
toad

Release 0.0.62

Feb 19, 2021

Contents

1	Installation	1
2	Tutorial	3
3	Contents	5
3.1	toad package	5
3.2	Submodules	5
3.3	Module contents	27
4	Indices and tables	29
	Python Module Index	31
	Index	33

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=[None])`

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*)

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.bin_plot` (*frame*, *x=None*, *target='target'*, *iv=True*, *annotate_format='.2f'*)

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

Returns bins' proportion and badrate plot

Return type Axes

3.2.5 toad.scorecard module

class toad.scorecard.**ScoreCard** (*pdo=60, rate=2, base_odds=35, base_score=750, card=None, combiner={}, transe=None, **kwargs*)
 Bases: sklearn.base.BaseEstimator, toad.utils.mixin.RulesMixin, toad.utils.mixin.BinsMixin

coef_
 coef of LR model

intercept_

n_features_

features_

combiner

fit (*X, y*)

Parameters

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

predict (*X, **kwargs*)

predict score :param X: X to predict :type X: 2D array-like :param return_sub: if need to return sub score, default *False* :type return_sub: bool

Returns predicted score DataFrame: sub score for each feature

Return type array-like

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-like/float

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

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

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

class toad.selection.**StatsModel** (*estimator='ols'*, *criterion='aic'*, *intercept=False*)

Bases: object

get_estimator (*name*)

stats (*X*, *y*)

get_criterion (*pre*, *y*, *k*)

t_value (*pre*, *y*, *X*, *coef*)

p_value (*t*, *n*)

loglikelihood (*pre*, *y*, *k*)

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 float, 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.badrates(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

name = 'indicator'

need_merge = False

dtype = None

wrapper (**args, **kwargs*)

`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 maximum 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

3.2.8 toad.transform module

class toad.transform.Transformer

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

Base class for transformers

fit ()

fit method, see details in *fit_* method

transform (*X*, **args*, ***kwargs*)

transform method, see details in *transform_* method

default_rule ()

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

rules

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

update rules

Parameters

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

class toad.transform.WOETransformerBases: *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***default_rule** ()**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** ()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

rules

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

```
ELSE_GROUP = 'else'
EMPTY_BIN = -1
NUMBER_EXP = re.compile('\\[(-inf|-?\\d+(\\.\\d+)?)\\s*[~]\\s*(inf|-?\\d+(\\.\\d+)?)\\]\\s*')
default_rule()
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()
    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
```


rules

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

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

default_rule ()

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 ()
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

rules

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

Example:

```
>>> (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()
... )
```

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.

append_func (*col, func*)

partitionby (*p*)

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

Parameters **p** (*Partition*) –

exec ()

process (*data*)

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

Bases: object

a placeholder to select dataframe

push (*op, value*)

replay (*data*)

isin (*other*)

isna ()

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

Bases: object

function class for processing

name

is_buildin

need_filter

filter (*data*)

toad.preprocessing.partition module

class toad.preprocessing.partition.**Partition**

Bases: object

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.**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'])
```

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')
```

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.utils module

toad.utils.func module

class toad.utils.func.**Parallel**

Bases: object

apply (*func*, *args=()*, *kwargs={}*)

join ()

toad.utils.func.**np_count** (*arr*, *value*, *default=None*)

toad.utils.func.**has_nan** (*arr*)

toad.utils.func.**np_unique** (*arr*, ***kwargs*)

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

toad.utils.func.**fillna** (*feature*, *by=-1*)

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.inter_feature (feature, splits)
toad.utils.func.is_continuous (series)
toad.utils.func.split_target (frame, target)
toad.utils.func.unpack_tuple (x)
toad.utils.func.generate_str (size=6, chars='ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789')
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.diff_time (base, target, format=None, time='day')
toad.utils.func.diff_time_frame (base, frame, format=None)
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 decorater class
    is_class = False
    setup (*args, **kwargs)
    call (*args, **kwargs)
    wrapper (*args, **kwargs)

class toad.utils.decorator.frame_exclude (*args, is_class=False, **kwargs)
    Bases: toad.utils.decorator.Decorator
    decorator for exclude columns
    wrapper (X, *args, exclude=None, **kwargs)

class toad.utils.decorator.select_dtypes (*args, is_class=False, **kwargs)
    Bases: toad.utils.decorator.Decorator
    decorator for select frame by dtypes
    wrapper (X, *args, select_dtypes=None, **kwargs)

class toad.utils.decorator.save_to_json (*args, is_class=False, **kwargs)
    Bases: toad.utils.decorator.Decorator
    support save result to json file
    wrapper (*args, to_json=None, **kwargs)

class toad.utils.decorator.load_from_json (*args, is_class=False, **kwargs)
    Bases: toad.utils.decorator.Decorator
    support load data from json file
    require_first = False
    wrapper (*args, from_json=None, **kwargs)

class toad.utils.decorator.support_dataframe (*args, is_class=False, **kwargs)
    Bases: toad.utils.decorator.Decorator
    decorator for supporting dataframe
    require_target = True
    target = 'target'
    wrapper (frame, *args, **kwargs)

class toad.utils.decorator.proxy_docstring (*args, is_class=False, **kwargs)
    Bases: toad.utils.decorator.Decorator
    method_name = None
```

toad.utils.mixin module

