
toad

Release 0.1.2

Jul 16, 2023

Contents

1	Installation	1
2	Tutorial	3
3	Contents	5
4	Indices and tables	33
5	Links	35
	Python Module Index	37
	Index	39

CHAPTER 1

Installation

via pip

```
pip install toad
```

via anaconda

```
conda install toad --channel conda-forge
```

via source code

```
python setup.py install
```


CHAPTER 2

Tutorial

A [basic tutorial](#) is provided.

3.1 toad package

3.2 Submodules

3.2.1 toad.detector module

Command line tools for detecting csv data

Team: ESC

Examples

```
python detector.py -i xxx.csv -o report.csv
```

```
toad.detector.getTopValues (series, top=5, reverse=False)
```

Get top/bottom n values

Parameters

- **series** (*Series*) – data series
- **top** (*number*) – number of top/bottom n values
- **reverse** (*bool*) – it will return bottom n values if True is given

Returns Series of top/bottom n values and percentage. ['value:percent', None]

Return type Series

```
toad.detector.getDescribe (series, percentiles=[0.25, 0.5, 0.75])
```

Get describe of series

Parameters

- **series** (*Series*) – data series

- **percentiles** – the percentiles to include in the output

Returns the describe of data include mean, std, min, max and percentiles

Return type Series

`toad.detector.countBlank(series, blanks=[])`

Count number and percentage of blank values in series

Parameters

- **series** (*Series*) – data series
- **blanks** (*list*) – list of blank values

Returns number of blanks str: the percentage of blank values

Return type number

`toad.detector.isNumeric(series)`

Check if the series's type is numeric

Parameters **series** (*Series*) – data series

Returns bool

`toad.detector.detect(dataframe)`

Detect data

Parameters **dataframe** (*DataFrame*) – data that will be detected

Returns report of detecting

Return type DataFrame

3.2.2 toad.merge module

`toad.merge.ChiMerge()`

Chi-Merge

Parameters

- **feature** (*array-like*) – feature to be merged
- **target** (*array-like*) – a array of target classes
- **n_bins** (*int*) – n bins will be merged into
- **min_samples** (*number*) – min sample in each group, if float, it will be the percentage of samples
- **min_threshold** (*number*) – min threshold of chi-square

Returns array of split points

Return type array

`toad.merge.DTMerge()`

Merge by Decision Tree

Parameters

- **feature** (*array-like*) –
- **target** (*array-like*) – target will be used to fit decision tree
- **nan** (*number*) – value will be used to fill nan

- **n_bins** (*int*) – n groups that will be merged into
- **min_samples** (*int*) – min number of samples in each leaf nodes

Returns array of split points

Return type array

`toad.merge.KMeansMerge()`
Merge by KMeans

Parameters

- **feature** (*array-like*) –
- **target** (*array-like*) – target will be used to fit kmeans model
- **nan** (*number*) – value will be used to fill nan
- **n_bins** (*int*) – n groups that will be merged into
- **random_state** (*int*) – random state will be used for kmeans model

Returns split points of feature

Return type array

`toad.merge.QuantileMerge()`
Merge by quantile

Parameters

- **feature** (*array-like*) –
- **nan** (*number*) – value will be used to fill nan
- **n_bins** (*int*) – n groups that will be merged into
- **q** (*array-like*) – list of percentage split points

Returns split points of feature

Return type array

`toad.merge.StepMerge()`
Merge by step

Parameters

- **feature** (*array-like*) –
- **nan** (*number*) – value will be used to fill nan
- **n_bins** (*int*) – n groups that will be merged into
- **clip_v** (*number | tuple*) – min/max value of clipping
- **clip_std** (*number | tuple*) – min/max std of clipping
- **clip_q** (*number | tuple*) – min/max quantile of clipping

Returns split points of feature

Return type array

`toad.merge.merge`
merge feature into groups

Parameters

- **feature** (*array-like*) –
- **target** (*array-like*) –
- **method** (*str*) – ‘dt’, ‘chi’, ‘quantile’, ‘step’, ‘kmeans’ - the strategy to be used to merge feature
- **return_splits** (*bool*) – if needs to return splits
- **n_bins** (*int*) – n groups that will be merged into

Returns a array of merged label with the same size of feature array: list of split points

Return type array

3.2.3 toad.metrics module

`toad.metrics.KS(score, target)`
calculate ks value

Parameters

- **score** (*array-like*) – list of score or probability that the model predict
- **target** (*array-like*) – list of real target

Returns the max KS value

Return type float

`toad.metrics.KS_bucket(score, target, bucket=10, method='quantile', return_splits=False, **kwargs)`
calculate ks value by bucket

Parameters

- **score** (*array-like*) – list of score or probability that the model predict
- **target** (*array-like*) – list of real target
- **bucket** (*int*) – n groups that will bin into
- **method** (*str*) – method to bin score. *quantile* (default), *step*
- **return_splits** (*bool*) – if need to return splits of bucket

Returns DataFrame

`toad.metrics.KS_by_col(df, by='feature', score='score', target='target')`

`toad.metrics.SSE(y_pred, y)`
sum of squares due to error

`toad.metrics.MSE(y_pred, y)`
mean of squares due to error

`toad.metrics.AIC(y_pred, y, k, llf=None)`
Akaike Information Criterion

Parameters

- **y_pred** (*array-like*) –
- **y** (*array-like*) –
- **k** (*int*) – number of featuers

- **llf** (*float*) – result of log-likelihood function

`toad.metrics.BIC(y_pred, y, k, llf=None)`

Bayesian Information Criterion

Parameters

- **y_pred** (*array-like*) –
- **y** (*array-like*) –
- **k** (*int*) – number of features
- **llf** (*float*) – result of log-likelihood function

`toad.metrics.F1(score, target, split='best', return_split=False)`

calculate f1 value

Parameters

- **score** (*array-like*) –
- **target** (*array-like*) –

Returns best f1 score float: best splitter

Return type float

`toad.metrics.AUC(score, target, return_curve=False)`

AUC Score

Parameters

- **score** (*array-like*) – list of score or probability that the model predict
- **target** (*array-like*) – list of real target
- **return_curve** (*bool*) – if need return curve data for ROC plot

Returns auc score

Return type float

`toad.metrics.PSI(test, base, combiner=None, return_frame=False)`

calculate PSI

Parameters

- **test** (*array-like*) – data to test PSI
- **base** (*array-like*) – base data for calculate PSI
- **combiner** (*Combiner|list|dict*) – combiner to combine data
- **return_frame** (*bool*) – if need to return frame of proportion

Returns float|Series

`toad.metrics.matrix(y_pred, y, splits=None)`

confusion matrix of target

Parameters

- **y_pred** (*array-like*) –
- **y** (*array-like*) –
- **splits** (*float|list*) – split points of y_pred

Returns confusion matrix with true labels in rows and predicted labels in columns

