Статистический анализ данных

Лекция 6. Подготовка данных

Московский авиационный институт
«МАИ»

26 сентября 2021 г.

Подготовка данных (Data preprocessing)

1. Фундаментальные свойства данных
2. Виды данных
3. Предобработка данных
   - очистка данных (Data Cleaning)
   - сокращение данных (Data Reduction)
   - трансформация данных (Data Transformation)
   - интеграция данных (Data Integration)

Подготовка данных

More data beats clever algorithms, but better data beats more data (Peter Norvig)

computer scientists	data scientists
работает с алгоритмами	работает с данными
организовывают законы	открывают законы
ошибки — катастрофа	ошибки — естественны
их нанимают, чтобы они делали код	их нанимают, чтобы они находили закономерности

Нулевой этап решения задачи АД

Фундаментальные свойства данных

- **Доступность** (Accessibility)
- **Актуальность** (Timeliness)
- **Ценность** (Value added)
- **Истинность** (Believability)

Фактически это то, что нужно понять, узнать у заказчика

Перечислены не все свойства

Сбор данных (Data Collection)

На что смотреть:

- размеры, размерность, число элементарных порций (объектов), разреженность, разрешение + полнота
- семантика данных, идентификация отдельных элементов и порций данных + id объектов, связи между таблицами и т.п.
- структура данных, режим доступа к данным (online / offline), способ доступа, источник данных

Виды данных

- признаковые описания (матрица объект-признак)
- измерения
  - одномерные сигналы (ряды, звук и т.п.), последовательности, тексты
  - изображения
  - видео
- метрические данные (например, вместо признакового описания сайта измеряем близость между разными сайтами по пересечению их аудитории)
- данные в специальных форматах
  - графы
  - XML-файлы
  - пространственно-временные
  - сырые логи
  - и т.п.

Источники данных

Proprietary data sources	часто нельзя получить доступ
Government data sets	Data.gov
Academic data sets	при написании публикаций
Web search	есть лимиты и условия использования
Sensor data	Относительно дешевы, но специфичны

\+ Ваши данные – самые ценные

Common https://datasetsearch.research.google.com/

Gov https://ukdataservice.ac.uk/

Academic https://library.columbia.edu/services/research-data-services.html

For money https://www.bloomberg.com/professional/datasets/

Подготовка данных

	Свойства данных	**Что мешает этому свойству**	**Причины нарушения свойства**
1	Корректность (точность)	Аномалии «некорректности»	Погрешность приборов, ошибки при заполнении
2	Полнота	Пропуски м.б. разреженность объектов $<$ признаков	Недоступность данных, ошибки при заполнении, сбои при записи
3	Непротиворечивость (согласованность)	«противоречия»	Различные источники данных
4	Безызбыточность	Дубликаты Шум Излишняя дискретизация	Особенности интеграции, ошибки при заполнении
5	Ясность	«неясности»	Плохие хранение и подготовка
6	Структурированность Однородность	Сырые данные	Нет признаковых описаний Признаки в разных шкалах

Средства борьбы

Способ	Свойства
Очистка данных	1, 2
Сокращение данных	2, 4
Интеграция	3
Трансформация	4, 5, 6
Генерация признаков	6

Подготовка данных

Предобработка данных

— замена, модификация или удаление частей набора данных с целью повышения непротиворечивости, полноты, корректности и ясности набора данных, а также уменьшения избыточности

— процесс преобразования данных в форму, удобную для анализа

Выполняется на полном наборе данных
(и на контрольных объектах тоже)

Тонкость: не допустить утечки
(информации, не доступной при функционировании модели)
https://www.kaggle.com/alexisbcook/data-leakage

Что бывает в данных

	дата	пол	образование	сумма	платёжная строка	число просрочек	?????	x_m
0	12/01/2017	1	высшее	5000.0	0000	0	0	0.00000
1	13/01/2017	1	высшее	2500.0	0000	1	1	1.00000
2	13/01/2017	1	высшее	2500.0	001000	1	1	1.00000
3	13/01/2017	0		13675.0	111	3	3	0.00000
4	25/01/2017	0		NaN	0	0	0	0.00000
5		1	начальное	NaN	00	0	0	0.00000
6	02/02/2017	1	среднее	1000.0		0	0	0.00000
7	01/01/0001	13/01/2017	среднее	0.0		-7	-7	-0.00001

Что бывает в данных

	дата	пол	образование	сумма	платёжная строка	число просрочек	?????	x_m
0	12/01/2017	1	высшее	5000.0	0000	0	0	0.00000
1	13/01/2017	1	высшее	2500.0	0000	1	1	1.00000
2	13/01/2017	1	высшее	2500.0	001000	1	1	1.00000
3	13/01/2017	0		13675.0	111	3	3	0.00000
4	25/01/2017	0		NaN	0	0	0	0.00000
5		1	начальное	NaN	00	0	0	0.00000
6	02/02/2017	1	среднее	1000.0		0	0	0.00000
7	01/01/0001 13/01/2017		среднее	0.0		-7	-7	-0.00001

неясность

дубликаты

некорректность

ошибка

нечисловой признак

пропуски

дубликаты

выброс

Разделы предобработки данных

Очистка данных (Data Cleaning)

Обнаружение и удаление / замена

- аномалий / выбросов (Anomaly detection)
- пропусков (Missing data imputation)
- шумов (Noise identification)
- некорректных значений (Filter incorrect data)

Сокращение данных (Data Reduction)

- сэмплирование (Sampling)
- сокращение размерности (Dimensionality reduction)
- отбор признаков (Feature subset selection)
- отбор объектов (Instance selection)
- удаление дубликатов

Разделы предобработки данных

Трансформация данных (Data Transformation)

- Переименование признаков, объектов, значений признаков, преобразование типов
- Кодирование значений категориальных переменных (отдельная лекция)
- Дискретизация (Discretization / Binning)
- Нормализация
- Сглаживание
- Создание признаков (отдельная лекция)
- Агрегирование
- Обобщение
- Деформация значений

Интеграция данных (Data Integration)

- Объединение данных из разных источников

Переименования

Названия переменных (и их значения ?!) должны быть интуитивны

Они используются в том числе при передачи данных коллегам, презентации результатов и т.п.

	X5XX.	X5XV.	price($)	date
0	200$	Jan.1.2018	200	2018-01-01
1	150$	Feb.13.2017	150	2017-02-13
2	7000$		7000	NaT
3	110$	Jun.13.1996	110	1996-06-13

Преобразования типов данных

Нужно использовать типы, которые поддерживает Ваша среда программирования

```
df.rename(columns={'X5XX.': 'price($)'}, inplace=True)
df['price($)'] = df['price($)'].apply(lambda x: int(x[:-1]))
# по другому
df['price($)'] = df['price($)'].apply(lambda x: x.replace('$', '')).astype(int)
df['date'] = pd.to_datetime(df['X5XV.'], errors='coerce')
# как еще можно?
```

Кодировки

Как правило, компьютер работает с числами ⇒ категории представляем числами (векторами)

	ans	weather	ans_coded	weather_coded
0	yes	warm	1	0
1	no	cool	0	1
2	yes	cold	1	2
3	no	warm	0	0

```
dct = {'yes': 1, 'no': 0}
df['ans_coded'] = df['ans'].map(dct)
# быстрее?!
df['weather_coded'] = df.weather.factorize()[0]
```

