Open Source технологии для моделирования корпоративных кредитных рисков

Суржко Д.А., к.э.н., PRM

Специфика моделирования в корпоративном сегменте

Существенно более сложная структура моделей, которая может включать иерархию специализированных модулей:

- Количественные факторы;
- Качественные факторы;
- Модули государственной и групповой поддержки;
- Предупреждающие сигналы.

Разные типы целевых переменных:

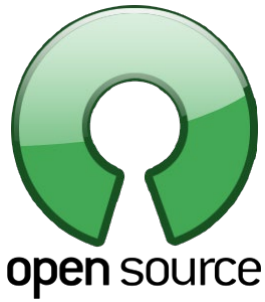
- Дефолты клиентов;
- Внешние рейтинги (PD);
- Классы кредитоспособности (экспертное ранжирование).

Ограниченные по размеру выборки требуют специализированных подходов по контролю «переобученности» у модели:

- Кросс-валидация;
- Регуляризация.

Наличие специализированных моделей калибровки для низкодефолтных сегментов (LDP), позволяющих «управлять» степенью консервативности полученных оценок.

Причины для использования Open Source



Отсутствие лицензионных затрат, регулярного процесса по продлению лицензий.

«Бесшовный» обмен результатами с дочерними компаниями, внешними аудиторами и т.п.

Поддержка наиболее современных методов анализа данных, спец. подходов для LDP, управление параллельными вычислениями.

Community которое готово оказать поддержку.

Поддержка интерактивных отчетов.

Возможность выбора языка разработки, в том числе, с поддержкой ООП, современных средств разработки и отладки.

«Золотой стандарт», в первую очередь, в ритейловых рисках.

Интеграция решений по моделированию в кредитный процесс.

Ориентация на потоковое построение «стандартизованных» моделей.

Примеры применения Open Source решений:

- ✓ **Интерактивные отчеты.**
- ✓ **Калибровка PD-моделей (R).**
- ✓ **Интерактивные отчеты (Python)**
- ✓ **WoE трансформация (Python).**
- ✓ **Параллельные Monte-Carlo вычисления (Python)**

Интерактивные отчеты

Мультифакторный анализ

Длинный список макроэкономических факторов сокращен до 3 факторов.

1. Изменение реального ВВП год к году **(dGDP)**;
2. Ставка по облигациям федерального займа **(OFZ)**;
3. Изменение курса **(USDRUB).**

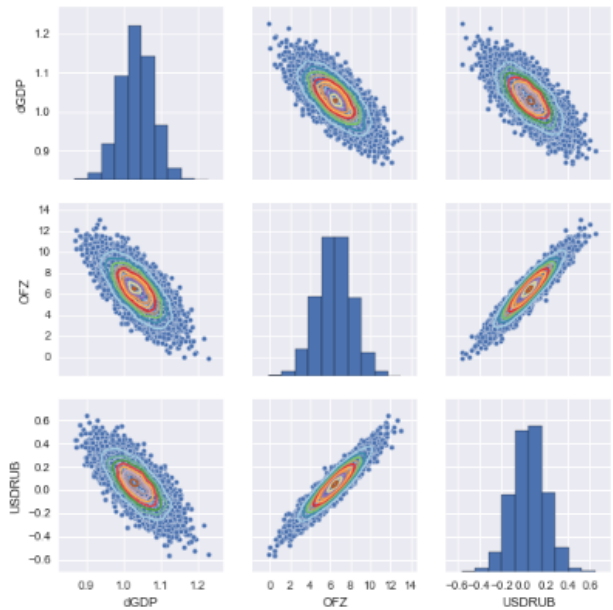
На следующем шаге рассчитывается матрица попарных корреляций факторов короткого листа и визуализируются их маржинальные распределения.

