Статистический анализ данных

Лекция 6. Создание признаков

Московский авиационный институт
«МАИ»

7 сентября 2021 г.

Создание признаков

Придумывать признаки трудно, требует много времени и глубоких знаний. «Прикладное машинное обучение», в основном, это конструирование признаков.

— *Эндрю Ын*

[Machine Learning and AI via Brain simulations. Stanford University]

Особенно важна эта тема та где нам нужно построить интерпретируемую модель ML
План

- постановка задачи
- признаки, типы признаков
- создание с помощью EDA
- типичные, хорошие признаки определённых типов

Создание признаков

Создание признаков (feature engineering / construction) — процесс придумывания способов описания данных с помощью простых значений, которые должны отражать характеристики объектов исследований, через которые могут выражаться целевые значения

Изначально объекты могут быть заданы непризнаковым описанием:

- измерения
- веб-страницы
- файлы
- участники соцсети
- и т.д.

Создание признаков

Процесс создания признакового пространства зависит от модели, которую будем использовать

OHE-кодирование предпочтительнее для линейных моделей, умное кодирование категорий — для деревьев

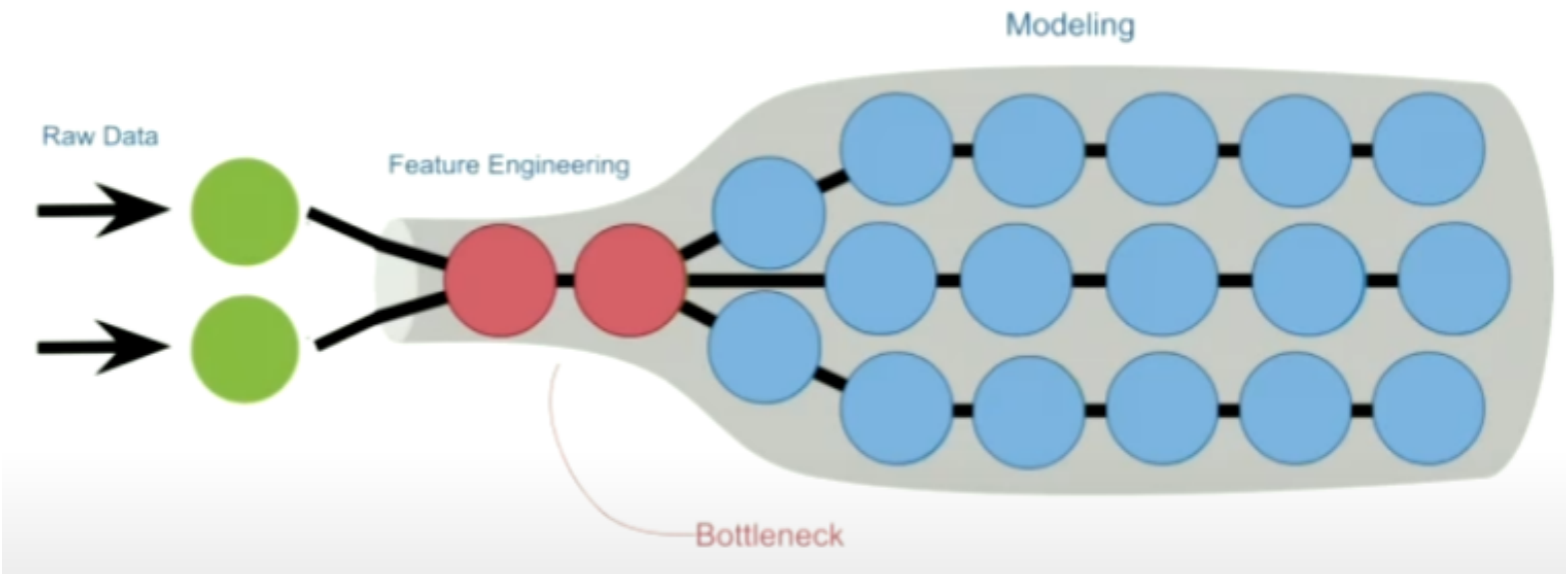
Выбросы можно не удалять для робастной модели

Следует использовать:

