Package: AutoScore (via r-universe)

August 25, 2024

Type Package

Title An Interpretable Machine Learning-Based Automatic Clinical Score Generator

Version 1.0.0

Date 2022-08-27

URL https://github.com/nliulab/AutoScore

BugReports https://github.com/nliulab/AutoScore/issues

Description A novel interpretable machine learning-based framework to automate the development of a clinical scoring model for predefined outcomes. Our novel framework consists of six modules: variable ranking with machine learning, variable transformation, score derivation, model selection, domain knowledge-based score fine-tuning, and performance evaluation.The details are described in our research paper<doi:10.2196/21798>. Users or clinicians could seamlessly generate parsimonious sparse-score risk models (i.e., risk scores), which can be easily implemented and validated in clinical practice. We hope to see its application in various medical case studies.

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports tableone, pROC, randomForest, ggplot2, knitr, Hmisc, car, coxed, dplyr, ordinal, survival, tidyr, plotly, magrittr, randomForestSRC, rlang, survAUC, survminer

Depends R (>= 3.5.0)

VignetteBuilder knitr

Suggests rpart, rmarkdown

Repository https://nliulab.r-universe.dev

RemoteUrl https://github.com/nliulab/autoscore

RemoteRef HEAD

RemoteSha 1f89dc1b7002e62ef053dc139da8217d98171f10

Contents

`add_baseline` *Internal Function: Add baselines after second-step logistic regression (part of AutoScore Module 3)*

Description

Internal Function: Add baselines after second-step logistic regression (part of AutoScore Module 3)

Usage

```
add_baseline(df, coef_vec)
```

Arguments

df	A data.frame used for logistic regression
coef_vec	Generated from logistic regression

Value

Processed vector for generating the scoring table

`assign_score` *Internal Function: Automatically assign scores to each subjects given new data set and scoring table (Used for intermediate and final evaluation)*

Description

Internal Function: Automatically assign scores to each subjects given new data set and scoring table (Used for intermediate and final evaluation)

Usage

```
assign_score(df, score_table)
```

Arguments

df	A data.frame used for testing, where variables keep before categorization
score_table	A vector containing the scoring table

Value

Processed data.frame with assigned scores for each variables

AutoScore_fine_tuning *AutoScore STEP(iv): Fine-tune the score by revising cut_vec with domain knowledge (AutoScore Module 5)*

Description

Domain knowledge is essential in guiding risk model development. For continuous variables, the variable transformation is a data-driven process (based on "quantile" or "kmeans"). In this step, the automatically generated cutoff values for each continuous variable can be fine-tuned by combining, rounding, and adjusting according to the standard clinical norm. Revised `cut_vec` will be input with domain knowledge to update scoring table. User can choose any cut-off values/any number of categories. Then final Scoring table will be generated. Run `vignette("Guide_book", package = "AutoScore")` to see the guidebook or vignette.

Usage

```
AutoScore_fine_tuning(
  train_set,
  validation_set,
  final_variables,
  cut_vec,
  max_score = 100,
  metrics_ci = FALSE
)
```

Arguments

`train_set`	A processed `data.frame` that contains data to be analyzed, for training.
`validation_set`	A processed `data.frame` that contains data for validation purpose.
`final_variables`	A vector containing the list of selected variables, selected from Step(ii) `AutoScore_parsimony`. Run `vignette("Guide_book", package = "AutoScore")` to see the guidebook or vignette.
`cut_vec`	Generated from STEP(iii) `AutoScore_weighting`.Please follow the guidebook
`max_score`	Maximum total score (Default: 100).
`metrics_ci`	whether to calculate confidence interval for the metrics of sensitivity, specificity, etc.

Value

Generated final table of scoring model for downstream testing

References

- Xie F, Chakraborty B, Ong MEH, Goldstein BA, Liu N. AutoScore: A Machine Learning-Based Automatic Clinical Score Generator and Its Application to Mortality Prediction Using Electronic Health Records. JMIR Medical Informatics 2020;8(10):e21798

See Also

AutoScore_rank, AutoScore_parsimony, AutoScore_weighting, AutoScore_testing,Run `vignette("Guide_book", package = "AutoScore")` to see the guidebook or vignette.

Examples

```
## Please see the guidebook or vignettes
```

AutoScore_fine_tuning_Ordinal

AutoScore STEP(iv) for ordinal outcomes: Fine-tune the score by revising `cut_vec` *with domain knowledge (AutoScore Module 5)*

Description

Domain knowledge is essential in guiding risk model development. For continuous variables, the variable transformation is a data-driven process (based on "quantile" or "kmeans"). In this step, the automatically generated cutoff values for each continuous variable can be fine-tuned by combining, rounding, and adjusting according to the standard clinical norm. Revised `cut_vec` will be input with domain knowledge to update scoring table. User can choose any cut-off values/any number of categories. Then final Scoring table will be generated. Run `vignette("Guide_book", package = "AutoScore")` to see the guidebook or vignette.

Usage

```
AutoScore_fine_tuning_Ordinal(
  train_set,
  validation_set,
  final_variables,
  link = "logit",
  cut_vec,
  max_score = 100,
  n_boot = 100,
  report_cindex = FALSE
)
```

Arguments

`train_set`	A processed `data.frame` that contains data to be analyzed, for training.
`validation_set`	A processed `data.frame` that contains data for validation purpose.
`final_variables`	A vector containing the list of selected variables, selected from Step(ii) AutoScore_parsimony_Ordinal.
`link`	The link function used to model ordinal outcomes. Default is `"logit"` for proportional odds model. Other options are `"cloglog"` (proportional hazards model) and `"probit"`.
`cut_vec`	Generated from STEP(iii) AutoScore_weighting_Ordinal.

max_score	Maximum total score (Default: 100).
n_boot	Number of bootstrap cycles to compute 95% CI for performance metrics.
report_cindex	Whether to report generalized c-index for model evaluation (Default:FALSE for faster evaluation).

Value

Generated final table of scoring model for downstream testing

References

- Saffari SE, Ning Y, Feng X, Chakraborty B, Volovici V, Vaughan R, Ong ME, Liu N, AutoScore-Ordinal: An interpretable machine learning framework for generating scoring models for ordinal outcomes, arXiv:2202.08407

See Also

AutoScore_rank_Ordinal, AutoScore_parsimony_Ordinal, AutoScore_weighting_Ordinal, AutoScore_testing_Ordinal.

Examples

```
## Please see the guidebook or vignettes
```

AutoScore_fine_tuning_Survival

AutoScore STEP(iv) for survival outcomes: Fine-tune the score by revising cut_vec with domain knowledge (AutoScore Module 5)

Description

Domain knowledge is essential in guiding risk model development. For continuous variables, the variable transformation is a data-driven process (based on "quantile" or "kmeans"). In this step, the automatically generated cutoff values for each continuous variable can be fine-tuned by combining, rounding, and adjusting according to the standard clinical norm. Revised cut_vec will be input with domain knowledge to update scoring table. User can choose any cut-off values/any number of categories. Then final Scoring table will be generated. Run vignette("Guide_book", package = "AutoScore") to see the guidebook or vignette.

Usage

```
AutoScore_fine_tuning_Survival(
  train_set,
  validation_set,
  final_variables,
  cut_vec,
  max_score = 100,
  time_point = c(1, 3, 7, 14, 30, 60, 90)
)
```

Arguments

train_set	A processed data.frame that contains data to be analyzed, for training.
validation_set	A processed data.frame that contains data for validation purpose.
final_variables	A vector containing the list of selected variables, selected from Step(ii) AutoScore_parsimony. Run vignette("Guide_book", package = "AutoScore") to see the guidebook or vignette.
cut_vec	Generated from STEP(iii) AutoScore_weighting_Survival().Please follow the guidebook
max_score	Maximum total score (Default: 100).
time_point	The time points to be evaluated using time-dependent AUC(t).

Value

Generated final table of scoring model for downstream testing

References

- Xie F, Ning Y, Yuan H, et al. AutoScore-Survival: Developing interpretable machine learning-based time-to-event scores with right-censored survival data. J Biomed Inform. 2022;125:103959. doi:10.1016/j.jbi.2021.103959

See Also

AutoScore_rank_Survival, AutoScore_parsimony_Survival, AutoScore_weighting_Survival, AutoScore_testing_Survival.