```
class toad.utils.mixin.RulesMixin
    Bases: object
```

```

default_rule()

rules

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

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

update (*args, **kwargs)
    update rules

    Parameters

    • rules (dict) – dictionary of rules

    • from_json (str | IOBase) – json file of rules

class toad.utils.mixin.BinsMixin
    Bases: object

    EMPTY_BIN = -1

    ELSE_GROUP = 'else'

    NUMBER_EXP = re.compile('\[(-inf|-?\d+(\.\d+)?)\s*[~]\s*(inf|-?\d+(\.\d+)?)\]\s*')

    classmethod parse_bins (bins)
        parse labeled bins to array

    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

```

3.3 Module contents

CHAPTER 4

Indices and tables

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

t

- `toad`, [27](#)
- `toad.detector`, [5](#)
- `toad.merge`, [6](#)
- `toad.metrics`, [8](#)
- `toad.plot`, [10](#)
- `toad.preprocessing.partition`, [23](#)
- `toad.preprocessing.process`, [22](#)
- `toad.scorecard`, [11](#)
- `toad.selection`, [12](#)
- `toad.stats`, [15](#)
- `toad.transform`, [17](#)
- `toad.utils.decorator`, [26](#)
- `toad.utils.func`, [24](#)
- `toad.utils.mixin`, [26](#)

A

after_export() (toad.scorecard.ScoreCard method), 12
 AIC() (in module toad.metrics), 8
 append_func() (toad.preprocessing.process.Processing method), 23
 apply() (toad.preprocessing.process.Processing method), 22
 apply() (toad.utils.func.Parallel 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), 24
 bin_plot() (in module toad.plot), 10
 bin_to_number() (in module toad.utils.func), 25
 bin_to_score() (toad.scorecard.ScoreCard method), 12
 BinsMixin (class in toad.utils.mixin), 27

C

call() (toad.utils.decorator.Decorator method), 26
 ChiMerge() (in module toad.merge), 6
 clip() (in module toad.utils.func), 25
 coef_ (toad.scorecard.ScoreCard attribute), 11
 column_quality() (in module toad.stats), 16
 Combiner (class in toad.transform), 19
 combiner (toad.scorecard.ScoreCard attribute), 11
 corr_plot() (in module toad.plot), 10
 countBlank() (in module toad.detector), 6

D

Decorator (class in toad.utils.decorator), 26
 default_rule() (toad.transform.Combiner method), 20
 default_rule() (toad.transform.GBDTTransformer method), 21

default_rule() (toad.transform.Transformer method), 17
 default_rule() (toad.transform.WOETTransformer method), 18
 default_rule() (toad.utils.mixin.RulesMixin method), 26
 detect() (in module toad.detector), 6
 diff_time() (in module toad.utils.func), 25
 diff_time_frame() (in module toad.utils.func), 25
 drop_corr() (in module toad.selection), 13
 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
 dtype (toad.stats.indicator attribute), 16

E

ELSE_GROUP (toad.transform.Combiner attribute), 19
 ELSE_GROUP (toad.utils.mixin.BinsMixin attribute), 27
 EMPTY_BIN (toad.transform.Combiner attribute), 20
 EMPTY_BIN (toad.utils.mixin.BinsMixin attribute), 27
 entropy() (in module toad.stats), 15
 entropy_cond (in module toad.stats), 15
 exec() (toad.preprocessing.process.Processing method), 23
 export() (toad.transform.Combiner method), 20
 export() (toad.transform.GBDTTransformer method), 21
 export() (toad.transform.Transformer method), 17
 export() (toad.transform.WOETTransformer method), 18
 export() (toad.utils.mixin.RulesMixin method), 27

F

F (class in toad.preprocessing.process), 23
 F1() (in module toad.metrics), 9
 feature_splits() (in module toad.utils.func), 24
 features_ (toad.scorecard.ScoreCard attribute), 11
 fillna() (in module toad.utils.func), 24

`filter()` (*toad.preprocessing.process.F* method), 23
`fit()` (*toad.scorecard.ScoreCard* method), 11
`fit()` (*toad.transform.Combiner* method), 20
`fit()` (*toad.transform.GBDTTransformer* method), 21
`fit()` (*toad.transform.Transformer* method), 17
`fit()` (*toad.transform.WOETransformer* method), 18
`fit_()` (*toad.transform.Combiner* method), 19
`fit_()` (*toad.transform.GBDTTransformer* method), 21
