
risksutils Documentation

Dmitry Shulchevskii

февр. 13, 2018

Содержание:

1	Установка	3
1.1	Визуализация	3
1.2	Reference	10
	Содержание модулей Python	13

В данном проекте содержится набор скриптов полезных в скоринге.

Для установки воспользуйтесь pip:

```
pip install risksutils
```

1.1 Визуализация

В модуле собраны скрипты для визуализации данных. В основном они полезны для анализа задачи бинарной классификации.

Для начала сгенерируем игрушечный пример.

```
In [1]: from sklearn.datasets import make_classification
import pandas as pd
import numpy as np

np.random.seed(42)

N=10000

X, y = make_classification(n_features=4, n_samples=N)

df = pd.DataFrame(X, columns=['feature_%d' % i for i in range(X.shape[1])])
df['y'] = y
df['sample_date'] = np.random.choice(pd.date_range('2025-01-01', '2025-12-31'), N)
df['category_feature'] = np.random.choice(['foo', 'bar', np.nan], N)

df.head(2)
```

```
Out[1]: feature_0 feature_1 feature_2 feature_3 y sample_date category_feature
0  1.522650 -0.934560 -0.465022  0.058874  0  2025-09-20          bar
1  1.048103 -0.746806  0.436853  0.859628  1  2025-07-15          nan
```

В игрушечной выборке содержатся:

- несколько признаков feature_*
- бинарная целевая переменная - y

- поле с датой - `sample_date`
- категориальное поле `category_feature`

Для работы скриптов потребуется пакет `holoviews`. Большинство скриптов - функции, возвращающие одну картинку, а точнее `Overlay` с различными точками, линиями, ...

```
In [2]: import holoviews as hv
        hv.extension('matplotlib')
```

```
<IPython.core.display.HTML object>
```

Проиллюстрируем работу некоторых функций

```
In [3]: from risksutils.visualization import woe_line, woe_stab, cross_tab, isotonic, distribution
```

1.1.1 woe_line

```
In [4]: woe_line(df=df, feature='feature_2', target='y', num_buck=30)
```

```
Out[4]: :Overlay
        .Weight_of_evidence.I      :Scatter [feature_2] (woe)
        .Confident_Intervals.I     :ErrorBars [feature_2] (woe,woe_u,woe_b)
        .Logistic_interpolations.I :Curve [feature_2] (logreg)
```

Данная функция разбивает числовой признак `feature` на `num_buck` бакетов. И в каждом бакете считается Weight of Evidence $= \ln\left(\frac{\text{доля 1 в бакете}}{\text{доля 0 в бакете}}\right) - \ln\left(\frac{\text{доля 1 во всей выборке}}{\text{доля 0 во всей выборке}}\right)$.

Если в бакете доля объектов класса 1 совпадает с долей 1 во всей выборке, то $WoE = 0$. Если в бакете присутствуют только объекты одного класса, то WoE будет равно бесконечности – из-за взятия логарифма. В данной функции доля объектов каждого класса ограничивается 0.001 - снизу и 0.999 - сверху.

На самом графике `woe_line` показана зависимость WoE в бакете от среднего значения признака `feature` в нем.

На примере графика выше можно сказать, что среди объектов со значением `feature_2 > 1` гораздо чаще присутствует 1.

1.1.2 isotonic

```
In [5]: isotonic(df=df, predict='feature_2', target='y')
```

```
Out[5]: :Overlay
        .Isotonic.I                :Curve [predict] (isotonic)
        .Confident_Intervals.I     :Area [predict] (ci_l,ci_h)
```

График `isotonic` похож на график `woe_line` - так же отображается зависимость частоты объектов класса 1 от значений признака, только явная разбивка на бакеты отсутствует. Построение зависимости основано на `Isotonic Regression`, которая восстанавливает монотонную зависимость.