- знание предметной области
- EDA
- автоматизированные системы

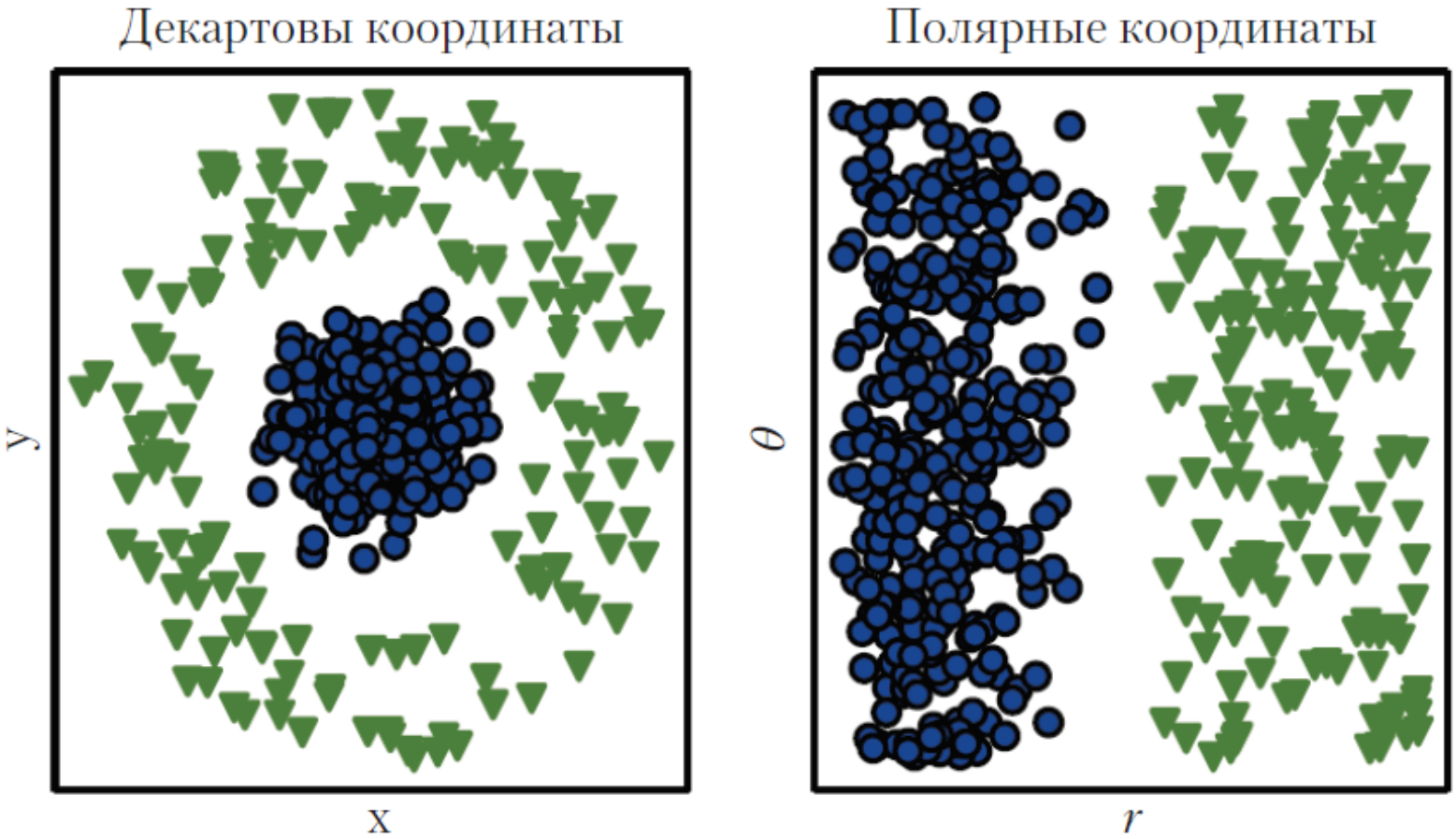
Создание признаков

Признаковое описание — всё, что знает модель о данных



https://www.youtube.com/watch?v=vwiVKm5_nRA

Пример FE



Гудфеллоу и др. Глубокое обучение

Создание признаков

Признаки (Features)

$$f : X \to S$$

Признак пол
Клиенты $\to \{\text{М}, \text{Ж}\}$

Признак доход
Клиенты $\to \{\ldots, 10000, 20000, \ldots, NA\}$

Значения признака могут быть не определены
это тоже важная информация!