Examples

```
## Please see the guidebook or vignettes
```

AutoScore_parsimony *AutoScore STEP(ii): Select the best model with parsimony plot (AutoScore Modules 2+3+4)*

Description

AutoScore STEP(ii): Select the best model with parsimony plot (AutoScore Modules 2+3+4)

Usage

```
AutoScore_parsimony(
  train_set,
  validation_set,
  rank,
  max_score = 100,
  n_min = 1,
  n_max = 20,
  cross_validation = FALSE,
  fold = 10,
  categorize = "quantile",
  quantiles = c(0, 0.05, 0.2, 0.8, 0.95, 1),
  max_cluster = 5,
  do_trace = FALSE,
  auc_lim_min = 0.5,
  auc_lim_max = "adaptive"
)
```

Arguments

train_set	A processed data.frame that contains data to be analyzed, for training.
validation_set	A processed data.frame that contains data for validation purpose.
rank	the raking result generated from AutoScore STEP(i) AutoScore_rank
max_score	Maximum total score (Default: 100).
n_min	Minimum number of selected variables (Default: 1).
n_max	Maximum number of selected variables (Default: 20).
cross_validation	If set to TRUE, cross-validation would be used for generating parsimony plot, which is suitable for small-size data. Default to FALSE
fold	The number of folds used in cross validation (Default: 10). Available if cross_validation = TRUE.
categorize	Methods for categorize continuous variables. Options include "quantile" or "kmeans" (Default: "quantile").
quantiles	Predefined quantiles to convert continuous variables to categorical ones. (Default: c(0, 0.05, 0.2, 0.8, 0.95, 1)) Available if categorize = "quantile".
max_cluster	The max number of cluster (Default: 5). Available if categorize = "kmeans".
do_trace	If set to TRUE, all results based on each fold of cross-validation would be printed out and plotted (Default: FALSE). Available if cross_validation = TRUE.
auc_lim_min	Min y_axis limit in the parsimony plot (Default: 0.5).
auc_lim_max	Max y_axis limit in the parsimony plot (Default: "adaptive").

Details

This is the second step of the general AutoScore workflow, to generate the parsimony plot to help select a parsimonious model. In this step, it goes through AutoScore Module 2,3 and 4 multiple times and to evaluate the performance under different variable list. The generated parsimony plot would give researcher an intuitive figure to choose the best models. If data size is small (ie, <5000), an independent validation set may not be a wise choice. Then, we suggest using cross-validation to maximize the utility of data. Set `cross_validation=TRUE`. Run `vignette("Guide_book", package = "AutoScore")` to see the guidebook or vignette.

Value

List of AUC value for different number of variables

References

- Xie F, Chakraborty B, Ong MEH, Goldstein BA, Liu N, AutoScore: A Machine Learning-Based Automatic Clinical Score Generator and Its Application to Mortality Prediction Using Electronic Health Records, JMIR Med Inform 2020;8(10):e21798, doi: 10.2196/21798

See Also

AutoScore_rank, AutoScore_weighting, AutoScore_fine_tuning, AutoScore_testing, Run `vignette("Guide_book", package = "AutoScore")` to see the guidebook or vignette.

Examples

```
# see AutoScore Guidebook for the whole 5-step workflow
data("sample_data")
names(sample_data)[names(sample_data) == "Mortality_inpatient"] <- "label"
out_split <- split_data(data = sample_data, ratio = c(0.7, 0.1, 0.2))
train_set <- out_split$train_set
validation_set <- out_split$validation_set
ranking <- AutoScore_rank(train_set, ntree=100)
AUC <- AutoScore_parsimony(
train_set,
validation_set,
rank = ranking,
max_score = 100,
n_min = 1,
n_max = 20,
categorize = "quantile",
quantiles = c(0, 0.05, 0.2, 0.8, 0.95, 1)
)
```

`AutoScore_parsimony_Ordinal` *AutoScore STEP(ii) for ordinal outcomes: Select the best model with parsimony plot (AutoScore Modules 2+3+4)*

Description

AutoScore STEP(ii) for ordinal outcomes: Select the best model with parsimony plot (AutoScore Modules 2+3+4)

Usage

```
AutoScore_parsimony_Ordinal(
  train_set,
  validation_set,
  rank,
  link = "logit",
  max_score = 100,
  n_min = 1,
  n_max = 20,
  cross_validation = FALSE,
  fold = 10,
  categorize = "quantile",
  quantiles = c(0, 0.05, 0.2, 0.8, 0.95, 1),
  max_cluster = 5,
  do_trace = FALSE,
  auc_lim_min = 0.5,
  auc_lim_max = "adaptive"
)
```

Arguments

`train_set`	A processed `data.frame` that contains data to be analyzed, for training.
`validation_set`	A processed `data.frame` that contains data for validation purpose.
`rank`	The raking result generated from AutoScore STEP(i) for ordinal outcomes (`AutoScore_rank_Ordinal`).
`link`	The link function used to model ordinal outcomes. Default is `"logit"` for proportional odds model. Other options are `"cloglog"` (proportional hazards model) and `"probit"`.
`max_score`	Maximum total score (Default: 100).
`n_min`	Minimum number of selected variables (Default: 1).
`n_max`	Maximum number of selected variables (Default: 20).
`cross_validation`	If set to `TRUE`, cross-validation would be used for generating parsimony plot, which is suitable for small-size data. Default to `FALSE`

fold	The number of folds used in cross validation (Default: 10). Available if cross_validation = TRUE.
categorize	Methods for categorize continuous variables. Options include "quantile" or "kmeans" (Default: "quantile").
quantiles	Predefined quantiles to convert continuous variables to categorical ones. (Default: c(0, 0.05, 0.2, 0.8, 0.95, 1)) Available if categorize = "quantile".
max_cluster	The max number of cluster (Default: 5). Available if categorize = "kmeans".
do_trace	If set to TRUE, all results based on each fold of cross-validation would be printed out and plotted (Default: FALSE). Available if cross_validation = TRUE.
auc_lim_min	Min y_axis limit in the parsimony plot (Default: 0.5).
auc_lim_max	Max y_axis limit in the parsimony plot (Default: "adaptive").

Details

This is the second step of the general AutoScore workflow for ordinal outcomes, to generate the parsimony plot to help select a parsimonious model. In this step, it goes through AutoScore Module 2,3 and 4 multiple times and to evaluate the performance under different variable list. The generated parsimony plot would give researcher an intuitive figure to choose the best models. If data size is small (eg, <5000), an independent validation set may not be a wise choice. Then, we suggest using cross-validation to maximize the utility of data. Set cross_validation=TRUE.

Value

List of mAUC (ie, the average AUC of dichotomous classifications) value for different number of variables

References

- Saffari SE, Ning Y, Feng X, Chakraborty B, Volovici V, Vaughan R, Ong ME, Liu N, AutoScore-Ordinal: An interpretable machine learning framework for generating scoring models for ordinal outcomes, arXiv:2202.08407

See Also

AutoScore_rank_Ordinal, AutoScore_weighting_Ordinal, AutoScore_fine_tuning_Ordinal, AutoScore_testing_Ordinal.

Examples

```
## Not run:
# see AutoScore-Ordinal Guidebook for the whole 5-step workflow
data("sample_data_ordinal") # Output is named `label`
out_split <- split_data(data = sample_data_ordinal, ratio = c(0.7, 0.1, 0.2))
train_set <- out_split$train_set
validation_set <- out_split$validation_set
ranking <- AutoScore_rank_Ordinal(train_set, ntree=100)
mAUC <- AutoScore_parsimony_Ordinal(
```

```
  train_set = train_set, validation_set = validation_set,
  rank = ranking, max_score = 100, n_min = 1, n_max = 20,
  categorize = "quantile", quantiles = c(0, 0.05, 0.2, 0.8, 0.95, 1)
)

## End(Not run)
```

`AutoScore_parsimony_Survival` *AutoScore STEP(ii) for survival outcomes: Select the best model with parsimony plot (AutoScore Modules 2+3+4)*

Description

AutoScore STEP(ii) for survival outcomes: Select the best model with parsimony plot (AutoScore Modules 2+3+4)

Usage

```
AutoScore_parsimony_Survival(
  train_set,
  validation_set,
  rank,
  max_score = 100,
  n_min = 1,
  n_max = 20,
  cross_validation = FALSE,
  fold = 10,
  categorize = "quantile",
  quantiles = c(0, 0.05, 0.2, 0.8, 0.95, 1),
  max_cluster = 5,
  do_trace = FALSE,
  auc_lim_min = 0.5,
  auc_lim_max = "adaptive"
)
```

Arguments

`train_set`	A processed `data.frame` that contains data to be analyzed, for training.
`validation_set`	A processed `data.frame` that contains data for validation purpose.
`rank`	the raking result generated from AutoScore STEP(i) for survival outcomes (`AutoScore_rank_Survival`).
`max_score`	Maximum total score (Default: 100).
`n_min`	Minimum number of selected variables (Default: 1).
`n_max`	Maximum number of selected variables (Default: 20).
`cross_validation`	If set to `TRUE`, cross-validation would be used for generating parsimony plot, which is suitable for small-size data. Default to `FALSE`

fold	The number of folds used in cross validation (Default: 10). Available if cross_validation = TRUE.
categorize	Methods for categorize continuous variables. Options include "quantile" or "kmeans" (Default: "quantile").
quantiles	Predefined quantiles to convert continuous variables to categorical ones. (Default: c(0, 0.05, 0.2, 0.8, 0.95, 1)) Available if categorize = "quantile".
max_cluster	The max number of cluster (Default: 5). Available if categorize = "kmeans".
do_trace	If set to TRUE, all results based on each fold of cross-validation would be printed out and plotted (Default: FALSE). Available if cross_validation = TRUE.
auc_lim_min	Min y_axis limit in the parsimony plot (Default: 0.5).
auc_lim_max	Max y_axis limit in the parsimony plot (Default: "adaptive").