Использовать `isotonic` совместно с доверительными интервалами удобно для проверки совпадения прогноза с фактическими данными. Так, как обычно, предполагается монотонное влияние прогноза на целевую переменную.

1.1.3 distribution

```
In [6]: distribution(df=df, feature='feature_2', date='sample_date', num_buck=4)
```

```
Out[6]: :NdOverlay [bucket]
        :Spread [sample_date] (objects_rate,obj_rate_l,obj_rate_u)
```


Данная диаграмма отражает изменение распределений признака `feature` во времени `date`. Признак дискретизируется разбивкой на бакеты. Затем выборка разбивается на группы (по умолчанию на месяца – параметр `date_freq`), и в каждой группе считается доля объектов из каждого бакета. По данному графику `distribution` удобно обнаруживать изменения в расчете признака во времени.

1.1.4 woe_stab

```
In [7]: woe_stab(df=df, feature='feature_2', target='y', date='sample_date', num_buck=3)
```

```
Out[7]: :Overlay
        .Confident_Intervals.I :NdOverlay [bucket]
        :Spread [sample_date] (woe,woe_b,woe_u)
        .Weight_of_evidence.I :NdOverlay [bucket]
        :Curve [sample_date] (woe)
```

На данном графике отображается изменение влияния признака `feature` на целевую переменную `target` во времени `date`. Для этого признак разбивается на бакеты и для каждой временной группы считаются значения WoE.

В данном игрушечном примере видно, что влияние `feature_2` на `y` стабильно по времени. Это и должно быть, так как мы сгенерировали поле `sample_date` случайно и независимо от остальной выборки.

1.1.5 cross_tab

```
In [8]: cross_tab(df, 'feature_2', 'category_feature', 'y', num_buck1=3, num_buck2=3)
```

```
Out[8]: (<pandas.io.formats.style.Styler at 0x7f0f2219ec88>,
        <pandas.io.formats.style.Styler at 0x7f0f2217be48>)
```

Данный скрипт отличается тем, что возвращает не объект `holoviews`, а набор `pandas.dataframe`, а точнее набор `Styler` - `dataframe` с визуальными настройками.

В `cross_tab` визуализируется совместное влияние пары признаков на целевую переменную (аналогично `pandas.crosstab`). Каждый признак разбивается на бакеты и считается доля объектов класса 1 в каждой комбинации пары бакетов - первая таблица. А так же считается общее количество объектов - вторая таблица. Вместе с этим выводятся агрегированные статистики - последние строка и столбец.

1.1.6 Настройка графиков

Для придания графикам различных свойств используются настройки `holoview`. Каждый тип графика состоит из набора базовых диаграмм, например, `woe_line` - это наложенные (`Overlay`) друг на друга диаграммы: * Диаграмма рассеивания (`Scatter`) со значениями WoE; * Диаграмма ошибок (`ErrorBars`) со значениями доверительный интервалов для WoE; * Линия (`Curve`) с результатом зависимости целевой переменной от признака из логистической регрессии.

Для того, чтобы вывести структуру диаграммы нужно вызвать `print` от неё.

```
In [9]: diagram = woe_line(df=df, feature='feature_2', target='y', num_buck=30)
        print(diagram)
```

```
:Overlay
        .Weight_of_evidence.I :Scatter [feature_2] (woe)
        .Confident_Intervals.I :ErrorBars [feature_2] (woe,woe_u,woe_b)
        .Logistic_interpolations.I :Curve [feature_2] (logreg)
```

В выводе присутствуют в дополнение пользовательские названия диаграмм, например, «`Confident_Intervals`» для `ErrorBars`, а так же названия осей.