Если алгоритм позволяет работать с исходными категориями, то можно не кодировать

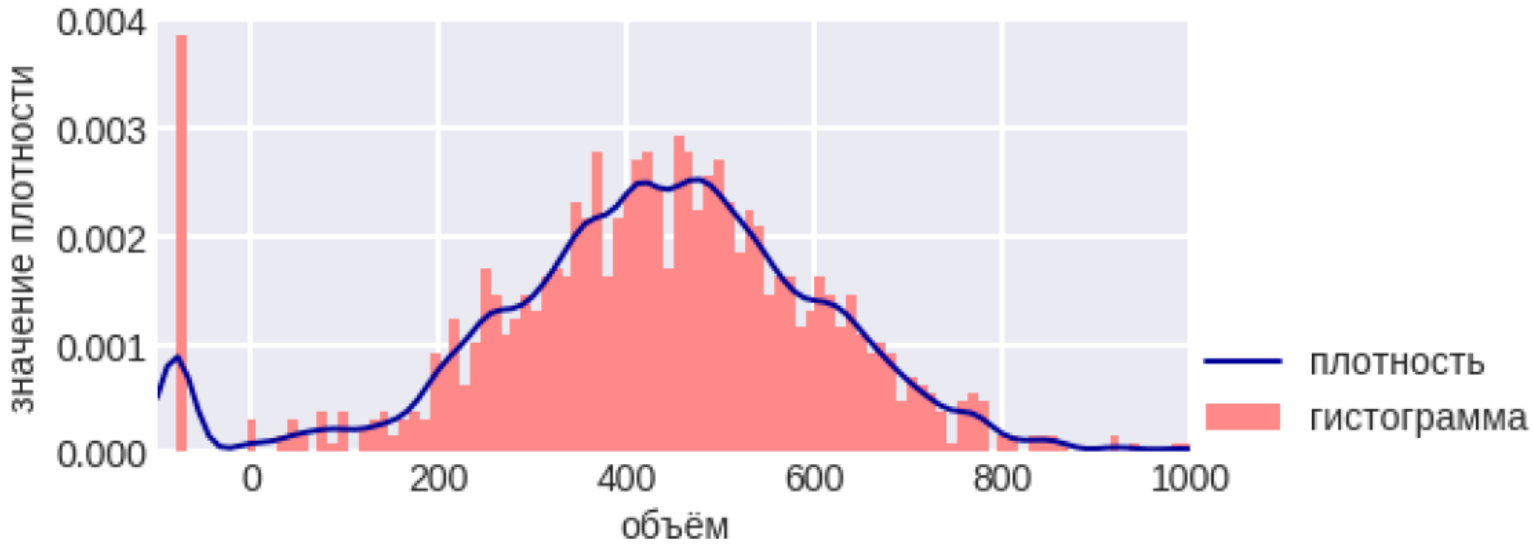
Корректировка значений

	время	давление	в.давл.	н.давл.	время
0	23:10	120/80	120	80	2018-09-13 23:10:00
1	10 часов	120/70	120	70	2018-09-13 10:00:00
2	7:40	110/70	110	70	2018-09-13 07:40:00

```
tmp = df['давление'].str.split('/')
df['в.давл.'] = tmp.apply(lambda x: x[0])
df['н.давл.'] = tmp.apply(lambda x: x[1])
# быстрее
df[['в.давл.', 'н.давл.']] = pd.DataFrame(df['давление'].str.split('/',
                          1).tolist(), columns = ['Давление_в','Давление_н'])
# тоже быстрее
df[['в.давл.', 'н.давл.']] = df['Давление'].str.split('/', expand=True)
# очень быстро, если нет ошибок в данных
st = '/'.join(df['Давление'])
df[['в.давл.', 'н.давл.']] = pd.DataFrame(np.array(st.split('/')).reshape(-1, 2))
```

Пропуски — как выглядят в данных

- пустые значения
- специальные значения (NA, NaN, null, ...)
- специальный код (–999, mean, число за пределами значения признака)




```
df[name].isnull().sum() # число «нанов»
df[name].count() # число не «нанов»
```

Пропуски — что делать

- **оставляем**

  (но не все модели могут работать с пропусками)

- **удаляем описания объектов с пропусками / признаки**

  (радикальная мера, которая редко используется)

  df.dropna(how='any', axis=1)

- **заменяем на фиксированное значение**

  (например, если признак бинарный, то на 0.5,
  значение –999, как правило, плохое — является выбросом)

  df.fillna(-1)

- **заменяем на легко вычислимое значение**

  (среднее, медиана, мода)

  df.fillna(df.mean())

Пропуски

- **восстановление значения**

  (построение специальной модели для восстановления)

```
from sklearn.preprocessing import Imputer
imputer = Imputer(missing_values='NaN',
                  strategy='mean', axis=0)
vals = imputer.fit_transform(df[['сумма']])
```

- **экспертная замена**
- **решаем отдельную задачу** для восстановления пропущенных значений признака, объявляя его целевой переменной

Пропуски — итеративная процедура (IterativeImputer)

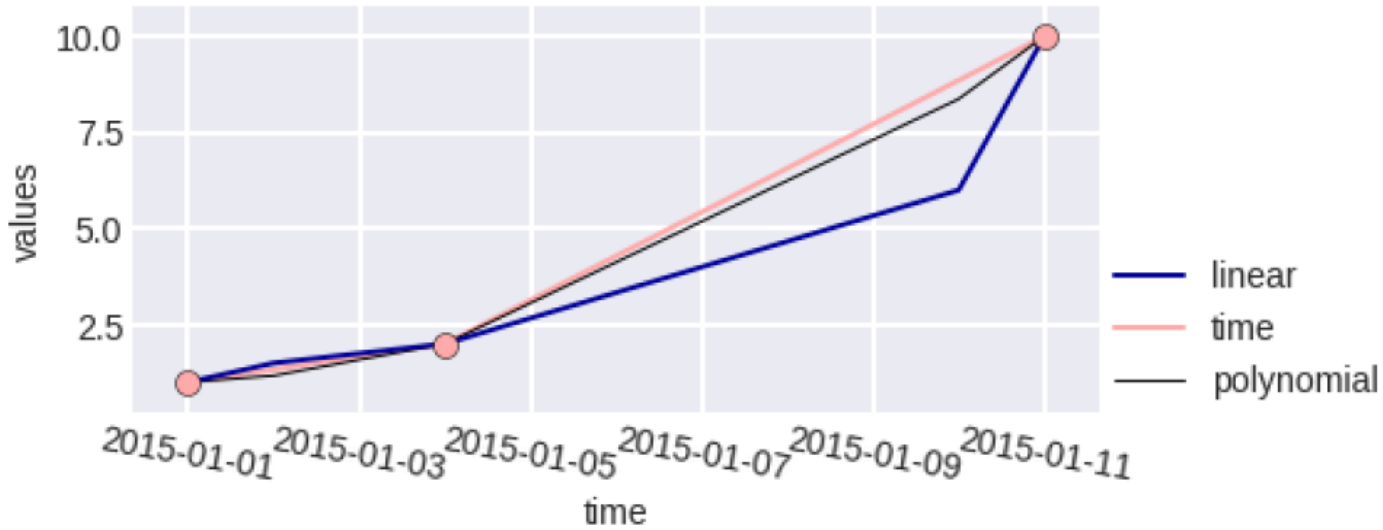
https://scikit-learn.org/stable/modules/impute.html

Хорошо добавить признак, который считает число пропусков для объекта (можно отдельно для каждого признака)

Пропуски в последовательностях, сигналах

	ts	ts2	ts3	ts4
2015-01-01	1.0	1.0	1.00	1.00
2015-01-02	NaN	1.5	1.33	1.17
2015-01-04	2.0	2.0	2.00	2.00
2015-01-10	NaN	6.0	8.86	8.37
2015-01-11	10.0	10.0	10.0	10.0

```
ts = pd.Series(vals, index=index)
ts2 = ts.interpolate()
ts3 = ts.interpolate(method='time')
ts4 = ts.interpolate(method='polynomial',
order=2)
```



Пропуски — тонкости

- **добавлять характеристический признак пропусков «is_nan»**
  тогда модель сама определит оптимальное значение для заполнения
- **заполнять пропуски лучше после генерации признаков**
  иначе возникают дополнительные неопределённости
- **не допускать ликов при заполнении пропусков**
  Общий пайплайн: предобработка данных + классификация
  при заполнении пропусков нельзя брать информацию из будущего
  Пример: нельзя считать среднее по всей выборке, включая тестовую

Пропуски

исходные

	площадь	цена	цена/кв.м.
0	82.0	5200000.0	63414.634146
1	70.0	4400000.0	62857.142857
2	74.0	4200000.0	56756.756757
3	60.0	NaN	NaN
4	NaN	3600000.0	NaN
5	NaN	NaN	NaN

перед созданием

	площадь	цена	цена/кв.м.
	82.0	5200000.0	63414.634146
	70.0	4400000.0	62857.142857
	74.0	4200000.0	56756.756757
	60.0	4350000.0	72500.000000
	71.5	3600000.0	50349.650350
	71.5	4350000.0	60839.160839

после создания

	площадь	цена	цена/кв.м.
	82.0	5200000.0	63414.6
	70.0	4400000.0	62857.1
	74.0	4200000.0	56756.8
	60.0	4350000.0	61009.5
	71.5	3600000.0	61009.5
	71.5	4350000.0	61009.5

```
df['цена/кв.м.'] = df['цена'] / df['площадь']
df.fillna(df.mean())
```

Пропуски

Важно понимать природу пропуска:

- **значение может не быть доступно**
  Пример: клиент банка не указал в анкете свой возраст
  отсутствие информации – тоже информация!
- **значение может не существовать**
  Пример: «Доход» для детей моложе 16 $\Rightarrow = 0$
- **значение не является числом**
  Пример: деление на ноль $0/0 = \text{NaN}$ (средняя покупка в категории товаров)
- **значение вызвано предобработкой данных**
  Пример: при конкатенации таблиц — несуществующие колонки
- **характер (распределение) пропусков на обучении и тесте должен быть одинаковый**
  Пример: изменение модели датчиков и уменьшение числа пропусков

Пропуски

Можно посмотреть, зависит ли факт пропуска от других данных

	data	площадь	target
0	train	82.0	0
1	train	NaN	1
2	train	74.0	0
3	test	60.0	0
4	test	NaN	1
5	test	50.0	0

целевой признак — характеристический признак пропуска

выкидываем «площадь» и решаем задачу для target, который учитывает пропуски

$$df['target'] = df['площадь'].isnull().astype(int)$$

Шум

Зашумлённые данные (Noisy Data)

Что делать (аналогия с пропусками)

- **оставляем** (но будет погрешность при моделировании)
- **удаляем сильно зашумлённые признаки**
- **удаляем сильно зашумлённые объекты**
- **замена аномальных значений**

