Лекция 14:
Отбор признаков

Задача отбора признаков

- Дана выборка:
  - $Z = \{x_1, \dots, x_l\}$, включает l объектов из X, где каждый x может быть представлена вектором из p признаков $[f_1(x), f_2(x), \dots, f_p(x)]$
  - В случае обучения с учителем дан еще вектор откликов $\{y_1, \dots, y_l\}$
  - Необходимо исключить «лишние» признаки, при этом максимально сохранив (но обычно с потерями) информацию о наблюдениях:

$$[\cancel{f_1(x)}, f_2(x), f_3(x), \dots, \cancel{f_{p-1}(x)}, f_p(x)]$$

- Зачем?
  - Борьба с проклятием размерности, удаление шумовых и зависимых признаков,
  - Упрощение прогнозной модели – борьба с переобучением, повышение стабильности
  - Повышение вычислительной эффективности и уменьшение используемой памяти

Фильтрация признаков в задачах с числовым откликом

- Фильтрация – признаки отбираются независимо друг от друга:
  - Оценивается сила или значимость связи признак-отклик, все признаки упорядочиваются по этой оценке и отбираются по порогу или топ n штук
- Если числовой отклик то:
  - для числовых признаков для проверки линейных зависимостей можно использовать корреляцию Пирсона: $r_{xy} = \frac{cov_{xy}}{\sigma_x \sigma_y}$
  - для нелинейных зависимостей и числового предиктора – ранговые корреляции, например, Спирмана: $\rho_{xy} = 1 - \frac{6}{l(l-1)(l+1)} \sum_{i=1}^{l} \big(r(x_i) - r(y_i)\big)^2$
  - если категориальный предиктор принимает B различных значений $\{v_1, \dots, v_B\}$ или он числовой, но дискретезирован на B групп (бакеты, квантили или оптимальные отрезки), то как в деревьях решений можно использовать уменьшение вариации отклика или критерий Фишера:

$$F = \left(\frac{SS_{model}}{SS_{error}}\right)\left(\frac{l-B}{B-1}\right) \sim F_{B-1,l-B}, \text{ где } SS_{total} = \sum_{i=1}^{l}(y_i - \bar{y})^2,$$

$$SS_{error} = \sum_{b=1}^{B} \sum_{i:x_i=v_b} (y_i - \overline{y_b})^2, \; SS_{model} = SS_{total} - SS_{error}$$

Фильтрация признаков в задачах с категориальным откликом

- Если отклик принимает *K* значений то для категориальных предикторов с *B* различными значениями или для дискретизированных числовых признаков с *B* группами:
  - Обозначим: $p_k = P(y = k)$ – априорная вероятность значения отклика *k*,

    $l_b = |\{x_i = v_b\}|$ - число наблюдений с b-м значением предиктора,

    $p_{kb} = P(y = k|x_i = v_b)$ – условная вероятность значения отклика *k* для наблюдений с b-м значением предиктора
  - Статистические критерии для анализа таблиц сопряженности, например, $\chi^2 = \sum_{k=1}^{K}\sum_{b=1}^{B}\frac{l_b(p_{kb}-p_k)^2}{p_k} \sim \chi^2_{(B-1)(K-1)}$
  - Критерии на основе индекса gini:

$$\Delta Gini(x) = 1 - \sum_{k=1}^{K} p_k^2 - \sum_{b=1}^{B}\frac{l_b}{l}\left(1 - \sum_{k=1}^{K} p_{kb}^2\right)$$

  - Критерии на основе взаимной информации (разность между энтропией отклика и ожидаемой условной энтропией отклика):

$$IG = -\sum_{k=1}^{K} p_k \log_2 p_k + \sum_{b=1}^{B}\frac{l_b}{l}\sum_{k=1}^{K} p_{kb}\log_2 p_{kb}$$

Фильтрация признаков в задачах с бинарным откликом

- Можно рассматривать бинарный отклик:
  - как частный случай числового со значениями 0 и 1 или как частный случай категориального с двумя значениями и пользоваться соответствующими инструментами, рассмотренными выше
- Более эффективно пользоваться для фильтрации переменных в задачах с бинарным откликом (если категориальный или диксретезированный предиктор принимает B значений $\{v_1, \dots, v_B\}$)
  - отношением шансов и WOE (чем ближе WOE к 0 тем менее значимая переменная) или сглаженным WOE:

$$WOE(x) = \sum_{b=1}^{B} WOE_b,$$

$$\mathrm{WOE}_b = \log_2\left(\frac{count((y=1)\mathrm{and}(x=v_b))/count(y=1)}{count((y=0)\mathrm{and}\,(x=v_b))/count(y=0)}\right)$$

  - Информационной важностью предиктора:

$$IV = \sum_{b=1}^{B} (P(x=v_b|y=1) - P(x=v_b|y=0))WOE_b$$

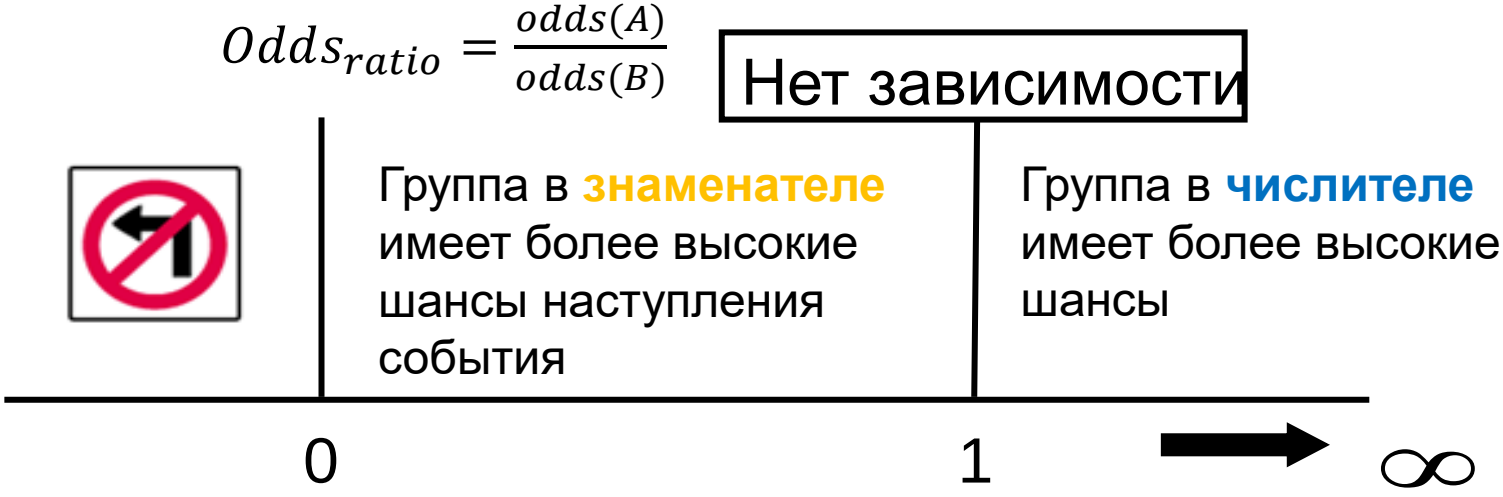
Оценка «силы» ассоциации между предиктором и бинарным откликом

- **Шанс** (это не вероятность) – отношение вероятностей события к не событию:

$$Odds = \frac{p_{event}}{p_{nonevent}}$$

- **Отношение шансов** (тоже не вероятность) показывает насколько вероятнее в терминах шансов появления события в группе A (соответствующей набору значений предикторов) по сравнению с другой группой B:

$$Odds_{ratio} = \frac{odds(A)}{odds(B)}$$



Сравнение вероятностей и шансов

	Заболел		
	Да	**Нет**	**Total**
Прививка	60	20	80
Без прививки	90	10	100
Total	150	30	180

Всего **Заболел** Без прививки ÷ Всего исходов Без прививки

Вероятность Заболел Без прививки =**90**÷**100**=**0.9**

Сравнение вероятностей и шансов

	Заболел		
	Да	**Нет**	**Total**
Прививка	60	20	80
Без прививки	90	10	100
Total	150	30	180

Вероятность **Заболел** Без прививки =0.90

÷

Вероятность **Не заболел** Без прививки =0.10

Шанс **Заболеть** Без прививки = **0.90÷0.10=9**

Без прививки шанс заболеть в 9 раз выше чем с прививкой

Сравнение вероятностей и шансов

	Заболел		
	Да	**Нет**	**Total**
Прививка	60	20	80
Без прививки	90	10	100
Total	150	30	180

Шанс Заболеть **с прививкой**=**3** ÷ Шанс Заболеть **Без прививки**=**9**

Отношение шансов =**3**÷**9**=**0.3333**

Шансов заболеть с прививкой в 3 раза меньше чем без

Отношение шансов в логистической регрессии

- Используется для оценки влияния переменной на отклик и показывает как изменятся шансы при изменении i-ой переменной на 1 (равно exp от коэффициента):

$$\text{logit}(p) = \log(odds) = w_0 + w_i x_i + \sum_{j \neq i} w_j x_j \Rightarrow$$

$$odds = \exp(w_0 + w_i x_i + \sum_{j \neq i} w_j x_j)$$

$$\text{logit}(p`) = \log(odds`) = w_0 + w_i(x_i + 1) + \sum_{j \neq i} w_j x_j \Rightarrow$$

$$odds` = \exp(w_0 + w_i(x_i + 1) + \sum_{j \neq i} w_j x_j)$$

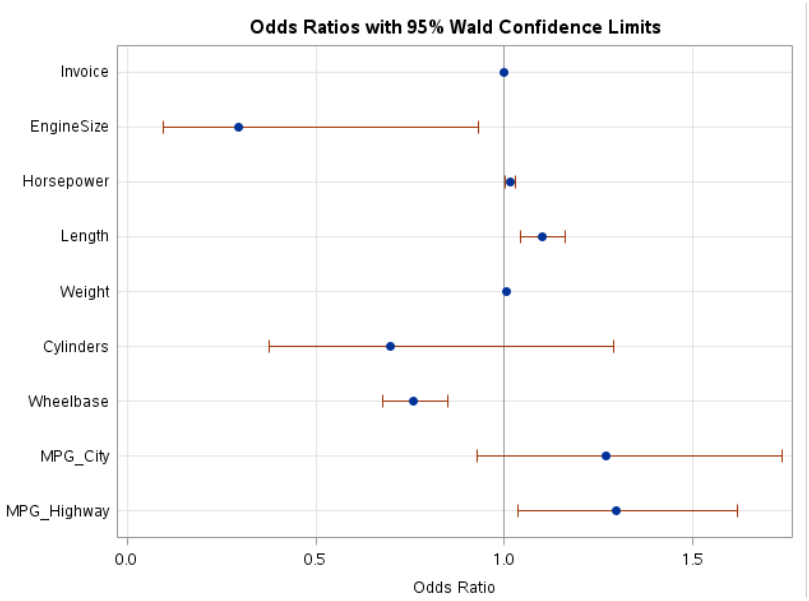
$$odds_{ratio} = \frac{odds'}{odds} = \exp(w_i)$$

- Если больше 1 – шансы увеличиваются, если меньше, то уменьшаются, интерпретация как в пуассоновской регрессии

Отношение шансов и важность переменных в логистической регрессии

exp(.)