Для настройки графиков можно воспользоваться магической командой `%%opts`.

```
In [10]: %%opts Curve [xrotation=45 yaxis=None] (color='red') Scatter (marker='s' s=100)
         diagram
```

```
Out[10]: :Overlay
         .Weight_of_evidence.I      :Scatter [feature_2] (woe)
         .Confident_Intervals.I     :ErrorBars [feature_2] (woe,woe_u,woe_b)
         .Logistic_interpolations.I :Curve [feature_2] (logreg)
```

Синтаксис команды следующий: `%%opts Diagram [plotting options] (style options) {normalization}`:

- plotting options (те, что в квадратных скобках) отвечают за функциональное наполнение графиков, например, опцией `xrotation=45` мы повернули подписи у оси `x` на 45 градусов, а за счет `yaxis=None` убрали ось `y`. Заметим, что эти настройки были применены к типу `Curve`, но повлияли на всю диаграмму.
- style options (в круглых скобках) изменяют визуальное оформление диаграмм. С помощью `color='red'` поменялся цвет у `Curve`, а с помощью `marker='s' s=100` мы сделали у `Scatter` маркеры в виде квадратов (`square`) и размера 100.
- normalization (в фигурных скобках) отвечает за связь разных диаграмм между собой. Далее мы рассмотрим пример.

В jupyter notebook-ах для настроек работает автодополнение, например:

- `%%opts C<TAB>` выдаст подсказки `Collator`, `Contours`, `Curve`;
- `%%opts Curve [xaxis=None sh<TAB>` выдает `show_frame=`, `show_grid=`, ...

Для более подробного описания настроек можно вызвать справку, например, `hv.help(hv.Overlay)`, а так же посмотреть примеры из документации.

Помимо настройки `%%opts` (с двумя процентами) так же есть и настройка `%opts` с одним они различаются следующим:

- `%%opts` - локальные настройки, применяются ко всем диаграммам, созданным в данной ячейке;
- `%opts` - глобальные настройки, применяется ко всем диаграммам в ноутбуке.

```
In [11]: simple_curve = hv.Curve([1, 3, 2, 4])
         diagram + simple_curve
```

```
Out[11]: :Layout
         .Woe_line.Feature_2 :Overlay
         .Weight_of_evidence.I      :Scatter [feature_2] (woe)
         .Confident_Intervals.I     :ErrorBars [feature_2] (woe,woe_u,woe_b)
         .Logistic_interpolations.I :Curve [feature_2] (logreg)
         .Curve.I                :Curve [x] (y)
```

В примере выше я вывел вместе две диаграммы: `diagram` и созданную `Curve`, при этом настройки у `diagram` сохранились, так как мы их применили ячейкой выше, а у новой диаграммы они остались прежними. Если мы теперь захотим поменять настройки только у одной кривой из двух, можно воспользоваться её именем.

```
In [12]: print(diagram + simple_curve)
```

```
:Layout
.Woe_line.Feature_2 :Overlay
.Weight_of_evidence.I :Scatter [feature_2] (woe)
.Confident_Intervals.I :ErrorBars [feature_2] (woe,woe_u,woe_b)
.Logistic_interpolations.I :Curve [feature_2] (logreg)
.Curve.I :Curve [x] (y)
```

```
In [13]: %%opts Curve.Logistic_interpolations (color='green') Curve (color='black')
         diagram + simple_curve
```

```
Out[13]: :Layout
         .Woe_line.Feature_2 :Overlay
         .Weight_of_evidence.I :Scatter [feature_2] (woe)
         .Confident_Intervals.I :ErrorBars [feature_2] (woe,woe_u,woe_b)
         .Logistic_interpolations.I :Curve [feature_2] (logreg)
         .Curve.I :Curve [x] (y)
```

Помимо настроек `%%opts` так же доступны настройки `%%output`, позволяющие менять размер и вывод

```
In [14]: %%output size=50
         diagram
```

```
Out[14]: :Overlay
         .Weight_of_evidence.I :Scatter [feature_2] (woe)
         .Confident_Intervals.I :ErrorBars [feature_2] (woe,woe_u,woe_b)
         .Logistic_interpolations.I :Curve [feature_2] (logreg)
```

Из полезных настроек: с помощью `%%output filename=` можно сохранить картинку в файл.