In [9]:

```
g = sns.pairplot(rez_sim[0:10000])
g.map_offdiag(sns.kdeplot, cmap="Paired", n_levels=10);

rez_sim.corr()
```

Out[9]:

	dGDP	OFZ	USDRUB
dGDP	1.000000	-0.724015	-0.714036
OFZ	-0.724015	1.000000	0.942236
USDRUB	-0.714036	0.942236	1.000000



Структура отчета: ключевые этапы и результаты

Текст отчета: описание экономической сути проведенного анализа

Программный код реализующий:

- обработку данных
- стат. анализ
- построение графиков

Результаты проведенного анализа

Графические иллюстрации

Полный пересчет (воспроизведение) отчета по одной «кнопке»

Калибровка PD-моделей на ЦТ (1/2)

Дано:

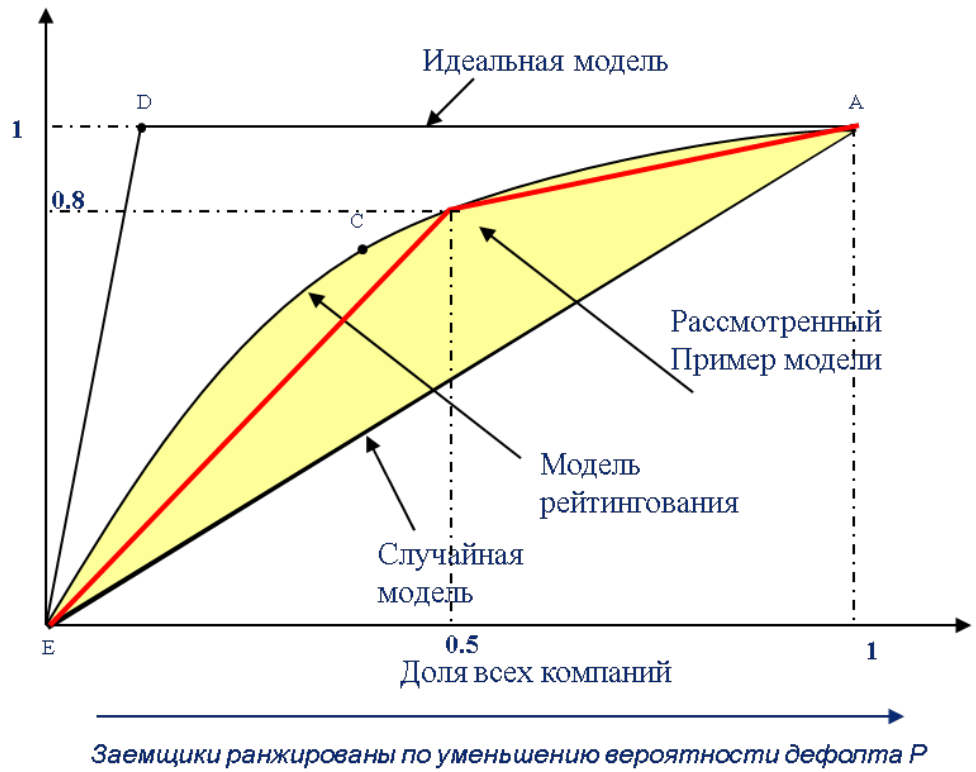
- Средняя частота дефолтов в портфеле (прогнозная либо среднециклическая);
- Оценка предсказательной способности модели (Gini);
- Распределение портфеля по рейтинговым классам (скорам)

Задача:

- Присвоить оценки PD каждому рейтинговому классу (скору).

Суть подходов калибровки на среднюю частоту дефолтов заключается в следующем:

вероятности дефолтов распределяются по рейтинговой шкале пропорционально предсказательной силе модели, т.е., чем выше дискриминирующая сила модели, тем больше дефолтов приходится на «плохие» рейтинги => большая часть «средней частоты дефолтов» будет распределена на «плохие» рейтинги => «плохие» рейтинги получают более высокие PD.



Функционал LDPD пакета

- ✓ Поддерживаются наиболее широко распространенные модели калибровки на ЦТ:
  - Quasi Moment Matching (D. Tasche);
  - M. van der Burgt algorithm.

- ✓ Возможность калибровки по рейтинговым классам (дискретные PD) либо по скорам (непрерывные PD, output алгоритма – параметры логистической функции).

- ✓ Поддержка специализированного подхода K.Pluto, D. Tasche для калибровки LDP портфелей (одно-периодная и много-периодная версии модели).