Details

This is the second step of the general AutoScore-Survival workflow for ordinal outcomes, to generate the parsimony plot to help select a parsimonious model. In this step, it goes through AutoScore-Survival Module 2,3 and 4 multiple times and to evaluate the performance under different variable list. The generated parsimony plot would give researcher an intuitive figure to choose the best models. If data size is small (eg, <5000), an independent validation set may not be a wise choice. Then, we suggest using cross-validation to maximize the utility of data. Set cross_validation=TRUE.

Value

List of iAUC (ie, the integrated AUC by integral under a time-dependent AUC curve for different number of variables

References

- Xie F, Ning Y, Yuan H, et al. AutoScore-Survival: Developing interpretable machine learning-based time-to-event scores with right-censored survival data. J Biomed Inform. 2022;125:103959. doi:10.1016/j.jbi.2021.103959

See Also

AutoScore_rank_Survival, AutoScore_weighting_Survival, AutoScore_fine_tuning_Survival, AutoScore_testing_Survival.

Examples

```
## Not run:
# see AutoScore-Survival Guidebook for the whole 5-step workflow
data("sample_data_survival")
out_split <- split_data(data = sample_data_survival, ratio = c(0.7, 0.1, 0.2))
train_set <- out_split$train_set
validation_set <- out_split$validation_set
ranking <- AutoScore_rank_Survival(train_set, ntree=10)
iAUC <- AutoScore_parsimony_Survival(
```

```
  train_set = train_set, validation_set = validation_set,
  rank = ranking, max_score = 100, n_min = 1, n_max = 20,
  categorize = "quantile", quantiles = c(0, 0.05, 0.2, 0.8, 0.95, 1)
)

## End(Not run)
```

`AutoScore_rank` *AutoScore STEP(i): Rank variables with machine learning (AutoScore Module 1)*

Description

AutoScore STEP(i): Rank variables with machine learning (AutoScore Module 1)

Usage

```
AutoScore_rank(train_set, validation_set = NULL, method = "rf", ntree = 100)
```

Arguments

`train_set`	A processed `data.frame` that contains data to be analyzed, for training.
`validation_set`	A processed `data.frame` that contains data to be analyzed, only for auc-based ranking.
`method`	method for ranking. Options: 1. 'rf' - random forest (default), 2. 'auc' - auc-based (required validation set). For "auc", univariate models will be built based on the train set, and the variable ranking is constructed via the AUC performance of corresponding univariate models on the validation set ('validation_set').
`ntree`	Number of trees in the random forest (Default: 100).

Details

The first step in the AutoScore framework is variable ranking. We use random forest (RF), an ensemble machine learning algorithm, to identify the top-ranking predictors for subsequent score generation. This step correspond to Module 1 in the AutoScore paper.

Value

Returns a vector containing the list of variables and its ranking generated by machine learning (random forest)

References

- Breiman, L. (2001), Random Forests, Machine Learning 45(1), 5-32
- Xie F, Chakraborty B, Ong MEH, Goldstein BA, Liu N. AutoScore: A Machine Learning-Based Automatic Clinical Score Generator and Its Application to Mortality Prediction Using Electronic Health Records. JMIR Medical Informatics 2020;8(10):e21798

See Also

AutoScore_parsimony, AutoScore_weighting, AutoScore_fine_tuning, AutoScore_testing, Run vignette("Guide_book", package = "AutoScore") to see the guidebook or vignette.

Examples

```
# see AutoScore Guidebook for the whole 5-step workflow
data("sample_data")
names(sample_data)[names(sample_data) == "Mortality_inpatient"] <- "label"
ranking <- AutoScore_rank(sample_data, ntree = 50)
```

AutoScore_rank_Ordinal

AutoScore STEP (i) for ordinal outcomes: Generate variable ranking list by machine learning (AutoScore Module 1)

Description

AutoScore STEP (i) for ordinal outcomes: Generate variable ranking list by machine learning (AutoScore Module 1)

Usage

```
AutoScore_rank_Ordinal(train_set, ntree = 100)
```

Arguments

train_set	A processed data.frame that contains data to be analyzed, for training.
ntree	Number of trees in the random forest (Default: 100).

Details

The first step in the AutoScore framework is variable ranking. We use random forest (RF) for multiclass classification to identify the top-ranking predictors for subsequent score generation. This step corresponds to Module 1 in the AutoScore-Ordinal paper.

Value

Returns a vector containing the list of variables and its ranking generated by machine learning (random forest)

References

- Breiman, L. (2001), Random Forests, Machine Learning 45(1), 5-32
- Saffari SE, Ning Y, Feng X, Chakraborty B, Volovici V, Vaughan R, Ong ME, Liu N, AutoScore-Ordinal: An interpretable machine learning framework for generating scoring models for ordinal outcomes, arXiv:2202.08407

See Also

AutoScore_parsimony_Ordinal, AutoScore_weighting_Ordinal, AutoScore_fine_tuning_Ordinal, AutoScore_testing_Ordinal.

Examples

```
## Not run:
# see AutoScore-Ordinal Guidebook for the whole 5-step workflow
data("sample_data_ordinal") # Output is named `label`
ranking <- AutoScore_rank_ordinal(sample_data_ordinal, ntree = 50)

## End(Not run)
```

AutoScore_rank_Survival

AutoScore STEP (1) for survival outcomes: Generate variable ranking List by machine learning (Random Survival Forest) (AutoScore Module 1)

Description

AutoScore STEP (1) for survival outcomes: Generate variable ranking List by machine learning (Random Survival Forest) (AutoScore Module 1)

Usage

```
AutoScore_rank_Survival(train_set, ntree = 50)
```

Arguments

train_set	A processed data.frame that contains data to be analyzed, for training.
ntree	Number of trees in the random forest (Default: 100).

Details

The first step in the AutoScore framework is variable ranking. We use Random Survival Forest (RSF) for survival outcome to identify the top-ranking predictors for subsequent score generation. This step correspond to Module 1 in the AutoScore-Survival paper.

Value

Returns a vector containing the list of variables and its ranking generated by machine learning (random forest)

References

- Ishwaran, H., Kogalur, U. B., Blackstone, E. H., & Lauer, M. S. (2008). Random survival forests. The annals of applied statistics, 2(3), 841-860.
- Xie F, Ning Y, Yuan H, et al. AutoScore-Survival: Developing interpretable machine learning-based time-to-event scores with right-censored survival data. J Biomed Inform. 2022;125:103959. doi:10.1016/j.jbi.2021.103959

See Also

AutoScore_parsimony_Survival, AutoScore_weighting_Survival, AutoScore_fine_tuning_Survival, AutoScore_testing_Survival.

Examples

```
## Not run:
# see AutoScore-Survival Guidebook for the whole 5-step workflow
data("sample_data_survival") # Output is named `label_time` and `label_status`
ranking <- AutoScore_rank_Survival(sample_data_survival, ntree = 50)

## End(Not run)
```

AutoScore_testing *AutoScore STEP(v): Evaluate the final score with ROC analysis (AutoScore Module 6)*

Description

AutoScore STEP(v): Evaluate the final score with ROC analysis (AutoScore Module 6)

Usage

```
AutoScore_testing(
  test_set,
  final_variables,
  cut_vec,
  scoring_table,
  threshold = "best",
  with_label = TRUE,
  metrics_ci = TRUE
)
```

Arguments

test_set	A processed data.frame that contains data for testing purpose. This data.frame should have same format as train_set (same variable names and outcomes)

`final_variables`
: A vector containing the list of selected variables, selected from Step(ii) `AutoScore_parsimony`. Run `vignette("Guide_book", package = "AutoScore")` to see the guidebook or vignette.

`cut_vec`
: Generated from STEP(iii) `AutoScore_weighting`.Please follow the guidebook

`scoring_table`
: The final scoring table after fine-tuning, generated from STEP(iv) `AutoScore_fine_tuning`.Please follow the guidebook

`threshold`
: Score threshold for the ROC analysis to generate sensitivity, specificity, etc. If set to "best", the optimal threshold will be calculated (Default:"best").

`with_label`
: Set to TRUE if there are labels in the test_set and performance will be evaluated accordingly (Default:TRUE). Set it to "FALSE" if there are not "label" in the "test_set" and the final predicted scores will be the output without performance evaluation.

`metrics_ci`
: whether to calculate confidence interval for the metrics of sensitivity, specificity, etc.

Value

A data frame with predicted score and the outcome for downstream visualization.