`fit_()` (*toad.transform.WOETransformer* method), 18
`fit_transform()` (*toad.transform.Combiner* method), 20
`fit_transform()` (*toad.transform.GBDTTransformer* method), 21
`fit_transform()` (*toad.transform.Transformer* method), 17
`fit_transform()` (*toad.transform.WOETransformer* method), 18
`flatten_columns()` (*in module toad.utils.func*), 25
`format_bins()` (*toad.transform.Combiner* class method), 20
`format_bins()` (*toad.utils.mixin.BinsMixin* class method), 27
`frame_exclude` (*class in toad.utils.decorator*), 26

G

`GBDTTransformer` (*class in toad.transform*), 21
`generate_str()` (*in module toad.utils.func*), 25
`generate_target()` (*in module toad.utils.func*), 25
`get_criterion()` (*toad.selection.StatsModel* method), 12
`get_dummies()` (*in module toad.utils.func*), 25
`get_estimator()` (*toad.selection.StatsModel* method), 12
`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), 22

H

`has_nan()` (*in module toad.utils.func*), 24

I

`indicator` (*class in toad.stats*), 16
`inter_feature()` (*in module toad.utils.func*), 24
`intercept_` (*toad.scorecard.ScoreCard* attribute), 11
`is_buildin` (*toad.preprocessing.process.F* attribute), 23
`is_class` (*toad.utils.decorator.Decorator* attribute), 26
`is_continuous()` (*in module toad.utils.func*), 25
`isin()` (*toad.preprocessing.process.Mask* method), 23
`isna()` (*toad.preprocessing.process.Mask* method), 23
`isNumeric()` (*in module toad.detector*), 6

`iter_df()` (*in module toad.utils.func*), 24
`IV` (*in module toad.stats*), 15

J

`join()` (*toad.utils.func.Parallel* method), 24

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

L

`load()` (*toad.transform.Combiner* method), 20
`load()` (*toad.transform.GBDTTransformer* method), 22
`load()` (*toad.transform.Transformer* method), 17
`load()` (*toad.transform.WOETransformer* method), 18
`load()` (*toad.utils.mixin.RulesMixin* method), 27
`load_from_json` (*class in toad.utils.decorator*), 26
`loglikelihood()` (*toad.selection.StatsModel* method), 12

M

`Mask` (*class in toad.preprocessing.process*), 23
`matrix()` (*in module toad.metrics*), 9
`merge` (*in module toad.merge*), 7
`method_name` (*toad.utils.decorator.proxy_docstring* attribute), 26
`MSE()` (*in module toad.metrics*), 8

N

`n_features_` (*toad.scorecard.ScoreCard* attribute), 11
`name` (*toad.preprocessing.process.F* attribute), 23
`name` (*toad.stats.indicator* attribute), 16
`need_filter` (*toad.preprocessing.process.F* attribute), 23
`need_merge` (*toad.stats.indicator* attribute), 16
`np_count()` (*in module toad.utils.func*), 24
`np_unique()` (*in module toad.utils.func*), 24
`NUMBER_EXP` (*toad.transform.Combiner* attribute), 20
`NUMBER_EXP` (*toad.utils.mixin.BinsMixin* attribute), 27

P

`p_value()` (*toad.selection.StatsModel* method), 12
`Parallel` (*class in toad.utils.func*), 24
`parse_bins()` (*toad.transform.Combiner* class method), 20
`parse_bins()` (*toad.utils.mixin.BinsMixin* class method), 27
`Partition` (*class in toad.preprocessing.partition*), 23
`partition()` (*toad.preprocessing.partition.Partition* method), 23
`partition()` (*toad.preprocessing.partition.TimePartition* method), 24

[partition\(\)](#) (*toad.preprocessing.partition.ValuePartition* method), 24
[partitionby\(\)](#) (*toad.preprocessing.process.Processing* method), 23
[predict\(\)](#) (*toad.scorecard.ScoreCard* method), 11
[predict_proba\(\)](#) (*toad.scorecard.ScoreCard* method), 11
[proba_to_score\(\)](#) (*toad.scorecard.ScoreCard* method), 11
[probability\(\)](#) (in module *toad.stats*), 15
[process\(\)](#) (*toad.preprocessing.process.Processing* method), 23
[Processing](#) (class in *toad.preprocessing.process*), 22
[proportion_plot\(\)](#) (in module *toad.plot*), 10
[proxy_docstring](#) (class in *toad.utils.decorator*), 26
[PSI\(\)](#) (in module *toad.metrics*), 9
[push\(\)](#) (*toad.preprocessing.process.Mask* method), 23

Q