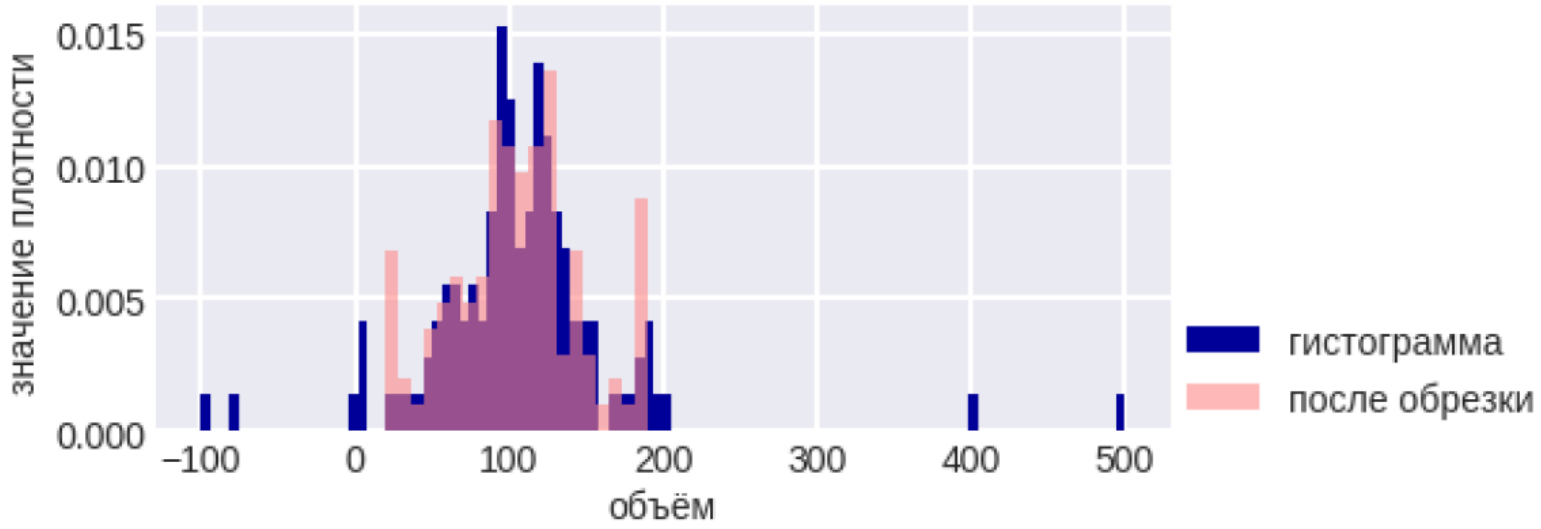
могут нести важную информацию!

Главный вопрос: «Почему в данных есть это?»

Причины

- **ошибка сбора данных** (погрешность прибора, ввода и т.п.)
- **ошибка обработки данных**
- **свойство данных** (выброс — зарплата CEO)

Винсоризация (Winsorizing)



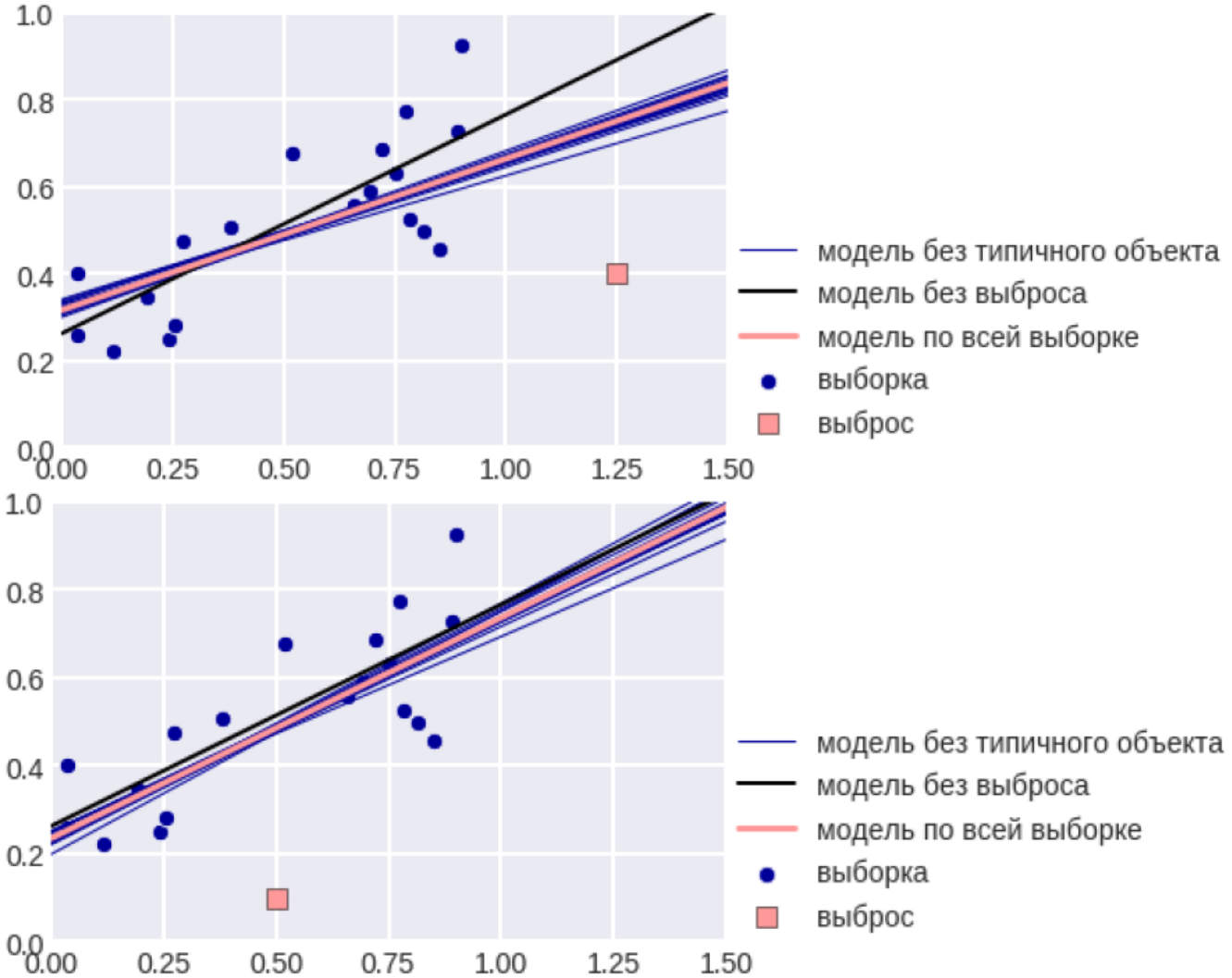

```
x2 = df.x.clip(lower = df.x.quantile(0.05),
upper = df.x.quantile(0.95))

import scipy
x3 = scipy.stats.mstats.winsorize(x, limits = 0.05)
```

Зашумлённые данные — Тонкости

- если есть шум, можем ли доверять и другим признакам
- выбросы в обычном признаке / целевом
- плохо для линейных моделей, но есть модели «устойчивые к выбросам»

Пример: прогноз количества покупок товара в интернет-магазине, пришёл оптовик за покупками и мы получаем выброс

В чём разница между этими выбросами?



Второй случай чаще встречается на практике

Агрегирование (Aggregation)

f_1	f_2	f_3	f_4	f_5	f_mean	f_std	f_max	f_min
14	15	18	23	10	16.0	4.85	23	10
36	13	14	21	16	20.0	9.46	36	13
10	14	16	17	20	15.4	3.71	20	10
13	20	15	25	13	17.2	5.22	25	13

Слагаемые, например, замеры разными датчиками и т.п.

```
df['f_mean'] = df[cols].mean(axis=1)
df['f_std'] = df[cols].std(axis=1) #.round(2)
df['f_max'] = df[cols].max(axis=1)
df['f_min'] = df[cols].min(axis=1)
```

Иногда новые признаки просто сортировка значений в строке

Обобщение (Generalization)

	товар	group_1	group_2
0	стол	офис	дерево
1	тетрадь	офис	бумага
2	горшок	дом	пластик
3	стул kv-15	дом	пластик

Создание новых описательных признаков

Применяется редко из-за трудоемкости и требования знания предметной области

Интеграция данных (Data Integration)

Обычно — из разных источников

	client	date	contract
0	1001	12.01.05	200547
1	1002	14.01.05	200545
2	1003	15.01.05	200558
3	1004	16.01.05	200565

	client	age	account
0	1001	34	12000
1	1002	52	0
2	1003	25	10000
3	1004	33	NaN

	contract	sum
0	200547	100000
1	200565	200000

	client	date	contract	age	account	sum
0	1001	12.01.05	200547	34	12000	100000
1	1002	14.01.05	200545	52	0	200000
2	1003	15.01.05	200558	25	10000	
3	1004	16.01.05	200565	33	NaN	

```
df.merge(df2, how = 'left').merge(df3, how = 'left')
```

Интеграция данных

Анкета

id	пол	возраст	сумма	карт
12	М	34	10000	0
15	М	23	50000	1
37	Ж	37	90000	2

БКИ

id	дата	сумма	delay
12	10.11.12	1000	0
12	01.02.13	2000	1
15	19.10.11	1000	0
15	05.03.12	2000	0
15	03.07.13	3000	1
15	09.09.13	2000	0
37	23.11.13	5000	1

сумма + веса

среднее

максимум

минимум

медиана

Использование интеграции или нет

id	user	target
0	1	0
1	3	1
2	6	0
3	7	0
4	8	1

id	user	activity
0	1	10.0
1	1	20.5
2	3	10.4
3	3	18.0
4	3	3.0

На уровень транзакций

id	user	activity	target
0	1	10.0	0
1	1	20.5	0
2	3	10.4	1
3	3	18.0	1
4	3	3.0	1

Агрегаты

id	user	ac_mean	ac_std	ac_max	ac_min	target
0	1	15.25	7.42	20.5	10.0	0
1	3	10.47	7.50	18.0	3.0	1
2	6	9.83	7.52	17.0	2.0	0
3	7	7.25	3.89	10.0	4.5	0
4	8	9.00	12.73	18.0	0.0	1

Интеграция

Использование интеграции или нет

Агрегаты

```
tmp = data2.groupby('user',)['activity']
tmp = tmp.agg('ac_mean':mean, 'ac_std':std, 'ac_max':max,
              'ac_min':min)
data = data.merge(tmp.reset_index(), on='user')
```

На уровень транзакций

```
data2.merge(data, on='user').head()
```

Второй способ, конечно, менее эффективен, но помогает при блендинге результатов с первым способом.

