr::row_number() > 700)

test_data$pred <- predict(rf_model, newdata = test_data, type = "prob")[, 2]

# Fairness evaluation
# We will use sex as the sensitive attribute and day_28_flg as the outcome.
# We choose threshold = 0.41 so that the overall FPR is around 5%.

# Evaluate Positive Predictive Parity
eval_pos_pred_parity(
  data = test_data,
  outcome = "day_28_flg",
  group = "gender",
  probs = "pred",
  cutoff = 0.41
)
```

`eval_pred_equality` *Examine Predictive Equality of a Model*

Description

This function evaluates predictive equality, a fairness metric that compares the False Positive Rate (FPR) between groups defined by a sensitive attribute. It assesses whether individuals from different groups are equally likely to be incorrectly flagged as positive when they are, in fact, negative.

Usage

```
eval_pred_equality(
  data,
  outcome,
  group,
  probs,
  cutoff = 0.5,
  confint = TRUE,
  alpha = 0.05,
  bootstraps = 2500,
  digits = 2,
  message = TRUE
)
```

Arguments

`data`	Data frame containing the outcome, predicted outcome, and sensitive attribute
`outcome`	Name of the outcome variable, it must be binary
`group`	Name of the sensitive attribute
`probs`	Name of the predicted outcome variable
`cutoff`	Threshold for the predicted outcome, default is 0.5
`confint`	Whether to compute 95% confidence interval, default is TRUE
`alpha`	The 1 - significance level for the confidence interval, default is 0.05
`bootstraps`	Number of bootstrap samples, default is 2500
`digits`	Number of digits to round the results to, default is 2
`message`	Whether to print the results, default is TRUE

Value

A list containing the following elements:

- FPR_Group1: False Positive Rate for the first group
- FPR_Group2: False Positive Rate for the second group
- FPR_Diff: Difference in False Positive Rate
- FPR_Ratio: Ratio in False Positive Rate If confidence intervals are computed (`confint = TRUE`):
- FPR_Diff_CI: A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the difference in False Positive Rate
- FPR_Ratio_CI: A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the ratio in False Positive Rate

See Also

`eval_pos_pred_parity`, `eval_neg_pred_parity`, `eval_stats_parity`

Examples

```
library(fairmetrics)
library(dplyr)
library(magrittr)
library(randomForest)
data("mimic_preprocessed")
set.seed(123)
```

```
train_data <- mimic_preprocessed %>%
  dplyr::filter(dplyr::row_number() <= 700)
# Fit a random forest model
rf_model <- randomForest::randomForest(factor(day_28_flg) ~ ., data = train_data, ntree = 1000)
# Test the model on the remaining data
test_data <- mimic_preprocessed %>%
  dplyr::mutate(gender = ifelse(gender_num == 1, "Male", "Female")) %>%
  dplyr::filter(dplyr::row_number() > 700)

test_data$pred <- predict(rf_model, newdata = test_data, type = "prob")[, 2]

# Fairness evaluation
# We will use sex as the sensitive attribute and day_28_flg as the outcome.
# We choose threshold = 0.41 so that the overall FPR is around 5%.

# Evaluate Predictive Equality
eval_pred_equality(
  data = test_data,
  outcome = "day_28_flg",
  group = "gender",
  probs = "pred",
  cutoff = 0.41
)
```

---

eval_stats_parity *Examine Statistical Parity of a Model*

Description

This function assesses *statistical parity* - also known as *demographic parity* - in the predictions of a binary classifier across two groups defined by a sensitive attribute. Statistical parity compares the rate at which different groups receive a positive prediction, irrespective of the true outcome. It reports the Positive Prediction Rate (PPR) for each group, their differences, ratios, and bootstrap-based confidence regions.

Usage