Некоторые значения можно восстановить по другим признакам
(например, пол)

Создание признаков

Типы числовых признаков

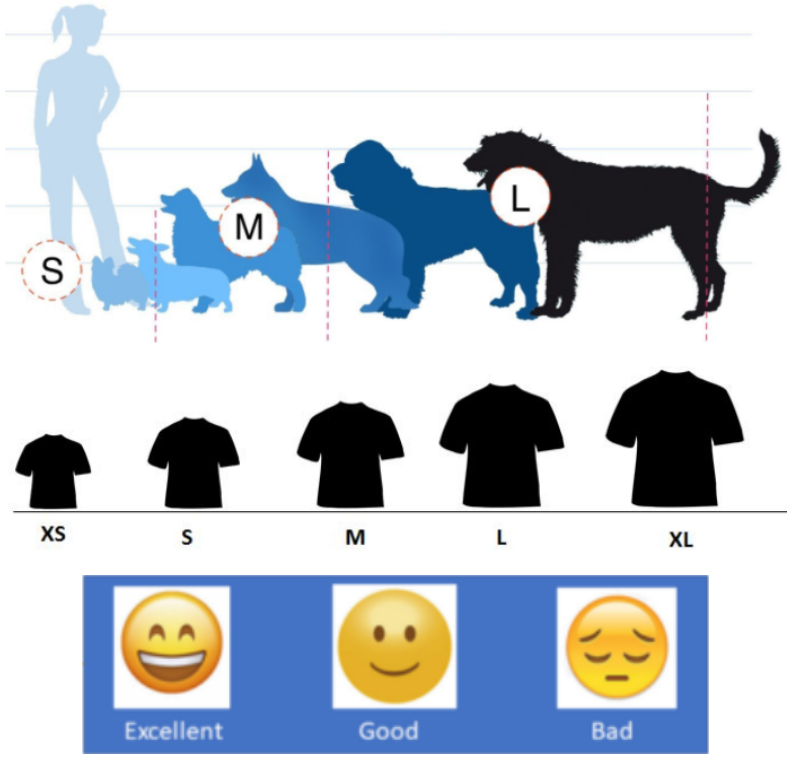
- вещественные
  — значения вещественные числа (измерения или характеристики чего-то), часто выделяют дискретные (discrete) и непрерывные (continuous variables)
  - интервальные (Interval)
  - относительные (Ration)
- категориальные (categorical variables)
  — значение переменной — принадлежность к одной из категорий (level), множество категорий конечно (часто фиксировано)
  - неупорядоченные категориальные (номинальные – Nominal, факторные)
  - порядковые (Ordinal) или упорядоченные категориальные
- бинарные

Создание признаков

Тип признака	Операции	Трансформации	Примеры
номинальные	$==$ перестановки mode, entropy, contingency, correlation, X2-test	перестановка	ID, пол, цвет, профессия
порядковые	$>$ median, percentiles, rank correlation, run tests, sign tests	монотонное преобразование	оценка, рейтинг
интервальные	$+,-$ mean, standard deviation, Pearson's correlation, t and F tests	$A \cdot X + B$	дата, температура по Цельсию
относительные	$\cdot, /$ geometric mean, harmonic mean, percent variation	$A \cdot X$	возраст, масса, длина, цена, температура по Кельвину

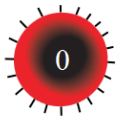
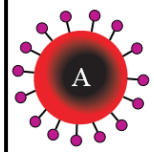
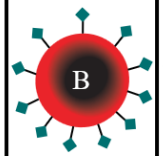
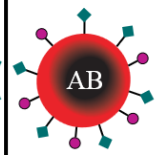
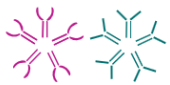
Категориальные