Нормировки (Data Normalization)

Для большинства алгоритмов машинного обучения необходимо, чтобы все признаки были вещественными и «в одной шкале»

- **Стандартизация** (Z-score Normalization / Variance Scaling)

$$\{u_i\}_{i\in I} \to \left\{ \frac{u_i - \operatorname{mean}\{u_t\}_{t\in I}}{\operatorname{std}\{u_t\}_{t\in I}} \right\}_{i\in I}$$

- **Нормировка на отрезок** (Min-Max Normalization)

$$\{u_i\}_{i\in I} \to \left\{ \frac{u_i - \min\{u_t\}_{t\in I}}{\max\{u_t\}_{t\in I} - \min\{u_t\}_{t\in I}} \right\}_{i\in I}$$

- **Нормировка по максимуму**

$$\{u_i\}_{i\in I} \to \left\{ \frac{u_i}{\max\{u_t\}_{t\in I}} \right\}_{i\in I}$$

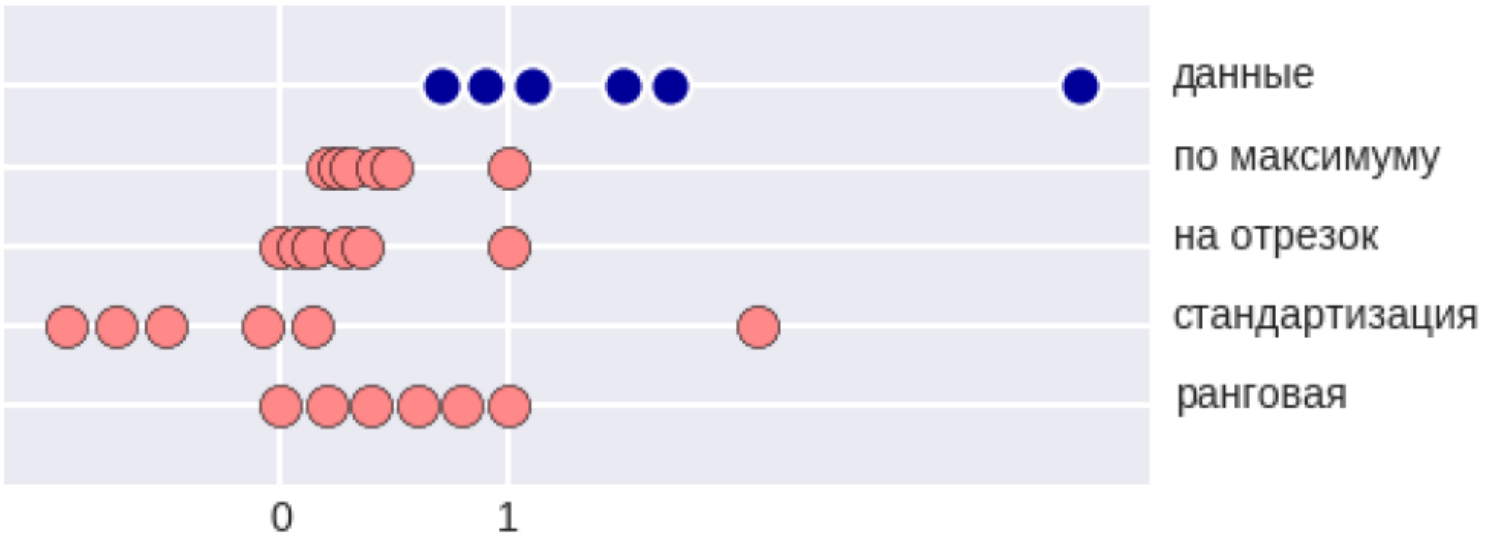
Нормировка

- **Decimal Scaling Normalization**

$$N_{\text{ds}}(x) = \frac{x}{10^{\min\{i:10^i>x\}}}$$

суть: 17,87 $\to$ 0,1787, используется редко

- **Ранговая нормировка** (tiedrank, rankdata)
1; 2; 3; 4; 10 $\to$ 1/5; 2/5; 3/5; 4/5; 5/5



Нормировка

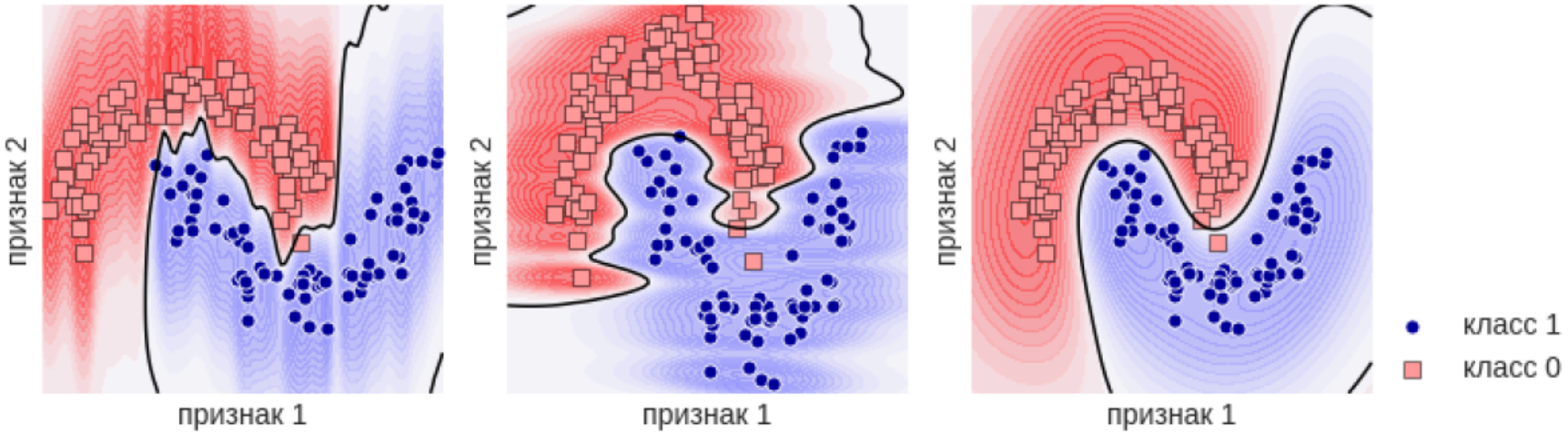
```
X = X - np.min(X)
X = X / (np.max(X) - np.min(X))

X = X - np.mean(X)
X = X / np.std(X)

import sklearn.preprocessing as prp

X['minmax'] = prp.minmax_scale(X['name'])

X['standart'] =
preproc.StandardScaler().fit_transform(X[['name']])
```

Ранговая нормировка (разные способы)

id	data	average	min	max	dense	ordinal
0	1	1.5	1	2	1	1
1	2	4.0	3	5	2	3
2	2	4.0	3	5	2	4
3	5	6.0	6	6	3	6
4	2	4.0	3	5	2	5
5	1	1.5	1	2	1	2
6	10	7.0	7	7	4	7

```
import scipy.stats as ss
from scipy.stats import rankdata
for method in ['average', 'min', 'max', 'dense', 'ordinal']:
df[method] = ss.rankdata(df.feat, method = method)
```