References

- Xie F, Chakraborty B, Ong MEH, Goldstein BA, Liu N. AutoScore: A Machine Learning-Based Automatic Clinical Score Generator and Its Application to Mortality Prediction Using Electronic Health Records. JMIR Medical Informatics 2020;8(10):e21798

See Also

`AutoScore_rank`, `AutoScore_parsimony`, `AutoScore_weighting`, `AutoScore_fine_tuning`, `print_roc_performance`, Run `vignette("Guide_book", package = "AutoScore")` to see the guidebook or vignette.

Examples

```
## Please see the guidebook or vignettes
```

AutoScore_testing_Ordinal

AutoScore STEP(v) for ordinal outcomes: Evaluate the final score (AutoScore Module 6)

Description

AutoScore STEP(v) for ordinal outcomes: Evaluate the final score (AutoScore Module 6)

Usage

```
AutoScore_testing_Ordinal(
  test_set,
  final_variables,
  link = "logit",
  cut_vec,
  scoring_table,
  with_label = TRUE,
  n_boot = 100
)
```

Arguments

test_set	A processed data.frame that contains data for testing purpose. This data.frame should have same format as train_set (same variable names and outcomes)
final_variables	A vector containing the list of selected variables, selected from Step(ii) AutoScore_parsimony_Ordinal.
link	The link function used to model ordinal outcomes. Default is "logit" for proportional odds model. Other options are "cloglog" (proportional hazards model) and "probit".
cut_vec	Generated from STEP(iii) AutoScore_weighting_Ordinal.
scoring_table	The final scoring table after fine-tuning, generated from STEP(iv) AutoScore_fine_tuning_Ordinal.Ple follow the guidebook
with_label	Set to TRUE if there are labels in the test_set and performance will be evaluated accordingly (Default:TRUE).
n_boot	Number of bootstrap cycles to compute 95% CI for performance metrics.

Value

A data frame with predicted score and the outcome for downstream visualization.

References

- Saffari SE, Ning Y, Feng X, Chakraborty B, Volovici V, Vaughan R, Ong ME, Liu N, AutoScore-Ordinal: An interpretable machine learning framework for generating scoring models for ordinal outcomes, arXiv:2202.08407

See Also

AutoScore_rank_Ordinal, AutoScore_parsimony_Ordinal, AutoScore_weighting_Ordinal, AutoScore_fine_tuning_Ordinal.

Examples

```
## Please see the guidebook or vignettes
```

`AutoScore_testing_Survival` *AutoScore STEP(v) for survival outcomes: Evaluate the final score with ROC analysis (AutoScore Module 6)*

Description

AutoScore STEP(v) for survival outcomes: Evaluate the final score with ROC analysis (AutoScore Module 6)

Usage

```
AutoScore_testing_Survival(
  test_set,
  final_variables,
  cut_vec,
  scoring_table,
  threshold = "best",
  with_label = TRUE,
  time_point = c(1, 3, 7, 14, 30, 60, 90)
)
```

Arguments

`test_set`	A processed `data.frame` that contains data for testing purpose. This `data.frame` should have same format as `train_set` (same variable names and outcomes)
`final_variables`	A vector containing the list of selected variables, selected from Step(ii) `AutoScore_parsimony`. Run `vignette("Guide_book", package = "AutoScore")` to see the guidebook or vignette.
`cut_vec`	Generated from STEP(iii) `AutoScore_weighting_Survival()`.Please follow the guidebook
`scoring_table`	The final scoring table after fine-tuning, generated from STEP(iv) `AutoScore_fine_tuning`.Please follow the guidebook
`threshold`	Score threshold for the ROC analysis to generate sensitivity, specificity, etc. If set to "best", the optimal threshold will be calculated (Default:"best").
`with_label`	Set to TRUE if there are labels(‘label_time‘ and ‘label_status‘) in the test_set and performance will be evaluated accordingly (Default:TRUE).
`time_point`	The time points to be evaluated using time-dependent AUC(t).

Value

A data frame with predicted score and the outcome for downstream visualization.

References

- Xie F, Ning Y, Yuan H, et al. AutoScore-Survival: Developing interpretable machine learning-based time-to-event scores with right-censored survival data. J Biomed Inform. 2022;125:103959. doi:10.1016/j.jbi.2021.103959

See Also

AutoScore_rank_Survival, AutoScore_parsimony_Survival, AutoScore_weighting_Survival, AutoScore_fine_tuning_Survival.

Examples

```
## Please see the guidebook or vignettes
```

AutoScore_weighting *AutoScore STEP(iii): Generate the initial score with the final list of variables (Re-run AutoScore Modules 2+3)*

Description

AutoScore STEP(iii): Generate the initial score with the final list of variables (Re-run AutoScore Modules 2+3)

Usage

```
AutoScore_weighting(
  train_set,
  validation_set,
  final_variables,
  max_score = 100,
  categorize = "quantile",
  max_cluster = 5,
  quantiles = c(0, 0.05, 0.2, 0.8, 0.95, 1),
  metrics_ci = FALSE
)
```

Arguments

train_set	A processed data.frame that contains data to be analyzed, for training.
validation_set	A processed data.frame that contains data for validation purpose.
final_variables	A vector containing the list of selected variables, selected from Step(ii)AutoScore_parsimony. Run vignette("Guide_book", package = "AutoScore") to see the guidebook or vignette.
max_score	Maximum total score (Default: 100).

categorize	Methods for categorize continuous variables. Options include "quantile" or "kmeans" (Default: "quantile").
max_cluster	The max number of cluster (Default: 5). Available if categorize = "kmeans".
quantiles	Predefined quantiles to convert continuous variables to categorical ones. (Default: c(0, 0.05, 0.2, 0.8, 0.95, 1)) Available if categorize = "quantile".
metrics_ci	whether to calculate confidence interval for the metrics of sensitivity, specificity, etc.

Value

Generated cut_vec for downstream fine-tuning process STEP(iv) AutoScore_fine_tuning.

References

- Xie F, Chakraborty B, Ong MEH, Goldstein BA, Liu N. AutoScore: A Machine Learning-Based Automatic Clinical Score Generator and Its Application to Mortality Prediction Using Electronic Health Records. JMIR Medical Informatics 2020;8(10):e21798

See Also

AutoScore_rank, AutoScore_parsimony, AutoScore_fine_tuning, AutoScore_testing, Run vignette("Guide_book", package = "AutoScore") to see the guidebook or vignette.

AutoScore_weighting_Ordinal

AutoScore STEP(iii) for ordinal outcomes: Generate the initial score with the final list of variables (Re-run AutoScore Modules 2+3)

Description

AutoScore STEP(iii) for ordinal outcomes: Generate the initial score with the final list of variables (Re-run AutoScore Modules 2+3)

Usage

```
AutoScore_weighting_Ordinal(
  train_set,
  validation_set,
  final_variables,
  link = "logit",
  max_score = 100,
  categorize = "quantile",
  quantiles = c(0, 0.05, 0.2, 0.8, 0.95, 1),
  max_cluster = 5,
  n_boot = 100
)
```

Arguments

train_set	A processed data.frame that contains data to be analyzed, for training.
validation_set	A processed data.frame that contains data for validation purpose.
final_variables	A vector containing the list of selected variables, selected from Step(ii) AutoScore_parsimony_Ordinal.
link	The link function used to model ordinal outcomes. Default is "logit" for proportional odds model. Other options are "cloglog" (proportional hazards model) and "probit".
max_score	Maximum total score (Default: 100).
categorize	Methods for categorize continuous variables. Options include "quantile" or "kmeans" (Default: "quantile").
quantiles	Predefined quantiles to convert continuous variables to categorical ones. (Default: c(0, 0.05, 0.2, 0.8, 0.95, 1)) Available if categorize = "quantile".
max_cluster	The max number of cluster (Default: 5). Available if categorize = "kmeans".
n_boot	Number of bootstrap cycles to compute 95% CI for performance metrics.

Value

Generated cut_vec for downstream fine-tuning process STEP(iv) AutoScore_fine_tuning_Ordinal.

References

- Saffari SE, Ning Y, Feng X, Chakraborty B, Volovici V, Vaughan R, Ong ME, Liu N, AutoScore-Ordinal: An interpretable machine learning framework for generating scoring models for ordinal outcomes, arXiv:2202.08407

See Also

AutoScore_rank_Ordinal, AutoScore_parsimony_Ordinal, AutoScore_fine_tuning_Ordinal, AutoScore_testing_Ordinal.