- ✓ Функционал по оценке implied Gini (D. Tasche), графическому отображению полученной калибровки.

- ✓ Наличие пакета с симметричным функционалом в Python.

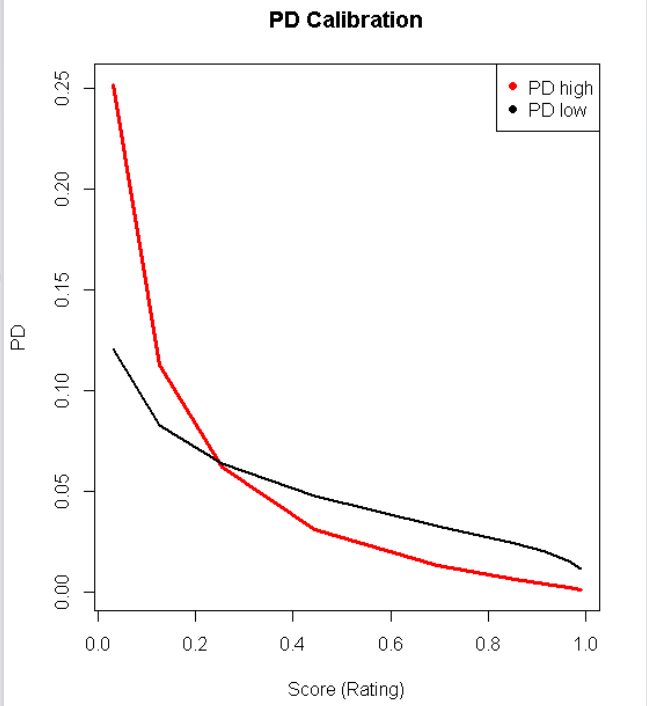
Калибровка в R-пакете LDPD – рейтинговые классы

~/irb2016 - RStudio

File Edit Code View Plots Session Build Debug Tools Help

example.R*

```
require(LDPD)

# Портфель по рейтинговым классам в шт. заемщиков
porfolio <- c(100, 200, 200, 400, 400, 100, 100, 50, 30)

# Оценка предсказательной способности (плохая 30% и хорошая модели 60%)
gini.bad <- 0.3
gini.good <- 0.6

# Средняя частота дефолтов в портфеле 5%
ct <- 0.05

# Наблюдаемая частота дефолтов в разрезе рейтинговы классов
# (используется для оценки Gini, в процессе калировки не участвует)

pd.historical <- c(0.005, 0.004, 0.003, 0.003, 0.002, 0.001, 0.001,
                   0.001, 0.001)

# Калибровка
qmm.bad <- QMMRecalibrate(ct, pd.historical, porfolio, NULL, gini.bad)
qmm.good <- QMMRecalibrate(ct, pd.historical, porfolio, NULL, gini.good)
```



Калибровка в R-пакете LDPD – скоринговые баллы

~/irb2016 - RStudio

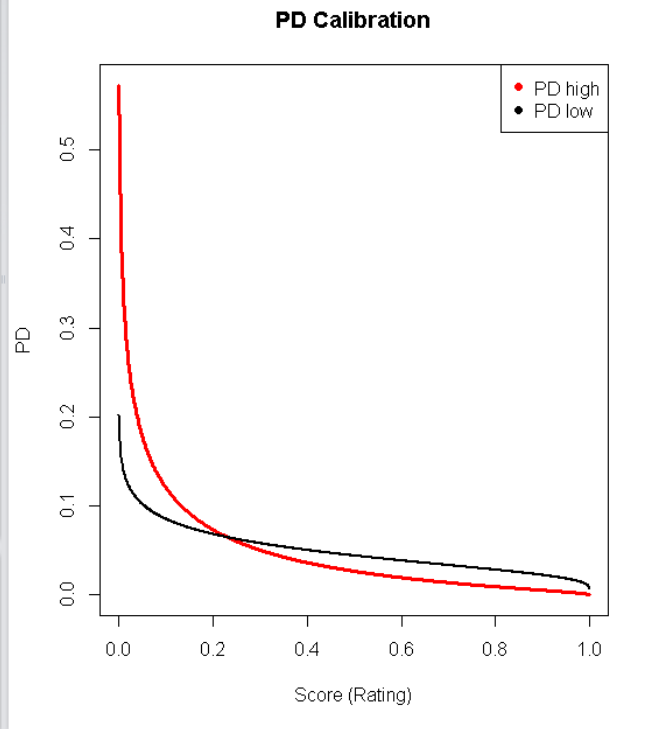
File Edit Code View Plots Session Build Debug Tools Help