```
eval_stats_parity(
  data,
  outcome,
  group,
  probs,
  cutoff = 0.5,
  confint = TRUE,
  bootstraps = 2500,
  alpha = 0.05,
  digits = 2,
  message = TRUE
)
```

Arguments

`data`	Data frame containing the outcome, predicted outcome, and sensitive attribute
`outcome`	Name of the outcome variable, it must be binary
`group`	Name of the sensitive attribute
`probs`	Name of the predicted outcome variable
`cutoff`	Threshold for the predicted outcome, default is 0.5
`confint`	Whether to compute 95% confidence interval, default is TRUE
`bootstraps`	Number of bootstrap samples, default is 2500
`alpha`	The 1 - significance level for the confidence interval, default is 0.05
`digits`	Number of digits to round the results to, default is 2
`message`	Whether to print the results, default is TRUE

Value

A list containing the following elements:

- PPR_Group1: Positive Prediction Rate for the first group
- PPR_Group2: Positive Prediction Rate for the second group
- PPR_Diff: Difference in Positive Prediction Rate
- PPR_Ratio: The ratio in Positive Prediction Rate between the two groups. If confidence intervals are computed (`confint` = `TRUE`):
- PPR_Diff_CI: A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the difference in Positive Prediction Rate
- PPR_Ratio_CI: A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the ratio in Positive Prediction Rate

See Also

`eval_cond_stats_parity`

Examples

```
library(fairmetrics)
library(dplyr)
library(magrittr)
library(randomForest)
data("mimic_preprocessed")
set.seed(123)
train_data <- mimic_preprocessed %>%
  dplyr::filter(dplyr::row_number() <= 700)
# Fit a random forest model
rf_model <- randomForest::randomForest(factor(day_28_flg) ~ ., data = train_data, ntree = 1000)
# Test the model on the remaining data
test_data <- mimic_preprocessed %>%
  dplyr::mutate(gender = ifelse(gender_num == 1, "Male", "Female")) %>%
  dplyr::filter(dplyr::row_number() > 700)
```

```
test_data$pred <- predict(rf_model, newdata = test_data, type = "prob")[, 2]

# Fairness evaluation
# We will use sex as the sensitive attribute and day_28_flg as the outcome.
# We choose threshold = 0.41 so that the overall FPR is around 5%.

# Evaluate Statistical Parity
eval_stats_parity(
  data = test_data,
  outcome = "day_28_flg",
  group = "gender",
  probs = "pred",
  cutoff = 0.41
)
```

`eval_treatment_equality`

Examine Treatment Equality of a Model

Description

This function evaluates *Treatment Equality*, a fairness criterion that assesses whether the ratio of false negatives to false positives is similar across groups (e.g., based on gender or race). Treatment Equality ensures that the model does not disproportionately favor or disadvantage any group in terms of the relative frequency of missed detections (false negatives) versus false alarms (false positives).

Usage

```
eval_treatment_equality(
  data,
  outcome,
  group,
  probs,
  cutoff = 0.5,
  confint = TRUE,
  alpha = 0.05,
  bootstraps = 2500,
  digits = 2,
  message = TRUE
)
```

Arguments

`data`	Data frame containing the outcome, predicted outcome, and sensitive attribute
`outcome`	Name of the outcome variable

group	Name of the sensitive attribute
probs	Predicted probabilities
cutoff	Cutoff value for the predicted probabilities
confint	Logical indicating whether to calculate confidence intervals
alpha	The 1 - significance level for the confidence interval, default is 0.05
bootstraps	Number of bootstraps to use for confidence intervals
digits	Number of digits to round the results to, default is 2
message	Whether to print the results, default is TRUE

Value

A list containing the following elements:

- False Negative / False Positive ratio for Group 1
- False Negative / False Positive ratio for Group 2
- Difference in False Negative / False Positive ratio
- Ratio in False Negative / False Positive ratio If confidence intervals are computed (`confint = TRUE`):
- A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the difference in False Negative / False Positive ratio
- A vector of length 2 containing the lower and upper bounds of the 95% confidence interval for the ratio in False Negative / False Positive ratio

See Also

eval_acc_parity, eval_bs_parity, eval_pos_pred_parity, eval_neg_pred_parity

Examples

```
library(fairmetrics)
library(dplyr)
library(magrittr)
library(randomForest)
# Data for tests
data("mimic_preprocessed")
set.seed(123)
train_data <- mimic_preprocessed %>%
  dplyr::filter(dplyr::row_number() <= 700)
# Fit a random forest model
rf_model <- randomForest::randomForest(factor(day_28_flg) ~ ., data = train_data, ntree = 1000)
# Test the model on the remaining data
test_data <- mimic_preprocessed %>%
  dplyr::mutate(gender = ifelse(gender_num == 1, "Male", "Female")) %>%
  dplyr::filter(dplyr::row_number() > 700)

test_data$pred <- predict(rf_model, newdata = test_data, type = "prob")[, 2]
```

```
# Fairness evaluation
# We will use sex as the sensitive attribute and day_28_flg as the outcome.