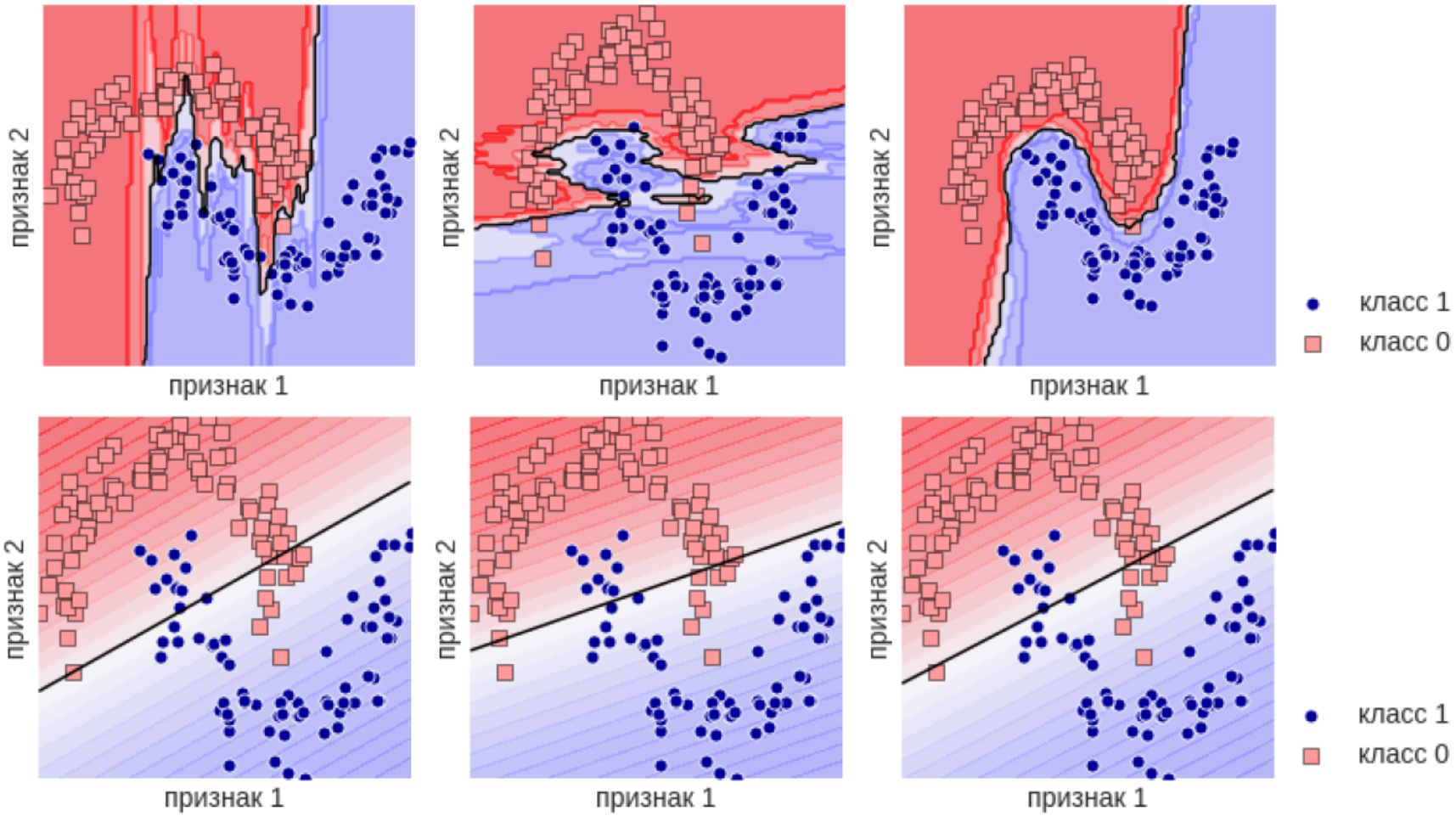
Зависимость моделей от масштаба



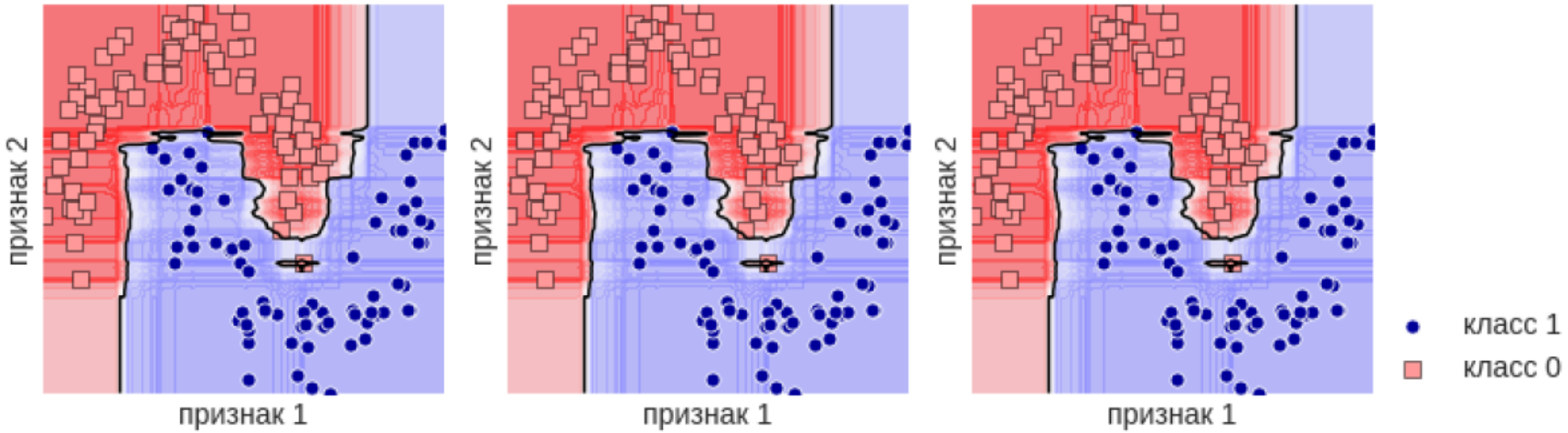
«признак 1» $\times$ 10, «признак 2» $\times$ 10, нормальный масштаб

Нормировка

Модели по-разному реагируют на изменение масштаба по признакам



Нормировка

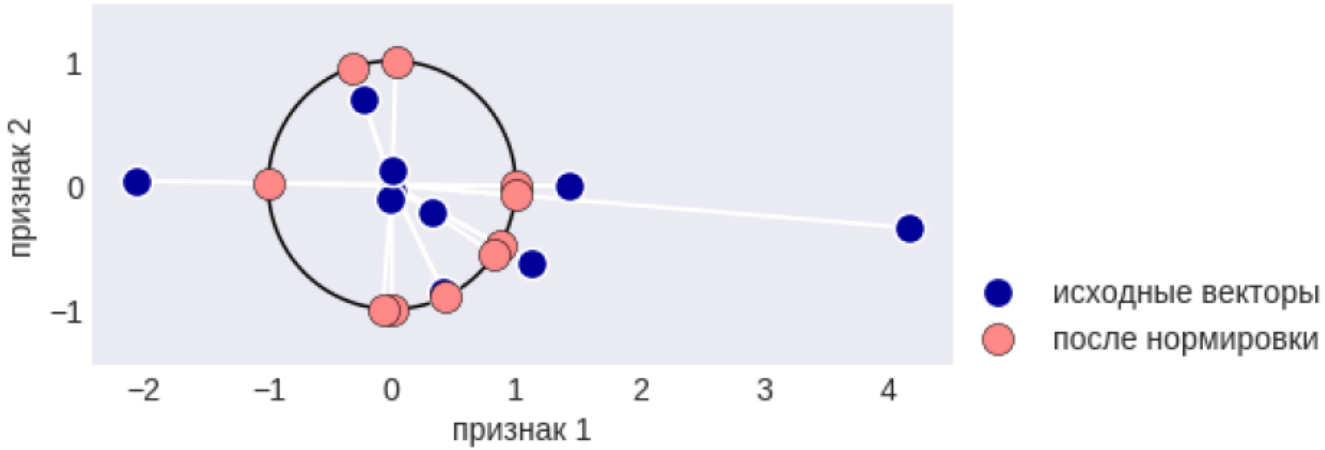


Нормировки — тонкости

- общий пайплайн: предобработка + классификация
  https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html
- параметры нормировок вычисляются по выборке
  использовать все данные не всегда корректно
- если вычислить параметры на обучении, то на контроле может не быть желаемого эффекта
  Пример: выход за пределы [0,1]

Нормировка

Если данные имеют смысл векторов (признаки однородны), то можно нормировать вектора




```
import sklearn.preprocessing as preproc
nrm = preproc.Normalizer()
X2 = nrm.fit_transform(X)
```

тонкости:

- не центрируйте разреженные данные

Нормировки в пределах группы

id	gr	sum	std
0	alpha	1	0.00
1	beta	4	1.12
2	alpha	0	−1.00
3	beta	0	−0.80
4	beta	1	−0.32
5	alpha	2	1.00

```
z_score = lambda x: (x - x.mean()) / x.std()
df['stand'] = df.groupby('gr').transform(z_score)
```

Трансформации

Box-Cox Transformation положительного признака

$$f_\lambda(x) = \begin{cases} \dfrac{x^\lambda - 1}{\lambda}, & \lambda > 0, \\ \ln(x) & \lambda = 0. \end{cases}$$

Как правило применяют, чтобы распределение признака стало похожим на нормальное, можно оценивать схожесть распределения преобразованных данных и нормального

```
x2 = np.log1p(x) # с предварительным +1
x2 = np.log(x)
```

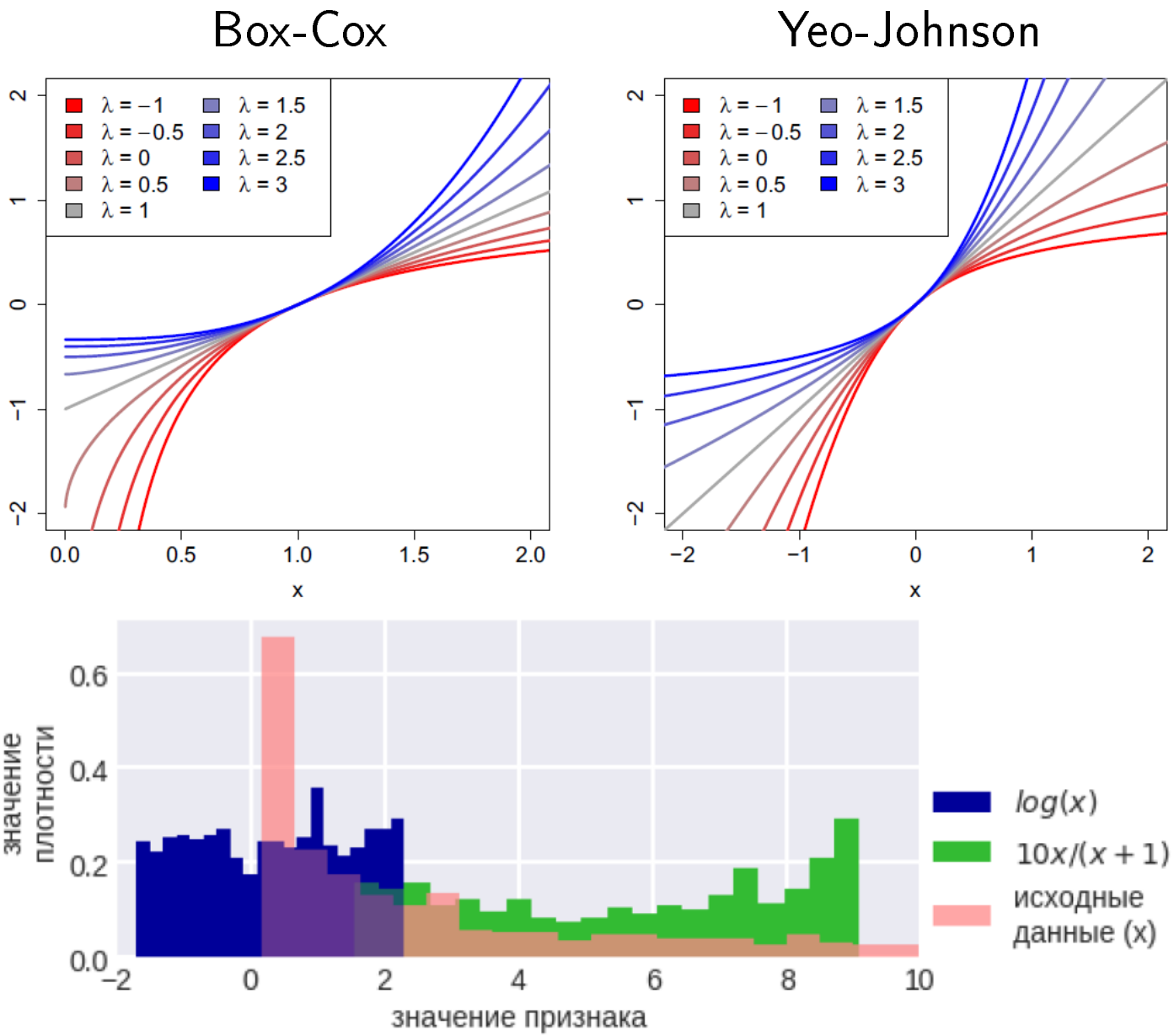
Преобразование Йео-Джонсона (и для $x < 0$)