1.1.7 Backend

В самом начале мы подключали backend matplotlib с помощью `hv.extension('matplotlib')`, но нам так же доступен и другой backend - `bokeh` <<https://bokeh.pydata.org/en/latest>>‘__.

```
In [15]: hv.extension('bokeh')
```

Data type cannot be displayed: application/vnd.bokehjs_load.v0+json, application/javascript

<IPython.core.display.HTML object>

```
In [16]: isotonic(df=df, predict='feature_2', target='y')
```

```
Out[16]: :Overlay
         .Isotonic.I :Curve [predict] (isotonic)
         .Confident_Intervals.I :Area [predict] (ci_l,ci_h)
```

В нем появляется возможность делать интерактивные диаграммы.

1.1.8 Совмещение диаграмм

Для визуализации данных бывает полезно выводить не по одной диаграмме а сразу несколько, и holoviews позволяет это сделать очень удобно.

Над диаграммами переопределены арифметические операции:

- “+” - `Layout` рисует диаграммы рядом друг с другом;
- “*” - `Overlay` накладывает диаграммы друг на друга.

```
In [17]: wl_2 = woe_line(df=df, feature='feature_2', target='y', num_buck=30)
         ws_2 = woe_stab(df=df, feature='feature_2', target='y', date='sample_date', num_buck=3)

         wl_2 + ws_2
```

```
Out[17]: :Layout
         .Woe_line.Feature_2 :Overlay
         .Weight_of_evidence.I :Scatter [feature_2] (woe)
         .Confident_Intervals.I :ErrorBars [feature_2] (woe,woe_u,woe_b)
         .Logistic_interpolations.I :Curve [feature_2] (logreg)
         .Woe_Stab.Feature_2 :Overlay
         .Confident_Intervals.I :NdOverlay [bucket]
```

```
:Spread [sample_date] (woe,woe_b,woe_u)
.Weight_of_evidence.I :NdOverlay [bucket]
:Curve [sample_date] (woe)
```

При рисовании диаграмм рядом друг с другом происходит совмещение осей (если они называются одинаково). Чтобы этого не происходило можно воспользоваться настройками `normalization` (те, что в фигурных скобках) - если добавить `%%opts Spread {+axiswise}`, то сцепление первого типа диаграммы `Spread` из второго графика `ws_2` пропадет.

Также есть удобная возможность рисовать сразу несколько диаграмм напрямую через конструктор `hv.Layout`.

```
In [18]: hv.extension('matplotlib')
```

```
<IPython.core.display.HTML object>
```

```
In [19]: %%opts Layout [hspace=1 vspace=0.5]
```

```
features = ['feature_' + str(i) for i in range(4)] + ['category_feature']
hv.Layout([woe_stab(df, f, 'y', 'sample_date', num_buck=3) for f in features]).cols(2)
```

```
Out[19]: :Layout
```

```
.Woe_Stab.Feature_0 :Overlay
.Confident_Intervals.I :NdOverlay [bucket]
:Spread [sample_date] (woe,woe_b,woe_u)
.Weight_of_evidence.I :NdOverlay [bucket]
:Curve [sample_date] (woe)
.Woe_Stab.Feature_1 :Overlay
.Confident_Intervals.I :NdOverlay [bucket]
:Spread [sample_date] (woe,woe_b,woe_u)
.Weight_of_evidence.I :NdOverlay [bucket]
:Curve [sample_date] (woe)
.Woe_Stab.Feature_2 :Overlay
.Confident_Intervals.I :NdOverlay [bucket]
:Spread [sample_date] (woe,woe_b,woe_u)
.Weight_of_evidence.I :NdOverlay [bucket]
:Curve [sample_date] (woe)
.Woe_Stab.Feature_3 :Overlay
.Confident_Intervals.I :NdOverlay [bucket]
:Spread [sample_date] (woe,woe_b,woe_u)
.Weight_of_evidence.I :NdOverlay [bucket]
:Curve [sample_date] (woe)
.Woe_Stab.Category_feature :Overlay
.Confident_Intervals.I :NdOverlay [bucket]
:Spread [sample_date] (woe,woe_b,woe_u)
.Weight_of_evidence.I :NdOverlay [bucket]
:Curve [sample_date] (woe)
```