Examples

```
## Not run:
data("sample_data_ordinal") # Output is named `label`
out_split <- split_data(data = sample_data_ordinal, ratio = c(0.7, 0.1, 0.2))
train_set <- out_split$train_set
validation_set <- out_split$validation_set
ranking <- AutoScore_rank_Ordinal(train_set, ntree=100)
num_var <- 6
final_variables <- names(ranking[1:num_var])
cut_vec <- AutoScore_weighting_Ordinal(
  train_set = train_set, validation_set = validation_set,
  final_variables = final_variables, max_score = 100,
  categorize = "quantile", quantiles = c(0, 0.05, 0.2, 0.8, 0.95, 1)
)

## End(Not run)
```

AutoScore_weighting_Survival

AutoScore STEP(iii) for survival outcomes: Generate the initial score with the final list of variables (Re-run AutoScore Modules 2+3)

Description

AutoScore STEP(iii) for survival outcomes: Generate the initial score with the final list of variables (Re-run AutoScore Modules 2+3)

Usage

```
AutoScore_weighting_Survival(
  train_set,
  validation_set,
  final_variables,
  max_score = 100,
  categorize = "quantile",
  max_cluster = 5,
  quantiles = c(0, 0.05, 0.2, 0.8, 0.95, 1),
  time_point = c(1, 3, 7, 14, 30, 60, 90)
)
```

Arguments

train_set	A processed data.frame that contains data to be analyzed, for training.
validation_set	A processed data.frame that contains data for validation purpose.
final_variables	A vector containing the list of selected variables, selected from Step(ii)AutoScore_parsimony. Run vignette("Guide_book", package = "AutoScore") to see the guidebook or vignette.
max_score	Maximum total score (Default: 100).
categorize	Methods for categorize continuous variables. Options include "quantile" or "kmeans" (Default: "quantile").
max_cluster	The max number of cluster (Default: 5). Available if categorize = "kmeans".
quantiles	Predefined quantiles to convert continuous variables to categorical ones. (Default: c(0, 0.05, 0.2, 0.8, 0.95, 1)) Available if categorize = "quantile".
time_point	The time points to be evaluated using time-dependent AUC(t).

Value

Generated cut_vec for downstream fine-tuning process STEP(iv) AutoScore_fine_tuning.

References

- Xie F, Ning Y, Yuan H, et al. AutoScore-Survival: Developing interpretable machine learning-based time-to-event scores with right-censored survival data. J Biomed Inform. 2022;125:103959. doi:10.1016/j.jbi.2021.103959

See Also

AutoScore_rank_Survival, AutoScore_parsimony_Survival, AutoScore_fine_tuning_Survival, AutoScore_testing_Survival.

Examples

```
## Not run:
data("sample_data_survival") #
out_split <- split_data(data = sample_data_survival, ratio = c(0.7, 0.1, 0.2))
train_set <- out_split$train_set
validation_set <- out_split$validation_set
ranking <- AutoScore_rank_Survival(train_set, ntree=5)
num_var <- 6
final_variables <- names(ranking[1:num_var])
cut_vec <- AutoScore_weighting_Survival(
  train_set = train_set, validation_set = validation_set,
  final_variables = final_variables, max_score = 100,
  categorize = "quantile", quantiles = c(0, 0.05, 0.2, 0.8, 0.95, 1),
  time_point = c(1,3,7,14,30,60,90)
)

## End(Not run)
```

---

change_reference *Internal Function: Change Reference category after first-step logistic regression (part of AutoScore Module 3)*

Description

Internal Function: Change Reference category after first-step logistic regression (part of AutoScore Module 3)

Usage

```
change_reference(df, coef_vec)
```

Arguments

df	A data.frame used for logistic regression
coef_vec	Generated from logistic regression

Value

Processed `data.frame` after changing reference category

`check_data` *AutoScore function for datasets with binary outcomes: Check whether the input dataset fulfill the requirement of the AutoScore*

Description

AutoScore function for datasets with binary outcomes: Check whether the input dataset fulfill the requirement of the AutoScore

Usage

```
check_data(data)
```

Arguments

`data`	The data to be checked

Value

No return value, the result of the checking will be printed out.

Examples

```
data("sample_data")
names(sample_data)[names(sample_data) == "Mortality_inpatient"] <- "label"
check_data(sample_data)
```

`check_data_ordinal` *AutoScore function for ordinal outcomes: Check whether the input dataset fulfil the requirement of the AutoScore*

Description

AutoScore function for ordinal outcomes: Check whether the input dataset fulfil the requirement of the AutoScore

Usage

```
check_data_ordinal(data)
```

Arguments

`data`	The data to be checked

Value

No return value, the result of the checking will be printed out.

Examples

```
data("sample_data_ordinal")
check_data_ordinal(sample_data_ordinal)
```

`check_data_survival` *AutoScore function for survival data: Check whether the input dataset fulfill the requirement of the AutoScore*

Description

AutoScore function for survival data: Check whether the input dataset fulfill the requirement of the AutoScore

Usage

```
check_data_survival(data)
```

Arguments

`data`	The data to be checked

Value

No return value, the result of the checking will be printed out.

Examples

```
data("sample_data_survival")
check_data_survival(sample_data_survival)
```

`check_link` *Internal function: Check link function*

Description

Internal function: Check link function

Usage

```
check_link(link)
```

Arguments

link	The link function used to model ordinal outcomes. Default is "logit" for proportional odds model. Other options are "cloglog" (proportional hazards model) and "probit".

---

check_predictor *Internal function: Check predictors*

Description

Internal function: Check predictors

Usage

```
check_predictor(data_predictor)
```

Arguments

data_predictor	Predictors to be checked

Value

No return value, the result of the checking will be printed out.

---

compute_auc_val *Internal function: Compute AUC based on validation set for plotting parsimony (AutoScore Module 4)*

Description

Compute AUC based on validation set for plotting parsimony

Usage

```
compute_auc_val(
  train_set_1,
  validation_set_1,
  variable_list,
  categorize,
  quantiles,
  max_cluster,
  max_score
)
```

Arguments

train_set_1	Processed training set
validation_set_1	Processed validation set
variable_list	List of included variables
categorize	Methods for categorize continuous variables. Options include "quantile" or "kmeans"
quantiles	Predefined quantiles to convert continuous variables to categorical ones. Available if categorize = "quantile".
max_cluster	The max number of cluster (Default: 5). Available if categorize = "kmeans".
max_score	Maximum total score

Value

A List of AUC for parsimony plot

---

compute_auc_val_ord *Internal function: Compute mean AUC for ordinal outcomes based on validation set for plotting parsimony*

Description

Compute mean AUC based on validation set for plotting parsimony

Usage

```
compute_auc_val_ord(
  train_set_1,
  validation_set_1,
  variable_list,
  link,
  categorize,
  quantiles,
  max_cluster,
  max_score
)
```

Arguments

train_set_1	Processed training set
validation_set_1	Processed validation set
variable_list	List of included variables

link	The link function used to model ordinal outcomes. Default is "logit" for proportional odds model. Other options are "cloglog" (proportional hazards model) and "probit".
categorize	Methods for categorize continuous variables. Options include "quantile" or "kmeans"
quantiles	Predefined quantiles to convert continuous variables to categorical ones. Available if categorize = "quantile".
max_cluster	The max number of cluster (Default: 5). Available if categorize = "kmeans".
max_score	Maximum total score

Value

A list of mAUC for parsimony plot

---

compute_auc_val_survival

Internal function for survival outcomes: Compute AUC based on validation set for plotting parsimony

---

Description

Compute AUC based on validation set for plotting parsimony (survival outcomes)

Usage

```
compute_auc_val_survival(
  train_set_1,
  validation_set_1,
  variable_list,
  categorize,
  quantiles,
  max_cluster,
  max_score
)
```

Arguments

train_set_1	Processed training set
validation_set_1	Processed validation set
variable_list	List of included variables
categorize	Methods for categorize continuous variables. Options include "quantile" or "kmeans"
quantiles	Predefined quantiles to convert continuous variables to categorical ones. Available if categorize = "quantile".
max_cluster	The max number of cluster (Default: 5). Available if categorize = "kmeans".
max_score	Maximum total score

Value

A List of AUC for parsimony plot

`compute_descriptive_table` *AutoScore function: Descriptive Analysis*

Description

Compute descriptive table (usually Table 1 in the medical literature) for the dataset.

Usage

```
compute_descriptive_table(df, ...)
```

Arguments

`df`	data frame after checking and fulfilling the requirement of AutoScore
`...`	additional parameters to pass to `print.TableOne` and `kable`.

Value

No return value and the result of the descriptive analysis will be printed out.

Examples

```
data("sample_data")
names(sample_data)[names(sample_data) == "Mortality_inpatient"] <- "label"
compute_descriptive_table(sample_data)
# Report median and IQR (instead of default mean and SD) for Age, and add a
# caption to printed table:
compute_descriptive_table(sample_data, nonnormal = "Age",
                          caption = "Table 1. Patient characteristics")
```

`compute_final_score_ord` *Internal function: Compute risk scores for ordinal data given variables selected, cut-off values and scoring table*

Description

Internal function: Compute risk scores for ordinal data given variables selected, cut-off values and scoring table

Usage

```
compute_final_score_ord(data, final_variables, cut_vec, scoring_table)
```

Arguments

data	A processed data.frame that contains data for validation or testing purpose. This data.frame must have variable label and should have same format as train_set (same variable names and outcomes)
final_variables	A vector containing the list of selected variables, selected from Step(ii) AutoScore_parsimony_Ordinal.
cut_vec	Generated from STEP(iii) AutoScore_weighting_Ordinal.
scoring_table	The final scoring table after fine-tuning, generated from STEP(iv) AutoScore_fine_tuning_Ordinal.Ple follow the guidebook

---

compute_mauc_ord *Internal function: Compute mAUC for ordinal predictions*

Description

Internal function: Compute mAUC for ordinal predictions

Usage

```
compute_mauc_ord(y, fx)
```

Arguments

y	An ordered factor representing the ordinal outcome, with length n and J categories.
fx	Either (i) a numeric vector of predictor (e.g., predicted scores) of length n or (ii) a numeric matrix of predicted cumulative probabilities with n rows and (J-1) columns.