$$\mathrm{yd}_\lambda(x) = \begin{cases} \dfrac{(x+1)^\lambda - 1}{\lambda}, & \lambda \neq 0,\ x \geq 0, \\ \log(x+1) & \lambda = 0,\ x \geq 0, \\ -\log(-x+1) & \lambda = 2,\ x \leq 0, \\ \dfrac{(-x+1)^{2-\lambda} - 1}{\lambda - 2}, & \lambda \neq 2,\ x \leq 0. \end{cases}$$

https://academic.oup.com/biomet/article/87/4/954/232908

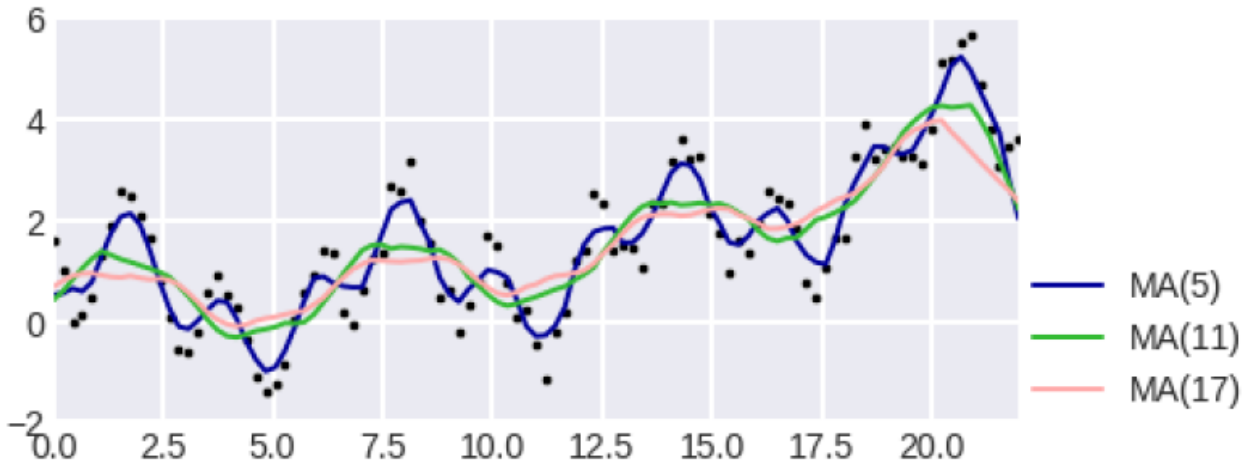
https://www.kaggle.com/kenmatsu4/yeo-johnson-conversion-was-applied

Трансформация



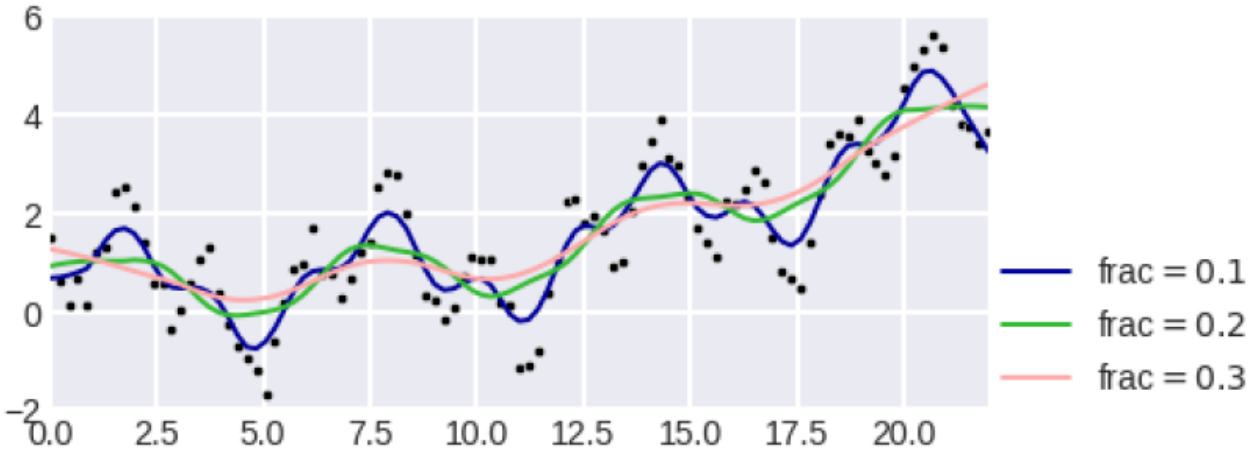
Сглаживание (Smoothing)

Moving Average




```
def smooth(y, box_pts):
        box = np.ones(box_pts)/box_pts
        y_smooth = np.convolve(y, box, mode='same')
        return y_smooth

y_MA = smooth(y, 5)
```

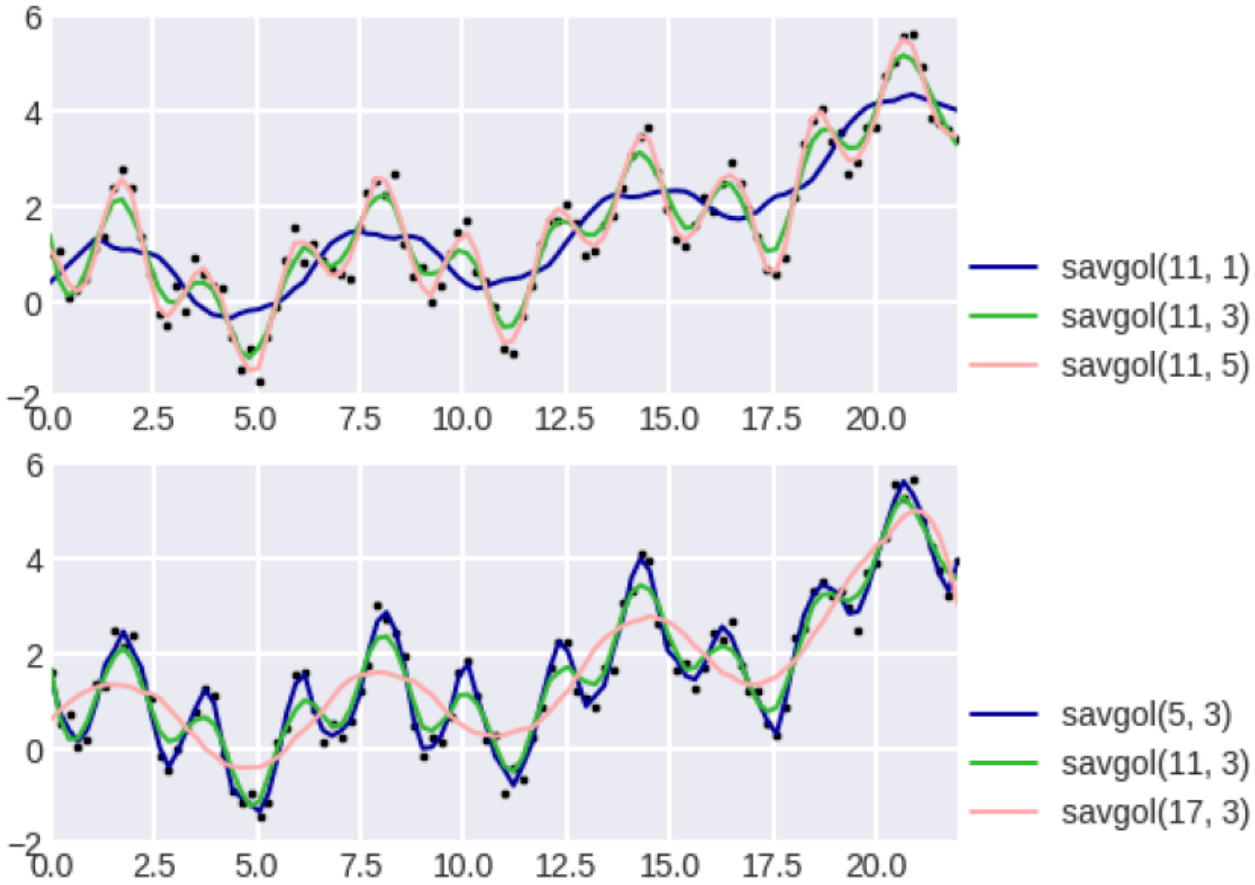
LOWESS (locally weighted scatterplot smoothing)




```
import statsmodels.api as sm
lowess = sm.nonparametric.lowess(y, x, frac=0.1)[:1]
```