# Evaluate Treatment Equality
eval_treatment_equality(
  data = test_data,
  outcome = "day_28_flg",
  group = "gender",
  probs = "pred",
  cutoff = 0.41,
  confint = TRUE,
  alpha = 0.05,
  bootstraps = 2500,
  digits = 2,
  message = FALSE
)
```

`get_fairness_metrics` *Compute Fairness Metrics for Binary Classification*

Description

This function evaluates a comprehensive set of fairness metrics for binary classification models across groups defined by a sensitive attribute (e.g., race, gender). It returns a unified data frame containing metric values, optionally with bootstrap confidence intervals.

Usage

```
get_fairness_metrics(
  data,
  outcome,
  group,
  group2 = NULL,
  condition = NULL,
  probs,
  confint = TRUE,
  cutoff = 0.5,
  bootstraps = 2500,
  alpha = 0.05,
  digits = 2
)
```

Arguments

`data`	A data frame containing the outcome, group, and predicted probabilities.
`outcome`	The name of the column containing the true binary outcome.
`group`	The name of the column representing the sensitive attribute (e.g., race, gender).

`group2`	Define if conditional statistical parity is desired. Name of a secondary group variable used for conditional fairness analysis.
`condition`	Define if conditional statistical parity is desired. If the conditional group is categorical, the condition supplied must be a character of the levels to condition on. If the conditional group is continuous, the conditions supplied must be a character containing the sign of the condition and the value to threshold the continuous variable (e.g. "<50", ">50", "<=50", ">=50").
`probs`	The name of the column with predicted probabilities.
`confint`	Logical indicating whether to calculate confidence intervals.
`cutoff`	Numeric threshold for classification. Default is 0.5.
`bootstraps`	Number of bootstrap samples. Default is 2500.
`alpha`	Significance level for confidence intervals. Default is 0.05.
`digits`	Number of digits to round the metrics to. Default is 2.

Details

Metrics Included::

- **Statistical Parity**: Difference in positive prediction rates across groups.
- **Conditional Statistical Parity** *(if group2 and condition are specified)*: Parity conditioned on a second group and value.
- **Equal Opportunity**: Difference in true positive rates (TPR) across groups.
- **Predictive Equality**: Difference in false positive rates (FPR) across groups.
- **Balance for Positive Class**: Checks whether the predicted probability distributions for positive outcomes are similar across groups.
- **Balance for Negative Class**: Same as above, but for negative outcomes.
- **Positive Predictive Parity**: Difference in positive predictive values (precision) across groups.
- **Negative Predictive Parity**: Difference in negative predictive values across groups.
- **Brier Score Parity**: Difference in Brier scores across groups.
- **Overall Accuracy Parity**: Difference in overall accuracy across groups.
- **Treatment Equality**: Ratio of false negatives to false positives across groups.

Value

A data frame with the evaluated fairness metrics.

Examples

```
library(fairmetrics)
library(dplyr)
library(randomForest)
library(magrittr)
data("mimic_preprocessed")
set.seed(123)
train_data <- mimic_preprocessed %>%
  dplyr::filter(dplyr::row_number() <= 700)
# Fit a random forest model
```

```
rf_model <- randomForest::randomForest(factor(day_28_flg) ~ ., data = train_data, ntree = 1000)
# Test the model on the remaining data
test_data <- mimic_preprocessed %>%
  dplyr::mutate(gender = ifelse(gender_num == 1, "Male", "Female"))%>%
  dplyr::filter(dplyr::row_number() > 700)

test_data$pred <- predict(rf_model, newdata = test_data, type = "prob")[, 2]

# Fairness evaluation
# We will use sex as the sensitive attribute and day_28_flg as the outcome.
# We choose threshold = 0.41 so that the overall FPR is around 5%.