Return type DataFrame

3.2.4 toad.plot module

`toad.plot.badrates_plot` (*frame*, *x=None*, *target='target'*, *by=None*, *freq=None*, *format=None*, *return_counts=False*, *return_proportion=False*, *return_frame=False*)

plot for badrate

Parameters

- **frame** (*DataFrame*) –
- **x** (*str*) – column in frame that will be used as x axis
- **target** (*str*) – target column in frame
- **by** (*str*) – column in frame that will be calculated badrate by it
- **freq** (*str*) – offset aliases string by pandas <http://pandas.pydata.org/pandas-docs/stable/timeseries.html#offset-aliases>
- **format** (*str*) – format string for time
- **return_counts** (*bool*) – if need return counts plot
- **return_frame** (*bool*) – if need return frame

Returns badrate plot Axes: counts plot Axes: proportion plot Dataframe: grouping detail data

Return type Axes

`toad.plot.corr_plot` (*frame*, *figure_size=(20, 15)*)

plot for correlation

Parameters **frame** (*DataFrame*) – frame to draw plot

Returns Axes

`toad.plot.proportion_plot` (*x=None*, *keys=None*)

plot for comparing proportion in different dataset

Parameters

- **x** (*Series/list*) – series or list of series data for plot
- **keys** (*str/list*) – keys for each data

Returns Axes

`toad.plot.roc_plot` (*score*, *target*, *compare=None*, *figsize=(14, 10)*)

plot for roc

Parameters

- **score** (*array-like*) – predicted score
- **target** (*array-like*) – true target
- **compare** (*array-like*) – another score for comparing with score

Returns Axes

`toad.plot.ks_plot` (*score*, *target*, *figsize=(14, 10)*)

plot for ks

Parameters

- **score** (*array-like*) – predicted score
- **target** (*array-like*) – true target
- **compare** (*array-like*) – another score for comparing with score

Returns Axes

`toad.plot.bin_plot` (*frame*, *x=None*, *target='target'*, *iv=True*, *annotate_format='.2f'*, *return_frame=False*)
 plot for bins

Parameters

- **frame** (*DataFrame*) –
- **x** (*str*) – column in frame that will be used as x axis
- **target** (*str*) – target column in frame
- **iv** (*bool*) – if need to show iv in plot
- **annotate_format** (*str*) – format str for axis annotation of chart
- **return_frame** (*bool*) – if need return bin frame

Returns contains good, bad, badrate, prop, y_prop, n_prop, woe, iv

Return type Dataframe

3.2.5 toad.scorecard module

class `toad.scorecard.ScoreCard` (*pdo=60*, *rate=2*, *base_odds=35*, *base_score=750*, *card=None*, *combiner={}*, *transer=None*, ***kwargs*)

Bases: `sklearn.base.BaseEstimator`, `toad.utils.mixin.RulesMixin`, `toad.utils.mixin.BinsMixin`

__init__ (*pdo=60*, *rate=2*, *base_odds=35*, *base_score=750*, *card=None*, *combiner={}*, *transer=None*, ***kwargs*)

Parameters

- **combiner** (*toad.Combiner*) –
- **transer** (*toad.WOETransformer*) –

coef_
 coef of LR model

fit (*X*, *y*)

Parameters

- **X** (*2D DataFrame*) –
- **Y** (*array-like*) –

predict (*X*, ***kwargs*)

predict score :param X: X to predict :type X: 2D-DataFrameldict :param return_sub: if need to return sub score of each feature :type return_sub: Bool :param default: default sub score for unknown feature, *min* (default), *max* :type default: strlnumber

Returns predicted score DataFrameldict: sub score for each feature

Return type array-like

get_reason (*X*, *base_effect*=None, *threshold_score*=None, *keep*=3)
calculate top-effect-of-features as reasons

Parameters

- **X** (*2D DataFrame*) – X to find reason
- **base_effect** (*Series*) – base effect score of each feature
- **threshold_score** (*float*) – threshold to find top k most important features, show the highest top k features when prediction score > threshold and show the lowest top k when prediction score <= threshold default is the sum of *base_effect* score
- **keep** (*int*) – top k most important reasons to keep, default 3

Returns top k most important reasons for each feature

Return type DataFrame

bin_to_score (*bins*, *return_sub*=False, *default*='min')
predict score from bins

predict_proba (*X*)
predict probability

Parameters **X** (*2D array-like*) – X to predict

Returns probability of all classes

Return type 2d array

proba_to_score (*prob*)
covert probability to score
 $\text{odds} = (1 - \text{prob}) / \text{prob}$ score = factor * log(odds) * offset

score_to_proba (*score*)
covert score to probability

Returns the probability of 1

Return type array-likefloat

woe_to_score (*woe*, *weight*=None)
calculate score by woe

after_load (*rules*)
after load card

after_export (*card*, *to_frame*=False, *to_json*=None, *to_csv*=None, ****kwargs**)
generate a scorecard object

Parameters

- **to_frame** (*bool*) – return DataFrame of card
- **to_json** (*str* | *IOBase*) – io to write json file
- **to_csv** (*filepath* | *IOBase*) – file to write csv

Returns dict

testing_frame (****kwargs**)
get testing frame with score

Returns testing frame with score

Return type DataFrame

3.2.6 toad.selection module

```
toad.selection.stepwise (frame, target='target', estimator='ols', direction='both', criterion='aic',
                        p_enter=0.01, p_remove=0.01, p_value_enter=0.2, intercept=False,
                        max_iter=None, return_drop=False, exclude=None)
```

stepwise to select features

Parameters

- **frame** (*DataFrame*) – dataframe that will be use to select
- **target** (*str*) – target name in frame
- **estimator** (*str*) – model to use for stats
- **direction** (*str*) – direction of stepwise, support ‘forward’, ‘backward’ and ‘both’, suggest ‘both’
- **criterion** (*str*) – criterion to statistic model, support ‘aic’, ‘bic’
- **p_enter** (*float*) – threshold that will be used in ‘forward’ and ‘both’ to keep features
- **p_remove** (*float*) – threshold that will be used in ‘backward’ to remove features
- **intercept** (*bool*) – if have intercept
- **p_value_enter** (*float*) – threshold that will be used in ‘both’ to remove features
- **max_iter** (*int*) – maximum number of iterate
- **return_drop** (*bool*) – if need to return features’ name who has been dropped
- **exclude** (*array-like*) – list of feature names that will not be dropped

Returns selected dataframe array: list of feature names that has been dropped

Return type *DataFrame*

```
toad.selection.drop_empty (frame, threshold=0.9, nan=None, return_drop=False, exclude=None)
```

drop columns by empty

Parameters

- **frame** (*DataFrame*) – dataframe that will be used
- **threshold** (*number*) – drop the features whose empty num is greater than threshold. if threshold is float, it will be use as percentage
- **nan** (*any*) – values will be look like empty
- **return_drop** (*bool*) – if need to return features’ name who has been dropped
- **exclude** (*array-like*) – list of feature names that will not be dropped

Returns selected dataframe array: list of feature names that has been dropped

Return type *DataFrame*

```
toad.selection.drop_var (frame, threshold=0, return_drop=False, exclude=None)
```

drop columns by variance

Parameters

- **frame** (*DataFrame*) – dataframe that will be used
- **threshold** (*float*) – drop features whose variance is less than threshold
- **return_drop** (*bool*) – if need to return features’ name who has been dropped