Odds Ratio Estimates			
Effect	Point Estimate	95% Wald Confidence Limits	
Invoice	1.000	1.000	1.000
EngineSize	0.295	0.094	0.931
Horsepower	1.016	1.003	1.029
Length	1.100	1.044	1.160
Weight	1.005	1.004	1.007
Cylinders	0.696	0.376	1.289
Wheelbase	0.757	0.676	0.849
MPG_City	1.270	0.929	1.736
MPG_Highway	1.295	1.036	1.618



Analysis of Maximum Likelihood Estimates					
Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	-10.7686	4.6784	5.2983	0.0213
Invoice	1	-0.00013	0.000028	21.9445	<.0001
EngineSize	1	-1.2200	0.5858	4.3365	0.0373
Horsepower	1	0.0156	0.00666	5.4867	0.0192
Length	1	0.0957	0.0270	12.6146	0.0004
Weight	1	0.00529	0.000908	33.9767	<.0001
Cylinders	1	-0.3625	0.3146	1.3275	0.2493
Wheelbase	1	-0.2778	0.0580	22.9685	<.0001
MPG_City	1	0.2389	0.1595	2.2421	0.1343
MPG_Highway	1	0.2584	0.1136	5.1710	0.0230

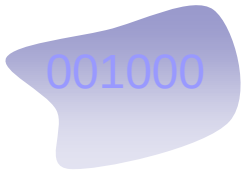
- Можно найти не только точечную оценку ОШ (OR), но и доверительный интервал
- Если он содержит 1, то доверительный интервал коэффициента содержит 0, т.е. предиктор не значимый
- Не учитывается разброс переменной

Категориальные предикторы

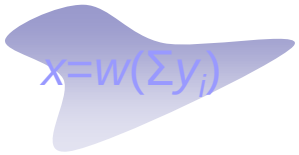
Пример переменной	Мощность	***Подход***
Physical characteristics Region Partnership status	10	Бинарное кодирование
Education level Urbanicity codes	100	**Преобразования или отображение на числовую шкалу**
State Ethnicity Employment classification	1000	
Postal Code Address	1000000	Связывание
Social security number	100000000	
text	*Бесконечность*	Текстовые модели

Подходы к кодировке категориальных признаков

Случайное кодирование



Бинарное кодирование



Преобразование
с учётом отклика

Unsupervised:

- Backward Difference Contrast [2][3]
- BaseN [6]
- Binary [5]
- Gray [14]
- Count [10]
- Hashing [1]
- Helmert Contrast [2][3]
- Ordinal [2][3]
- One-Hot [2][3]
- Rank Hot [15]
- Polynomial Contrast [2][3]
- Sum Contrast [2][3]

Supervised:

- CatBoost [11]
- Generalized Linear Mixed Model [12]
- James-Stein Estimator [9]
- LeaveOneOut [4]
- M-estimator [7]
- Target Encoding [7]
- Weight of Evidence [8]
- Quantile Encoder [13]
- Summary Encoder [13]

https://github.com/scikit-learn-contrib/category_encoders

Преобразование с учётом отклика

Level	N_i	ΣY_i	p_i
A	**1562**	**430**	**0.28**
B	**970**	**432**	**0.45**
C	**223**	**45**	**0.20**
D	**111**	**36**	**0.32**
E	**85**	**23**	**0.27**
F	**50**	**20**	**0.40**
G	**23**	**8**	**0.35**
H	**17**	**5**	**0.29**
I	**12**	**6**	**0.50**
J	**5**	**5**	**1.00**

«редкие значение» переменной - источник нестабильности, недостоверности в модели

Level* –** различные значения переменной ***X

N_i**–** число наблюдений, что ***X*** принимает ***i***-е значение

Σy_i - сумма бинарных откликов для наблюдений, где ***X*** принимает ***i***-е значение

$p_i = \Sigma y_i / N_i$- условная вероятность положительного отклика, если ***X*** принимает ***i***-е значение

Преобразование с учётом отклика

Level	N_i	ΣY_i	p_i
J	5	5	1.00
I	12	6	0.50
B	970	432	0.45
F	50	20	0.40
G	23	8	0.35
D	111	36	0.32
H	17	5	0.29
A	1562	430	0.28
E	85	23	0.27
C	223	45	0.20

Сортируем по $\boldsymbol{p_i}$, если значения $\boldsymbol{X}$ «рядом» после сортировки, то можно предположить, что они «похоже» влияют на отклик

Кодирование категориального признака порядковой (ординальной) переменной

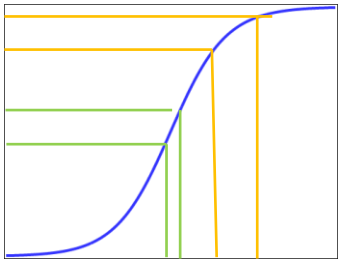
X'	N_i	ΣY_i	p_i
1	5	5	1.00
2	12	6	0.50
3	970	432	0.45
4	50	20	0.40
5	23	8	0.35
6	111	36	0.32
7	17	5	0.29
8	1562	430	0.28
9	85	23	0.27
10	223	45	0.20

Можем отобразить категориальный ***X*** на порядковую шкалу (новая переменная ***X'***), но не учитываем насколько похожи отклики и не решается проблема редких значений

Шансы

Level	N_i	ΣY_i	p_i	$\log(p_i/1-p_i)$	
J	5	5	1.00	.	
I	12	6	0.50	0.00	$\Delta p_i = 0.05 \Rightarrow$
B	970	432	0.45	-0.10	$\Delta\text{logit}(p_i)$=0.1
F	50	20	0.40	-0.18	
G	23	8	0.35	-0.27	
D	111	36	0.32	-0.32	
H	17	5	0.29	-0.38	$\Delta p_i = 0.05 \Rightarrow$
A	1562	430	0.28	-0.42	$\Delta\text{logit}(p_i)$=0.11
E	85	23	0.27	-0.43	
C	223	45	0.20	-0.60	

Рассмотрим логарифм шансов положительного отклика для ***i***-го значения ***X***, т.е. **logit** от p_i, сортировка не меняется, но более корректно учитываются различия в областях определенности (около 0 и 1) и неопределенности



Группировка значений переменной по шансам

Level	N_i	ΣY_i	p_i	$\log(p_i/1-p_i)$
J	5	5	1.00	.
I	12	6	0.50	0.00
B	970	432	0.45	-0.10
F	50	20	0.40	-0.18
G	23	8	0.35	-0.27
D	111	36	0.32	-0.32
H	17	5	0.29	-0.38
A	1562	430	0.28	-0.42
E	85	23	0.27	-0.43
C	223	45	0.20	-0.60

Можем агрегировать (например, с помощью одномерной кластеризации) «похожие» значения ***X***, объединив их в однородные (с точки зрения поведения отклика) группы …

Группировка значений переменной по шансам

X''	N_i	ΣY_i	p_i	$\log(p_i/1-p_i)$
CL1	1037	463	0.45	-0.09
CL2	134	44	0.33	-0.31
CL3	1664	458	0.28	-0.42
CL4	223	45	0.20	-0.60

… и создать новую категориальную переменную ***X''***, без редких уровней и с меньшим числом различных значений – уменьшаем число степеней свободы модели, увеличиваем стабильность модели и уменьшаем шанс переобучиться

Weight of evidence и шансы

- Снова **формула Байеса** (еще будем разбираться подробнее):
  - Обозначения : маленькие p – плотности, большие P – вероятности
  - $p(x)$ - плотность распределения наблюдений в пространстве признаков
  - $P(y)$ - априорная и $P(y|x)$ - апостериорная вероятности классов
  - $p(x|y)$ – функция правдоподобия
  - Совместная плотность распределения:

$$\boldsymbol{p(x,y) = p(x)P(y|x) = P(y)p(x|y) \Rightarrow P(y|x) = P(y)p(x|y)/p(x)}$$

- Логарифм условных шансов:

$$\log\left(\frac{P(y=1|x)}{P(y=0|x)}\right) = \log\left(\frac{P(y=1)p(x|y=1)/p(x)}{P(y=0)p(x|y=0)/p(x)}\right) = \log\left(\frac{P(y=1)}{P(y=0)}\right) + \log\left(\frac{p(x|y=1)}{p(x|y=0)}\right)$$

- WOE и «наивное» предположение (независимость переменных):

$$\log\left(\frac{P(y=1|x)}{P(y=0|x)}\right) = \log\left(\frac{P(y=1)}{P(y=0)}\right) + \sum_{j=1}^{p} \mathrm{WOE}(x_j), \quad \text{где } \mathrm{WOE}(x_j) = \log\left(\frac{p(x_j|y=1)}{p(x_j|y=0)}\right)$$

Априорные шансы (log равен 0, если выборка «сбалансирована»)

Вклад шансов каждой из p переменных

Weight of Evidence для категориальной переменной

- Пусть в задаче с бинарным откликом категориальная (или дискретизированная) переменная x_j принимает k различных значений $\{v_1, \dots, v_k\}$, тогда (опять по формуле Байеса):
  - $WOE(x_j)$ - j–й переменной: $WOE(x_j) = \sum_{i=1}^{k} \mathrm{WOE}_i(x_j)$,
  - где $WOE_i(x_j)$ - weight of evidence i–го значения v_i

$$\mathrm{WOE}_i(x_j) = \log\left(\frac{P(x_j = v_i | y = 1)}{P(x_j = v_i | y = 0)}\right) = \log\left(\frac{count((y = 1) \text{ and } (x_j = v_i))/count(y = 1)}{count((y = 0) \text{ and } (x_j = v_i))/count(y = 0)}\right)$$

- Интерпретация WOE:
  - ***WOE**i* **> 0**: (ОШ>1) i-e значение связано с более высоким шансом положительного отклика
  - ***WOE**i* **< 0**: (ОШ<1) i-e значение связано с более низким шансом положительного отклика
  - ***WOE**i* **= 0**: (ОШ=1) i-e значение не влияет на уровень отклика
  - ***WOE*** в целом оказывает *предиктивную силу* категориальной (или дискретезированной) переменной и ее отдельных значений

Weight of Evidence

Level	N_i	ΣY_i	p_i	WOE_i
J	5	5	1.00	.
I	12	6	0.50	0.71
B	970	432	0.45	0.49
F	50	20	0.40	0.3
G	23	8	0.35	0.08
D	111	36	0.32	-0.03
H	17	5	0.29	-0.17
A	1562	430	0.28	-0.26
E	85	23	0.27	-0.28
C	223	45	0.20	-0.67
	3058	1010		0.17

Не решает проблему редких уровней

Информационная важность переменных

- Дивергенция Кульбака-Лейблера (различающая информация, относительная энтропия и другие термины):
  - Ассиметричная мера расхождения двух распределений
  - Для дискретных распределений P и Q: $D_{KL}(P||Q) = \sum_i \ p_i \log(p_i/q_i)$
  - Симметричный вариант: $D_{KL}(P;Q) = D_{KL}(P||Q) + D_{KL}(Q||P)$
- IV **(информационная важность** или информационный индекс) :
  - Симметричная дивергенция (расстояние) Кульбака-Лейблера, которое показывает насколько отличаются распределения переменной (отдельных значений переменной) внутри положительного и отрицательного классов:
  - Пусть в задаче с бинарным откликом категориальная переменная x принимает k различных значений $\{v_1, \dots, v_k\}$, тогда:

$$IV(x) = \sum_{j=1}^{k} (P(x = v_j | y = 1) - P(x = v_j | y = 0)) WOE_j(x)$$

Information Value

Level	N_i	ΣY_i	p_i	IV_i
J	5	5	1.00	.
I	12	6	0.50	0.0021
B	970	432	0.45	0.0809
F	50	20	0.40	0.0015
G	23	8	0.35	0
D	111	36	0.32	0
H	17	5	0.29	0.0002
A	1562	430	0.28	0.033
E	85	23	0.27	0.0021
C	223	45	0.20	0.0284
	3058	1010		0.1482

- Эвристические пороги на IV:
  - Меньше 0.02 – незначимая переменная
  - 0.02 – 0.10 низкая прогнозная сила
  - 0.10 – 0.30 средняя прогнозная сила
  - 0.30 – 0.50 высокая прогнозная сила
  - Больше 0.50 – что-то пошло не так

Пример

```
import pandas as pd
import numpy as np
mydata = pd.read_csv("https://stats.idre.ucla.edu/stat/data/binary.csv")
```

```
mydata.head(10)
```

	admit	gre	gpa	rank
0	0	380	3.61	3
1	1	660	3.67	3
2	1	800	4.00	1
3	1	640	3.19	4
4	0	520	2.93	4
5	1	760	3.00	2
6	1	560	2.98	1
7	0	400	3.08	2
8	1	540	3.39	3
9	0	700	3.92	2

```
def iv_woe(data, target, bins=10, show_woe=False):

    #Empty Dataframe
    newDF,woeDF = pd.DataFrame(), pd.DataFrame()

    #Extract Column Names
    cols = data.columns

    #Run WOE and IV on all the independent variables
    for ivars in cols[~cols.isin([target])]:
        if (data[ivars].dtype.kind in 'bifc') and (len(np.unique(data[ivars]))>10):
            binned_x = pd.qcut(data[ivars], bins,  duplicates='drop')
            d0 = pd.DataFrame({'x': binned_x, 'y': data[target]})
        else:
            d0 = pd.DataFrame({'x': data[ivars], 'y': data[target]})
        d0 = d0.astype({"x": str})
        d = d0.groupby("x", as_index=False, dropna=False).agg({"y": ["count", "sum"]})
        d.columns = ['Cutoff', 'N', 'Events']
        d['% of Events'] = np.maximum(d['Events'], 0.5) / d['Events'].sum()
        d['Non-Events'] = d['N'] - d['Events']
        d['% of Non-Events'] = np.maximum(d['Non-Events'], 0.5) / d['Non-Events'].sum()
        d['WoE'] = np.log(d['% of Non-Events']/d['% of Events'])
        d['IV'] = d['WoE'] * (d['% of Non-Events']-d['% of Events'])
        d.insert(loc=0, column='Variable', value=ivars)
        print("Information value of " + ivars + " is " + str(round(d['IV'].sum(),6)))
        temp =pd.DataFrame({"Variable" : [ivars], "IV" : [d['IV'].sum()]}, columns = ["Variable", "IV"])
        newDF=pd.concat([newDF,temp], axis=0)
        woeDF=pd.concat([woeDF,d], axis=0)

        #Show WOE Table
        if show_woe == True:
            print(d)
    return newDF, woeDF
```

```
iv, woe = iv_woe(data = mydata, target = 'admit', bins=10, show_woe = True)
```