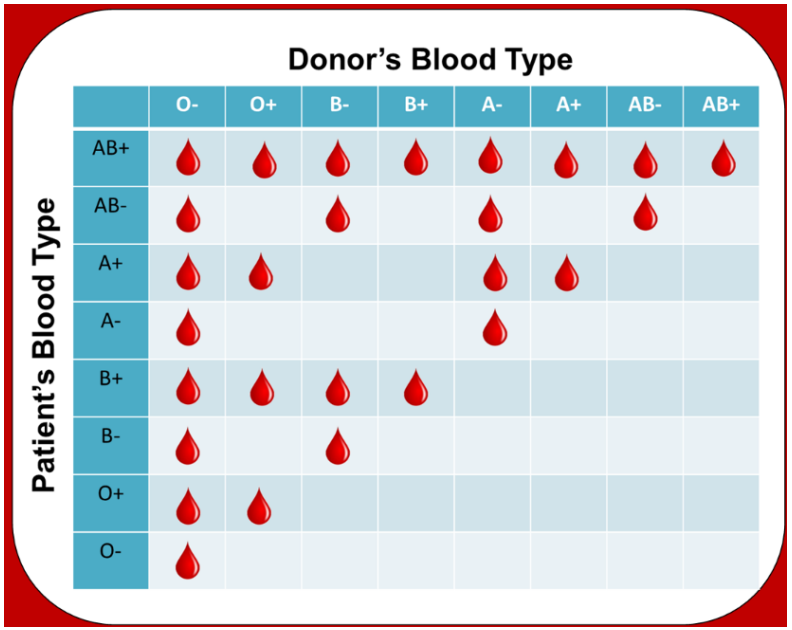
Неупорядоченные	Порядковые
жанры фильмов, имена актёров, категории товаров	Часто «кольцевой порядок»: времена года, дни недели, час



Создание признаков

С возможным правильным кодированием

	Группа 0 (I)	Группа A (II)	Группа B (III)	Группа AB (IV)
Тип эритроцитов				
Антитела в плазме	 α- и β-агглютинины	 β-агглютинины	 α-агглютинины	Нет
Антигены на эритроцитах	Нет	A-агглютиноген	B-агглютиноген	 A- и B-агглютиногены



Создание признаков

Простые типы нечисловых признаков

- временные отметки
- множества, наборы (например, пары координат)
- строки

id	дата	пол	образование	сумма	просрочки	платежи
0	10.01.18	1	высшее	5000	0	000
1	15.01.18	1	высшее	2500	1	001000
2	08.02.18	0		13675	3	111
3	11.02.18	0	среднее	NaN	0	0

Создание признаков

Признаки бывают

- исходные
- сгенерированные / производные

пример: возраст = текущая дата − дата рождения

Совет: даже если есть какой-то признак, сгенерируйте его по другим

Пример с возрастом: есть дата рождения, текущая, возраст $\Rightarrow$ создаём ещё одни признак возраст и сравниваем

Создание признаков

Дальнейший план

Всё что касается EDA:

Контекстные признаки
Служебные признаки
Утечки
Странности в данных

Отдельные виды признаков

Строковые
Категориальные
Вещественные
Временные
Географические
Деньги, количество, ...

Создание признаков

Контекстные признаки — это признаки, смысл которых явно прописан постановке задачи или понятен из контекста.

Смысл определяет:

- область значений
- примерное распределение в этой области

Пример: верхнее и нижнее давление

id	ap_hi	ap_low	ap_hi_new	ap_low_new	edit
0	150	1100	150	110	1
1	11	70	110	70	1
2	12	80	120	80	1
3	11	570	115	70	1
4	1	2080	120	80	1
5	130	90	130	90	0

Создание признаков

Контекстные признаки

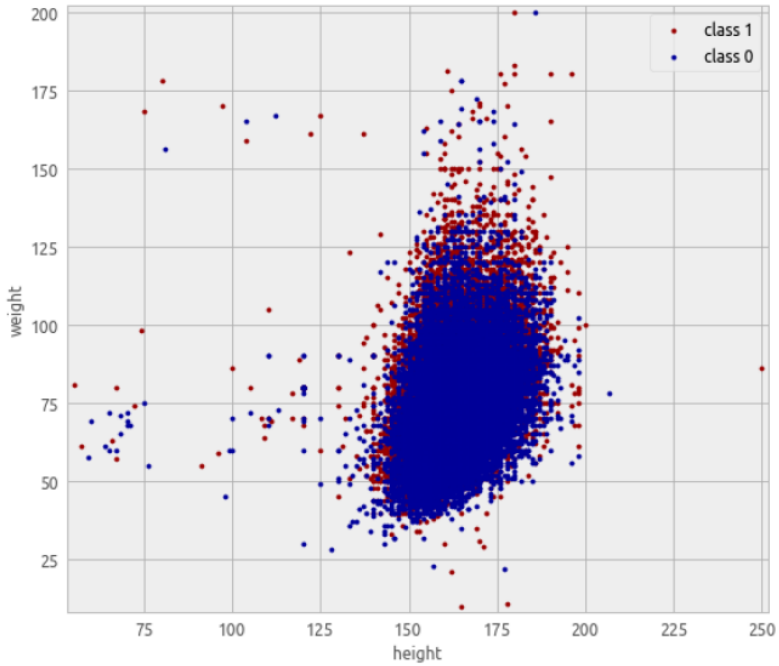
Признак	Гипотеза
число кликов	Максимальна в рабочие дни, в дневные часы
уровень дохода	Унимодальное распределение, значения положительные
температура	Лежит на отрезке $[36, 42]$
«верхн_XXX», «нижн_XXX»	«верхн_XXX» $\geq$ «нижн_XXX»