- **exclude** (*array-like*) – list of feature names that will not be dropped

Returns selected dataframe array: list of feature names that has been dropped

Return type DataFrame

```
toad.selection.drop_corr(frame, target=None, threshold=0.7, by='IV', return_drop=False, exclude=None)
```

drop columns by correlation

Parameters

- **frame** (*DataFrame*) – dataframe that will be used
- **target** (*str*) – target name in dataframe
- **threshold** (*float*) – drop features that has the smallest weight in each groups whose correlation is greater than threshold
- **by** (*array-like*) – weight of features that will be used to drop the features
- **return_drop** (*bool*) – if need to return features' name who has been dropped
- **exclude** (*array-like*) – list of feature names that will not be dropped

Returns selected dataframe array: list of feature names that has been dropped

Return type DataFrame

```
toad.selection.drop_iv(frame, target='target', threshold=0.02, return_drop=False, return_iv=False, exclude=None)
```

drop columns by IV

Parameters

- **frame** (*DataFrame*) – dataframe that will be used
- **target** (*str*) – target name in dataframe
- **threshold** (*float*) – drop the features whose IV is less than threshold
- **return_drop** (*bool*) – if need to return features' name who has been dropped
- **return_iv** (*bool*) – if need to return features' IV
- **exclude** (*array-like*) – list of feature names that will not be dropped

Returns selected dataframe array: list of feature names that has been dropped Series: list of features' IV

Return type DataFrame

```
toad.selection.drop_vif(frame, threshold=3, return_drop=False, exclude=None)
```

variance inflation factor

Parameters

- **frame** (*DataFrame*) –
- **threshold** (*float*) – drop features until all vif is less than threshold
- **return_drop** (*bool*) – if need to return features' name who has been dropped
- **exclude** (*array-like*) – list of feature names that will not be dropped

Returns selected dataframe array: list of feature names that has been dropped

Return type DataFrame

`toad.selection.select` (*frame*, *target*='target', *empty*=0.9, *iv*=0.02, *corr*=0.7, *return_drop*=False, *exclude*=None)
select features by rate of empty, iv and correlation

Parameters

- **frame** (*DataFrame*) –
- **target** (*str*) – target's name in dataframe
- **empty** (*number*) – drop the features which empty num is greater than threshold. if threshold is less than 1, it will be use as percentage
- **iv** (*float*) – drop the features whose IV is less than threshold
- **corr** (*float*) – drop features that has the smallest IV in each groups which correlation is greater than threshold
- **return_drop** (*bool*) – if need to return features' name who has been dropped
- **exclude** (*array-like*) – list of feature name that will not be dropped

Returns selected dataframe dict: list of dropped feature names in each step

Return type DataFrame

3.2.7 toad.stats module

`toad.stats.gini` (*target*)
get gini index of a feature

Parameters **target** (*array-like*) – list of target that will be calculate gini

Returns gini value

Return type number

`toad.stats.gini_cond`
get conditional gini index of a feature

Parameters

- **feature** (*array-like*) –
- **target** (*array-like*) –

Returns conditional gini value. If feature is continuous, it will return the best gini value when the feature bins into two groups

Return type number

`toad.stats.entropy` (*target*)
get infomation entropy of a feature

Parameters **target** (*array-like*) –

Returns information entropy

Return type number

`toad.stats.entropy_cond`
get conditional entropy of a feature

Parameters

- **feature** (*array-like*) –

- **target** (*array-like*) –

Returns conditional information entropy. If feature is continuous, it will return the best entropy when the feature bins into two groups

Return type number

`toad.stats.probability(target, mask=None)`
get probability of target by mask

`toad.stats.WOE(y_prob, n_prob)`
get WOE of a group

Parameters

- **y_prob** – the probability of grouped y in total y
- **n_prob** – the probability of grouped n in total n

Returns woe value

Return type number

`toad.stats.IV`
get the IV of a feature

Parameters

- **feature** (*array-like*) –
- **target** (*array-like*) –
- **return_sub** (*bool*) – if need return IV of each groups
- **n_bins** (*int*) – n groups that the feature will bin into
- **method** (*str*) – the strategy to be used to merge feature, default is 'dt'
- **()** (***kwargs*) – other options for merge function

`toad.stats.badrate(target)`
calculate badrate

Parameters **target** (*array-like*) – target array which *I* is bad

Returns float

`toad.stats.VIF(frame)`
calculate vif

Parameters **frame** (*ndarray/DataFrame*) –

Returns Series

class `toad.stats.indicator(*args, is_class=False, **kwargs)`
Bases: `toad.utils.decorator.Decorator`
indicator decorator

`toad.stats.column_quality(feature, target, name='feature', indicators=[], need_merge=False, **kwargs)`
calculate quality of a feature

Parameters

- **feature** (*array-like*) –
- **target** (*array-like*) –

- **name** (*str*) – feature’s name that will be setted in the returned Series
- **indicators** (*list*) – list of indicator functions
- **need_merge** (*bool*) – if need merge feature

Returns a list of quality with the feature’s name

Return type Series

```
toad.stats.quality(dataframe, target='target', cpu_cores=0, iv_only=False, indicators=['iv', 'gini',
'entropy', 'unique'], **kwargs)
```

get quality of features in data

Parameters

- **dataframe** (*DataFrame*) – dataframe that will be calculate quality
- **target** (*str*) – the target’s name in dataframe
- **iv_only** (*bool*) – *deprecated*. if only calculate IV
- **cpu_cores** (*int*) – the maximun number of CPU cores will be used, 0 means all CPUs will be used, -1 means all CPUs but one will be used.

Returns quality of features with the features’ name as row name

Return type DataFrame

```
toad.stats.feature_bin_stats(df_bin, feature, target)
```

calculate the detail info of a feature after bin

Parameters

- **df_bin** (*dataframe has featute and target columns*) –
- **feature** (*str*) –
- **target** (*str*) –

Returns contains good, bad, badrate, prop, y_prop, n_prop, woe, iv

Return type DataFrame

3.2.8 toad.transform module

class toad.transform.Transformer

Bases: sklearn.base.TransformerMixin, toad.utils.mixin.RulesMixin

Base class for transformers

fit (*X*, **args*, *update=False*, ***kwargs*)
fit method, see details in *fit_* method

transform (*X*, **args*, ***kwargs*)
transform method, see details in *transform_* method

__init__
Initialize self. See help(type(self)) for accurate signature.

export (***kwargs*)
export rules to dict or a json file

Parameters **to_json** (*str* | *IOBase*) – json file to save rules

Returns dictionary of rules

Return type dict

fit_transform (*X*, *y=None*, ***fit_params*)

Fit to data, then transform it.

Fits transformer to *X* and *y* with optional parameters *fit_params* and returns a transformed version of *X*.