Пример

```
woe
```

	Variable	Cutoff	N	Events	% of Events	Non-Events	% of Non-Events	WoE	IV
0	gre	(219.999, 440.0]	48	6	0.047244	42	0.153846	1.180625	0.125857
1	gre	(440.0, 500.0]	51	12	0.094488	39	0.142857	0.413370	0.019994
2	gre	(500.0, 520.0]	24	10	0.078740	14	0.051282	-0.428812	0.011774
3	gre	(520.0, 560.0]	51	15	0.118110	36	0.131868	0.110184	0.001516
4	gre	(560.0, 580.0]	29	6	0.047244	23	0.084249	0.578450	0.021406
5	gre	(580.0, 620.0]	53	21	0.165354	32	0.117216	-0.344071	0.016563
6	gre	(620.0, 660.0]	45	17	0.133858	28	0.102564	-0.266294	0.008333
7	gre	(660.0, 680.0]	20	9	0.070866	11	0.040293	-0.564614	0.017262
8	gre	(680.0, 740.0]	44	12	0.094488	32	0.117216	0.215545	0.004899
9	gre	(740.0, 800.0]	35	19	0.149606	16	0.058608	-0.937135	0.085278
0	gpa	(2.259, 2.9]	43	8	0.062992	35	0.128205	0.710622	0.046342
1	gpa	(2.9, 3.048]	37	11	0.086614	26	0.095238	0.094917	0.000819
2	gpa	(3.048, 3.17]	42	8	0.062992	34	0.124542	0.681634	0.041955
3	gpa	(3.17, 3.31]	42	10	0.078740	32	0.117216	0.397866	0.015308
4	gpa	(3.31, 3.395]	36	8	0.062992	28	0.102564	0.487478	0.019290
5	gpa	(3.395, 3.494]	40	14	0.110236	26	0.095238	-0.146246	0.002193
6	gpa	(3.494, 3.61]	41	16	0.125984	25	0.091575	-0.318998	0.010976
7	gpa	(3.61, 3.752]	39	20	0.157480	19	0.069597	-0.816578	0.071764
8	gpa	(3.752, 3.94]	42	13	0.102362	29	0.106227	0.037062	0.000143
9	gpa	(3.94, 4.0]	38	19	0.149606	19	0.069597	-0.765285	0.061230
0	rank	1	61	33	0.259843	28	0.102564	-0.929588	0.146204
1	rank	2	151	54	0.425197	97	0.355311	-0.179558	0.012548
2	rank	3	121	28	0.220472	93	0.340659	0.435110	0.052295
3	rank	4	67	12	0.094488	55	0.201465	0.757142	0.080997

```
iv
```

	Variable	IV
0	gre	0.312882
0	gpa	0.270020
0	rank	0.292044

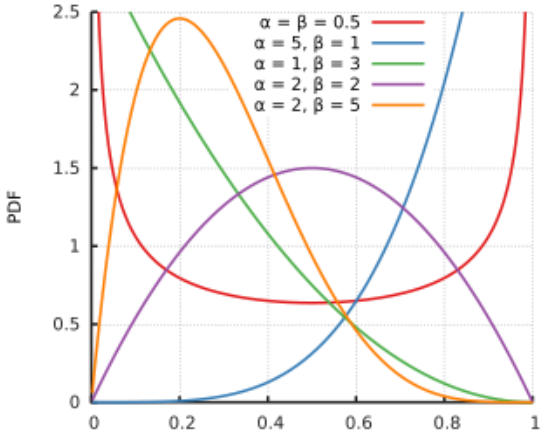
Сглаженное WOE для борьбы с редкими значениями

Level	N_i		ΣY_i		p_i	SWOE
J	5	+24	5	+8	0.45	0.5
I	12	+24	6	+8	0.39	0.25
B	970	+24	432	+8	0.44	0.48
F	50		20		0.38	0.21
G	23	+24	8	+8	0.34	0.05
D	111	+24	36	+8	0.33	-0.02
H	17	+24	5	+8	0.32	-0.06
A	1562		430		0.28	-0.26
E	85	+24	23	+8	0.28	-0.22
C	223	+24	45	+8	0.21	-0.59

- Для «исправления» ситуации с редкими значениями:
  - Добавим для каждого значения переменной «виртуальный» набор наблюдений (фиксированного размера) с вероятностью положительного отклика равной априорной.
  - Чем более редкий уровень, тем выше влияние априорного распределения.

Бета распределение для кодирования категориальных предикторов по выборке

- Бета распределение:
  - двухпараметрическое (альфа и бета)
  - мат. ожидание: $\frac{a}{a+b}$
  - используется для моделирования случайных величин, заданных на интервале.

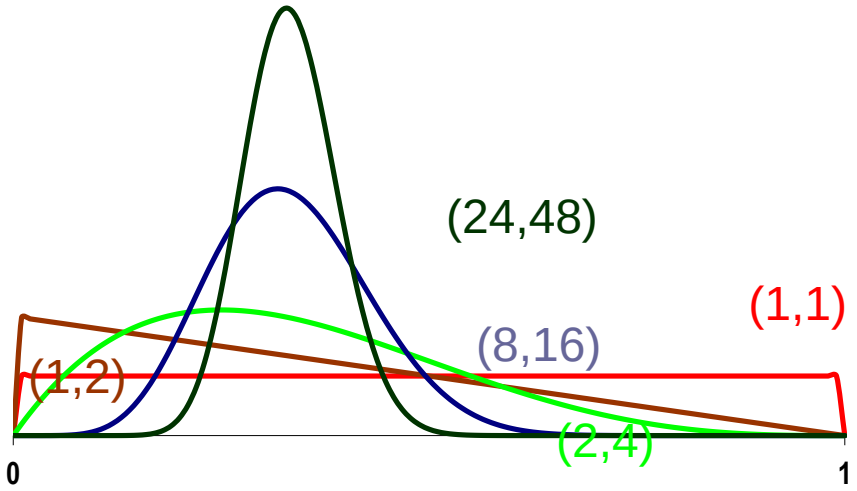


- Моделируем:
  - условную вероятность положительного отклика для фиксированного значения категориальной переменной как случайную величину $p_i \sim \text{Beta}\,(a_s, b_s)$,
  - a_0, b_0 - параметры априорного распределения
  - a_s, b_s пересчитываются последовательно, проходя всю выборку, где категориальная переменная принимает i-e значение, при этом …
  - $a_{s+1} = a_s + 1$, если встретили наблюдение с откликом $y = 1$,
  - $b_{s+1} = b_s + 1$, если встретили наблюдение с откликом $y = 0$,