Value

The mean AUC of J-1 cumulative AUCs (i.e., when evaluating the prediction of Y<=j, j=1,...,J-1).

compute_multi_variable_table

AutoScore function: Multivariate Analysis

Description

Generate tables for multivariate analysis

Usage

```
compute_multi_variable_table(df)
```

Arguments

df	data frame after checking

Value

result of the multivariate analysis

Examples

```
data("sample_data")
names(sample_data)[names(sample_data) == "Mortality_inpatient"] <- "label"
multi_table<-compute_multi_variable_table(sample_data)
```

compute_multi_variable_table_ordinal

AutoScore-Ordinal function: Multivariate Analysis

Description

Generate tables for multivariate analysis

Usage

```
compute_multi_variable_table_ordinal(df, link = "logit", n_digits = 3)
```

Arguments

df	data frame after checking
link	The link function used to model ordinal outcomes. Default is "logit" for proportional odds model. Other options are "cloglog" (proportional hazards model) and "probit".
n_digits	Number of digits to print for OR or exponentiated coefficients (Default:3).

Value

result of the multivariate analysis

Examples

```
data("sample_data_ordinal")
# Using just a few variables to demonstrate usage:
multi_table<-compute_multi_variable_table_ordinal(sample_data_ordinal[, 1:3])
```

---

compute_multi_variable_table_survival

AutoScore function for survival outcomes: Multivariate Analysis

---

Description

Generate tables for multivariate analysis for survival outcomes

Usage

```
compute_multi_variable_table_survival(df)
```

Arguments

df	data frame after checking

Value

result of the multivariate analysis for survival outcomes

Examples

```
data("sample_data_survival")
multi_table<-compute_multi_variable_table_survival(sample_data_survival)
```

---

compute_prob_observed

Internal function: Based on given labels and scores, compute proportion of subjects observed in each outcome category in given score intervals.

Description

Internal function: Based on given labels and scores, compute proportion of subjects observed in each outcome category in given score intervals.

Usage

```
compute_prob_observed(
  pred_score,
  link = "logit",
  max_score = 100,
  score_breaks = seq(from = 5, to = 70, by = 5)
)
```

Arguments

pred_score	A data.frame with outcomes and final scores generated from AutoScore_fine_tuning_Ordinal
link	The link function used to model ordinal outcomes. Default is "logit" for proportional odds model. Other options are "cloglog" (proportional hazards model) and "probit".
max_score	Maximum attainable value of final scores.
score_breaks	A vector of score breaks to group scores. The average predicted risk will be reported for each score interval in the lookup table. Users are advised to first visualise the predicted risk for all attainable scores to determine scores (see plot_predicted_risk)

---

compute_prob_predicted

Internal function: Based on given labels and scores, compute average predicted risks in given score intervals.

Description

Internal function: Based on given labels and scores, compute average predicted risks in given score intervals.

Usage

```
compute_prob_predicted(
  pred_score,
  link = "logit",
  max_score = 100,
  score_breaks = seq(from = 5, to = 70, by = 5)
)
```

Arguments

pred_score	A data.frame with outcomes and final scores generated from AutoScore_fine_tuning_Ordinal
link	The link function used to model ordinal outcomes. Default is "logit" for proportional odds model. Other options are "cloglog" (proportional hazards model) and "probit".

max_score	Maximum attainable value of final scores.
score_breaks	A vector of score breaks to group scores. The average predicted risk will be reported for each score interval in the lookup table. Users are advised to first visualise the predicted risk for all attainable scores to determine scores (see plot_predicted_risk)

---

compute_score_table *Internal function: Compute scoring table based on training dataset (AutoScore Module 3)*

Description

Compute scoring table based on training dataset

Usage

```
compute_score_table(train_set_2, max_score, variable_list)
```

Arguments

train_set_2	Processed training set after variable transformation (AutoScore Module 2)
max_score	Maximum total score
variable_list	List of included variables

Value

A scoring table

---

compute_score_table_ord *Internal function: Compute scoring table for ordinal outcomes based on training dataset*

Description

Compute scoring table based on training dataset

Usage

```
compute_score_table_ord(train_set_2, max_score, variable_list, link)
```

Arguments

train_set_2	Processed training set after variable transformation
max_score	Maximum total score
variable_list	List of included variables
link	The link function used to model ordinal outcomes. Default is "logit" for proportional odds model. Other options are "cloglog" (proportional hazards model) and "probit".

Value

A scoring table

---

compute_score_table_survival

Internal function: Compute scoring table for survival outcomes based on training dataset

---

Description

Compute scoring table for survival outcomes based on training dataset

Usage

```
compute_score_table_survival(train_set_2, max_score, variable_list)
```

Arguments

train_set_2	Processed training set after variable transformation (AutoScore Module 2)
max_score	Maximum total score
variable_list	List of included variables

Value

A scoring table

compute_uni_variable_table

AutoScore function: Univariable Analysis

Description

Perform univariable analysis and generate the result table with odd ratios.

Usage

```
compute_uni_variable_table(df)
```

Arguments

df	data frame after checking

Value

result of univariate analysis

Examples

```
data("sample_data")
names(sample_data)[names(sample_data) == "Mortality_inpatient"] <- "label"
uni_table<-compute_uni_variable_table(sample_data)
```

compute_uni_variable_table_ordinal

AutoScore-Ordinal function: Univariable Analysis

Description

Perform univariable analysis and generate the result table with odd ratios from proportional odds models.

Usage

```
compute_uni_variable_table_ordinal(df, link = "logit", n_digits = 3)
```

Arguments

df	data frame after checking
link	The link function used to model ordinal outcomes. Default is "logit" for proportional odds model. Other options are "cloglog" (proportional hazards model) and "probit".
n_digits	Number of digits to print for OR or exponentiated coefficients (Default:3).

Value

result of univariate analysis

Examples

```
data("sample_data_ordinal")
# Using just a few variables to demonstrate usage:
uni_table<-compute_uni_variable_table_ordinal(sample_data_ordinal[, 1:3])
```

`compute_uni_variable_table_survival` *AutoScore function for survival outcomes: Univariate Analysis*

Description

Generate tables for Univariate analysis for survival outcomes

Usage

```
compute_uni_variable_table_survival(df)
```

Arguments

`df`	data frame after checking

Value

result of the Univariate analysis for survival outcomes

Examples

```
data("sample_data_survival")
uni_table<-compute_uni_variable_table_survival(sample_data_survival)
```

`conversion_table` *AutoScore function: Print conversion table based on final performance evaluation*

Description

Print conversion table based on final performance evaluation

Usage

```
conversion_table(
  pred_score,
  by = "risk",
  values = c(0.01, 0.05, 0.1, 0.2, 0.5)
)
```

Arguments

pred_score	a vector with outcomes and final scores generated from AutoScore_testing
by	specify correct method for categorizing the threshold: by "risk" or "score".Default to "risk"
values	A vector of threshold for analyze sensitivity, specificity and other metrics. Default to "c(0.01,0.05,0.1,0.2,0.5)"

Value

No return value and the conversion will be printed out directly.

See Also

AutoScore_testing

---

conversion_table_ordinal

AutoScore function: Print conversion table for ordinal outcomes to map score to risk

---

Description

AutoScore function: Print conversion table for ordinal outcomes to map score to risk

Usage

```
conversion_table_ordinal(
  pred_score,
  link = "logit",
  max_score = 100,
  score_breaks = seq(from = 5, to = 70, by = 5),
  ...
)
```

Arguments

pred_score	A data.frame with outcomes and final scores generated from AutoScore_fine_tuning_Ordinal
link	The link function used to model ordinal outcomes. Default is "logit" for proportional odds model. Other options are "cloglog" (proportional hazards model) and "probit".
max_score	Maximum attainable value of final scores.
score_breaks	A vector of score breaks to group scores. The average predicted risk will be reported for each score interval in the lookup table. Users are advised to first visualise the predicted risk for all attainable scores to determine scores (see plot_predicted_risk)
...	Additional parameters to pass to kable.

Value

No return value and the conversion will be printed out directly.

See Also

AutoScore_testing_Ordinal

---

conversion_table_survival

AutoScore function for survival outcomes: Print conversion table

---

Description

Print conversion table for survival outcomes

Usage

```
conversion_table_survival(
  pred_score,
  score_cut = c(40, 50, 60),
  time_point = c(7, 14, 30, 60, 90)
)
```

Arguments

pred_score	a data frame with outcomes and final scores generated from AutoScore_testing_Survival
score_cut	Score cut-offs to be used for generating conversion table
time_point	The time points to be evaluated using time-dependent AUC(t).