Parameters

- **X** (*array-like of shape (n_samples, n_features)*) – Input samples.
- **y** (*array-like of shape (n_samples,) or (n_samples, n_outputs), default=None*) – Target values (None for unsupervised transformations).
- ****fit_params** (*dict*) – Additional fit parameters.

Returns **X_new** – Transformed array.

Return type ndarray array of shape (n_samples, n_features_new)

load (*rules*, *update=False*, ***kwargs*)

load rules from dict or json file

Parameters

- **rules** (*dict*) – dictionary of rules
- **from_json** (*str/IOBase*) – json file of rules
- **update** (*bool*) – if need to use updating instead of replacing rules

set_output (**, transform=None*)

Set output container.

See sphx_glr_auto_examples_miscellaneous_plot_set_output.py for an example on how to use the API.

Parameters **transform** (*{"default", "pandas"}, default=None*) – Configure output of *transform* and *fit_transform*.

- **"default"**: Default output format of a transformer
- **"pandas"**: DataFrame output
- **None**: Transform configuration is unchanged

Returns **self** – Estimator instance.

Return type estimator instance

update (**args, **kwargs*)

update rules

Parameters

- **rules** (*dict*) – dictionary of rules
- **from_json** (*str/IOBase*) – json file of rules

class toad.transform.WOETransformer

Bases: *toad.transform.Transformer*

WOE transformer

fit_ (*X*, *y*)

fit WOE transformer

Parameters

- **X** (*DataFrame/array-like*) –
- **y** (*str/array-like*) –
- **select_dtypes** (*str/numpy.dtypes*) – ‘object’, ‘number’ etc. only selected dtypes will be transform

transform_ (*rule, X, default='min'*)
transform function for single feature

Parameters

- **X** (*array-like*) –
- **default** (*str*) – ‘min’(default), ‘max’ - the strategy to be used for unknown group

Returns array-like

__init__
Initialize self. See help(type(self)) for accurate signature.

export (***kwargs*)
export rules to dict or a json file

Parameters **to_json** (*str/IOBase*) – json file to save rules

Returns dictionary of rules

Return type dict

fit (*X, *args, update=False, **kwargs*)
fit method, see details in *fit_* method

fit_transform (*X, y=None, **fit_params*)
Fit to data, then transform it.

Fits transformer to *X* and *y* with optional parameters *fit_params* and returns a transformed version of *X*.

Parameters

- **X** (*array-like of shape (n_samples, n_features)*) – Input samples.
- **y** (*array-like of shape (n_samples,) or (n_samples, n_outputs), default=None*) – Target values (None for unsupervised transformations).
- ****fit_params** (*dict*) – Additional fit parameters.

Returns **X_new** – Transformed array.

Return type ndarray array of shape (n_samples, n_features_new)

load (*rules, update=False, **kwargs*)
load rules from dict or json file

Parameters

- **rules** (*dict*) – dictionary of rules
- **from_json** (*str/IOBase*) – json file of rules
- **update** (*bool*) – if need to use updating instead of replacing rules

set_output (**, transform=None*)
Set output container.

See sphx_glr_auto_examples_misellaneous_plot_set_output.py for an example on how to use the API.

Parameters **transform** (`{ "default", "pandas" }, default=None`) – Configure output of *transform* and *fit_transform*.

- `"default"`: Default output format of a transformer
- `"pandas"`: DataFrame output
- `None`: Transform configuration is unchanged

Returns **self** – Estimator instance.

Return type estimator instance

transform (*X*, **args*, ***kwargs*)
transform method, see details in *transform_* method

update (**args*, ***kwargs*)
update rules

Parameters

- **rules** (*dict*) – dictionary of rules
- **from_json** (*str* | *IOBase*) – json file of rules

class toad.transform.Combiner

Bases: *toad.transform.Transformer*, *toad.utils.mixin.BinsMixin*

Combiner for merge data

fit_ (*X*, *y=None*, *method='chi'*, *empty_separate=False*, ***kwargs*)
fit combiner

Parameters

- **X** (*DataFrame* | *array-like*) – features to be combined
- **y** (*str* | *array-like*) – target data or name of target in *X*
- **method** (*str*) – the strategy to be used to merge *X*, same as *.merge*, default is *chi*
- **n_bins** (*int*) – counts of bins will be combined
- **empty_separate** (*bool*) – if need to combine empty values into a separate group

transform_ (*rule*, *X*, *labels=False*, *ellipsis=16*, ***kwargs*)
transform *X* by combiner

Parameters

- **X** (*DataFrame* | *array-like*) – features to be transformed
- **labels** (*bool*) – if need to use labels for resulting bins, *False* by default
- **ellipsis** (*int*) – max length threshold that labels will not be ellipsis, *None* for skipping ellipsis

Returns array-like

set_rules (*map*, *reset=False*)
set rules for combiner

Parameters

- **map** (*dict* | *array-like*) – map of splits
- **reset** (*bool*) – if need to reset combiner

Returns **self**

__init__

Initialize self. See help(type(self)) for accurate signature.

export (***kwargs*)

export rules to dict or a json file

Parameters **to_json** (*str* | *IOBase*) – json file to save rules

Returns dictionary of rules

Return type dict

fit (*X*, **args*, *update=False*, ***kwargs*)

fit method, see details in *fit*_method

fit_transform (*X*, *y=None*, ***fit_params*)

Fit to data, then transform it.

Fits transformer to *X* and *y* with optional parameters *fit_params* and returns a transformed version of *X*.

Parameters

- **X** (*array-like of shape (n_samples, n_features)*) – Input samples.
- **y** (*array-like of shape (n_samples,) or (n_samples, n_outputs), default=None*) – Target values (None for unsupervised transformations).
- ****fit_params** (*dict*) – Additional fit parameters.

Returns **X_new** – Transformed array.

Return type ndarray array of shape (n_samples, n_features_new)

classmethod format_bins (*bins*, *index=False*, *ellipsis=None*)

format bins to label

Parameters

- **bins** (*ndarray*) – bins to format
- **index** (*bool*) – if need index prefix
- **ellipsis** (*int*) – max length threshold that labels will not be ellipsis, *None* for skipping ellipsis

Returns array of labels

Return type ndarray

load (*rules*, *update=False*, ***kwargs*)

load rules from dict or json file

Parameters

- **rules** (*dict*) – dictionary of rules
- **from_json** (*str* | *IOBase*) – json file of rules
- **update** (*bool*) – if need to use updating instead of replacing rules

classmethod parse_bins (*bins*)

parse labeled bins to array

set_output (***, *transform=None*)

Set output container.

See sphx_glr_auto_examples_misellaneous_plot_set_output.py for an example on how to use the API.

Parameters **transform** (`{"default", "pandas"}`, `default=None`) – Configure output of *transform* and *fit_transform*.

- `"default"`: Default output format of a transformer
- `"pandas"`: DataFrame output
- `None`: Transform configuration is unchanged

Returns **self** – Estimator instance.