example.R | exmaple_score.R

```
1  require(LDPD)
2  # Портфель по рейтинговым классам в шт. заемщиков
3  porfolio <- seq(1000)
4
5  # Оценка предсказательной способности (плохая 30% и хорошая модели 60%)
6  gini.bad <- 0.3
7  gini.good <- 0.6
8
9  # Средняя частота дефолтов в портфеле 5%
10 ct <- 0.05
11
12 # Наблюдаемая частота дефолтов в разрезе рейтинговы классов
13 # (используется для оценки Gini, в процессе калировки не участвует)
14 pd.historical <- rep(0.01, 1000)
15
16 # Калибровка
17 qmm.bad <- QMMRecalibrate(ct, pd.historical, porfolio, NULL, gini.bad,
18                           rating.type='SCORE')
19 qmm.good <- QMMRecalibrate(ct, pd.historical, porfolio, NULL, gini.good,
20                            rating.type='SCORE')
21 qmm$alpha
22 qmm$beta
```

Console ~/irb2016/

```
> qmm$alpha
[1] -3.083913
> qmm$beta
[1] -0.623432
>
```



Что такое WoE трансформация?

Каждому интервалу непрерывной переменной (значению дискретной переменной) присваивается величина, полученная как соотношения «хороших» и «плохих» значений целевой переменной в данном интервале:

$$Weight\ of\ Evidence_i = \ln\left(\frac{DistributionGood_i}{DistributionBad_i}\right)$$

Преимущества данного вида трансформации переменных:

- ✓ Позволяет свести на одну «шкалу» дискретные и непрерывные переменные;
- ✓ «Автоматически» обрабатывает «дискретные элементы» непрерывных переменных (пропущенные значения, спец. значения).
- ✓ Учитывает нелинейную составляющую (при ее наличии) зависимости целевой от объясняющей переменных.
- ✓ Позволяет управлять «степенью переобученности» итоговой трансформации.

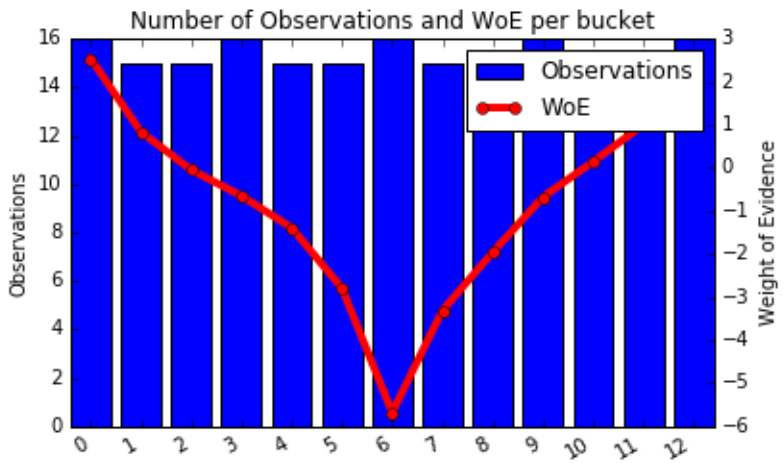
WoE – пример (1/2)

Пример линейной модели для завимости $Y = X^2$, $X \sim U[-1,1]$

```
from pywoe import WoE
import numpy as np
import pandas as pd
import statsmodels.formula.api as sm
%matplotlib inline
```

```
N = 200
x = (np.random.rand(N) - 0.5) * 2
PD = x * x
```

```
woe = WoE(v_type='c',  t_type='c')
woe.fit(pd.Series(x), pd.Series(PD))
_ = woe.plot()
```



WoE – пример (2/2)