Сглаженный WOE

Значения X

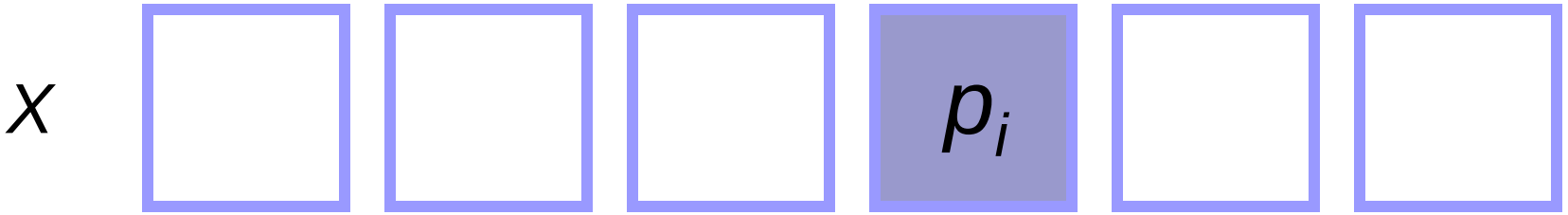
X | | | | p_i | | |

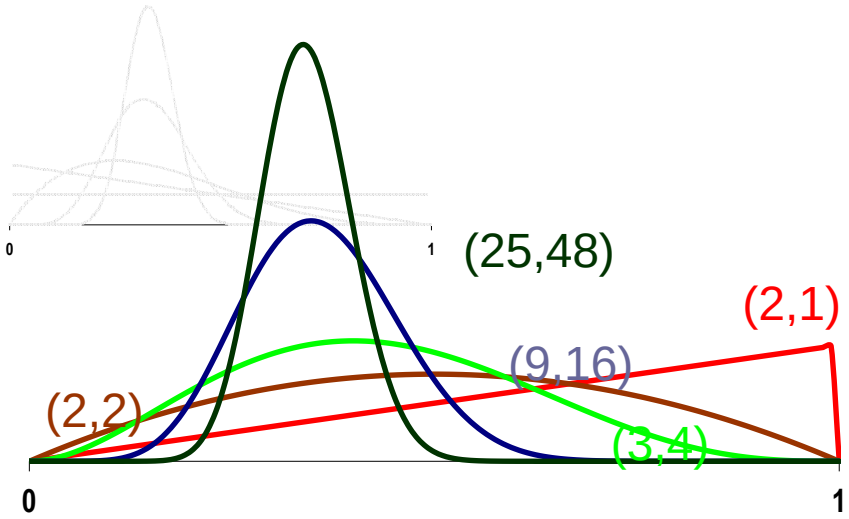


$$p_i \sim \text{Beta}\,(a, b)$$

Сглаженный WOE

Значение X



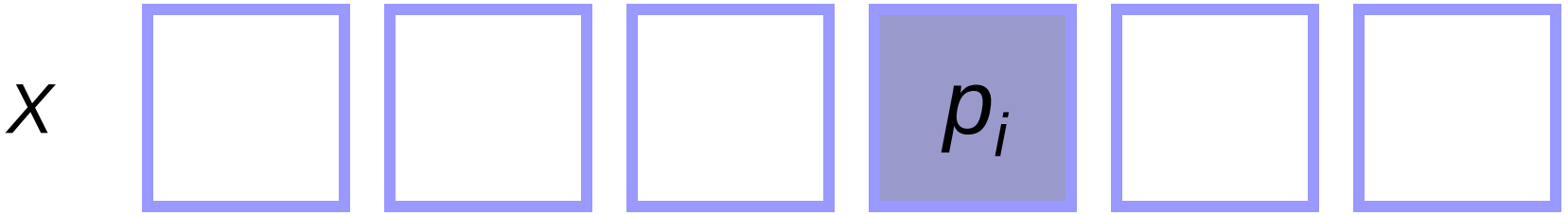


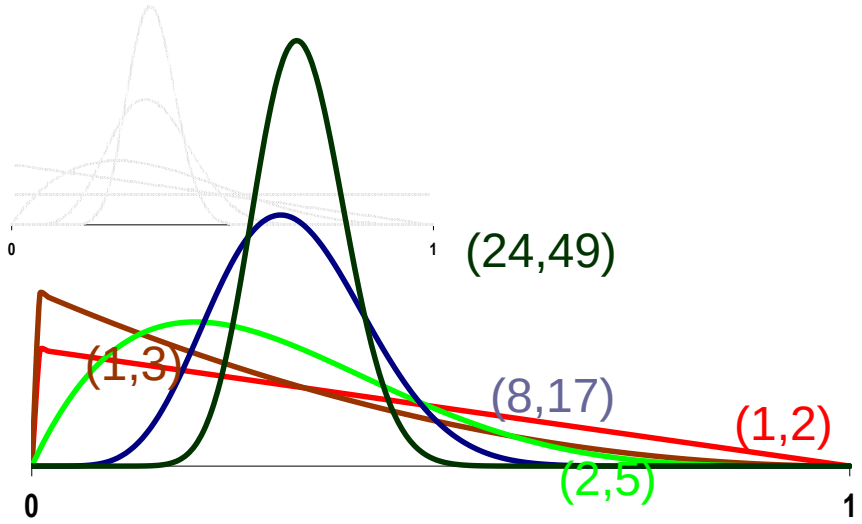
$$p_i \sim \text{Beta}\,(a, b)$$

$$p_i | y = 1 \sim \text{Beta}\,(a + 1, b)$$

Сглаженный WOE

Значения X





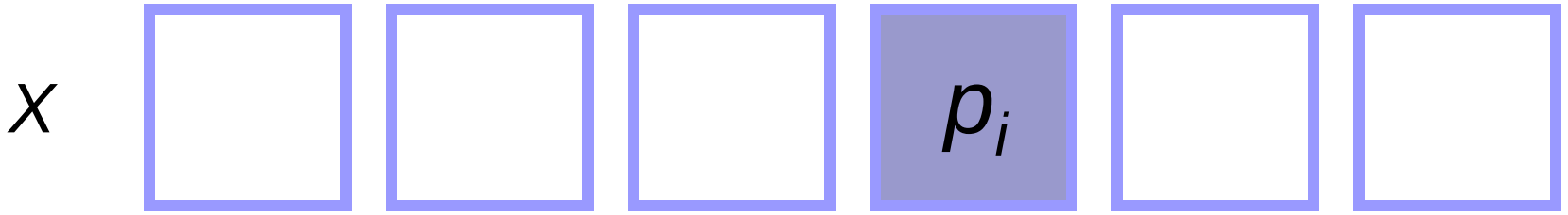
$$p_i \sim \text{Beta}\,(a, b)$$

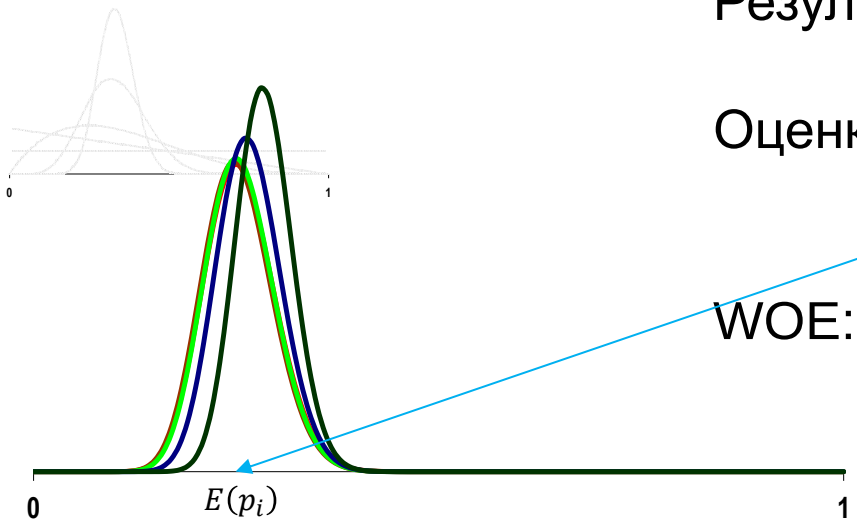
$$p_i | y = 1 \sim \text{Beta}\,(a + 1, b)$$

$$p_i | y = 0 \sim \text{Beta}\,(a, b + 1)$$

Сглаженный WOE

Значения X





Результат:

$$p_i \sim \text{Beta}\,(a + n_1, b + n_0)$$

Оценка:

$$E(p_i) = \frac{n_1 + a}{n_1 + n_0 + a + b}$$

WOE:

$$\log\left(\frac{E(p_i)}{1 - E(p_i)}\right)$$

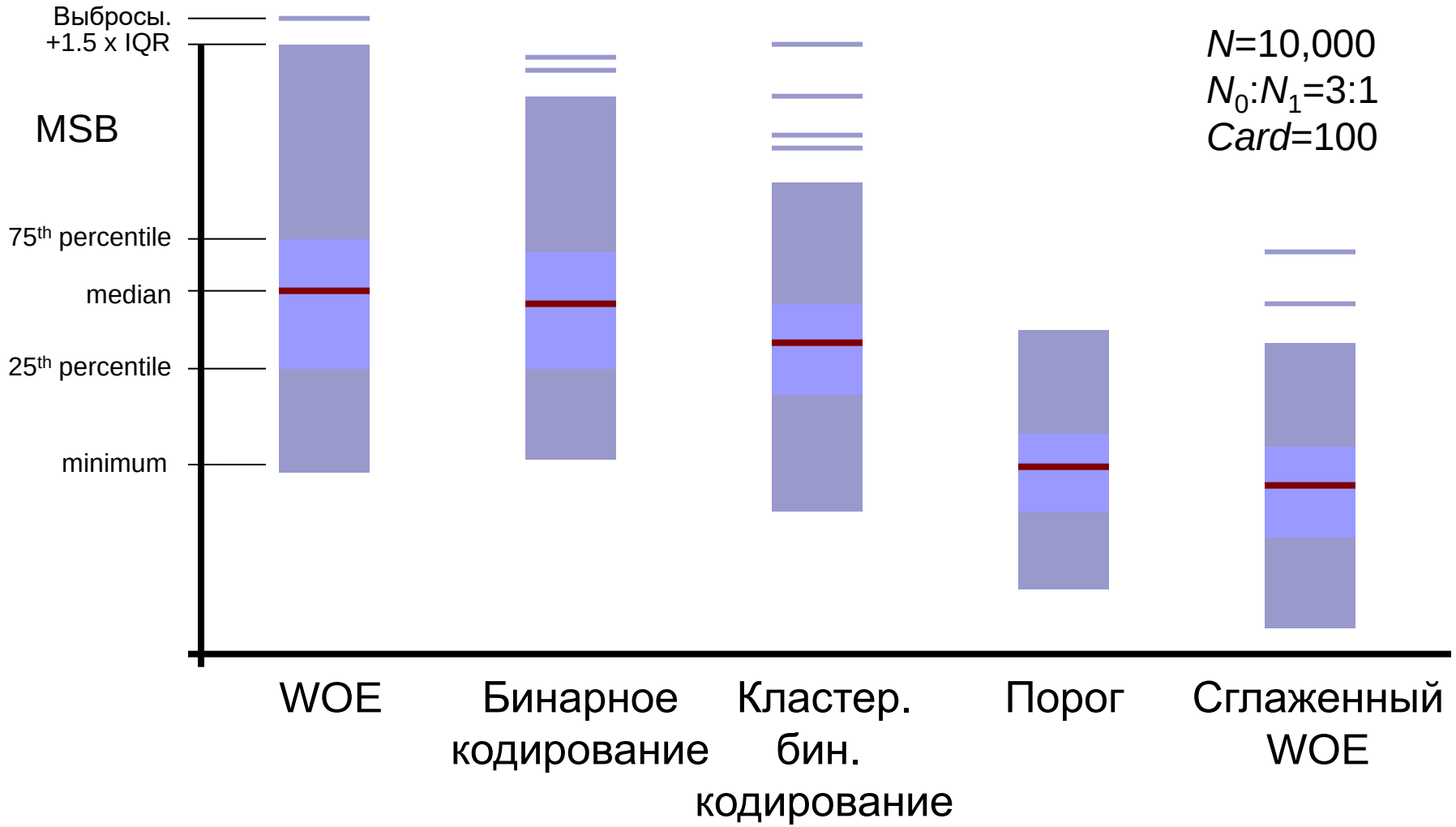
Наблюдаемое число событий, если x принимает i-е значение

$$SWOE_i(x) = \log\left(\frac{n_1 + c\rho_1}{n_0 + c(1 - \rho_1)}\right)$$

Априорная вероятность события

Параметр регуляризации

Имитационный эксперимент



Smoothed WOE можно обобщать для других типов отклика (и функций связи)

- Для числового отклика с квадратичной функцией потерь:

$$SWOE(x = v) = \frac{\sum_i y_i I[x_i = v] + c * E(y)}{\sum_i I[x_i = v] + c}$$

  - c – параметр сглаживания, индекс i для перебора наблюдений по всей выборке, $x_i = v$ – условие, что переменная x равна v для i – го наблюдения, y_i - числовой отклик i – го наблюдения
- Для много-классовой классификации через обобщения вида «каждый против всех»:

$$SWOE^k(x = v) = \log\left(\frac{\sum_i I[y_i = k] I[x_i = v] + c\rho_k}{\sum_i I[y_i \neq k] I[x_i = v] + c(1 - \rho_k)}\right)$$

  - параметр c, индекс i, условие $x_i = v$ – аналогично;
  - y_i - категориальный отклик i – го наблюдения
  - ρ_k - безусловная вероятность k класса в выборке

$$IV(x) = \sum_k \sum_v (P(x = v|y = k) - P(x = v|y \neq k)) SWOE^k(x = v)$$

Практическое применение WOE и IV

- Моделенезависимый отбор важных переменных:
  - Дискретизируем (например, на квантили или равные интервалы) числовые переменные, категориальные берем как есть
  - Для всех считаем IV, сортируем переменные по убыванию, оставляем топ k самых важных

$$x_{(1)}, x_{(2)}, \dots, x_{(k)}, x_{(k+1)} \dots, x_{(p)}, \text{ где } IV\left(x_{(s)}\right) \geq IV\left(x_{(s+1)}\right)$$

- Моделенезависимая оценка качества разбиения на тренировочную и проверочную выборку
  - Дискретизируем числовые переменные, категориальные как есть
  - Разбиваем набор данных $Z = Z_{test} \cup Z_{train}$ и по переменным считаем IV на тренировочном и тестовом наборе
  - Если для некоторой переменной x: $||IV_{test}(x) - IV_{train}(x)|| > \Delta$, то производим разбиение заново

Практическое применение WOE и IV

- Отображение категориальных переменных на числовую шкалу для сокращения числа степеней свободы и упрощения модели:

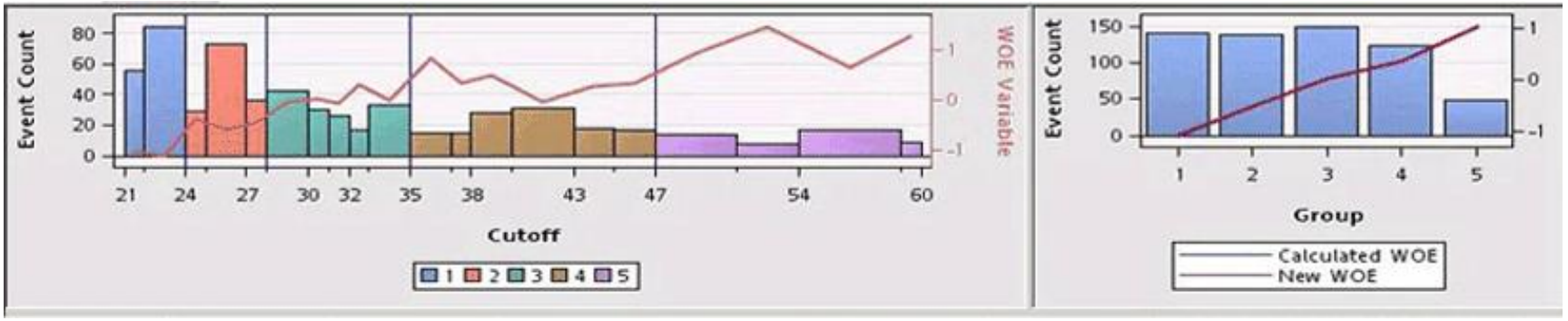
$$x_{new} = WOE(x_{old})$$

	admit	gre	gpa	rank
0	0	380	3.61	3
1	1	660	3.67	3
2	1	800	4.00	1
3	1	640	3.19	4

	Variable	Cutoff	WoE
0	rank	1	-0.929588
1	rank	2	-0.179558
2	rank	3	0.435110
3	rank	4	0.757142