Внутри `hv.Layout` мы создали лист с пятью диаграммами, а вызвав метод `cols(2)` нарисовали все в 2 колонки.

Настройка `hspace=1` позволяет сделать отступы между графиками, расположенными горизонтально друг от друга для того, чтобы уместились легенды, а `vspace=0.5` - между вертикально расположенными графиками.

1.1.9 Интерактивность

Один из мощных инструментов в `holoviews` - это создание интерактивных графиков, позволяющих с помощью виджетов перебирать различные диаграммы. Доступно два базовых типа:

- **HoloMap** - из словаря с ключом - название диаграммы, а значением самими диаграммами создается интерактивный график (пример ниже).
- **DynamicMap** - динамичная диаграммы, вычисляющая по положениям виджетов встроенную диаграмму. Для DynamicMap нужно задать функцию, которая это сделает и вычислению будут происходить только при запущенной сессии Python (за то не тратится место на хранение сразу всех диаграмм как у HoloMap).

In [20]: `hv.extension('bokeh')`

Data type cannot be displayed: application/vnd.bokehjs_load.v0+json, application/javascript

<IPython.core.display.HTML object>

In [21]: `hv.HoloMap({i: woe_line(df, 'feature_2', 'y', num_buck=i) for i in range(10, 100, 50)}, kdims=['buckets'])`

Out[21]: `HoloMap [buckets]`
`:Overlay`
`.Weight_of_evidence.I :Scatter [feature_2] (woe)`
`.Confident_Intervals.I :ErrorBars [feature_2] (woe,woe_u,woe_b)`
`.Logistic_interpolations.I :Curve [feature_2] (logreg)`

В данном примере мы внутри `hv.HoloMap` создали словарь с ключем `i` и значением - диаграммой `woe_line`, с разбивкой признака `feature_2` как раз на `i` бакетов. Теперь с помощью виджета можно посмотреть как меняется график при изменении количества бакетов. Видно, что график становится подробнее, вместе с тем растут доверительные интервалы у оцененных значений `woe`. (в документации `readthedocs` отображать виджеты не получается, для понимания того, как они работают можно посмотреть примеры со страницы [HoloMap](#))

Можно так же создавать сразу несколько виджетов, если ключ будет более сложным объектом `tuple` (в случае ниже пара - название признака и количество бакетов).