Return type estimator instance

transform (*X*, **args*, ***kwargs*)
transform method, see details in *transform_* method

update (**args*, ***kwargs*)
update rules

Parameters

- **rules** (*dict*) – dictionary of rules
- **from_json** (*str* | *IOBase*) – json file of rules

class `toad.transform.GBDTTransformer`

Bases: `toad.transform.Transformer`

GBDT transformer

__init__ ()
Initialize self. See `help(type(self))` for accurate signature.

fit_ (*X*, *y*, ***kwargs*)
fit GBDT transformer

Parameters

- **X** (*DataFrame* | *array-like*) –
- **y** (*str* | *array-like*) –
- **select_dtypes** (*str* | *numpy.dtypes*) – ‘object’, ‘number’ etc. only selected dtypes will be transform,

transform_ (*rules*, *X*)
transform woe

Parameters **X** (*DataFrame* | *array-like*) –

Returns array-like

export (***kwargs*)
export rules to dict or a json file

Parameters **to_json** (*str* | *IOBase*) – json file to save rules

Returns dictionary of rules

Return type dict

fit (*X*, **args*, *update=False*, ***kwargs*)
fit method, see details in *fit_* method

fit_transform (*X*, *y=None*, ***fit_params*)
Fit to data, then transform it.

Fits transformer to *X* and *y* with optional parameters *fit_params* and returns a transformed version of *X*.

Parameters

- **X** (*array-like of shape (n_samples, n_features)*) – Input samples.
- **y** (*array-like of shape (n_samples,) or (n_samples, n_outputs), default=None*) – Target values (None for unsupervised transformations).
- ****fit_params** (*dict*) – Additional fit parameters.

Returns **X_new** – Transformed array.

Return type ndarray array of shape (n_samples, n_features_new)

load (*rules, update=False, **kwargs*)
load rules from dict or json file

Parameters

- **rules** (*dict*) – dictionary of rules
- **from_json** (*str / IOBase*) – json file of rules
- **update** (*bool*) – if need to use updating instead of replacing rules

set_output (**, transform=None*)
Set output container.

See sphx_glr_auto_examples_miscellaneous_plot_set_output.py for an example on how to use the API.

Parameters transform (*{"default", "pandas"}, default=None*) – Configure output of *transform* and *fit_transform*.

- *"default"*: Default output format of a transformer
- *"pandas"*: DataFrame output
- *None*: Transform configuration is unchanged

Returns **self** – Estimator instance.

Return type estimator instance

transform (*X, *args, **kwargs*)
transform method, see details in *transform_* method

update (**args, **kwargs*)
update rules

Parameters

- **rules** (*dict*) – dictionary of rules
- **from_json** (*str / IOBase*) – json file of rules

3.2.9 toad.preprocessing module

toad.preprocessing.process module

class toad.preprocessing.process.**Processing** (*data*)
Bases: object
Examples:

```

>>> (Processing(data)
...   .groupby('id')
...   .partitionby(TimePartition(
...       'base_time',
...       'filter_time',
...       ['30d', '60d', '180d', '365d', 'all']
...   ))
...   .apply({'A': ['max', 'min', 'mean']})
...   .apply({'B': ['max', 'min', 'mean']})
...   .apply({'C': 'nunique'})
...   .apply({'D': {
...       'f': len,
...       'name': 'normal_count',
...       'mask': Mask('D').isin(['normal']),
...   }})
...   .apply({'id': 'count'})
...   .exec()
... )

```

__init__ (*data*)

Initialize self. See help(type(self)) for accurate signature.

groupby (*name*)

group data by name

Parameters **name** (*str*) – column name in data

apply (*f*)

apply functions to data

Parameters **f** (*dict/function*) – a config dict that keys are the column names and values are the functions, it will take the column series as the functions argument. if *f* is a function, it will take the whole dataframe as the argument.

partitionby (*p*)

partition data to multiple pieces, processing will process to all the pieces

Parameters **p** (*Partition*) –

class toad.preprocessing.process.**Mask** (*column=None*)

Bases: object

a placeholder to select dataframe

__init__ (*column=None*)

Initialize self. See help(type(self)) for accurate signature.

class toad.preprocessing.process.**F** (*f, name=None, mask=None*)

Bases: object

function class for processing

__init__ (*f, name=None, mask=None*)

Initialize self. See help(type(self)) for accurate signature.

toad.preprocessing.partition module

class toad.preprocessing.partition.**TimePartition** (*base, filter, times*)

Bases: toad.preprocessing.partition.Partition

partition data by time delta

Parameters

- **base** (*str*) – column name of base time
- **filter** (*str*) – column name of target time to be compared
- **times** (*list*) – list of time delta

Example:

```
>>> TimePartition('apply_time', 'query_time', ['30d', '90d', 'all'])
```

__init__ (*base, filter, times*)

Initialize self. See help(type(self)) for accurate signature.

partition (*data*)

partition data

Parameters **data** (*DataFrame*) – dataframe

Returns mask of partition data iterator -> str: suffix string of current partition

Return type iterator -> ndarray[bool]

class toad.preprocessing.partition.**ValuePartition** (*column*)

Bases: toad.preprocessing.partition.Partition

partition data by column values

Parameters **column** (*str*) – column name which will be used as partition

Example:

```
>>> ValuePartition('status')
```

__init__ (*column*)

Initialize self. See help(type(self)) for accurate signature.

partition (*data*)

partition data

Parameters **data** (*DataFrame*) – dataframe

Returns mask of partition data iterator -> str: suffix string of current partition

Return type iterator -> ndarray[bool]

3.2.10 toad.nn module

toad.nn.module module

class toad.nn.module.**Module**

Bases: torch.nn.modules.module.Module

base module for every model

Examples

```

>>> from toad.nn import Module
... from torch import nn
...
... class Net(Module):
...     def __init__(self, inputs, hidden, outputs):
...         super().__init__()
...         self.model = nn.Sequential(
...             nn.Linear(inputs, hidden),
...             nn.ReLU(),
...             nn.Linear(hidden, outputs),
...             nn.Sigmoid(),
...         )
...
...     def forward(self, x):
...         return self.model(x)
...
...     def fit_step(self, batch):
...         x, y = batch
...         y_hat = self(x)
...
...         # log into history
...         self.log('y', y)
...         self.log('y_hat', y_hat)
...
...         return nn.functional.mse_loss(y_hat, y)
...
... model = Net(10, 4, 1)
...
... model.fit(train_loader)

```

__init__()
define model struct

device
device of model

fit (*loader*, *trainer=None*, *optimizer=None*, *loss=None*, *early_stopping=None*, ***kwargs*)
train model

Parameters

- **loader** (*DataLoader*) – loader for training model
- **trainer** (*Trainer*) – trainer for training model
- **optimizer** (*torch.Optimier*) – the default optimizer is *Adam(lr = 1e-3)*
- **loss** (*Callable*) – could be called as ‘loss(y_hat, y)’
- **early_stopping** (*earlystopping*) – the default value is *loss_earlystopping*, you can set it to *False* to disable early stopping
- **epoch** (*int*) – number of epoch for training loop
- **callback** (*callable*) – callable function will be called every epoch

evaluate (*loader*, *trainer=None*)
evaluate model

Parameters

- **loader** (*DataLoader*) – loader for evaluate model

- **trainer** (*Trainer*) – trainer for evaluate model

fit_step (*batch*, *loss=None*, **args*, ***kwargs*)
step for fitting

Parameters

- **batch** (*Any*) – batch data from dataloader
- **loss** (*Callable*) – could be called as 'loss(y_hat, y)'