До трансформации

```
before_transform = sm.OLS( PD.reshape(-1,1), x.reshape(-1,1) ).fit()
before_transform.summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	-0.005
Method:	Least Squares	**F-statistic:**	1.789e-05
Date:	Tue, 18 Oct 2016	**Prob (F-statistic):**	0.997
Time:	17:03:38	**Log-Likelihood:**	-108.79
No. Observations:	200	**AIC:**	219.6
Df Residuals:	199	**BIC:**	222.9
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[95.0% Conf. Int.]
x1	0.0002	0.054	0.004	0.997	-0.106 0.107

Omnibus:	22.707	**Durbin-Watson:**	0.973
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	26.147
Skew:	0.855	**Prob(JB):**	2.10e-06
Kurtosis:	2.536	**Cond. No.**	1.00

Значимость переменной

После трансформации

```
after_transform = sm.OLS( PD.reshape(-1,1), woe.transform(pd.Series(x))['woe'] ).fit()
after_transform.summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.124
Model:	OLS	**Adj. R-squared:**	0.119
Method:	Least Squares	**F-statistic:**	28.08
Date:	Tue, 18 Oct 2016	**Prob (F-statistic):**	3.07e-07
Time:	17:40:31	**Log-Likelihood:**	-95.591
No. Observations:	200	**AIC:**	193.2
Df Residuals:	199	**BIC:**	196.5
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[95.0% Conf. Int.]
woe	0.0622	0.012	5.299	0.000	0.039 0.085

Omnibus:	31.115	**Durbin-Watson:**	0.396
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	41.442
Skew:	1.109	**Prob(JB):**	1.00e-09
Kurtosis:	3.227	**Cond. No.**	1.00

Возможности Open Source решения (PyWoE)

- ✓ Позволяет трансформировать все типы объясняющих переменных:
  - Непрерывные;
  - Дискретные;
  - Смешанные (непрерывные + спец. значения и/или missing).

- ✓ Поддерживает управляемую вручную «склейку» как дискретных, так и непрерывных переменных.

- ✓ Оптимизация разбиения на бакеты производится методом, препятствующем «переобученности» итоговой трансформации - кросс-валидация (по данным автора, функционал отсутствует в коммерческих решениях).

- ✓ В качестве целевой переменной могут быть использованы как дискретные значения (дефолт/не дефолт), так и непрерывные, например, для целей построения shadow bond PD моделей либо LGD моделей (по данным автора, функционал отсутствует в коммерческих решениях).

- ✓ Есть встроенная функция графического представления полученных результатов.

WoE – пример (1/4)