In [22]: `hv.HoloMap({(f, i): woe_line(df, f, 'y', num_buck=i)
for f in ['feature_1', 'feature_2']
for i in [10, 100]}, kdims=['feature', 'buckets'])`

```
/home/docs/checkouts/readthedocs.org/user_builds/risksutils/conda/latest/lib/python3.5/site-packages/bokeh/models/sources.py
"Current lengths: %s" % ", ".join(sorted(str((k, len(v))) for k, v in data.items())), BokehUserWarning))
/home/docs/checkouts/readthedocs.org/user_builds/risksutils/conda/latest/lib/python3.5/site-packages/bokeh/models/sources.py
"Current lengths: %s" % ", ".join(sorted(str((k, len(v))) for k, v in data.items())), BokehUserWarning))
/home/docs/checkouts/readthedocs.org/user_builds/risksutils/conda/latest/lib/python3.5/site-packages/bokeh/models/sources.py
"Current lengths: %s" % ", ".join(sorted(str((k, len(v))) for k, v in data.items())), BokehUserWarning))
/home/docs/checkouts/readthedocs.org/user_builds/risksutils/conda/latest/lib/python3.5/site-packages/bokeh/models/sources.py
"Current lengths: %s" % ", ".join(sorted(str((k, len(v))) for k, v in data.items())), BokehUserWarning))
/home/docs/checkouts/readthedocs.org/user_builds/risksutils/conda/latest/lib/python3.5/site-packages/bokeh/models/sources.py
"Current lengths: %s" % ", ".join(sorted(str((k, len(v))) for k, v in data.items())), BokehUserWarning))
/home/docs/checkouts/readthedocs.org/user_builds/risksutils/conda/latest/lib/python3.5/site-packages/bokeh/models/sources.py
"Current lengths: %s" % ", ".join(sorted(str((k, len(v))) for k, v in data.items())), BokehUserWarning))
/home/docs/checkouts/readthedocs.org/user_builds/risksutils/conda/latest/lib/python3.5/site-packages/bokeh/models/sources.py
"Current lengths: %s" % ", ".join(sorted(str((k, len(v))) for k, v in data.items())), BokehUserWarning))
/home/docs/checkouts/readthedocs.org/user_builds/risksutils/conda/latest/lib/python3.5/site-packages/bokeh/models/sources.py
"Current lengths: %s" % ", ".join(sorted(str((k, len(v))) for k, v in data.items())), BokehUserWarning))
/home/docs/checkouts/readthedocs.org/user_builds/risksutils/conda/latest/lib/python3.5/site-packages/bokeh/models/sources.py
"Current lengths: %s" % ", ".join(sorted(str((k, len(v))) for k, v in data.items())), BokehUserWarning))
/home/docs/checkouts/readthedocs.org/user_builds/risksutils/conda/latest/lib/python3.5/site-packages/bokeh/models/sources.py
"Current lengths: %s" % ", ".join(sorted(str((k, len(v))) for k, v in data.items())), BokehUserWarning))
```

Out[22]: `HoloMap [feature,buckets]`
`:Overlay`
`.Weight_of_evidence.I :Scatter [feature_2] (woe)`
`.Confident_Intervals.I :ErrorBars [feature_2] (woe,woe_u,woe_b)`
`.Logistic_interpolations.I :Curve [feature_2] (logreg)`

1.1.10 Другие возможности

С другими интересными возможностями работы диаграмм стоит обращаться к документации [holoviews](#).

1.2 Reference

```
class risksutils.visualization.InteractiveIsotonic(data, pdims, tdims, ddims=None, gdims=None,
                                                  calibrations_data=None)
```

Интерактивная визуализация точности прогноза вероятности

```
risksutils.visualization.cross_tab(df, feature1, feature2, target, num_buck1=10, num_buck2=10,
                                   min_sample=100, compute_iv=False)
```

Кросстабуляция пары признаков и бинарной целевой переменной

Аргументы:

df: pandas.DataFrame таблица с данными

feature1: str название признака 1

feature2: str название признака 2

target: str название целевой переменной

num_buck1: int количество бакетов для признака 1

num_buck2: int количество бакетов для признака 2

min_sample: int минимальное количество наблюдений для отображение доли целевой переменной в ячейке

compute_iv: bool нужно ли рассчитывать information value для признаков

Результат: (rates, counts): (pandas.Styler, pandas.Styler)

```
risksutils.visualization.distribution(df, feature, date, num_buck=10, date_freq='MS')
```

График изменения распределения признака по времени

Аргументы:

df: pandas.DataFrame таблица с данными

feature: str название признака

date: str название поля со временем

num_buck: int количество бакетов

date_freq: str Тип агрегации времени (по умолчанию „MS“ - начало месяца)

Результат: spreads: holoviews.NdOverlay

```
risksutils.visualization.isotonic(df, predict, target, calibrations_data=None)
```