Returns loss of this step

Return type Tensor

save (*path*)
save model

load (*path*)
load model

log (*key*, *value*)
log values to history

Parameters

- **key** (*str*) – name of message
- **value** (*Tensor*) – tensor of values

distributed (*backend=None*, ***kwargs*)
get distributed model

class toad.nn.module.**DistModule** (*module*, *device_ids=None*, *output_device=None*,
dim=0, *broadcast_buffers=True*, *process_group=None*,
bucket_cap_mb=25, *find_unused_parameters=False*,
check_reduction=False, *gradient_as_bucket_view=False*,
static_graph=False)

Bases: torch.nn.parallel.distributed.DistributedDataParallel

distributed module class

toad.nn.functional module

toad.nn.functional.**flooding** (*loss*, *b*)
flooding loss

toad.nn.functional.**focal_loss**
focal loss

Parameters

- **input** (*Tensor*) – N x C, C is the number of classes
- **target** (*Tensor*) – N, each value is the index of classes
- **alpha** (*Variable*) – balaced variant of focal loss, range is in [0, 1]
- **gamma** (*float*) – focal loss parameter
- **reduction** (*str*) – *mean*, *sum*, *none* for reduce the loss of each classes

toad.nn.functional.**binary_focal_loss**
binary focal loss

```
toad.nn.functional.focal_loss_for_numpy(input, target, alpha=1.0, gamma=2.0, reduction='mean')
    focal loss for numpy array

toad.nn.functional.label_smoothing(labels, smoothing=0.1)
    label smoothing
```

toad.nn.trainer module

```
class toad.nn.trainer.History
    Bases: object
    model history
    __init__()
        Initialize self. See help(type(self)) for accurate signature.
    log(key, value)
        log message to history

    Parameters
    • key (str) – name of message
    • value (Tensor) – tensor of values
```

```
class toad.nn.trainer.callback(*args, **kwargs)
    Bases: toad.utils.decorator.Decorator
    callback for trainer
```

Examples

```
>>> @callback
... def savemodel(model):
...     model.save("path_to_file")
...
... trainer.train(model, callback = savemodel)
```

```
__init__(*args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
```

```
class toad.nn.trainer.earlystopping(*args, **kwargs)
    Bases: toad.nn.trainer.callback.callback
```

Examples

```
>>> @earlystopping(delta = 1e-3, patience = 5)
... def auc(history):
...     return AUC(history['y_hat'], history['y'])
```

```
setup(delta=-0.001, patience=10, skip=0)
```

Parameters

- **delta** (*float*) – stop training if diff of new score is smaller than delta
- **patience** (*int*) – patience of rounds to stop training

- **skip** (*int*) – n rounds from starting training to warm up

get_best_state ()
get best state of model

reset ()

class toad.nn.trainer.Trainer (*model*, *loader=None*, *optimizer=None*, *loss=None*,
keep_history=None, *early_stopping=None*)

Bases: toad.nn.trainer.event.Event

trainer for training models

__init__ (*model*, *loader=None*, *optimizer=None*, *loss=None*, *keep_history=None*,
early_stopping=None)
initialization

Parameters

- **model** (*nn.Module*) – model will be trained
- **loader** (*torch.DataLoader*) – training data loader
- **optimizer** (*torch.Optimier*) – the default optimizer is *Adam(lr = 1e-3)*
- **loss** (*Callable*) – could be called as ‘loss(y_hat, y)’
- **early_stopping** (*earlystopping*) – the default value is *loss_earlystopping*, you can set it to *False* to disable early stopping
- **keep_history** (*int*) – keep the last n-th epoch logs, *None* will keep all

set_model (*model*)
setup model

distributed (*address=None*, *workers=4*, *gpu=False*)
setting distribution enviroment and initial a ray cluster connection

Parameters

- **address** (*string*) – the head of ray cluster address
- **workers** (*int*) – compute task’s resource
- **gpu** (*Booleans*) – whether use GPU, “True” or “False”

train (*loader=None*, *epoch=10*, ***kwargs*)

Parameters

- **loader** (*torch.DataLoader*) – training data loader
- **epoch** (*int*) – number of epoch for training loop
- **callback** (*list[Callback]*) – callable function will be called every epoch - parameters of callback

model (*nn.Module*): the training model history (*History*): history of total log records
epoch (*int*): current epoch number trainer (*Trainer*): self trainer

- **start** (*int*) – epoch start from n round
- **backward_rounds** (*int*) – backward after every n rounds

Returns the model with best performents

Return type *Module*

evaluate (*loader*, *callback=None*)
evaluate model

Parameters

- **loader** (*torch.DataLoader*) – evaluation data loader
- **callback** (*callable*) – callback function

3.2.11 toad.utils module

toad.utils.func module

`toad.utils.func.to_ndarray` (*s*, *dtype=None*)

`toad.utils.func.bin_by_splits` (*feature*, *splits*)
Bin feature by split points

`toad.utils.func.feature_splits` (*feature*, *target*)
find possibility split points

`toad.utils.func.iter_df` (*dataframe*, *feature*, *target*, *splits*)
iterate dataframe by split points

Returns iterator (df, splitter)

`toad.utils.func.split_target` (*frame*, *target*)

`toad.utils.func.save_json` (*contents*, *file*, *indent=4*)
save json file

Parameters

- **contents** (*dict*) – contents to save
- **file** (*str* | *IOBase*) – file to save

`toad.utils.func.read_json` (*file*)
read json file

`toad.utils.func.clip` (*series*, *value=None*, *std=None*, *quantile=None*)
clip series

Parameters

- **series** (*array-like*) – series need to be clipped
- **value** (*number* | *tuple*) – min/max value of clipping
- **std** (*number* | *tuple*) – min/max std of clipping
- **quantile** (*number* | *tuple*) – min/max quantile of clipping

`toad.utils.func.flatten_columns` (*columns*, *sep='_'*)
flatten multiple columns to 1-dim columns joined with '_'

`toad.utils.func.bin_to_number` (*reg=None*)

Returns func(string) -> number

Return type function

`toad.utils.func.generate_target` (*size*, *rate=0.5*, *weight=None*, *reverse=False*)
generate target for reject inference

Parameters

- **size** (*int*) – size of target
- **rate** (*float*) – rate of ‘1’ in target
- **weight** (*array-like*) – weight of ‘1’ to generate target
- **reverse** (*bool*) – if need reverse weight