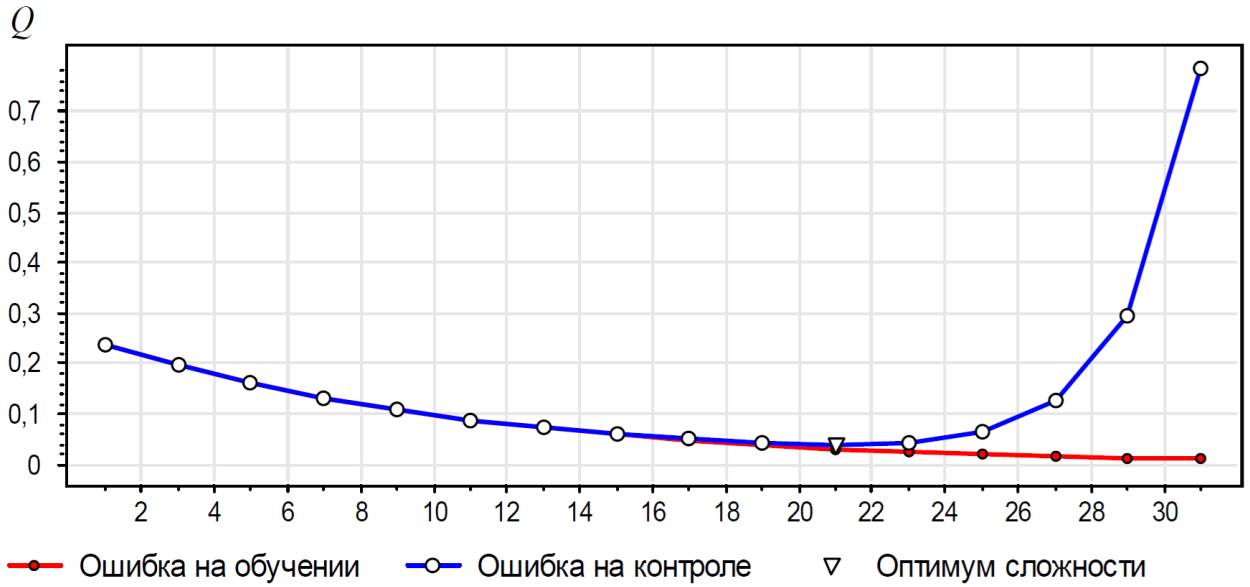
	admit	gre	gpa	WOE rank
0	0	380	3.61	0.4351
1	1	660	3.67	0.4351
2	1	800	4.00	-0.9295
3	1	640	3.19	0.7571

- Эффективная дискретизации переменных:
  - нахождение (часто «интерактивное) такого разбиения, чтобы максимизировать IV всей переменной и по возможности добиться монотонного роста WOE



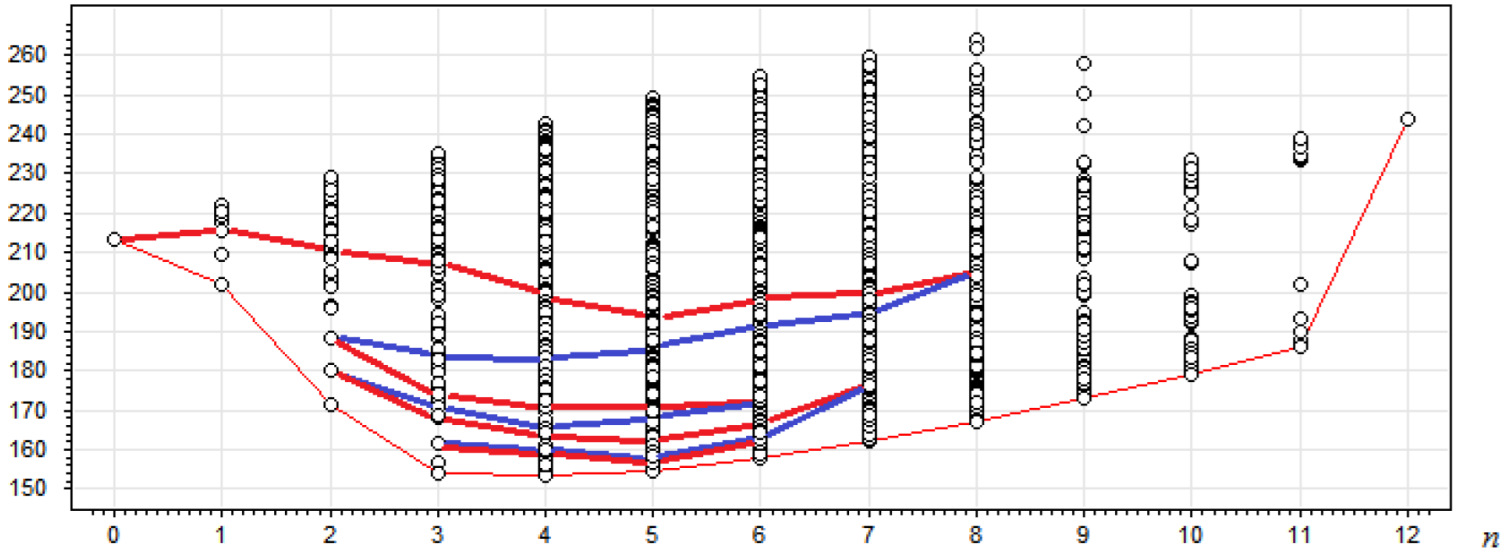
«Обертки» и пошаговый отбор переменных с учителем - критерии

Внутренний критерий и внешний критерий:



- Примеры критериев:
  - Внешние: кросс-валидационная и валидационная ошибка, информационные критерии AIC и BIC, статистические критерии, учитывающие число степеней свободы модели
  - Внутренние: эмпирический риск, статистики отклонения, MSE, MAE на тренировочном наборе

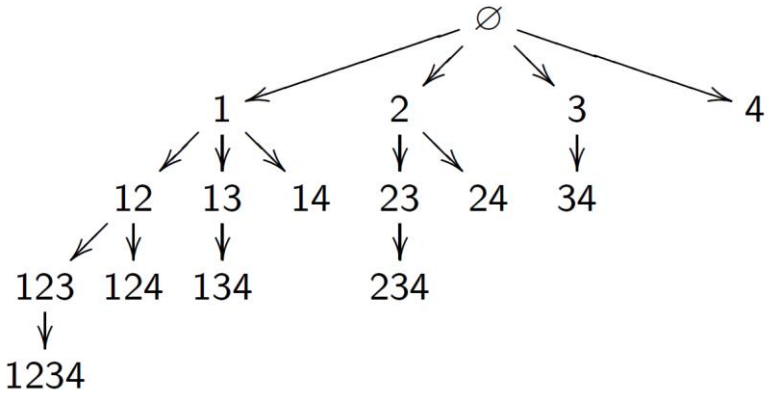
Пошаговый отбор переменных с учителем - стратегии

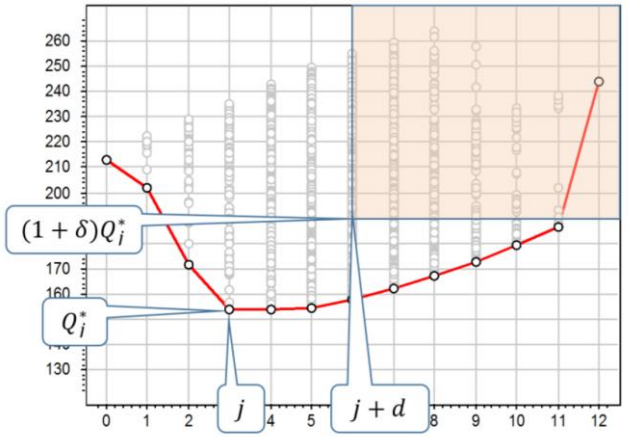


- Ранее рассмотренные пошаговые методы:
  - прямые (от простого к сложному), обратные (от сложного к простому), комбинированные – шаг(и) вперед и шаг(и) назад, SWAP пары переменных
- Что еще бывает?
  - Поиск в ширину и глубину, эволюционные алгоритмы, случайный поиск

Направленный перебор «в глубину»

- Дерево перебора признаков – от простых моделей к сложным:

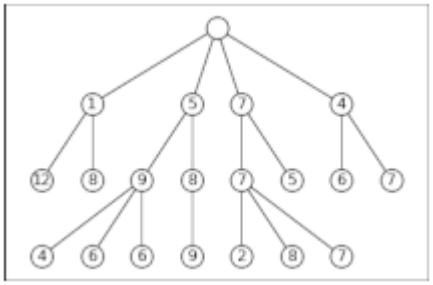




- Метод ветвей и границ – поиск «в глубину» с отсечением малоперспективных путей:
  - Пусть Q_j^*- лучшая уже найденная модель сложности j, тогда ветка с набором J отсекается, если найдется такой j, что $Q(J) \geq (1+\delta)Q_j^*$ и $|J| \geq j+d$, где δ и d – параметры
  - При шаге в глубину фильтровать (сортировать) переменные по важности для ускорения поиска
  - Все равно как правило сохраняется экспоненциальная сложность

Усеченный поиск в ширину (beam search)

- Основной принцип «неокончательных решений»:
  - На каждом шаге оставлять максимальную свободу выбора принятия решений
  - На каждой j-ой итерации строить не один набор, а множество из B_j наборов $R_j = \left\{J_j^1, \dots, J_j^{Bj}\right\}$,



где $|J_j^b| = j$, $b = 1, .., B_j$, $B_j \le B$ – параметр ширины поиска

- Алгоритм (итерации по j – сложность наборов)
  - Инициализация: R_1 - одноэлементные наборы, $Q^* = Q(\emptyset)$
  - Для всех $j = 1, \dots, n$
    1. Отсортировать $R_j = \left\{J_j^1, \dots, J_j^{Bj}\right\}$ по возрастанию критерия $Q(J_j)$
    2. Если $B_j > B$, то оставить B лучших наборов в R_j
    3. Если есть улучшение $Q(J_j^1) < Q^*$, то запомнить $j^* = j$, $Q^* = Q(J_j^1)$
    4. Если $j - j^* \ge d$ то вернуть $J_{j^*}^1$
    5. Породить следующий набор: $R_j = \left\{J \cup \{X_i\} | J \in R_j, X_i \in X \setminus J\right\}$

Эволюционные алгоритмы отбора признаков

- Терминология:
  - «Индивид» $J_j \subseteq \{f_i\}_{i=1}^{p}$ - текущее отобранное подмножество из всех p признаков на j шаге алгоритма
  - «Поколение» $R_j = \left(J_j^1, \ldots, J_j^{Bj}\right)$- вектор из B_j разных индивидов, отсортированных по убыванию критерия $Q(J_j)$
  - «Хромосома» $\beta(J)$ кодирует бинарным вектором признаки $\beta_i = [f_i \in J]$
  - Операция **crossover** «скрещивание» $\beta^{new} = \beta^{left} \times \beta^{right}$, например: $\beta_i^{\text{new}} = \rho_i \beta_i^{left} + (1 - \rho_i)\beta_i^{right}$, где $\rho_i \sim Uni((0,1))$ – случайный параметр
  - Операция **mutation** «мутация» $\beta_i^{new} = \sim\beta_i^{old}$, например: $\beta_i^{\text{new}} = \rho_i(1 - \beta_i^{old}) + (1 - \rho_i)\beta_i^{old}$, где $\rho_i \sim Uni((0,1))$, где $i \sim Uni(\{1, \ldots, p\})$
  - Операция **select** «селекция»: $R_{j+1}(B) = \text{top}_B(R_j)$
- Алгоритм:

$R_{j+1} = \text{select}\left[\text{crossover}\left(\text{mutation}(R_j)\right)\right]$ или $R_{j+1} = \text{select}\left[\text{mutation}\left(\text{crossover}(R_j)\right)\right]$

Обучение без учителя

- Иногда называют задачей «самоорганизации»
- Все признаки равнозначны, нет отклика $X = (x_1, \ldots, x_n)$, поэтому:
  - Обучение без учителя более «субъективное» (нет единых интуитивно понятных мер качества на основе оценки отклонения отклика от прогноза)
  - Поэтому менее «автоматизируемо», сложнее подбирать метапараметры и сравнивать полученные модели
- Важность этого направления велико:
  - «Истинный data mining» - ищет неизвестные заранее зависимости без «подсказок» эксперта
  - Много важных прикладных задач по сегментации, выявление скрытых характеристик, зависимостей между атрибутами и т.д.
  - Для больших данных получить качественно размеченный набор тяжело или невозможно, часто возникают задачи *semi-supervised learning* (частичное обучение), когда размечена лишь чатсь набора