Почему плохо решать задачи без знания предметной области

- Хакатон Газпромнефти (II место без ML)
- бозон Хиггса (Семенов + эксперты физики)
- РЖД и классификация дефектов (погуглите)
- РЖД и спрос на грузовые перевозки

Создание признаков

Диаграмма рассеивания пар контекстных признаков



аномальные зоны, концентрации значений

новые признаки:

- признак аномальности
- признак частого значения
- отклонение от регрессионной модели построенной по паре (показатель передания/недоедания)

Служебные признаки

могут не входить в явном виде в признаковую матрицу, но их значения определяются из способа организации данных

- номер строки (а также производные признаки, например, чётность номера строки)
- номер объекта в какой-то внутренней нумерации (например, id объекта), производные признаки от этого номера
- порция данных,если датасет разбит на несколько частей, например, train / test / valid
- константный признак
- характеристические признаки (выполняется ли какое-то свойство)
- характеристика особенностей данных (например, число пропусков на объекте)
- имена и характеристики записей (в задачах, где объекты хранятся отдельно, например как файлы изображений)

Казалось бы, логично не рассматривать такие признаки...

Служебные признаки

	Пол	Рост	Вес
1	М	170	80
20	Ж	NaN	70
23	М	167	75
33	М	NaN	NaN
40	Ж	180	65

	ind	Пол	Рост	Вес	nan	Пол=М
0	1	М	170	80	0	0
1	20	Ж	NaN	70	1	0
2	23	М	167	75	0	1
3	33	М	NaN	NaN	2	1
4	40	Ж	180	65	0	0

```
data.reset_index(inplace = True)
data['nan'] = data.isnull().sum(axis=1)
data['Пол=М'] = (data['Пол'] == 'М').astype(int)
data['Рост2'] = data['Рост'] / 10
data['Рост2'] = data['Рост'] - 10·np.floor(data['Рост2'].values)
```

Служебные признаки

Когда служебные признаки важны:

- особенность значения $\Leftarrow$ особенность объекта
  «круглый доход, круглый рост, вес, давление» — не знает точного значения
- улучшает качество
  «есть ли пропуск в признаке»,
  «сколько пропусков / аномальных значений»
- поиск утечек в данных
- организация эксперимента (разбиение на корзины и т.п.)

  Пример: области значений целевой переменной (маленькие, средние, большие) + StratifiedKFold

Служебные признаки

Утечка в данных — информация, которая повышает качество решения задачи машинного обучения, но теряет эти свойства при тестировании на независимом и правильно организованном контроле (при эксплуатации алгоритма)

Пол	Рост	Вес	Класс
М	170	80	0
Ж	NA	70	0
М	167	75	0
М	NA	NA	1
Ж	180	65	1

Утечка приводит к переобучению

Служебные признаки

Виды утечек

1. **зависимость от служебных признаков, в том числе**
   - от порядка (номера строки)
   - от организации данных (от каталога)
   - способа представления (названия файлов, времени его создания и т.п.)
   - от особенностей (наличия пропусков, дубликатов и т.п.)
2. **неявное использование информации**
   - о целевом векторе
   - из будущего или настоящего
3. **содержание ответа в исходных данных, как правило, из-за заглядывания в будущее**
   (пример про номер страницы и число страниц в сессии)

Странности в данных

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
1143	1586	23351	1	150	90.0	150	90	3	2	0	0	1	1
1503	2122	16534	1	164	73.0	164	73	1	1	0	0	1	1
3420	4838	14516	1	100	70.0	100	70	1	1	0	0	1	0
3735	5278	17642	1	120	70.0	120	70	1	1	0	0	1	0
3799	5378	23434	1	150	61.0	150	61	1	3	0	0	1	1
4212	5946	16110	1	120	80.0	120	80	1	1	0	0	1	0
7058	10053	21025	1	140	90.0	140	90	3	1	0	0	1	1
7305	10412	15859	1	120	80.0	120	80	1	1	0	0	1	0

Что странного?