WoE трансформация в Python

In [12]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pywoe import WoE
%matplotlib inline
```

Формируем случайный набор данных, включающий:

- Непрерывные данные.
- Дискретные специальные значения (1).
- Пропущенные значения (NaN).

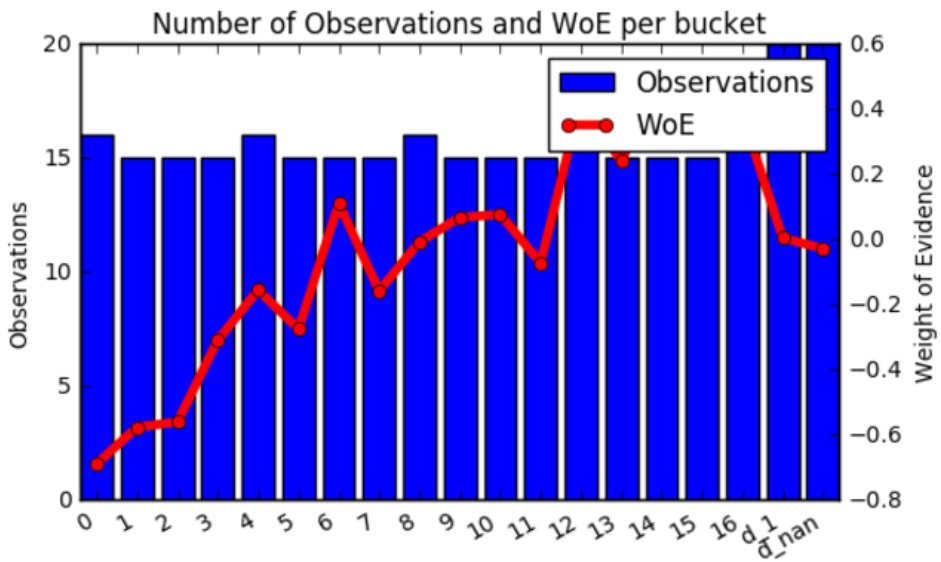
In [51]:
```
# Set target type: 'b' for default/non-default, 'c' for continuous pd values
t_type = 'c'
# Set sample size
N = 300
# Random variables
x1 = np.random.rand(N)
x2 = np.random.rand(N)
if t_type == 'b':
    y = np.where(np.random.rand(N) + x1 + x2 > 2, 1, 0)
else:
    y = np.random.rand(N) + x1 + x2
    y = (y - np.min(y)) / (np.max(y) - np.min(y)) / 2
# Inserting special values
x1[0:20] = float('nan')
x1[60:80] = float(1)
```

WoE – пример (2/4)

Трансформация

Создаем объект класса WoE, указваем спец. значения, тип трансформируемой и целевой переменных. Обучаем трансформации на переменной x1.

In [48]:
```
woe = WoE(spec_values={1: '1'}, v_type='c',  t_type=t_type)
woe.fit(pd.Series(x1), pd.Series(y))
f1 = woe.plot()
```

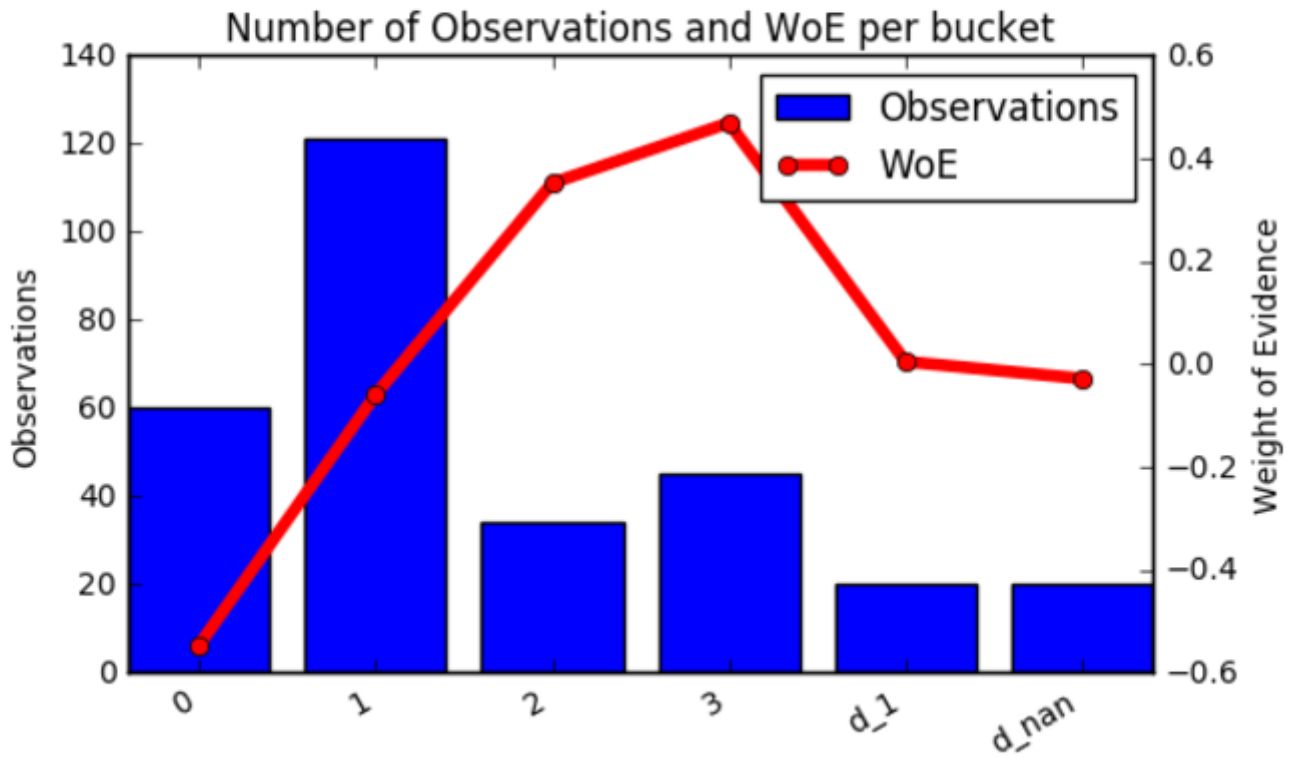


WoE – пример (3/4)

Оптимизируем количество бакетов

Ищем оптимальное количество бакетов (степень оптимизации определяется методом кросс-валидации)

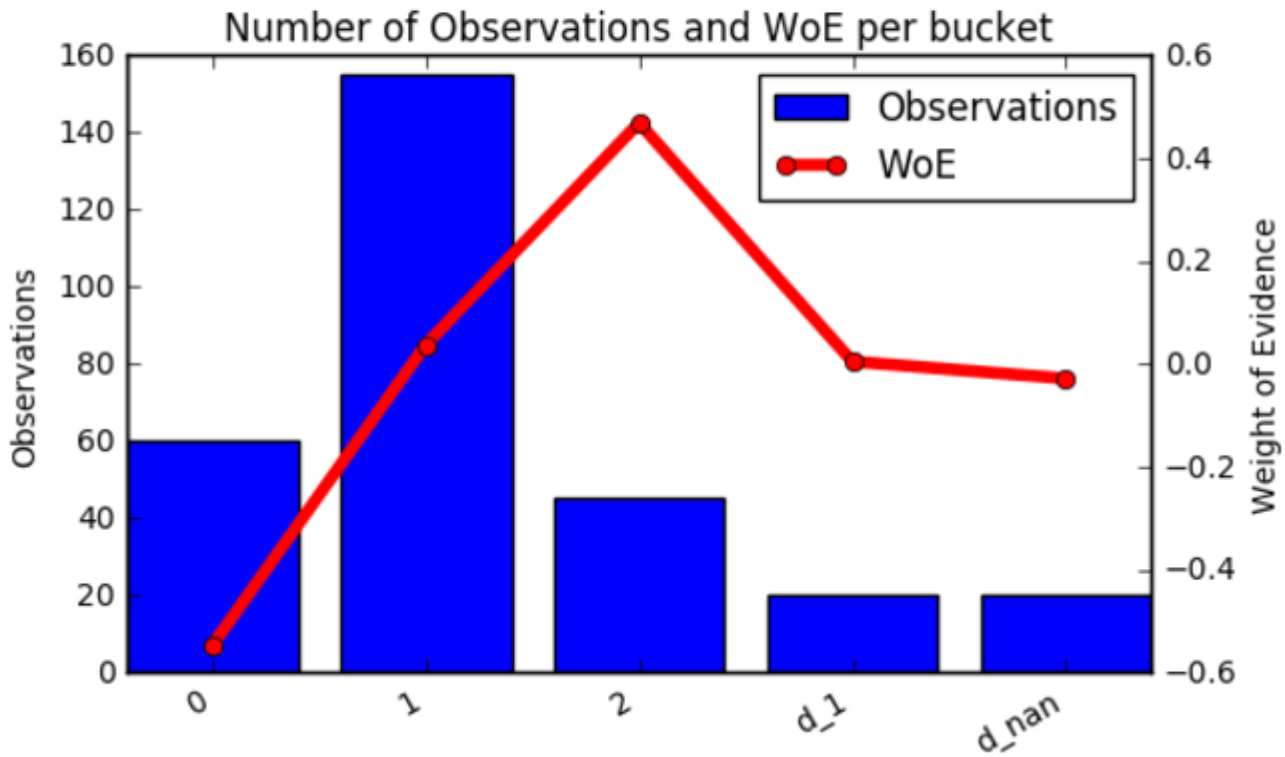
In [49]:
