Package ‘AutoScore’

June 16, 2021

Type Package
Title An Interpretable Machine Learning-Based Automatic Clinical Score Generator
Version 0.2.0
Date 2021-06-10
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 7.1.1
Imports tableone, pROC, randomForest, ggplot2, knitr
Depends R (>= 2.10)
VignetteBuilder knitr
Suggests rmarkdown
NeedsCompilation no

Author Feng Xie [aut, cre] (<https://orcid.org/0000-0002-0215-667X>),
Yilin Ning [aut] (<https://orcid.org/0000-0002-6758-4472>),
Han Yuan [aut] (<https://orcid.org/0000-0002-2674-6068>),
Ehsan Saffari [aut] (<https://orcid.org/0000-0002-6473-4375>),
Bibhas Chakraborty [aut] (<https://orcid.org/0000-0002-7366-0478>),
Nan Liu [aut] (<https://orcid.org/0000-0003-3610-4883>)}
Maintainer  Feng Xie <xief@u.duke.nus.edu>
Repository  CRAN
Date/Publication  2021-06-16 09:50:04 UTC

R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>add_baseline</td>
<td>2</td>
</tr>
<tr>
<td>assign_score</td>
<td>3</td>
</tr>
<tr>
<td>AutoScore_fine_tuning</td>
<td>3</td>
</tr>
<tr>
<td>AutoScore_parsimony</td>
<td>5</td>
</tr>
<tr>
<td>AutoScore_rank</td>
<td>7</td>
</tr>
<tr>
<td>AutoScore_testing</td>
<td>8</td>
</tr>
<tr>
<td>AutoScore_weighting</td>
<td>9</td>
</tr>
<tr>
<td>change_reference</td>
<td>10</td>
</tr>
<tr>
<td>check_data</td>
<td>11</td>
</tr>
<tr>
<td>compute_auc_val</td>
<td>11</td>
</tr>
<tr>
<td>compute_descriptive_table</td>
<td>12</td>
</tr>
<tr>
<td>compute_multi_variable_table</td>
<td>13</td>
</tr>
<tr>
<td>compute_score_table</td>
<td>13</td>
</tr>
<tr>
<td>compute Uni_variable_table</td>
<td>14</td>
</tr>
<tr>
<td>get_cut_vec</td>
<td>14</td>
</tr>
<tr>
<td>plot_roc_curve</td>
<td>15</td>
</tr>
<tr>
<td>print_roc_performance</td>
<td>16</td>
</tr>
<tr>
<td>print_scoring_table</td>
<td>16</td>
</tr>
<tr>
<td>sample_data</td>
<td>17</td>
</tr>
<tr>
<td>sample_data_small</td>
<td>17</td>
</tr>
<tr>
<td>split_data</td>
<td>18</td>
</tr>
<tr>
<td>transform_df_fixed</td>
<td>18</td>
</tr>
</tbody>
</table>

Index  

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)
**assign_score**

**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 subject given new data set and scoring table (Used for intermediate and final evaluation)*

---

**Description**

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

**Usage**

```r
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 variable

---

**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.
AutoScore_fine_tuning

Usage

AutoScore_fine_tuning(
  train_set,
  validation_set,
  final_variables,
  cut_vec,
  max_score = 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. 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).

Value

Generated final table of scoring model for downstream testing

References


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_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**

```r
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
)
```

**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 ranking 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`. 
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


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 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, 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), 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

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)
### Description

Domain knowledge is essential in guiding risk model development. For continuous variables, the variable transformation is a data-driven process (based on "quantile", "kmeans" or "decision_tree"). 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

```r
AutoScore_testing(
  test_set,
  final_variables,
  cut_vec,
  scoring_table,
  threshold = "best",
  with_label = 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.

### Value

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

References


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_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)
)

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").
change_reference

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".

Value

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

References


See Also

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

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: Check whether the input dataset fulfill the requirement of the AutoScore

Description
AutoScore function: 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)

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
)
**compute_descriptive_table**

**AutoScore function: Descriptive Analysis**

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

**Usage**

```r
compute_descriptive_table(df)
```

**Arguments**

df  
data frame after checking and fulfilling the requirement of AutoScore

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

**Examples**

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

*AutoScore function: Multivariate Analysis*

**Description**
Generate tables for multivariate analysis

**Usage**
```r
compute_multi_variable_table(df)
```

**Arguments**
- `df` data frame after checking

**Value**
result of the multivariate analysis

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

compute_score_table

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

**Description**
Compute scoring table based on training dataset

**Usage**
```r
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_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**

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

---

**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"
)
```
**plot_roc_curve**

**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`

---

**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.
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")

Arguments
- label: outcome variable
- score: predicted score
- threshold: Threshold for analyze sensitivity, specificity and other metrics. Default to "best"

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
### sample_data

**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.


**Usage**

```r
clean_data
```

**Format**

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


**Usage**

```r
clean_data_small
```

**Format**

An object of class `data.frame` with 1000 rows and 22 columns.
**split_data**  
*AutoScore function: Automatically splitting dataset to train, validation and test set*

**Description**  
*AutoScore function: Automatically splitting dataset to train, validation and test set*

**Usage**  
`split_data(data, ratio, cross_validation = 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

**Value**
Returns a list containing training, validation and testing set

**Examples**
```
data("sample_data")
set.seed(4)
#large sample size
out_split <- split_data(data = sample_data, ratio = c(0.7, 0.1, 0.2))
#small sample size (for cross-validation)
out_split <- split_data(data = sample_data, ratio = c(0.7, 0, 0.3), cross_validation = 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)`

---

`transform_df_fixed` is an internal function that categorizes continuous variables based on `cut_vec` for AutoScore Module 2.
**transform_df_fixed**

**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

* datasets
  sample_data, 17
  sample_data_small, 17

add_baseline, 2
assign_score, 3
AutoScore_fine_tuning, 3, 6–10, 16, 17
AutoScore_parsimony, 4, 5, 7–10
AutoScore_rank, 4–6, 7, 9, 10
AutoScore_testing, 4, 6, 7, 8, 10, 16
AutoScore_weighting, 4, 6–9, 9, 17
change_reference, 10
check_data, 11
compute_auc_val, 11
compute_descriptive_table, 12
compute_multi_variable_table, 13
compute_score_table, 13
compute_uni_variable_table, 14
get_cut_vec, 14
plot_roc_curve, 15
print_roc_performance, 9, 16
print_scoring_table, 16

sample_data, 17
sample_data_small, 17
split_data, 18
transform_df_fixed, 18