Странности в данных

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
1143	1586	23351	1	150	90.0	150	90	3	2	0	0	1	1
1503	2122	16534	1	164	73.0	164	73	1	1	0	0	1	1
3420	4838	14516	1	100	70.0	100	70	1	1	0	0	1	0
3735	5278	17642	1	120	70.0	120	70	1	1	0	0	1	0
3799	5378	23434	1	150	61.0	150	61	1	3	0	0	1	1
4212	5946	16110	1	120	80.0	120	80	1	1	0	0	1	0
7058	10053	21025	1	140	90.0	140	90	3	1	0	0	1	1
7305	10412	15859	1	120	80.0	120	80	1	1	0	0	1	0

рост = верхнее давление, вес = нижнее

Использование EDA для генерации признаков

Разведывательный анализ данных (EDA), кроме основной цели «понять, как устроены данные», имеет ещё несколько важных целей:

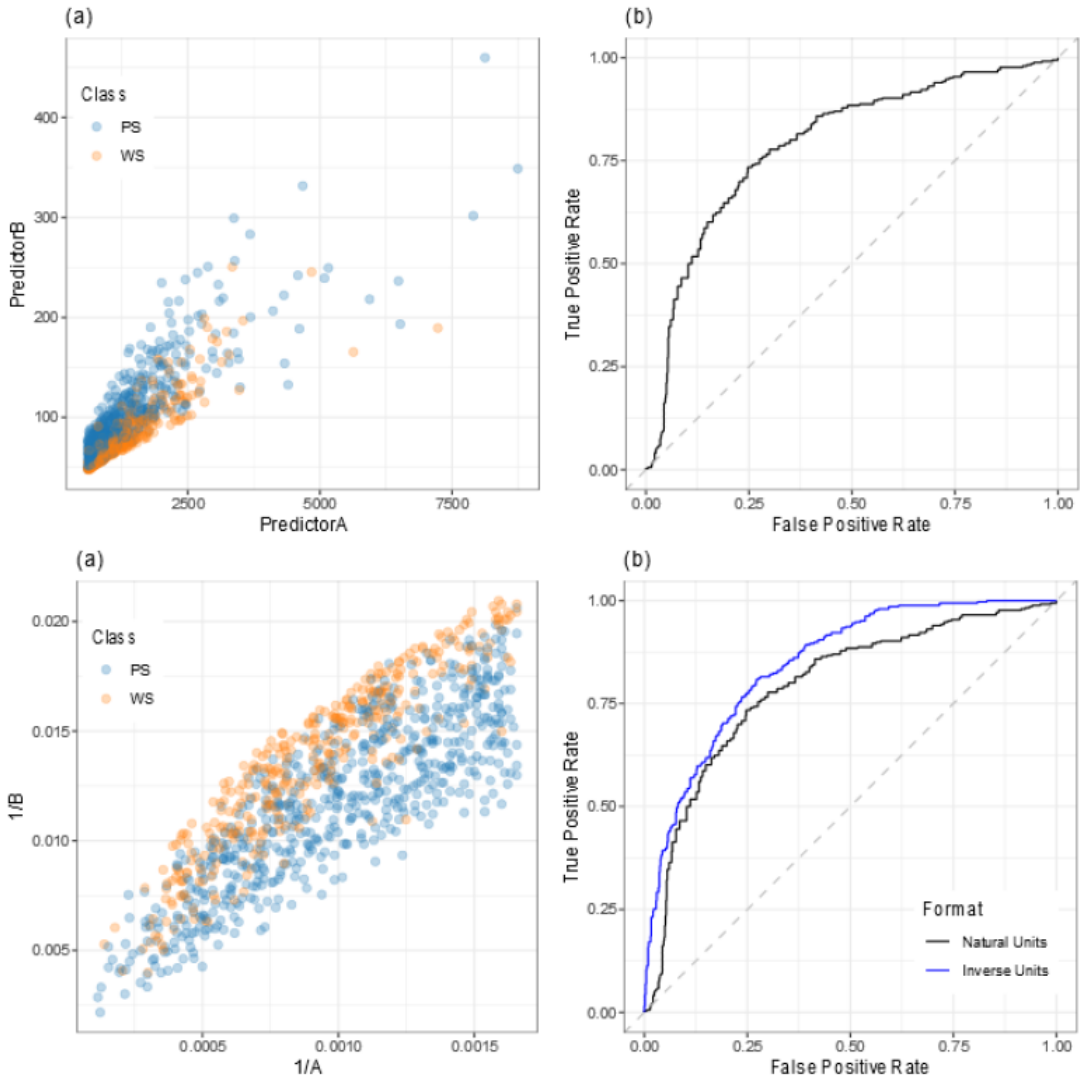
- найти «волшебные признаки» (которые почти решают задачу, или с помощью которых чётко видны какие-то паттерны)
- понять, как «меняются признаки» (насколько можно верить данным или синтезируемым признаковым описаниям при изменении времени или категорий объектов)
- использовать контекст (= экспертные знания, нашу интуицию и т.п.)
- выявить информационные утечки (не содержат ли описания явно/неявно лишнюю информацию)
- построить простые бенчмарки (алгоритмы в несколько строчек кода, которые надежны, интерпретируемы и т.п.)