Value

conversion table and the it will also be printed out directly.

See Also

AutoScore_testing_Survival

`estimate_p_mat` *Internal function: generate probability matrix for ordinal outcomes given thresholds, linear predictor and link function*

Description

Internal function: generate probability matrix for ordinal outcomes given thresholds, linear predictor and link function

Usage

```
estimate_p_mat(theta, z, link)
```

Arguments

`theta`	numeric vector of thresholds
`z`	numeric vector of linear predictor
`link`	The link function used to model ordinal outcomes. Default is `"logit"` for proportional odds model. Other options are `"cloglog"` (proportional hazards model) and `"probit"`.

`evaluate_model_ord` *Internal function: Evaluate model performance on ordinal data*

Description

Internal function: Evaluate model performance on ordinal data

Usage

```
evaluate_model_ord(label, score, n_boot, report_cindex = TRUE)
```

Arguments

`label`	outcome variable
`score`	predicted score
`n_boot`	Number of bootstrap cycles to compute 95% CI for performance metrics.
`report_cindex`	If generalized c-index should be reported alongside mAUC (Default:FALSE).

Value

Returns a list of the mAUC (mauc) and generalized c-index (cindex, if requested for) and their 95

eva_performance_iauc *Internal function survival outcome: Calculate iAUC for validation set*

Description

Internal function survival outcome: Calculate iAUC for validation set

Usage

```
eva_performance_iauc(score, validation_set, print = TRUE)
```

Arguments

score	Predicted score
validation_set	Dataset for generating performance
print	Whether to print out the final iAUC result

extract_or_ci_ord *Extract OR, CI and p-value from a proportional odds model*

Description

Extract OR, CI and p-value from a proportional odds model

Usage

```
extract_or_ci_ord(model, n_digits = 3)
```

Arguments

model	An ordinal regression model fitted using clm.
n_digits	Number of digits to print for OR or exponentiated coefficients (Default:3).

find_one_inds *Internal function: Find column indices in design matrix that should be 1*

Description

Internal function: Find column indices in design matrix that should be 1

Usage

```
find_one_inds(x_inds)
```

Arguments

x_inds	A list of column indices corresponding to each final variable.

find_possible_scores *Internal function: Compute all scores attainable.*

Description

Internal function: Compute all scores attainable.

Usage

```
find_possible_scores(final_variables, scoring_table)
```

Arguments

final_variables	A vector containing the list of selected variables.
scoring_table	The final scoring table after fine-tuning.

Value

Returns a numeric vector of all scores attainable.

get_cut_vec *Internal function: Calculate cut_vec from the training set (AutoScore Module 2)*

Description

Internal function: Calculate cut_vec from the training set (AutoScore Module 2)

Usage

```
get_cut_vec(
  df,
  quantiles = c(0, 0.05, 0.2, 0.8, 0.95, 1),
  max_cluster = 5,
  categorize = "quantile"
)
```

Arguments

df	training set to be used for calculate the cut vector
quantiles	Predefined quantiles to convert continuous variables to categorical ones. (Default: c(0, 0.05, 0.2, 0.8, 0.95, 1)) Available if categorize = "quantile".
max_cluster	The max number of cluster (Default: 5). Available if categorize = "kmeans".
categorize	Methods for categorize continuous variables. Options include "quantile" or "kmeans" (Default: "quantile").

Value

cut_vec for transform_df_fixed

group_score *Internal function: Group scores based on given score breaks, and use friendly names for first and last intervals.*

Description

Internal function: Group scores based on given score breaks, and use friendly names for first and last intervals.

Usage

```
group_score(score, max_score, score_breaks)
```

Arguments

score	numeric vector of scores.
max_score	Maximum attainable value of final scores.
score_breaks	A vector of score breaks to group scores. The average predicted risk will be reported for each score interval in the lookup table. Users are advised to first visualise the predicted risk for all attainable scores to determine scores (see plot_predicted_risk)

---

induce_informative_missing

Internal function: induce informative missing to sample data in the package to demonstrate how AutoScore handles missing as a separate category

---

Description

Internal function: induce informative missing to sample data in the package to demonstrate how AutoScore handles missing as a separate category

Usage

```
induce_informative_missing(
  df,
  vars_to_induce = c("Lab_A", "Vital_A"),
  prop_missing = 0.4
)
```

Arguments

df	A data.frame of sample data.
vars_to_induce	Names of variables to induce informative missing in. Default is c("Lab_A", "Vital_A").
prop_missing	Proportion of missing to induce for each vars_to_induce. Can be a single value for a common proportion for all variables (default is 0.4), or a vector with same length as vars_to_induce.

Details

Assume subjects with normal values (i.e., values close to the median) are more likely to not have measurements.

Value

Returns df with selected columns modified to have missing.

induce_median_missing *Internal function: induce informative missing in a single variable*

Description

Internal function: induce informative missing in a single variable

Usage

```
induce_median_missing(x, prop_missing)
```

Arguments

x	Variable to induce missing in.
prop_missing	Proportion of missing to induce for each vars_to_induce. Can be a single value for a common proportion for all variables (default is 0.4), or a vector with same length as vars_to_induce.

inv_cloglog *Internal function: Inverse cloglog link*

Description

Internal function: Inverse cloglog link

Usage

```
inv_cloglog(x)
```

Arguments

x	A numeric vector.

inv_logit *Internal function: Inverse logit link*

Description

Internal function: Inverse logit link

Usage

```
inv_logit(x)
```

Arguments

x	A numeric vector.

inv_probit *Internal function: Inverse probit link*

Description

Internal function: Inverse probit link

Usage

```
inv_probit(x)
```

Arguments

x	A numeric vector.

make_design_mat *Internal function: Based on* find_one_inds*, make a design matrix to compute all scores attainable.*

Description

Internal function: Based on find_one_inds, make a design matrix to compute all scores attainable.

Usage

```
make_design_mat(one_inds)
```

Arguments

one_inds	Output from find_one_inds.

plot_auc *Internal function: Make parsimony plot*

Description

Internal function: Make parsimony plot

Usage

```
plot_auc(
  AUC,
  variables,
  num = seq_along(variables),
  auc_lim_min,
  auc_lim_max,
  ylab = "Mean Area Under the Curve",
  title = "Parsimony plot on the validation set"
)
```

Arguments

AUC	A vector of AUC values (or mAUC for ordinal outcomes).
variables	A vector of variable names
num	A vector of indices for AUC values to plot. Default is to plot all.
auc_lim_min	Min y_axis limit in the parsimony plot (Default: 0.5).
auc_lim_max	Max y_axis limit in the parsimony plot (Default: "adaptive").
ylab	Title of y-axis
title	Plot title

---

plot_importance *Internal Function: Print plotted variable importance*

Description

Internal Function: Print plotted variable importance

Usage

```
plot_importance(ranking)
```

Arguments

ranking	vector output generated by functions: AutoScore_rank, AutoScore_rank_Survival or AutoScore_rank_Ordinal

See Also

AutoScore_rank, AutoScore_rank_Survival, AutoScore_rank_Ordinal

plot_predicted_risk *AutoScore function for binary and ordinal outcomes: Plot predicted risk*

Description

AutoScore function for binary and ordinal outcomes: Plot predicted risk

Usage

```
plot_predicted_risk(
  pred_score,
  link = "logit",
  max_score = 100,
  final_variables,
  scoring_table,
  point_size = 0.5
)
```

Arguments

pred_score	Output from AutoScore_testing (for binary outcomes) or AutoScore_testing_Ordinal (for ordinal outcomes).
link	(For ordinal outcome only) The link function used in ordinal regression, which must be the same as the value used to build the risk score. Default is "logit" for proportional odds model.
max_score	Maximum total score (Default: 100).
final_variables	A vector containing the list of selected variables, selected from Step(ii) AutoScore_parsimony (for binary outcomes) or AutoScore_parsimony_Ordinal (for ordinal outcomes).
scoring_table	The final scoring table after fine-tuning, generated from STEP(iv) AutoScore_fine_tuning (for binary outcomes) or AutoScore_fine_tuning_Ordinal (for ordinal outcomes).
point_size	Size of points in the plot. Default is 0.5.

plot_roc_curve *Internal Function: Plotting ROC curve*

Description

Internal Function: Plotting ROC curve

Usage

```
plot_roc_curve(prob, labels, quiet = TRUE)
```

Arguments

prob	Predicate probability
labels	Actual outcome(binary)
quiet	if set to TRUE, there will be no trace printing

Value

No return value and the ROC curve will be plotted.