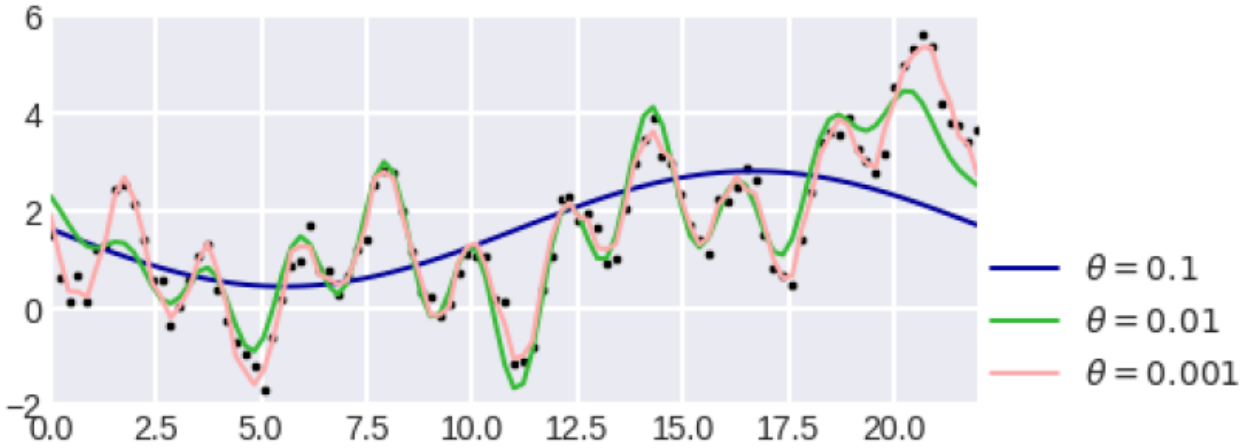
подгоняет полиномом определённой степени в окрестности x

Savitzky–Golay filter




```
from scipy.signal import savgol_filter y_sg = savgol_filter(y, 11, 2)
```

Преобразование Фурье (Fast Fourier Transform)




```
import scipy.fftpack
N = len(y)
w = scipy.fftpack.rfft(y)
spectrum = w**2

theta = 0.1
w2 = w.copy()
w2[spectrum < (theta*spectrum.max())] = 0
y2 = scipy.fftpack.irfft(w2)
```

Можно просто: заменять выбросы на сглаженные значения

Дискритизация

Дискретизация (биннинг, квантование)

— переход от вещественного признака к порядковому за счёт кодирования интервалов одним значением

Улучшает интерпретацию

Позволяет решать задачу простыми алгоритмами (качество ухудшается)

categories	counts	freqs
ребенок	8	0.16
юноша	9	0.18
молодой человек	7	0.14
мужчина	26	0.52
пожилой	0	0.00

```
bins = pd.cut(df[name], 5)
points = [0, 12, 18, 25, 50, 100]
labels = ['ребёнок', 'юноша', 'молодой
человек', 'мужчина', 'пожилой']
factors = pd.cut(ages, points,
labels = labels)
factors.describe()
```

Способы дискретизации

Equal-width (distance) partitioning

Делим область значения признаков на области-интервалы равной длины м.б. в другой шкале, например, в логарифмической

categories	counts	freqs
$(5.96, 15.0)$	14	0.2000
(15.0,24.0)	21	0.3000
$(24.0, 33.0)$	15	0.2143
$(33.0, 42.0)$	20	0.2857

factors = pd.cut(ages, 4)

Equal-width partitioning

\+ простая быстрая реализация

– неравномерные бины

Есть средства бинаризации:

```
from sklearn.cluster import Binarizer
bn = Binarizer(threshold=0.9)
new = bn.transform(old)
```

Equal-depth (frequency) partitioning

Делим область значения признаков на области-интервалы: в каждую попало одинаковое число точек

categories	counts	freqs
(5.99, 17.0)	18	0.2571
(17.0,25.0)	17	0.2429
(25.0, 34.0)	18	0.2571
(34.0, 42.0)	17	0.2429

factors = pd.qcut(ages, 4)

Equal-depth partitioning

\+ равномерные бины

```
X['name'].quantile([.2, .4, .6, .8]) # пороги-квантили
x2 = pd.qcut(x, 10, labels=False) # квантильная дискретизация
```

Кластеризация

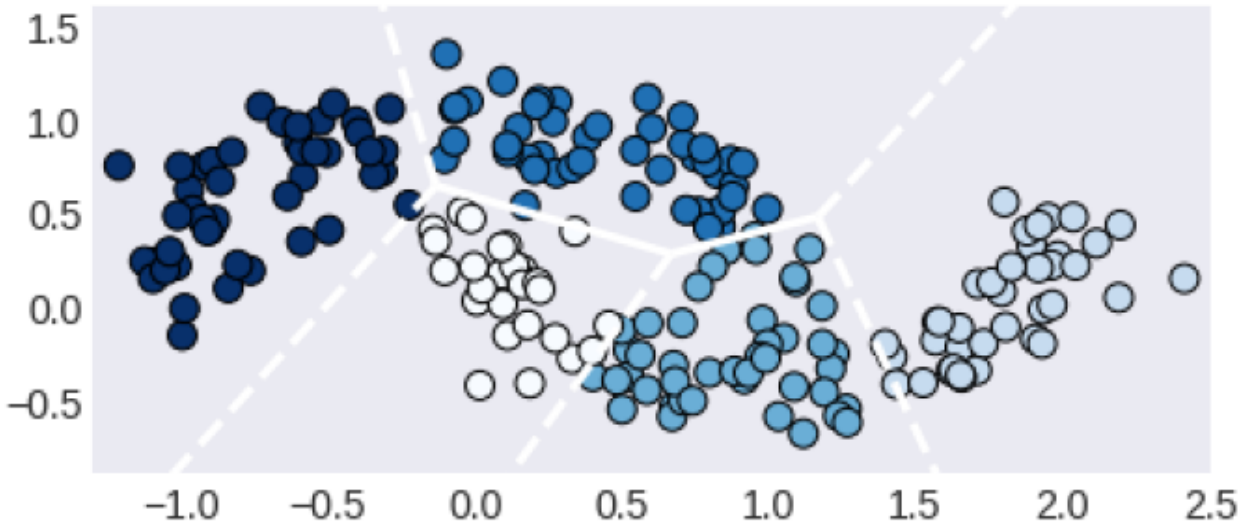

```
from sklearn.cluster import KMeans
model = KMeans(n_clusters=4, random_state=0)
a = model.fit_predict(x)
c = np.sort(model.cluster_centers_[:, 0])
c = (c[1:] + c[:-1]) / 2
c = np.concatenate([min(x), c, max(x)])
```

Дискритизация

Если не угадать с кластеризацией...

Дискритизация

Кластеризация



Если есть группа однородных признаков, то можно с помощью кластеризации получить новый категориальный признак «номер кластера» или «расстояние до центра кластера»

Дискритизация

Способы кодирования при дискретизации:

- первым / последним значением из бина
- средним (арифметическим, медианой и т.п)
- номером бина

Пример: разобьём на два бина **[1, 1, 1, 2, 2]; [4, 4, 7, 8, 9]**

Кодировки:

первым: 1, 1, 1, 1, 1, + 4, 4, 4, 4, 4

последним: 2, 2, 2, 2, 2, + 9, 9, 9, 9, 9

mean: 1.4, 1.4, 1.4, 1.4, 1.4, + 6.4, 6.4, 6.4, 6.4, 6.4

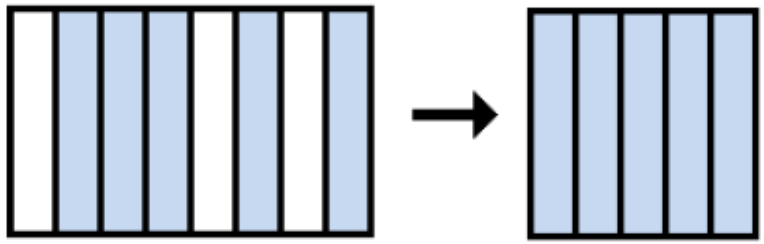
медианой: 1, 1, 1, 1, 1, + 7, 7, 7, 7, 7

номером: 1, 1, 1, 1, 1, + 2, 2, 2, 2, 2

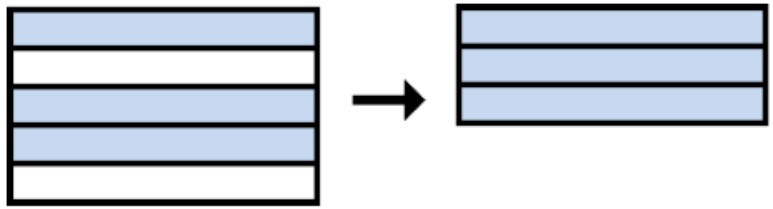
Сокращение данных

Сокращение данных (Data Reduction)
— уменьшение объёма исходных данных, сохраняя полезную информацию