[quality\(\)](#) (in module *toad.stats*), 16
[QuantileMerge\(\)](#) (in module *toad.merge*), 7

R

[read_json\(\)](#) (in module *toad.utils.func*), 25
[replay\(\)](#) (*toad.preprocessing.process.Mask* method), 23
[require_first](#) (*toad.utils.decorator.load_from_json* attribute), 26
[require_target](#) (*toad.utils.decorator.support_dataframe* attribute), 26
[roc_plot\(\)](#) (in module *toad.plot*), 10
[rules](#) (*toad.transform.Combiner* attribute), 20
[rules](#) (*toad.transform.GBDTTransformer* attribute), 22
[rules](#) (*toad.transform.Transformer* attribute), 17
[rules](#) (*toad.transform.WOETransformer* attribute), 19
[rules](#) (*toad.utils.mixin.RulesMixin* attribute), 27
[RulesMixin](#) (class in *toad.utils.mixin*), 26

S

[save_json\(\)](#) (in module *toad.utils.func*), 25
[save_to_json](#) (class in *toad.utils.decorator*), 26
[score_to_proba\(\)](#) (*toad.scorecard.ScoreCard* method), 11
[ScoreCard](#) (class in *toad.scorecard*), 11
[select\(\)](#) (in module *toad.selection*), 14
[select_dtypes](#) (class in *toad.utils.decorator*), 26
[set_rules\(\)](#) (*toad.transform.Combiner* method), 19
[setup\(\)](#) (*toad.utils.decorator.Decorator* method), 26
[split_target\(\)](#) (in module *toad.utils.func*), 25
[SSE\(\)](#) (in module *toad.metrics*), 8
[stats\(\)](#) (*toad.selection.StatsModel* method), 12
[StatsModel](#) (class in *toad.selection*), 12
[StepMerge\(\)](#) (in module *toad.merge*), 7
[stepwise\(\)](#) (in module *toad.selection*), 12

T

[support_dataframe](#) (class in *toad.utils.decorator*), 26
[t_value\(\)](#) (*toad.selection.StatsModel* method), 12
[target](#) (*toad.utils.decorator.support_dataframe* attribute), 26
[testing_frame\(\)](#) (*toad.scorecard.ScoreCard* method), 12
[TimePartition](#) (class in *toad.preprocessing.partition*), 23
[to_ndarray\(\)](#) (in module *toad.utils.func*), 24
[toad](#) (module), 27
[toad.detector](#) (module), 5
[toad.merge](#) (module), 6
[toad.metrics](#) (module), 8
[toad.plot](#) (module), 10
[toad.preprocessing.partition](#) (module), 23
[toad.preprocessing.process](#) (module), 22
[toad.scorecard](#) (module), 11
[toad.selection](#) (module), 12
[toad.stats](#) (module), 15
[toad.transform](#) (module), 17
[toad.utils.decorator](#) (module), 26
[toad.utils.func](#) (module), 24
[toad.utils.mixin](#) (module), 26
[transform\(\)](#) (*toad.transform.Combiner* method), 21
[transform\(\)](#) (*toad.transform.GBDTTransformer* method), 22
[transform\(\)](#) (*toad.transform.Transformer* method), 17
[transform\(\)](#) (*toad.transform.WOETransformer* method), 19
[transform_\(\)](#) (*toad.transform.Combiner* method), 19
[transform_\(\)](#) (*toad.transform.GBDTTransformer* method), 21
[transform_\(\)](#) (*toad.transform.WOETransformer* method), 18
[Transformer](#) (class in *toad.transform*), 17

U

[unpack_tuple\(\)](#) (in module *toad.utils.func*), 25
[update\(\)](#) (*toad.transform.Combiner* method), 21
[update\(\)](#) (*toad.transform.GBDTTransformer* method), 22
[update\(\)](#) (*toad.transform.Transformer* method), 17
[update\(\)](#) (*toad.transform.WOETransformer* method), 19
[update\(\)](#) (*toad.utils.mixin.RulesMixin* method), 27

V

[ValuePartition](#) (class in *toad.preprocessing.partition*), 24
[VIF\(\)](#) (in module *toad.stats*), 16

W

`WOE()` (in module *toad.stats*), [15](#)

`woe_to_score()` (*toad.scorecard.ScoreCard* method), [12](#)

`WOETransformer` (class in *toad.transform*), [18](#)

`wrapper()` (*toad.stats.indicator* method), [16](#)

`wrapper()` (*toad.utils.decorator.Decorator* method), [26](#)

`wrapper()` (*toad.utils.decorator.frame_exclude* method), [26](#)

`wrapper()` (*toad.utils.decorator.load_from_json* method), [26](#)

`wrapper()` (*toad.utils.decorator.save_to_json* method), [26](#)

`wrapper()` (*toad.utils.decorator.select_dtypes* method), [26](#)

`wrapper()` (*toad.utils.decorator.support_dataframe* method), [26](#)