Основные задачи обучения без учителя

- Поиск **скрытых** признаков, характеристик или структур (групп или зависимостей) в **неразмеченных** данных:

$$z_1 = F_1(x_1, \dots, x_n), \dots, z_k = F_k(x_1, \dots, x_n)$$

- Основные задачи обучения без учителя:
  - **Оценивание плотности распределения:** $F(x_1, \dots, x_n) \approx p(x)$.
  - **Кластеризация:** $z_i = F_i(x_1, \dots, x_n)$, z_i - кластеры, F_i - функции принадлежности или индикаторные функции.
  - **Поиск зависимостей и структур в пространстве признаков (в том числе отбор):** $z_i = F_i(x_1, \dots, x_n)$ - либо новые признаки (для проекции, визуализации, сокращения), либо структуры взаимосвязей признаков (кластеры переменных, ассоциативные правила), либо подмножество исходных признаков, описывающих основные вариации (отбор)
  - **Выявление аномалий:** $z(x) = F_i(x_1, \dots, x_n)$ - оценка аномальности наблюдения, формальные определения разные
  - Иногда один и тот же метод используется для решения нескольких задач, например: SOM – кластеризация и проекция, EM – оценка плотности и кластеризация, kernel PCA – проекция и поиск аномалий

Пошаговый отбор без учителя

- Можно реализовать пошаговые стратегии отбора (как с учителем), но в качестве критерия брать не долю описанной вариации отклика, а долю описанной вариации не включенных в модель переменных, простой линейный пример:
  - X_{in}- множество переменных, включенных в модель
  - X_{out} - множество переменных, не включенных в модель
  - критерий для оценки шага:

$$tr\left(X_{out}^{T}\left(I-\boldsymbol{X_{in}}\left(\boldsymbol{X_{in}^{T}X_{in}}\right)^{-1}\boldsymbol{X_{in}^{T}}\right)X_{out}\right)\rightarrow \min$$

  - похоже на линейную регрессию с матрицей проекции $H=X_{in}\left(X_{in}^{T}X_{in}\right)^{-1}X_{in}^{T}$, но проецируем не отклик, а невключенные переменные X_{out}, т.е. оценивается как X_{in} описывают не вариацию отклика, а вариацию оставшихся переменных
  - Критерий «внутренний» – не учитывает сложность модели, поэтому обычно делается пошаговый перебор с ограничением на число включенных переменных, а потом весь путь проверяется по внешнему критерию

Пример использования

Пример:

- Преобразуем транзакционный набор ASSOC в матрицу клиент-продукт:

```
import pandas as pd
```

```
assoc = pd.read_csv("assoc.csv")
```

```
assoc = assoc.groupby("CUSTOMER")["PRODUCT"].value_counts().unstack(fill_value=0)
assoc
```

PRODUCT	apples	artichok	avocado	baguette	bordeaux	bourbon	chicken	coke	corned_b	cracker	ham	heinel
CUSTOMER												
0.0	0	0	0	0	0	1	0	0	1	0	1	
1.0	0	0	0	1	0	0	0	0	1	1	0	
2.0	0	1	1	0	0	0	0	0	0	1	1	
3.0	0	0	0	0	0	1	0	1	0	0	1	
4.0	1	0	1	0	0	0	0	0	1	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	

Пример

$$\mathbf{C}_{11} = \mathbf{X}_1^\top \mathbf{X}_1,\ \mathbf{C}_{12} = \mathbf{X}_1^\top \mathbf{X}_2,\ \text{and}\ \mathbf{C}_{21} = \mathbf{X}_2^\top \mathbf{X}_1.$$

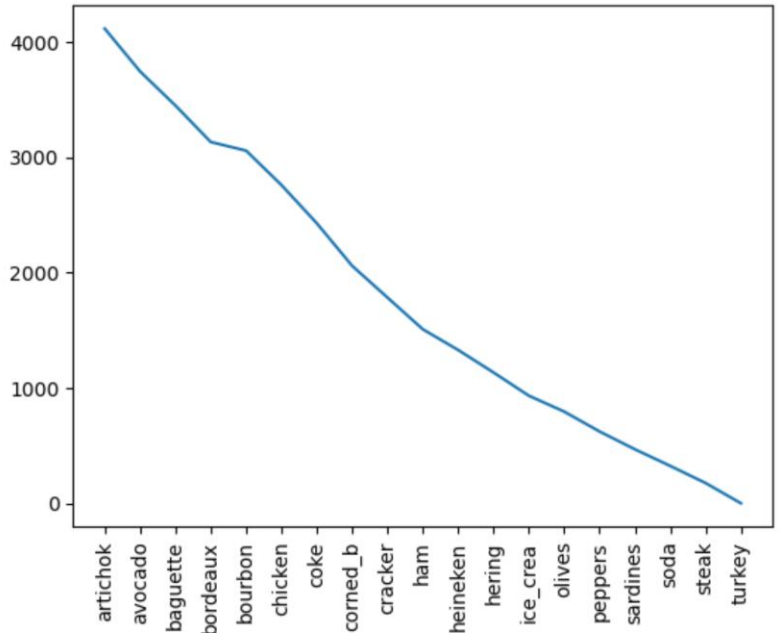
$$\mathbf{C} = \mathbf{X}^\top \mathbf{X} = \begin{pmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} \\ \mathbf{C}_{21} & \mathbf{C}_{22} \end{pmatrix}$$

$$\mathbf{X}_2^\top \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1 \right)^{-1} \mathbf{X}_1^\top \right) \mathbf{X}_2 = \mathbf{C}_{22} - \mathbf{C}_{21} \mathbf{C}_{11}^{-1} \mathbf{C}_{12}$$

```
assoc = assoc - assoc.mean()
matrix = assoc.values
```

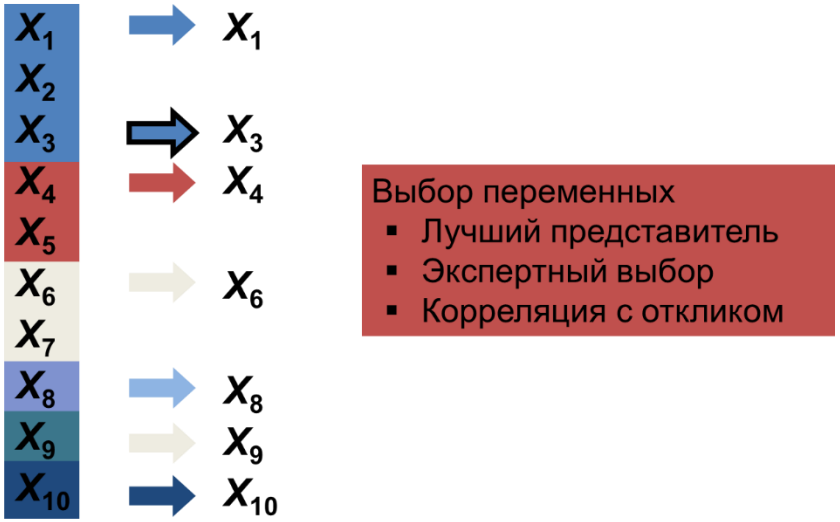
```
history = []
for i in range(1, len(assoc.columns)):
    X1, X2 = matrix[:, :i], matrix[:, i:]
    C11 = X1.T @ X1
    C12 = X1.T @ X2
    C21 = X2.T @ X1
    C22 = X2.T @ X2
    x = C22 - C21 @ np.linalg.inv(C11) @ C12
    score = np.trace(x)
    history.append(score)
```

```
plt.plot(assoc.columns[1:], history)
plt.xticks(rotation='vertical')
pass
```



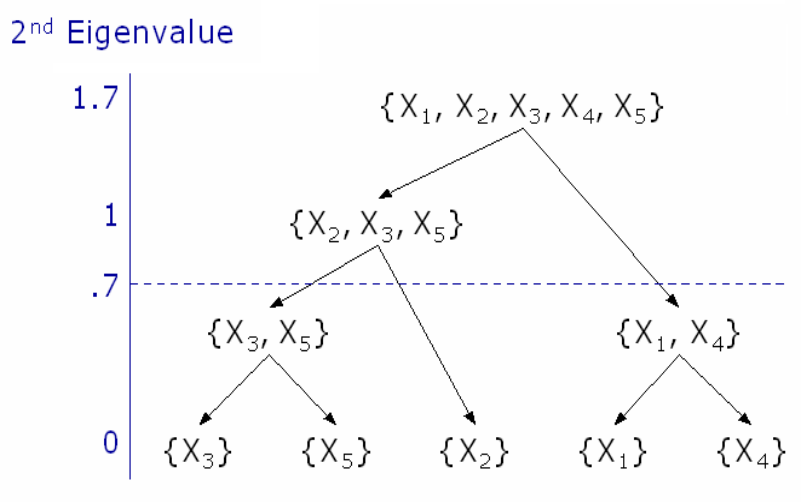
Фильтрация признаков в задачах без учителя

- Можно посчитать попарные взаимные оценки силы связей предикторов, но как выбрать лучших представителей?
- Методы кластеризации переменных без учителя:
  - группировка переменных в иерархические кластеры так, чтобы в одном кластере переменные были **максимально коррелированы**, а кластеры между собой нет
  - затем выбирается лучший представитель кластера

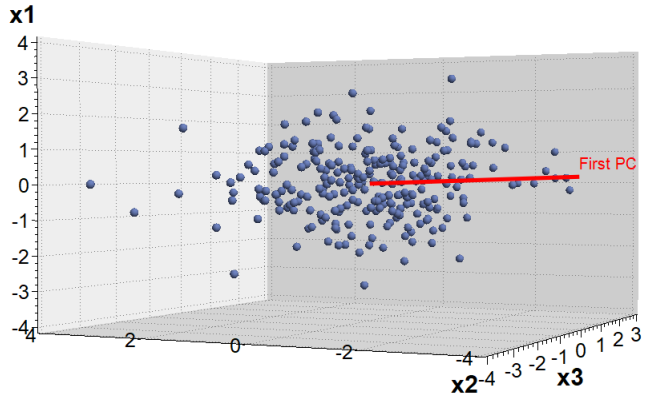


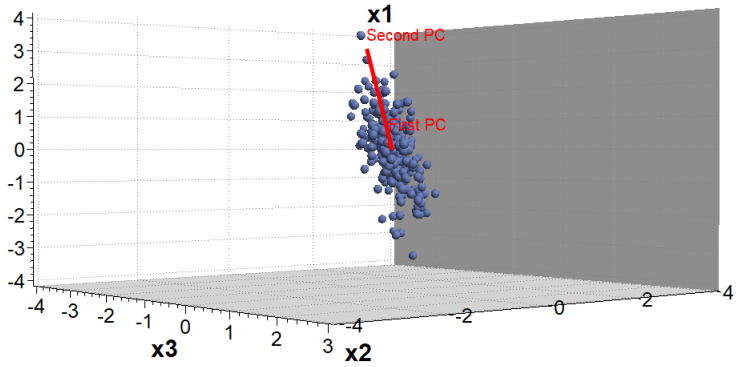
Суть алгоритма группировки переменных

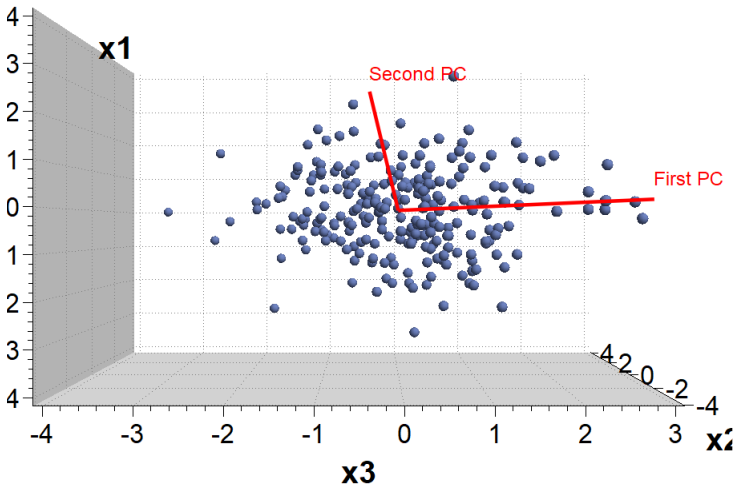
- Нисходящая иерархическая кластеризация, сначала все переменные в одном кластере
- Затем повторяется в цикле процесс:
  1. Выбор кластера (группы коррелирующих переменных) для разбиения
  2. Деление кластера на два с помощью метода гл. комп. с вращением
  3. Перераспределение переменных по кластерам