---

plot_survival_km *AutoScore function for survival outcomes: Print scoring performance (KM curve)*

Description

Print scoring performance (KM curve) for survival outcome

Usage

```
plot_survival_km(
  pred_score,
  score_cut = c(40, 50, 60),
  risk.table = TRUE,
  title = NULL,
  legend.title = "Score",
  xlim = c(0, 90),
  break.x.by = 30,
  ...
)
```

Arguments

pred_score	Generated from STEP(v)AutoScore_testing_Survival()
score_cut	Score cut-offs to be used for the analysis
risk.table	Allowed values include: TRUE or FALSE specifying whether to show or not the risk table. Default is TRUE.
title	Title displayed in the KM curve
legend.title	Legend title displayed in the KM curve
xlim	limit for x
break.x.by	Threshold for analyze sensitivity,
...	additional parameters to pass to ggsurvplot .

Value

No return value and the KM performance will be plotted.

See Also

AutoScore_testing_Survival

---

print_performance_ci_survival

AutoScore function for survival outcomes: Print predictive performance with confidence intervals

---

Description

Print iAUC, c-index and time-dependent AUC as the predictive performance

Usage

```
print_performance_ci_survival(score, validation_set, time_point, n_boot = 100)
```

Arguments

score	Predicted score
validation_set	Dataset for generating performance
time_point	The time points to be evaluated using time-dependent AUC(t).
n_boot	Number of bootstrap cycles to compute 95% CI for performance metrics.

Value

No return value and the ROC performance will be printed out directly.

See Also

AutoScore_testing_Ordinal

---

print_performance_ordinal

AutoScore function for ordinal outcomes: Print predictive performance

---

Description

Print mean area under the curve (mAUC) and generalised c-index (if requested)

Usage

```
print_performance_ordinal(label, score, n_boot = 100, report_cindex = FALSE)
```

Arguments

label	outcome variable
score	predicted score
n_boot	Number of bootstrap cycles to compute 95% CI for performance metrics.
report_cindex	Whether to report generalized c-index for model evaluation (Default:FALSE for faster evaluation).

Value

No return value and the ROC performance will be printed out directly.

See Also

AutoScore_testing_Ordinal

---

print_performance_survival

AutoScore function for survival outcomes: Print predictive performance

---

Description

Print mean area under the curve (mAUC) and generalised c-index (if requested)

Usage

```
print_performance_survival(score, validation_set, time_point)
```

Arguments

score	Predicted score
validation_set	Dataset for generating performance
time_point	The time points to be evaluated using time-dependent AUC(t).

Value

No return value and the ROC performance will be printed out directly.

See Also

AutoScore_testing_Ordinal

print_roc_performance *AutoScore function: Print receiver operating characteristic (ROC) performance*

Description

Print receiver operating characteristic (ROC) performance

Usage

```
print_roc_performance(label, score, threshold = "best", metrics_ci = FALSE)
```

Arguments

label	outcome variable
score	predicted score
threshold	Threshold for analyze sensitivity, specificity and other metrics. Default to "best"
metrics_ci	whether to calculate confidence interval for the metrics of sensitivity, specificity, etc.

Value

No return value and the ROC performance will be printed out directly.

See Also

AutoScore_testing

print_scoring_table *AutoScore Function: Print scoring tables for visualization*

Description

AutoScore Function: Print scoring tables for visualization

Usage

```
print_scoring_table(scoring_table, final_variable)
```

Arguments

scoring_table	Raw scoring table generated by AutoScore step(iv) AutoScore_fine_tuning
final_variable	Final included variables

Value

Data frame of formatted scoring table

See Also

AutoScore_fine_tuning, AutoScore_weighting

---

sample_data *20000 simulated ICU admission data, with the same distribution as the data in the MIMIC-III ICU database*

Description

20000 simulated samples, with the same distribution as the data in the MIMIC-III ICU database. It is used for demonstration only in the Guidebook. Run `vignette("Guide_book", package = "AutoScore")` to see the guidebook or vignette.

- Johnson, A., Pollard, T., Shen, L. et al. MIMIC-III, a freely accessible critical care database. Sci Data 3, 160035 (2016).

Usage

```
sample_data
```

Format

An object of class `data.frame` with 20000 rows and 22 columns.

---

sample_data_ordinal *Simulated ED data with ordinal outcome*

Description

Simulated data for 20,000 inpatient visits with demographic information, healthcare resource utilisation and associated laboratory tests and vital signs measured in the emergency department (ED). Data were simulated based on the dataset analysed in the AutoScore-Ordinal paper, and only includes a subset of variables (with masked variable names) for the purpose of demonstrating the AutoScore framework for ordinal outcomes.

Usage

```
sample_data_ordinal
```

Format

An object of class `data.frame` with 20000 rows and 21 columns.

References

- Saffari SE, Ning Y, Feng X, Chakraborty B, Volovici V, Vaughan R, Ong ME, Liu N, AutoScore-Ordinal: An interpretable machine learning framework for generating scoring models for ordinal outcomes, arXiv:2202.08407

`sample_data_ordinal_small` *Simulated ED data with ordinal outcome (small sample size)*

Description

5,000 observations randomly sampled from `sample_data_ordinal`. It is used for demonstration only in the Guidebook.

Usage

```
sample_data_ordinal_small
```

Format

An object of class `data.frame` with 5000 rows and 21 columns.

`sample_data_small` *1000 simulated ICU admission data, with the same distribution as the data in the MIMIC-III ICU database*

Description

1000 simulated samples, with the same distribution as the data in the MIMIC-III ICU database. It is used for demonstration only in the Guidebook. Run `vignette("Guide_book", package = "AutoScore")` to see the guidebook or vignette.

- Johnson, A., Pollard, T., Shen, L. et al. MIMIC-III, a freely accessible critical care database. Sci Data 3, 160035 (2016).

Usage

```
sample_data_small
```

Format

An object of class `data.frame` with 1000 rows and 22 columns.

sample_data_survival *20000 simulated MIMIC sample data with survival outcomes*

Description

20000 simulated samples, with the same distribution as the data in the MIMIC-III ICU database. Data were simulated based on the dataset analysed in the AutoScore-Survival paper. It is used for demonstration only in the Guidebook. Run `vignette("Guide_book", package = "AutoScore")` to see the guidebook or vignette.

- Johnson, A., Pollard, T., Shen, L. et al. MIMIC-III, a freely accessible critical care database. Sci Data 3, 160035 (2016).

Usage

```
sample_data_survival
```

Format

An object of class `data.frame` with 20000 rows and 23 columns.

sample_data_survival_small *1000 simulated MIMIC sample data with survival outcomes*

Description

1000 simulated samples, with the same distribution as the data in the MIMIC-III ICU database. Data were simulated based on the dataset analysed in the AutoScore-Survival paper. It is used for demonstration only in the Guidebook. Run `vignette("Guide_book", package = "AutoScore")` to see the guidebook or vignette.

- Johnson, A., Pollard, T., Shen, L. et al. MIMIC-III, a freely accessible critical care database. Sci Data 3, 160035 (2016).

Usage

```
sample_data_survival_small
```

Format

An object of class `data.frame` with 1000 rows and 23 columns.

sample_data_with_missing

20000 simulated ICU admission data with missing values

Description

20000 simulated samples with missing values, which can be used for demostrating AutoScore workflow dealing with missing values.

- Johnson, A., Pollard, T., Shen, L. et al. MIMIC-III, a freely accessible critical care database. Sci Data 3, 160035 (2016).

Usage

```
sample_data_with_missing
```

Format

An object of class `data.frame` with 20000 rows and 23 columns.

split_data

AutoScore Function: Automatically splitting dataset to train, validation and test set, possibly stratified by label

Description

AutoScore Function: Automatically splitting dataset to train, validation and test set, possibly stratified by label

Usage

```
split_data(data, ratio, cross_validation = FALSE, strat_by_label = FALSE)
```

Arguments

`data`	The dataset to be split
`ratio`	The ratio for dividing dataset into training, validation and testing set. (Default: c(0.7, 0.1, 0.2))
`cross_validation`	If set to `TRUE`, cross-validation would be used for generating parsimony plot, which is suitable for small-size data. Default to `FALSE`
`strat_by_label`	If set to `TRUE`, data splitting is stratified on the outcome variable. Default to `FALSE`

Value

Returns a list containing training, validation and testing set

Examples

```
data("sample_data")
names(sample_data)[names(sample_data) == "Mortality_inpatient"] <- "label"
set.seed(4)
#large sample size
out_split <- split_data(data = sample_data, ratio = c(0.7, 0.1, 0.2))
#small sample size
out_split <- split_data(data = sample_data, ratio = c(0.7, 0, 0.3),
                        cross_validation = TRUE)
#large sample size, stratified
out_split <- split_data(data = sample_data, ratio = c(0.7, 0.1, 0.2),
                        strat_by_label = TRUE)
```

---

transform_df_fixed *Internal function: Categorizing continuous variables based on cut_vec (AutoScore Module 2)*

Description

Internal function: Categorizing continuous variables based on cut_vec (AutoScore Module 2)

Usage

```
transform_df_fixed(df, cut_vec)
```

Arguments

df	dataset(training, validation or testing) to be processed
cut_vec	fixed cut vector

Value

Processed data.frame after categorizing based on fixed cut_vec

Index