```
woe2 = woe.optimize(max_depth=5)
f2 = woe2.plot()
```



WoE – пример (4/4)

"Склеиваем" второй и третий бакеты

In [50]:
```
woe3 = woe2.merge('2')
f3 = woe3.plot()
```



Портфельные модели

Область применения портфельных моделей:

- Проведение прямого и обратного стресс-тестов;
- Оценка Экономического капитала Банка.

Большинство портфельных моделей, учитывающих:

- Концентрационные риски;
- Миграции рейтингов

требуют применения методов Monte-Carlo.

Корпоративные портфели по количеству наблюдений существенно меньше ритейловых, однако, для следующих целей при применении метода МС необходим существенный объем вычислений:

- Расчет вклада каждого заемщика в величину ЭК/Стресс-потери
- Меры риска, оценка которых производится в «хвосте» распределения (Expected Short Fall).

В задачах МС параллельные вычисления дают близкий к линейному выигрыш в скорости вычислений.

Большинство коммерческих решений не предоставляют достаточно гибких и простых инструментов для параллельных вычислений в отличии от Python и R.

Параллельные вычисления в Python – пример MC

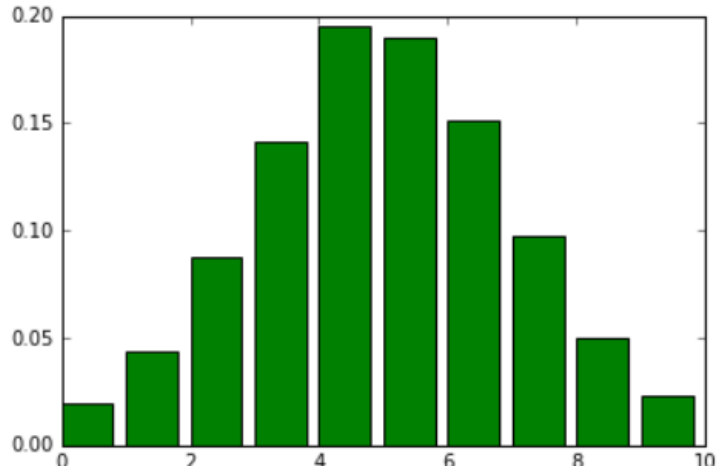
```
def mc_func(N):
    sims = np.random.normal(0, 1, N)
    out = np.histogram(sims, bins=breaks)
    return out[0] / N
```

Функция, производящая симуляции методом Monte-Carlo