Примеры EDA

- Data mining in action
  Лекция https://hackmd.io/@aguschin/r1Z3nfidr
  ipynb https://github.com/data-mining-in-action/DMIA_Sport_2019_Autumn/tree/master/seminars/EDA
- https://www.kaggle.com/deffro/eda-is-fun
- https://www.kaggle.com/warkingleo2000/first-step-on-kaggle
- https://www.kaggle.com/ash316/eda-to-prediction-dietanic
- https://www.kaggle.com/artgor/eda-feature-engineering-and-everything
- https://www.kaggle.com/artgor/eda-and-models
- https://www.kaggle.com/vbmokin/eda-for-tabular-data-advanced-techniques

Использование EDA для генерации признаков

(обратные признаки для линейной модели)



Строковые признаки

id	keyword	location	text	target
50	ablaze	AFRICA	#AFRICANBAZE: Breaking news:Nigeria flag set ablaze in Aba. http://t.co/2nndBGwyEi	1
52	ablaze	Philadelphia, PA	Crying out for more! Set me ablaze	0
53	ablaze	London, UK	On plus side LOOK AT THE SKY LAST NIGHT IT WAS ABLAZE http://t.co/qqsmshaJ3N	0

отдельная тема — обработка текстов

Категориальные признаки

1. Автоматическое определение категориальности

- если значения — строки
- если мало уникальных значений

id	city	sex	income
0	Moscow	M	110
1	London	F	200
2	London	M	130
3	Paris	M	120
4	Moscow	F	180

```
def find_cat(data):
for name in data.columns:
    s =''
    s += name
    if (type(data[name][0]) == str):
        s += ' строка,'
    if (data[name].nunique() <= 3):
        s += ' мало уникальных'
    if (s!=name):
        print (s)
```

2. Создание новых категориальных признаков

- конъюнкция признаков
  может быть очень полезно: kNN, линейные алгоритмы
- создание новых признаков из знаний о предметной области

id	city	sex	income	city + sex
0	Moscow	M	110	Moscow + M
1	London	F	200	London +F
2	London	M	130	London + M
3	Paris	M	120	Paris + M
4	Moscow	F	180	Moscow + F

```
def make_conj(data, feature1, feature2):
        data[feature1 + ' + ' + feature2] =
                data[feature1].astype(str) + ' + ' + data[feature2].astype(str)
        return (data)

# Пример использования

make_conj(data, 'city', 'sex')
```

Категориальные признаки

3. Простейшее кодирование — по номеру категории **Label Encoding**

- **Лексикографический порядок (sklearn)**
  sklearn.preprocessing.LabelEncoder
  - не подходит для линейных алгоритмов
  - проблема новых категорий (средним?)
- **В порядке появления (pandas)**
  pandas. factorize
  Это удобно! Можно использовать сортировки по признакам
  $\Rightarrow$ по порядку индуцированным каким-то признаком
- **Случайное кодирование**
  dict + map
  многократное случайное кодирование иногда хорошо работает с RF