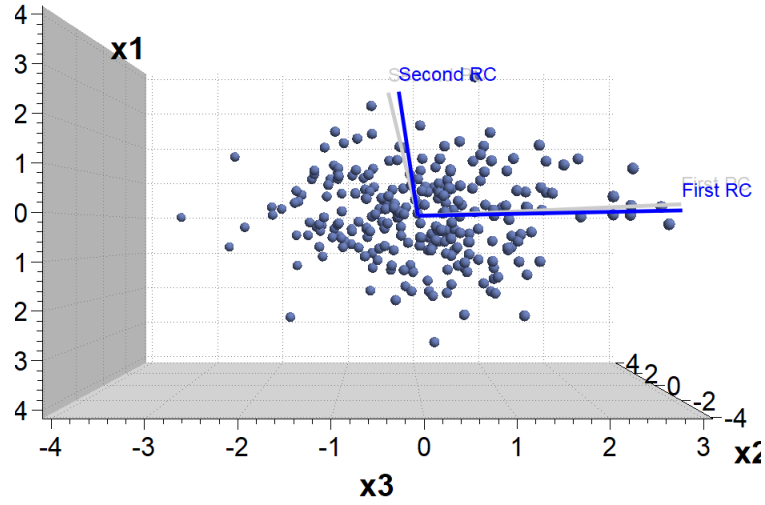


Шаг разбиения





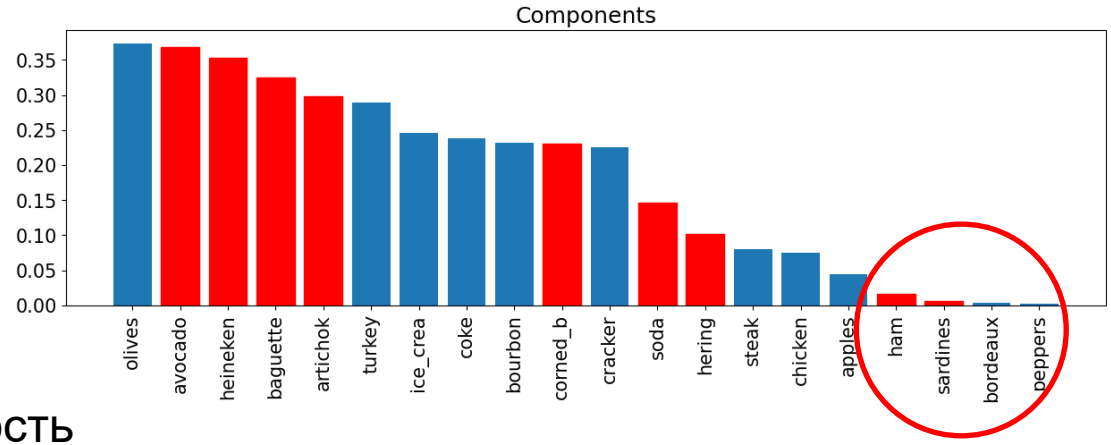




Поворот главных компонет

Поворот главных компонент

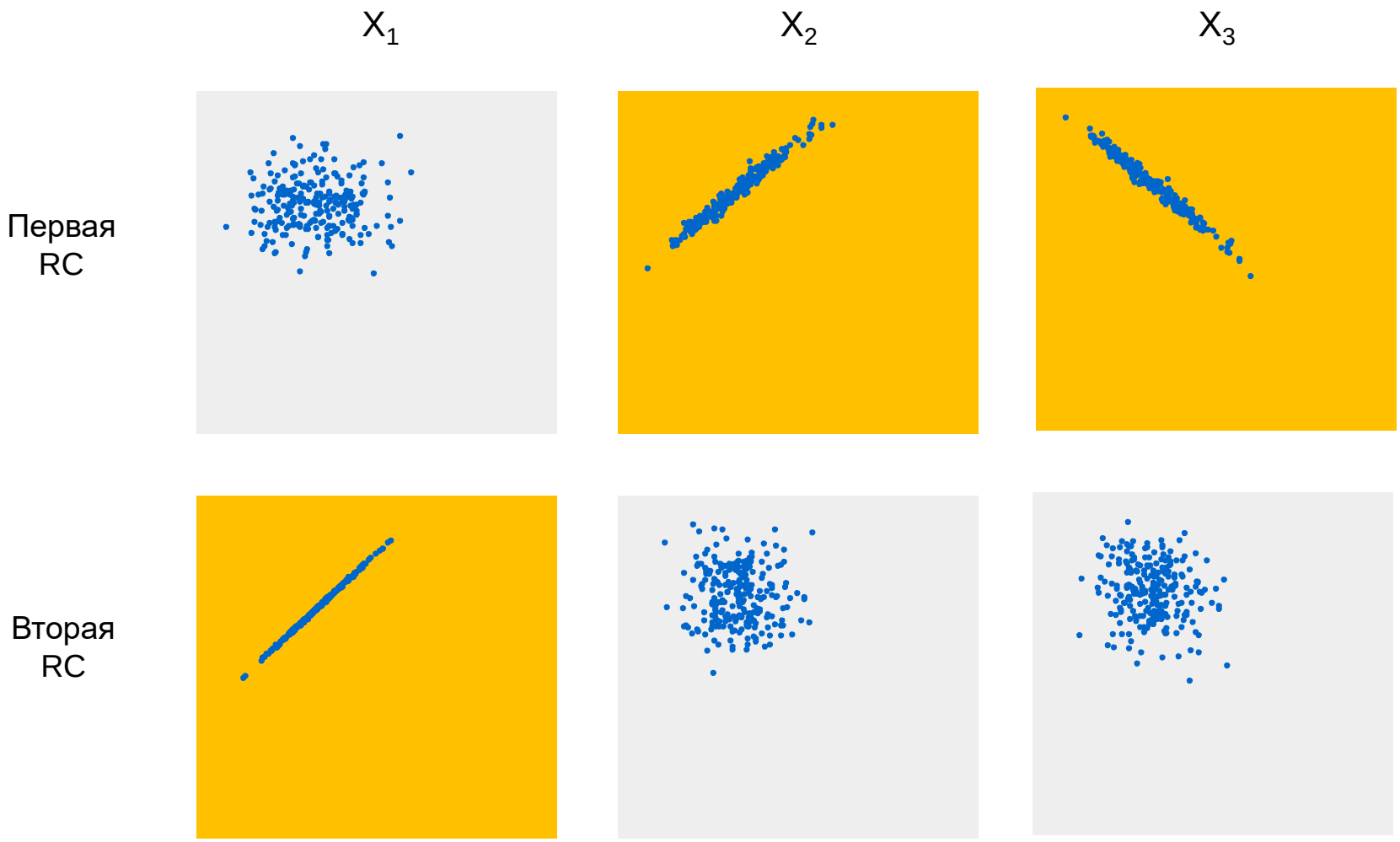
- Проблема PCA - много близких к 0 коэффициентов у переменных, плохая интерпретируемость
- Цель поворота:
  - Перейти к новому максимально близкому базису с большим числом 0 весов (увеличить определенность) и улучшить интерпретируемость



- VARIMAX:
  - максимизирует сумму квадратов корреляций между переменными и факторами (гл. компонентами), осуществляя поворот исходной матрицы факторов $V_{p\times k}$ с помощью матрицы $R_{VARIMAX}$, такой что:

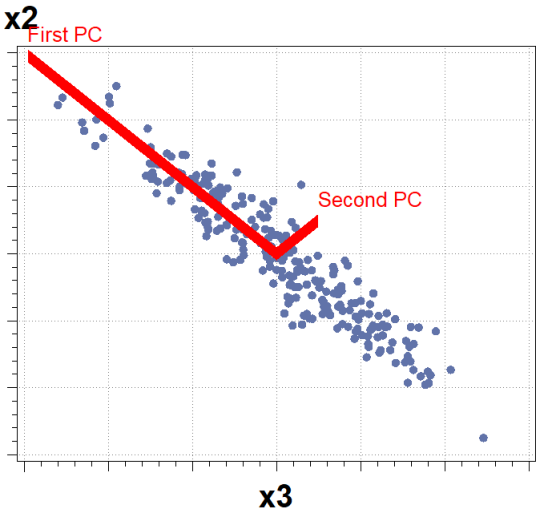
$$R_{VARIMAX} = \underset{R}{\operatorname{argmax}} \left(\frac{1}{p} \sum_{j=1}^{k} \sum_{i=1}^{p} (VR)^4{}_{ij} - \sum_{j=1}^{k} \left(\frac{1}{p} \sum_{i=1}^{p} (VR)^2{}_{ij} \right) \right)$$

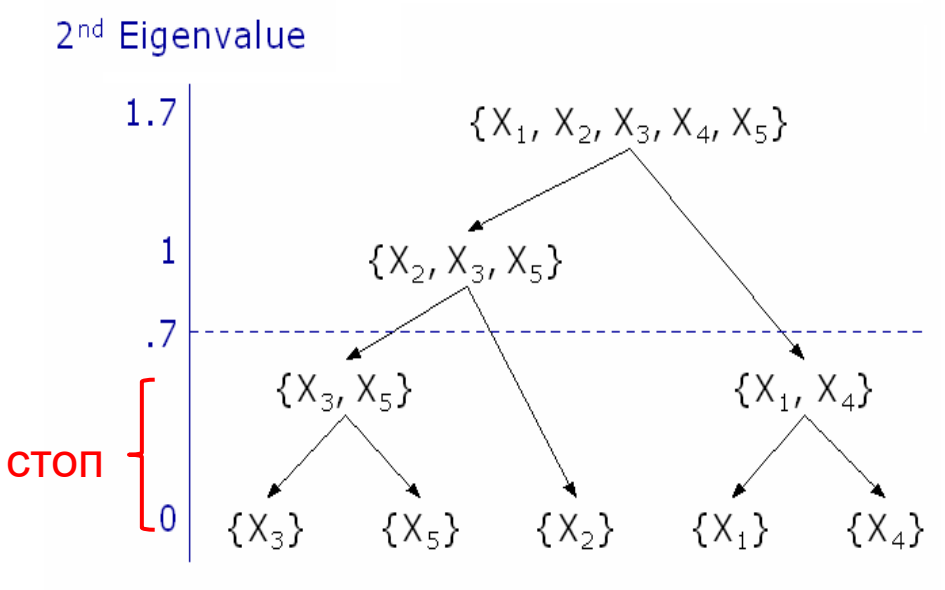
Перераспределение переменных



Разбивать ли дальше и какой кластер выбрать?

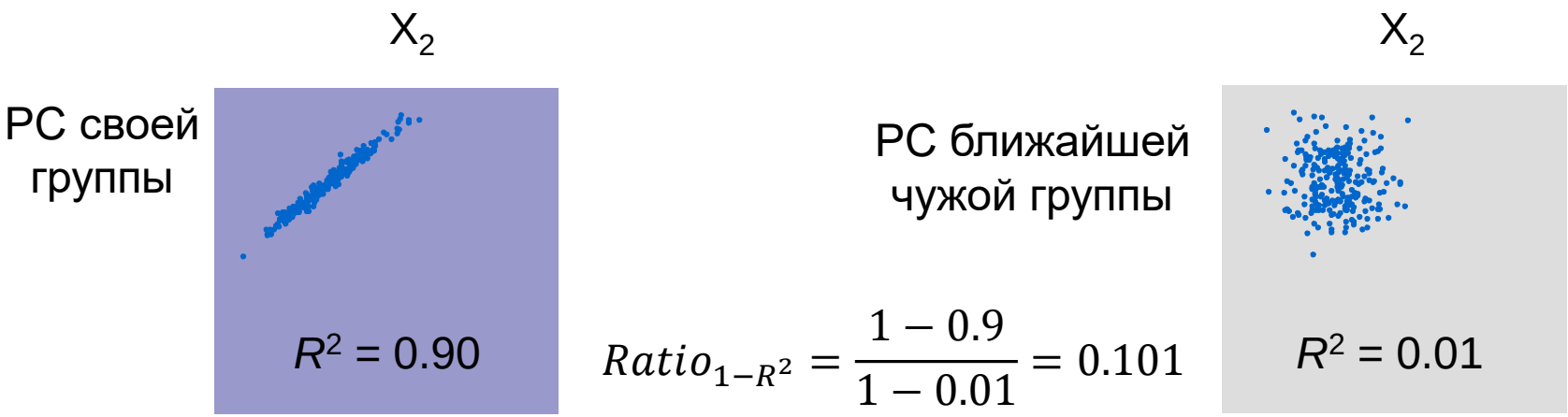
Первое с
Второе с





- Варианты критерия остановки:
  - Задать порог на допустимое второе с.зн.
  - Задать максимально допустимое число кластеров
  - Задать необходимый минимум описанной вариации – сумма первых с.зн по всем листьям, деленных на размерность пространства признаков