```
import numpy as np
import matplotlib.pyplot as plt
from mc_func import mc_func
from multiprocessing import Pool
%matplotlib inline

NP = 4 # Кол-во  потоков
NS = 1000 # Кол-во симуляций в каждом потоке

with Pool(NP) as p:
     parallel_out = p.map(mc_func, [NS] * NP, chunksize=1)


hist = np.vstack(parallel_out).mean(axis=0)


plt.bar(np.arange(hist.size), hist, color='g', label='Observations')
plt.show()
```

Распараллеливание вычислений

Склейка результатов



Склейка результатов

Заключение.

В условиях усиления регуляторного надзора за результатами моделирования, в том числе, в контексте внедрения Базельских стандартов и Стандарта IFRS 9, Open Source решения могут стать единым стандартом в части:

Библиотек статистических тестов работы моделей (PD, LGD, EAD), которым «доверяют» аудиторы, банки и регуляторы, в том числе, на:

- ✓ Точность ранжирования.
- ✓ Точность калибровки.
- ✓ Стабильность.

Стандартных интерактивных отчетов по оценке работы моделей и их составляющих.

Базовых элементов средств разработки и калибровки моделей.

Ссылки:

- ✓ **Пакет PD-калибровки:**
  - ✓ **R: https://cran.r-project.org/web/packages/LDPD/index.html**
  - ✓ **Python: https://github.com/Densur/LDPD**

- ✓ **Пакет расчета WoE-трансформации:**
  - ✓ **Python: https://github.com/Densur/PyWoE**

- ✓ **Интерпретатор R + пакеты для анализа и обработки данных + интерактивные отчеты (Jupyter): https://www.continuum.io/downloads**