# Get Fairness Metrics
get_fairness_metrics(
 data = test_data,
 outcome = "day_28_flg",
 group = "gender",
 group2 = "age",
 condition = ">=60",
 probs = "pred",
 confint = TRUE,
 cutoff = 0.41,
 alpha = 0.05
)
```

mimic — *Clinical data from the MIMIC-II database for a case study on indwelling arterial catheters*

Description

The Indwelling Arterial Catheter Clinical dataset contains clinical data for 1776 patients from the MIMIC-II clinical database. It was the basis for the article: Hsu DJ, et al. The association between indwelling arterial catheters and mortality in hemodynamically stable patients with respiratory failure: A propensity score analysis. Chest, 148(6):1470–1476, Aug. 2015. This dataset was also used by Raffa et al. in Chapter 5 "Data Analysis" of the forthcoming book: Secondary Analysis of Electronic Health Records, published by Springer in 2016.

Usage

```
mimic
```

Format

A data frame with 1776 rows and 46 variables:

`aline_flg` Integer, indicates if IAC was used (1 = yes, 0 = no) `icu_los_day` Double, length of stay in ICU (days)

`hospital_los_day` Integer, length of stay in hospital (days)

`age` Double, age at baseline (years)

`gender_num` Integer, patient gender (1 = male; 0 = female)

`weight_first` Double, first weight (kg)

`bmi` Double, patient BMI

`sapsi_first` Integer, first SAPS I score

`sofa_first` Integer, first SOFA score

`service_unit` Character, type of service unit (FICU, MICU, SICU)

`service_num` Integer, service as a numeric value (0 = MICU or FICU, 1 = SICU)

`day_icu_intime` Character, day of week of ICU admission

`day_icu_intime_num` Integer, day of week of ICU admission (numeric)

`hour_icu_intime` Integer, hour of ICU admission (24hr clock)

`hosp_exp_flg` Integer, death in hospital (1 = yes, 0 = no)

`icu_exp_flg` Integer, death in ICU (1 = yes, 0 = no)

`day_28_flg` Integer, death within 28 days (1 = yes, 0 = no)

`mort_day_censored` Double, day post ICU admission of censoring or death (days)

`censor_flg` Integer, censored or death (0 = death, 1 = censored)

`sepsis_flg` Integer, sepsis present (0 = no, 1 = yes)

`chf_flg` Integer, congestive heart failure (0 = no, 1 = yes)

`afib_flg` Integer, atrial fibrillation (0 = no, 1 = yes)

`renal_flg` Integer, chronic renal disease (0 = no, 1 = yes)

`liver_flg` Integer, liver disease (0 = no, 1 = yes)

`copd_flg` Integer, chronic obstructive pulmonary disease (0 = no, 1 = yes)

`cad_flg` Integer, coronary artery disease (0 = no, 1 = yes)

`stroke_flg` Integer, stroke (0 = no, 1 = yes)

`mal_flg` Integer, malignancy (0 = no, 1 = yes)

`resp_flg` Integer, respiratory disease (non-COPD) (0 = no, 1 = yes)

`map_1st` Double, mean arterial pressure (mmHg)

`hr_1st` Integer, heart rate

`temp_1st` Double, temperature (F)

`spo2_1st` Integer, S_pO_2 (percent)

`abg_count` Integer, arterial blood gas count (number of tests)

`wbc_first` Double, first white blood cell count (K/uL)

`hgb_first` Double, first hemoglobin (g/dL)

`platelet_first` Integer, first platelets (K/u)

`sodium_first` Integer, first sodium (mEq/L)

`potassium_first`	Double, first potassium (mEq/L)
`tco2_first`	Double, first bicarbonate (mEq/L)
`chloride_first`	Integer, first chloride (mEq/L)
`bun_first`	Integer, first blood urea nitrogen (mg/dL)
`creatinine_first`	Double, first creatinine (mg/dL)
`po2_first`	Integer, first PaO_2 (mmHg)
`pco2_first`	Integer, first PaCO_2 (mmHg)
`iv_day_1`	Double, input fluids by IV on day 1 (mL)

Source

https://physionet.org/content/mimic2-iaccd/1.0/

`mimic_preprocessed` *Preprocessed Clinical Data from the MIMIC-II Database*

Description

This version of the `mimic` dataset has been cleaned by removing columns with more than 10% missing data, imputing remaining missing values with the median, and dropping columns highly correlated with the outcome. It is designed for use in fairness-aware machine learning tasks and streamlined analysis.

Usage

```
mimic_preprocessed
```

Format

A data frame with fewer variables than the original due to preprocessing. Number of rows: 1776.

Source

https://physionet.org/content/mimic2-iaccd/1.0/

See Also

`mimic`

Index