- отбор признаков (отдельная тема)

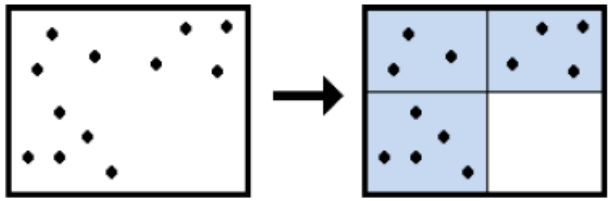


- отбор объектов (Instance Selection)
  редко используется, как правило, по анализу или экспертами (пример с ремонтами в РЖД)

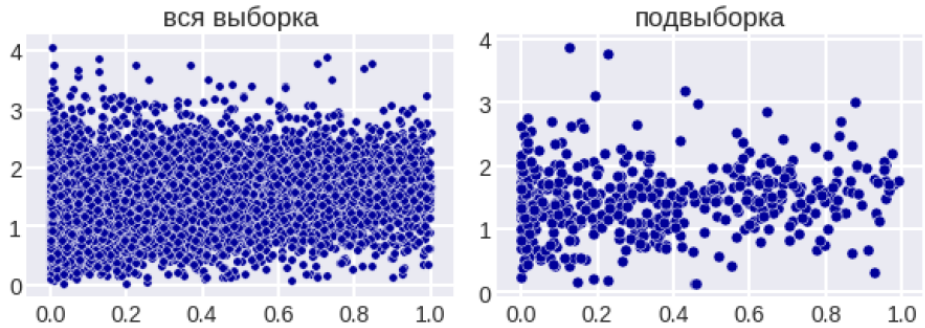


- удаление дубликатов, «пустых» данных

- дискретизация, огрубление информации (Discretization) увеличение шага дискретизации, перевод вещественных признаков в дискретные



- сэмплированое (Sampling)



- сокращение размерности (Dimensionality reduction)
  - факторный анализ (factor analysis)
  - метод главных компонент (PCA), SVD, случайные проекции
  - нелинейные модели: LLE, ISOMAP
  - многомерное шкалирование (MDS)

Сокращение данных

Цели сокращения данных

- удаление лишних (нерелевантных) данных
- повышение качества решения задачи
- уменьшение стоимости данных
- увеличение скорости последующего анализа (в частности, настройка моделей)
- повышение интерпретируемости моделей

Удаление дубликатов

```
df['is_dup'] = df.duplicated(subset=['date', 'sum'], keep='first')
```

```
df.drop_duplicates(subset=['date', 'sum'], keep='first')
```

id	date	sum	k	is_dup
0	01.01.2012	55	1.2	False
1	02.03.2012	117	4.3	False
0	01.01.2012	55	1.5	True
0	02.03.2012	117	0.2	True

id	date	sum	k	is_dup
0	01.01.2012	55	1.2	False
1	02.03.2012	117	4.3	False

Тонкости

- дубликаты могут быть по подмножеству признаков (шум)
- факт дублирования м.б. важен (лучше установить причину)
- совет: смотреть данные, отсортированные по отдельным признакам

Сокращение данных

По числу уникальных значений, можно догадаться, что категориальные признаки равны

id	a	b	c	d	e
0	1	0.05	B	1.12	0.76
1	1	0.76	B	1.12	0.96
2	2	0.05	F	0.78	0.79
3	1	0.56	B	1.12	0.96
4	3	0.47	A	1.09	0.15
5	2	0.22	F	0.78	0.79
6	2	0.69	F	0.78	0.66
7	3	0.59	A	1.09	0.25
8	2	0.43	F	0.78	0.79

Сокращение данных

```
2    4
1    3
3    2
Name: a,dtype: int64
0.05    2
0.56    1
0.43    1
0.47    1
0.76    1
0.22    1
0.59    1
0.69    1
Name: b,dtype: int64
```

```
F    4
B    3
A    2
Name: c,dtype: int64
0.78    4
1.12    3
1.09    2
Name: d,dtype: int64
0.79    3
0.96    2
0.15    1
0.76    1
0.66    1
0.66    1
0.25    1
Name: e,dtype: int64
```

Сокращение данных — сэмплирование

Цели

- для более быстрого поиска оптимальных параметров
- составляющая часть алгоритма (RF)
- для получения выборки, обладающей специальными свойствами

Способы

- Без возвратов (Simple random sampling without replacement)
- С возвратами (Simple random sampling with replacement)
- Сбалансированное (Balanced sampling) — сэмплирование при котором подвыборка будет удовлетворять некоторому заранее заданному условию (например, 90% объектов будет соответствовать пациентам старше 60 лет)

Сэмплирование

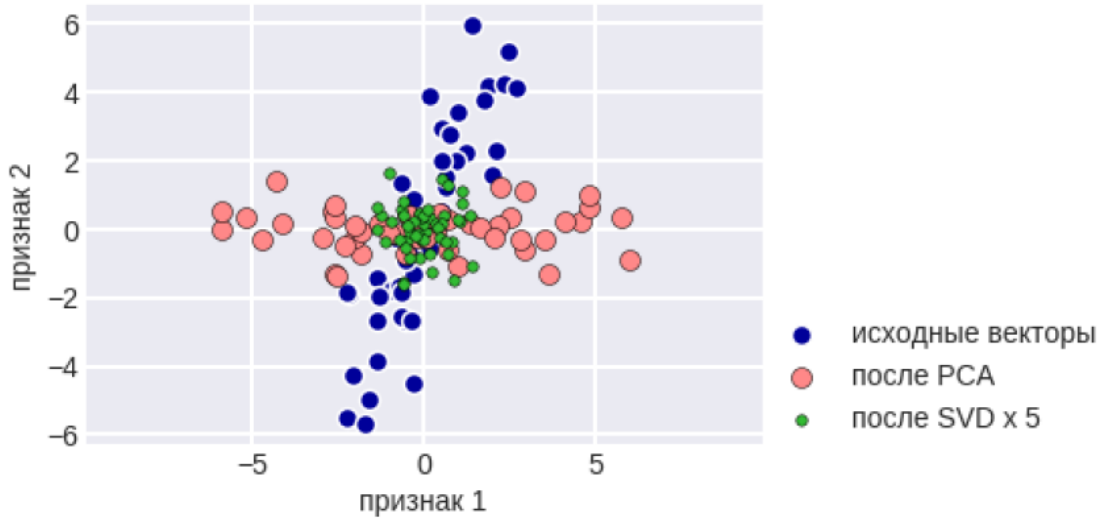
- Кластерное (Cluster sampling) — предварительно данные разбиваются на кластеры и выбирается подмножество кластеров.
- Стратифицированное (Stratified sampling) — предварительно данные разбиваются на кластеры, в каждом кластере отдельно осуществляется сэмплирование, таким образом в подвыборку попадают представители всех кластеров

Может производиться с предварительно заданными вероятностями выбора каждого объекта
Вероятности выбираются исходя из:

- сложности классификации объектов (например, для получения трудных подвыборок
- весам в функциях ошибок
- свежести данных / доверия к данным / стоимости данных

Сокращение размерности (Dimensionality reduction)

PCA (отдельная лекция)




```
from sklearn.decomposition import PCA
pca_transformer = PCA()
X2 = pca_transformer.fit_transform(X)

X = X - X.mean(axis=0)
U, L, V = svd(X)
```

Сокращение размерности

Для изображений (Изначальный размер изображения 300×451)

k=5 k=10 k=20

```
from numpy.linalg import
svd U, L, V = svd(image)
k = 5
plt.imshow(U[:,:k].dot(np.diag(L[:k]).dot(V[:k,:])),
cmap=plt.cm.gray)
```

Итоги

План работы с данными (до разбиения train/test)

- удаление
  – служебных признаков и признаков, содержащих утечки (информацию о целевом векторе или из будущего)
  – категориальных переменных, если число уникальных значений $\simeq$ число объектов или одно уникальное значение
  – признаков, которые в будущем не будем собирать
- преобразование типов данных
- предобработка строк (общий регистр), устранение дубликатов
- обработка редких категорий (если не использовать данные)
- заполнение пропусков (если не использовать данные, т.е. константами)
- генерация признаков (если не использовать данные)

Итоги

После разбиения на обучение и контроль, т.е. не заглядывая в test

- масштабирование признаков (в том числе стандартизация), деформации, OHE
- обработка редких категорий
- заполнение пропусков
- генерация признаков

Выводы

- Предобработка данных нужна, чтобы данные обладали желаемым свойством
- Предобработка может производиться с учётом модели (особенно, если тесно связана с конструированием признаков)
- качественные данные $\Rightarrow$ качественная модель (очень трудозатратно, м.б. до 90% времени)
- Главный вопрос: «Почему в данных есть это?»
- Не забываем о pipeline https://www.kaggle.com/alexisbcook/pipelines

Итоги

Ссылки

- J. Brownlee. Data preparation for machine learning. eBook, 2020
- Fraboni Ec «Data preprocessing» https://www.slideshare.net/FraboniEc/data-preprocessing-61426734

Примеры

- https://www.kaggle.com/jolasa/eda-anda-data-preparation-4th-place
- https://medium.com/analytics-vidhya/part-1-data-preparation-made-easy-with-python-e2c024402327
- https://www.kaggle.com/iamleonie/intro-to-time-series-forecasting