Выбор представителей кластеров



- Лучший представитель это переменная, которая:
  - максимально хорошо объясняет разброс в своей группе – максимально коррелирующая с первым с.в. своего кластера
  - максимально плохо объясняет разброс в любой другой группе – минимально значение максимальной корреляции с первым с.в. по всем остальным кластерам
  - Отношение $1 - R^2$:

$$Ratio_{1-R^2} = \frac{1 - R^2_{own\ cluster}}{1 - R^2_{next\ closest}}$$

Пример использования

```
from varclushi import VarClusHi # varclushi package
```

```
clusters = VarClusHi(assoc, maxeigval2=1, maxclus=None) # assoc has products as columns
clusters.varclus()
```

```
clusters.rsquare
```

	Cluster	Variable	RS_Own	RS_NC	RS_Ratio
0	0	artichok	0.331882	0.073638	0.721228
1	0	avocado	0.542912	0.093912	0.504463
2	0	baguette	0.325976	0.059756	0.716861
3	0	olives	0.521717	0.110853	0.537912
4	0	turkey	0.269857	0.076499	0.790625
5	0	bourbon	0.277107	0.036053	0.749930
6	1	coke	0.784811	0.083987	0.234919
7	1	ice_crea	0.784811	0.080811	0.234108
8	2	apples	0.360347	0.076063	0.692313
9	2	corned_b	0.463746	0.075951	0.580331

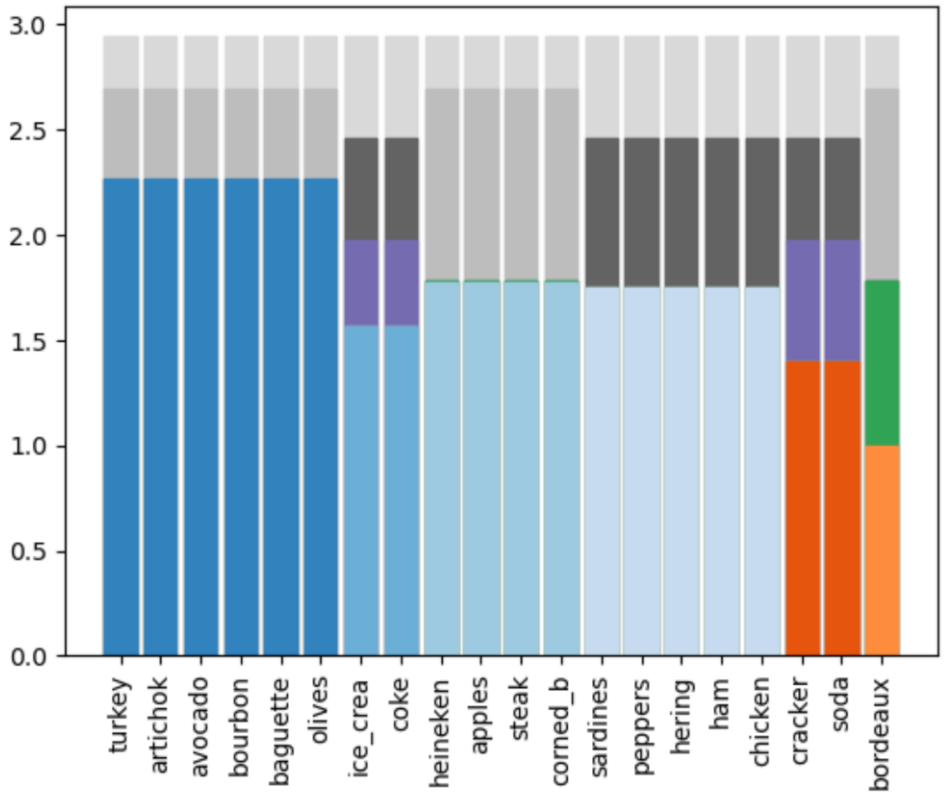
```
clusters.info
```

	Cluster	N_Vars	Eigval1	Eigval2	VarProp
0	0	6	2.269452	0.993312	0.378242
1	1	2	1.569622	0.430378	0.784811
2	2	4	1.775050	0.903608	0.443763
3	3	5	1.747298	0.960767	0.349460
4	4	2	1.401720	0.598280	0.700860
5	5	1	1.000000	0.000000	1.000000

Пример использования

```
# The library doesn't provide the iterations
# So let's collect them manually
hierarchy = list()
variance_explained = dict()
for i in range(len(clusters.info), 0, -1):
    vch = VarClusHi(assoc, maxeigval2=1, maxclus=i)
    vch.varclus()
    for _, x in vch.clusters.items(): # number, cluster
        cluster = frozenset(x.clus)
        hierarchy.append(cluster)
        variance_explained[cluster] = x.eigval1
```

```
# Building linkage matrix for scipy.dendrogram is tedious
# Lets draw it manually
plt.xticks(rotation='vertical')
cmap = plt.cm.tab20c(np.linspace(0, 1, len(hierarchy)))
for i, cluster in enumerate(hierarchy):
    x = list(cluster)
    y = np.ones(len(x)) * variance_explained[cluster]
    for bar in plt.bar(x, y, zorder=-i):
        bar.set_color(cmap[i])
```

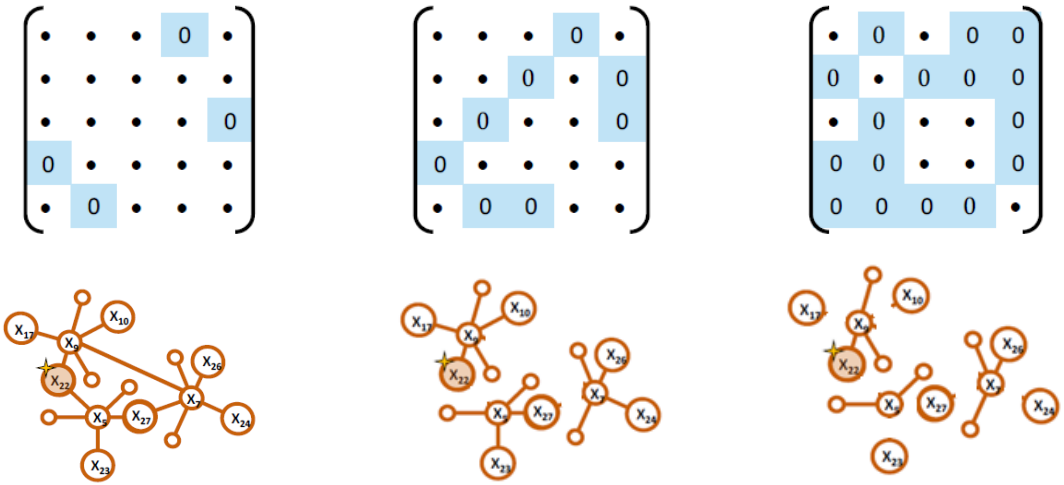


Отбор признаков без учителя в линейных моделях с LASSO

- При обучении линейных моделей с учителем:
  - регуляризация L1 и Elastic Net «отбирает» признаки
- А можно ли без учителя? Да – Graphical LASSO (пусть $x_i \sim N(0, \Sigma)$):

$$\hat{\Theta} = \underset{\Theta \geq 0}{\operatorname{argmin}} \left(\operatorname{tr}(S\Theta) - \log|\Theta| + \lambda \sum_{j \neq k} |\Theta_{jk}| \right)$$

  - $\hat{\Theta}$ - обратная ковариационная матрица, где S - наблюдаемая ковариационная матрицы, λ - параметр регуляризации, чем он больше, тем $\hat{\Theta}$ более разрежена и меньше дуг в графе связей переменных :



Пример

```
from sklearn.covariance import GraphicalLasso
```

Регуляризация

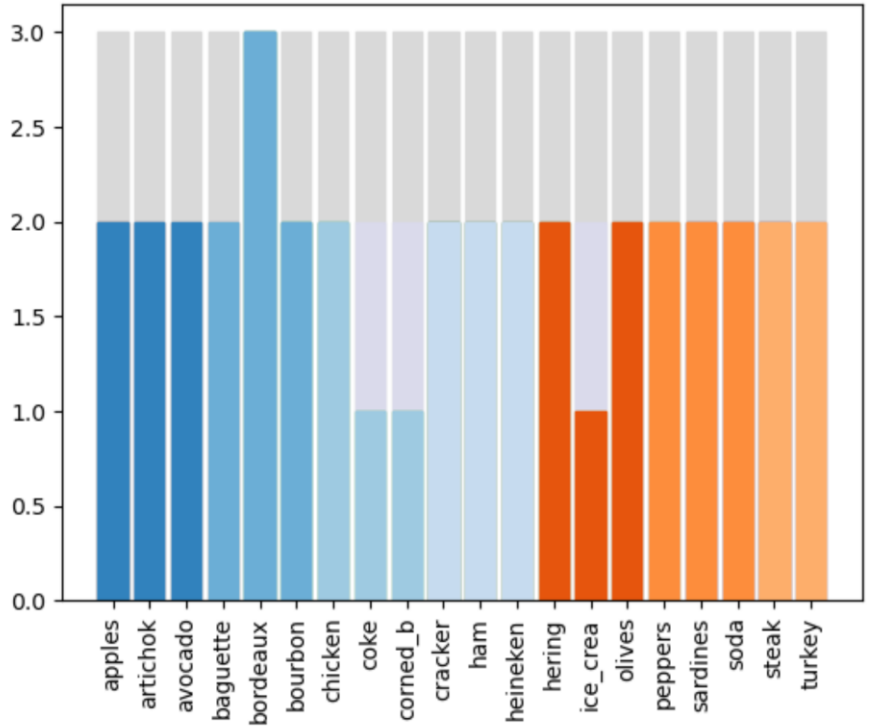
```
cov = GraphicalLasso(alpha=0.01).fit(assoc)
pd.DataFrame(columns=assoc.columns, index=assoc.columns, data=cov.covariance_)
```

PRODUCT PRODUCT	apples	artichok	avocado	baguette	bordeaux	bourbon	chicken	coke	corned_b
apples	0.227114	-0.021079	0.015019	0.018002	0.000000	-0.029258	-0.022309	-0.011946	0.025641
artichok	-0.021079	0.226149	0.087879	0.024196	0.000000	-0.036957	-0.014775	-0.028545	-0.032747
avocado	0.015019	0.087879	0.240205	0.062205	0.000000	-0.064977	-0.049531	-0.053192	-0.049661
baguette	0.018002	0.024196	0.062205	0.238251	0.000000	-0.050282	-0.035487	-0.038649	-0.052448
bordeaux	0.000000	0.000000	0.000000	0.000000	0.068461	0.000000	0.000000	0.000000	0.000000
bourbon	-0.029258	-0.036957	-0.064977	-0.050282	0.000000	0.260165	0.024992	0.027112	0.003663
chicken	-0.022309	-0.014775	-0.049531	-0.035487	0.000000	0.024992	0.222754	0.035286	0.009449
coke	-0.011946	-0.028545	-0.053192	-0.038649	0.000000	0.027112	0.035286	0.213072	-0.043645
corned_b	0.025641	-0.032747	-0.049661	-0.052448	0.000000	0.003663	0.009449	-0.043645	0.240249

Пример

```
from scipy.sparse.csgraph import connected_components
```

```
L = 0.001
RATE = 10
STEPS = 4
hierarchy = list()
step = dict()
for i in range(STEPS, 0, -1):
    cov = GraphicalLasso(alpha=L*STEP**i, max_iter=1000).fit(assoc)
    distance = cov.covariance_ > 0
    # Find clusters for current lambda
    n, clusters = connected_components(distance, directed=False)
    # Repeat the dendrogram process
    # but variance_explained -> step
    for j in range(clusters.max() + 1):
        cluster = frozenset(assoc.columns[clusters==j])
        hierarchy.append(cluster)
        step[cluster] = STEPS - i
```



Восходящая кластеризация переменных - последовательный поиск обратной ковариационной матрицы с домножением параметра регуляризации при каждом шаге

По вертикали – число шагов.