Returns array

```
toad.utils.func.get_dummies (dataframe, exclude=None, binary_drop=False, **kwargs)
get dummies
```

toad.utils.decorator module

```
class toad.utils.decorator.Decorator (*args, is_class=False, **kwargs)
    Bases: object
    base decorator class

    __init__ (*args, is_class=False, **kwargs)
        Initialize self. See help(type(self)) for accurate signature.

class toad.utils.decorator.frame_exclude (*args, is_class=False, **kwargs)
    Bases: toad.utils.decorator.Decorator
    decorator for exclude columns

class toad.utils.decorator.select_dtypes (*args, is_class=False, **kwargs)
    Bases: toad.utils.decorator.Decorator
    decorator for select frame by dtypes

class toad.utils.decorator.save_to_json (*args, is_class=False, **kwargs)
    Bases: toad.utils.decorator.Decorator
    support save result to json file

class toad.utils.decorator.load_from_json (*args, is_class=False, **kwargs)
    Bases: toad.utils.decorator.Decorator
    support load data from json file

class toad.utils.decorator.support_dataframe (*args, is_class=False, **kwargs)
    Bases: toad.utils.decorator.Decorator
    decorator for supporting dataframe

class toad.utils.decorator.proxy_docstring (*args, is_class=False, **kwargs)
    Bases: toad.utils.decorator.Decorator

class toad.utils.decorator.support_numpy (*args, is_class=False, **kwargs)
    Bases: toad.utils.decorator.Decorator
    decorator for supporting numpy array to use torch function

class toad.utils.decorator.xgb_loss (*args, is_class=False, **kwargs)
    Bases: toad.utils.decorator.Decorator
    decorator for converting function to xgb supported loss function
```

Parameters

- **loss_func** (*callable*) – loss function

- ****kwargs** – other arguments for loss function except *pred* and *label*

Examples:

```
>>> @xgb_loss(**kwargs)
>>> def loss_func(pred, label, **kwargs):
>>>     ...
>>>     return loss
>>>
>>> # or use `xgb_loss` directly
>>> xgb_func = xgb_loss(**kwargs)(loss_func)
>>>
>>> # use in xgb
>>> model = xgb.XGBClassifier(objective = xgb_func)
```

class toad.utils.decorator.**performance**(*args, is_class=False, **kwargs)

Bases: *toad.utils.decorator.Decorator*

decorator for analysis code performance

Parameters **loop** (*int*) – loop times, default *1*

Examples: >>> @performance(loop = 100) >>> def func(): >>> ... # code >>> return res >>> >>> func() >>>
>>> # or use *performance* in *with* statement >>> with performance(): >>> ... # code

toad.utils.mixin module

3.3 Module contents

CHAPTER 4

Indices and tables

- `genindex`
- `modindex`
- `search`

CHAPTER 5

Links

FiboRule

t

- [toad](#), [32](#)
- [toad.detector](#), [5](#)
- [toad.merge](#), [6](#)
- [toad.metrics](#), [8](#)
- [toad.nn.functional](#), [27](#)
- [toad.nn.module](#), [25](#)
- [toad.nn.trainer](#), [28](#)
- [toad.plot](#), [10](#)
- [toad.preprocessing.partition](#), [24](#)
- [toad.preprocessing.process](#), [23](#)
- [toad.scorecard](#), [11](#)
- [toad.selection](#), [13](#)
- [toad.stats](#), [15](#)
- [toad.transform](#), [17](#)
- [toad.utils.decorator](#), [31](#)
- [toad.utils.func](#), [30](#)
- [toad.utils.mixin](#), [32](#)

Symbols

`__init__` (*toad.transform.Combiner* attribute), 20
`__init__` (*toad.transform.Transformer* attribute), 17
`__init__` (*toad.transform.WOETransformer* attribute), 19
`__init__` () (*toad.nn.module.Module* method), 26
`__init__` () (*toad.nn.trainer.History* method), 28
`__init__` () (*toad.nn.trainer.Trainer* method), 29
`__init__` () (*toad.nn.trainer.callback* method), 28
`__init__` () (*toad.preprocessing.partition.TimePartition* method), 25
`__init__` () (*toad.preprocessing.partition.ValuePartition* method), 25
`__init__` () (*toad.preprocessing.process.F* method), 24
`__init__` () (*toad.preprocessing.process.Mask* method), 24
`__init__` () (*toad.preprocessing.process.Processing* method), 24
`__init__` () (*toad.scorecard.ScoreCard* method), 11
`__init__` () (*toad.transform.GBDTTransformer* method), 22
`__init__` () (*toad.utils.decorator.Decorator* method), 31

A

`after_export` () (*toad.scorecard.ScoreCard* method), 12
`after_load` () (*toad.scorecard.ScoreCard* method), 12
`AIC` () (*in module toad.metrics*), 8
`apply` () (*toad.preprocessing.process.Processing* method), 24
`AUC` () (*in module toad.metrics*), 9

B

`badrate` () (*in module toad.stats*), 16
`badrate_plot` () (*in module toad.plot*), 10
`BIC` () (*in module toad.metrics*), 9

`bin_by_splits` () (*in module toad.utils.func*), 30
`bin_plot` () (*in module toad.plot*), 11
`bin_to_number` () (*in module toad.utils.func*), 30
`bin_to_score` () (*toad.scorecard.ScoreCard* method), 12
`binary_focal_loss` (*in module toad.nn.functional*), 27

C

`callback` (*class in toad.nn.trainer*), 28
`ChiMerge` () (*in module toad.merge*), 6
`clip` () (*in module toad.utils.func*), 30
`coef` (*toad.scorecard.ScoreCard* attribute), 11
`column_quality` () (*in module toad.stats*), 16
`Combiner` (*class in toad.transform*), 20
`corr_plot` () (*in module toad.plot*), 10
`countBlank` () (*in module toad.detector*), 6

D

`Decorator` (*class in toad.utils.decorator*), 31
`detect` () (*in module toad.detector*), 6
`device` (*toad.nn.module.Module* attribute), 26
`DistModule` (*class in toad.nn.module*), 27
`distributed` () (*toad.nn.module.Module* method), 27
`distributed` () (*toad.nn.trainer.Trainer* method), 29
`drop_corr` () (*in module toad.selection*), 14
`drop_empty` () (*in module toad.selection*), 13
`drop_iv` () (*in module toad.selection*), 14
`drop_var` () (*in module toad.selection*), 13
`drop_vif` () (*in module toad.selection*), 14
`DTMerge` () (*in module toad.merge*), 6

E

`earlystopping` (*class in toad.nn.trainer*), 28
`entropy` () (*in module toad.stats*), 15
`entropy_cond` (*in module toad.stats*), 15
`evaluate` () (*toad.nn.module.Module* method), 26
`evaluate` () (*toad.nn.trainer.Trainer* method), 29
`export` () (*toad.transform.Combiner* method), 21

`export()` (*toad.transform.GBDTTransformer* method), 22
`export()` (*toad.transform.Transformer* method), 17
`export()` (*toad.transform.WOETransformer* method), 19