Визуализация точности прогноза вероятности

Аргументы:

df: pandas.DataFrame таблица с данными

predict: str прогнозная вероятность

target: str бинарная (0, 1) целевая переменная

calibrations_data: pandas.DataFrame таблица с калибровками

Результат: area * curve * [curve] : holoviews.Overlay

`risksutils.visualization.woe_line(df, feature, target, num_buck=10)`

График зависимости WoE от признака

Аргументы:

`df`: `pandas.DataFrame` таблица с данными

`feature`: `str` название признака

`target`: `str` название целевой переменной

`num_buck`: `int` количество бакетов

Результат: `scatter * errors * line: holoviews.Overlay`

`risksutils.visualization.woe_stab(df, feature, target, date, num_buck=10, date_freq='MS')`

График стабильности WoE признака по времени

Аргументы:

`df`: `pandas.DataFrame` таблица с данными

`feature`: `str` название признака

`target`: `str` название целевой переменной

`date`: `str` название поля со временем

`num_buck`: `int` количество бакетов

`date_freq`: `str` Тип агрегации времени (по умолчанию „MS“ - начало месяца)

Результат: `curves * spreads: holoviews.Overlay`

`risksutils.metrics.information_value(df, feature, target, num_buck=10)`

information value признака с целевой переменной `target`

Аргументы:

`df`: `pandas.DataFrame` таблица с данными

`feature`: `str` признак

`target`: `str` целевая переменная

`num_buck`: `numeric` количество бакетов

Результат: `information value: float`

Пример использования

```
>>> import pandas as pd
>>> df = pd.DataFrame({'foo': [1, 1, 1, np.nan, np.nan],
...                    'bar': [0, 0, 1, 0, 1]})
>>> information_value(df, 'foo', 'bar')
0.11552453009332433
```

`risksutils.metrics.information_value_binormal(auc)`

information value из биномиального приближения через AUC

Аргументы:

AUC: `float` Area Under Roc Curve

Результат: `information value: float`

Пример использования

```
>>> information_value_binormal(0.5)
0.0
```

`risksutils.metrics.stability_index(df, feature, date, num_buck=10, date_freq='MS')`

Stability index для всех последовательных пар дат

Аргументы:

`df`: `pandas.DataFrame` таблица с данными

`feature`: `str` признак

`date`: `str` название поля со временем

`num_buck`: `numeric` количество бакетов

`date_freq`: `str` Тип агрегации времени (по умолчанию „MS“ - начало месяца)

Результат: `pd.Series`

Пример использования

```
>>> df = pd.DataFrame({
...     'dt': pd.Series(['2000-01-01', '2000-01-01', '2000-01-01',
...                     '2000-02-01', '2000-02-02', '2000-02-01',
...                     '2000-04-02', '2000-04-03'],
...                     dtype='datetime64[ns]'),
...     'foo': ['a', 'a', np.nan, 'a', 'b', 'b', 'a', 'b']
... })
>>> stability_index(df, 'foo', 'dt')
dt
2000-02-01    6.489979
2000-04-01    0.115525
Name: si, dtype: float64
```

Содержание модулей Python

r

`risksutils.metrics`, 11

`risksutils.visualization`, 10

C

`cross_tab()` (в модуле `risksutils.visualization`), [10](#)

D

`distribution()` (в модуле `risksutils.visualization`), [10](#)

I

`information_value()` (в модуле `risksutils.metrics`),
[11](#)

`information_value_binormal()` (в модуле
`risksutils.metrics`), [11](#)

`InteractiveIsotonic` (класс в
`risksutils.visualization`), [10](#)

`isotonic()` (в модуле `risksutils.visualization`), [10](#)

R

`risksutils.metrics` (модуль), [11](#)

`risksutils.visualization` (модуль), [10](#)

S

`stability_index()` (в модуле `risksutils.metrics`), [11](#)

W

`woe_line()` (в модуле `risksutils.visualization`), [10](#)

`woe_stab()` (в модуле `risksutils.visualization`), [11](#)