F

F (*class in toad.preprocessing.process*), 24
`F1()` (*in module toad.metrics*), 9
`feature_bin_stats()` (*in module toad.stats*), 17
`feature_splits()` (*in module toad.utils.func*), 30
`fit()` (*toad.nn.module.Module* method), 26
`fit()` (*toad.scorecard.ScoreCard* method), 11
`fit()` (*toad.transform.Combiner* method), 21
`fit()` (*toad.transform.GBDTTransformer* method), 22
`fit()` (*toad.transform.Transformer* method), 17
`fit()` (*toad.transform.WOETransformer* method), 19
`fit_()` (*toad.transform.Combiner* method), 20
`fit_()` (*toad.transform.GBDTTransformer* method), 22
`fit_()` (*toad.transform.WOETransformer* method), 18
`fit_step()` (*toad.nn.module.Module* method), 27
`fit_transform()` (*toad.transform.Combiner* method), 21
`fit_transform()` (*toad.transform.GBDTTransformer* method), 22
`fit_transform()` (*toad.transform.Transformer* method), 18
`fit_transform()` (*toad.transform.WOETransformer* method), 19
`flatten_columns()` (*in module toad.utils.func*), 30
`flooding()` (*in module toad.nn.functional*), 27
`focal_loss` (*in module toad.nn.functional*), 27
`focal_loss_for_numpy()` (*in module toad.nn.functional*), 27
`format_bins()` (*toad.transform.Combiner* class method), 21
`frame_exclude` (*class in toad.utils.decorator*), 31

G

GBDTTransformer (*class in toad.transform*), 22
`generate_target()` (*in module toad.utils.func*), 30
`get_best_state()` (*toad.nn.trainer.earlystopping* method), 29
`get_dummies()` (*in module toad.utils.func*), 31
`get_reason()` (*toad.scorecard.ScoreCard* method), 11
`getDescribe()` (*in module toad.detector*), 5
`getTopValues()` (*in module toad.detector*), 5
`gini()` (*in module toad.stats*), 15
`gini_cond` (*in module toad.stats*), 15
`groupby()` (*toad.preprocessing.process.Processing* method), 24

H

History (*class in toad.nn.trainer*), 28

I

indicator (*class in toad.stats*), 16
`isNumeric()` (*in module toad.detector*), 6
`iter_df()` (*in module toad.utils.func*), 30
IV (*in module toad.stats*), 16

K

KMeansMerge (*in module toad.merge*), 7
KS() (*in module toad.metrics*), 8
KS_bucket() (*in module toad.metrics*), 8
KS_by_col() (*in module toad.metrics*), 8
ks_plot() (*in module toad.plot*), 10

L

`label_smoothing()` (*in module toad.nn.functional*), 28
`load()` (*toad.nn.module.Module* method), 27
`load()` (*toad.transform.Combiner* method), 21
`load()` (*toad.transform.GBDTTransformer* method), 23
`load()` (*toad.transform.Transformer* method), 18
`load()` (*toad.transform.WOETransformer* method), 19
`load_from_json` (*class in toad.utils.decorator*), 31
`log()` (*toad.nn.module.Module* method), 27
`log()` (*toad.nn.trainer.History* method), 28

M

Mask (*class in toad.preprocessing.process*), 24
matrix() (*in module toad.metrics*), 9
merge (*in module toad.merge*), 7
Module (*class in toad.nn.module*), 25
MSE() (*in module toad.metrics*), 8

P

`parse_bins()` (*toad.transform.Combiner* class method), 21
`partition()` (*toad.preprocessing.partition.TimePartition* method), 25
`partition()` (*toad.preprocessing.partition.ValuePartition* method), 25
`partitionby()` (*toad.preprocessing.process.Processing* method), 24
performance (*class in toad.utils.decorator*), 32
`predict()` (*toad.scorecard.ScoreCard* method), 11
`predict_proba()` (*toad.scorecard.ScoreCard* method), 12
`proba_to_score()` (*toad.scorecard.ScoreCard* method), 12
`probability()` (*in module toad.stats*), 16
Processing (*class in toad.preprocessing.process*), 23
`proportion_plot()` (*in module toad.plot*), 10

proxy_docstring (class in toad.utils.decorator), 31
 PSI () (in module toad.metrics), 9

Q

quality () (in module toad.stats), 17
 QuantileMerge () (in module toad.merge), 7

R

read_json () (in module toad.utils.func), 30
 reset () (toad.nn.trainer.earlystopping method), 29
 roc_plot () (in module toad.plot), 10

S

save () (toad.nn.module.Module method), 27
 save_json () (in module toad.utils.func), 30
 save_to_json (class in toad.utils.decorator), 31
 score_to_proba () (toad.scorecard.ScoreCard method), 12
 ScoreCard (class in toad.scorecard), 11
 select () (in module toad.selection), 14
 select_dtypes (class in toad.utils.decorator), 31
 set_model () (toad.nn.trainer.Trainer method), 29
 set_output () (toad.transform.Combiner method), 21
 set_output () (toad.transform.GBDTTransformer method), 23
 set_output () (toad.transform.Transformer method), 18
 set_output () (toad.transform.WOETransformer method), 19
 set_rules () (toad.transform.Combiner method), 20
 setup () (toad.nn.trainer.earlystopping method), 28
 split_target () (in module toad.utils.func), 30
 SSE () (in module toad.metrics), 8
 StepMerge () (in module toad.merge), 7
 stepwise () (in module toad.selection), 13
 support_dataframe (class in toad.utils.decorator), 31
 support_numpy (class in toad.utils.decorator), 31

T

testing_frame () (toad.scorecard.ScoreCard method), 12
 TimePartition (class in toad.preprocessing.partition), 24
 to_ndarray () (in module toad.utils.func), 30
 toad (module), 32
 toad.detector (module), 5
 toad.merge (module), 6
 toad.metrics (module), 8
 toad.nn.functional (module), 27
 toad.nn.module (module), 25
 toad.nn.trainer (module), 28
 toad.plot (module), 10

toad.preprocessing.partition (module), 24
 toad.preprocessing.process (module), 23
 toad.scorecard (module), 11
 toad.selection (module), 13
 toad.stats (module), 15
 toad.transform (module), 17
 toad.utils.decorator (module), 31
 toad.utils.func (module), 30
 toad.utils.mixin (module), 32
 train () (toad.nn.trainer.Trainer method), 29
 Trainer (class in toad.nn.trainer), 29
 transform () (toad.transform.Combiner method), 22
 transform () (toad.transform.GBDTTransformer method), 23
 transform () (toad.transform.Transformer method), 17
 transform () (toad.transform.WOETransformer method), 20
 transform_ () (toad.transform.Combiner method), 20
 transform_ () (toad.transform.GBDTTransformer method), 22
 transform_ () (toad.transform.WOETransformer method), 19
 Transformer (class in toad.transform), 17

U

update () (toad.transform.Combiner method), 22
 update () (toad.transform.GBDTTransformer method), 23
 update () (toad.transform.Transformer method), 18
 update () (toad.transform.WOETransformer method), 20

V

ValuePartition (class in toad.preprocessing.partition), 25
 VIF () (in module toad.stats), 16

W

WOE () (in module toad.stats), 16
 woe_to_score () (toad.scorecard.ScoreCard method), 12
 WOETransformer (class in toad.transform), 18

X

xgb_loss (class in toad.utils.decorator), 31