Категориальные признаки

id	city	sex	income	city_le	city_fz	city_rnd	sex_le	sex_fz	sex_rnd
0	Moscow	M	110	1	0	0.63	1	0	0.22
1	London	F	200	0	1	0.75	0	1	0.20
2	London	M	130	0	1	0.75	1	0	0.22
3	Paris	M	120	2	2	0.50	1	0	0.22
4	Moscow	F	180	1	0	0.63	0	1	0.20

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()

for name in cols:
        # лексикографический порядок
        data[name + '_le'] = le.fit_transform(data[name])
        # в порядке упоминания
        data[name + '_fz'] = pd.factorize(data[name])[0]
        # случайно
        dct = dict(zip(data[name].unique(),
                np.random.rand(data[name].nunique()).round(2)))

        data[name + '_rnd'] = data[name].map(dct)
```

4. **Dummy-кодирование** / One-hot-encoding

id	city	sex	income	city=Moscow	city=London	city=Paris	sex=M	sex=F
0	Moscow	M	110	1	0	0	1	0
1	London	F	200	0	1	0	0	1
2	London	M	130	0	1	0	1	0
3	Paris	M	120	0	0	1	1	0
4	Moscow	F	180	1	0	0	0	1

Категориальные признаки

OneHotEncoder (по умолчанию sparse)

```
from sklearn import preprocessing

ohe = preprocessing.OneHotEncoder(sparse=False)
tmp = ohe.fit_transform(data[cols]).astype(int)
tmp = pd.DataFrame(tmp, columns=['OHE_' + str(i)
                                 for i in range(tmp.shape[1])])
data = pd.concat([data, tmp], axis=1)
```

Ручное решение

```
def code_myohe(data, feature):
    for i in data[feature].unique():
        data[feature + '=' + i] = (data[feature] == i).astype(int)
    for name in cols:
        code_myohe(data, name)
```

Самый простой способ

```
pd.get_dummies(data)
```

Категориальные признаки

One-hot-encoding

- **Хороши для линейных алгоритмов**
  можно кодировать $N - 1$ категорию
- **Все признаки в одной шкале** (на $[0, 1]$)
  но есть тонкость, что категории разной мощности
- **Большое число категорий $\Rightarrow$ сильно разреженные матрицы**
  часто используют sparse-формат
- **Плохо, что после OHE значительная часть признаков — бинарные**, описывают категориальные зависимости
  некоторые алгоритмы (RF) могут терять качество
- **В задачах часто встречаются признаки с большим числом категорий**
- **практически нет проблемы новых категорий**

Проблема мелких и новых категорий

— возникает почти для всех способов кодирования

Часто: мелкие категории ⇒ в одну
Здесь: можно не кодировать!
простая и понятная интерпретация

Новую категорию можно сразу относить в группу к мелким или в самую «опасную» для нас категорию

5. По значениям вещественного признака

id	city	sex	income	city_mean_income	sex_mean_income
0	Moscow	M	110	150	123.3
1	London	F	200	170	195.0
2	London	M	130	170	123.3
3	Paris	M	120	120	123.3
4	Moscow	F	180	150	195.0

```
def code_mean(data, cat_feature, real_feature):
    """
    кодирование средним значением
    """
    mn = data.groupby(cat_feature)[real_feature].mean()
    return data[cat_feature].map(mn)
```

```
for name in cols:
    data[name + '_mean_income'] = code_mean(data, name, 'income')
```

Категориальные признаки

5. По значениям вещественного признака

- **естественная интерпретация:**
  товары какой категории дороже
- **можно использовать другие статистики**
  они не всегда логичны и хуже интерпретируются
- можно кодировать по разным признакам

6. По значениям самого признака

- **просто по мощности** Count Encoding
- **по частоте** Frequency Encoding

id	city	sex	income	city_vc	sex_vc	city_vcn	sex_vcn
0	Moscow	M	110	2	3	0.4	0.6
1	London	F	200	2	2	0.4	0.4
2	London	M	130	2	3	0.4	0.6
3	Paris	M	120	1	3	0.2	0.6
4	Moscow	F	180	2	3	0.4	0.4

```
data[name + '_vc'] = data[name].map(data[name].value_counts())
data[name + '_vcn'] = data[name].map(data[name].value_counts(normalize=True))
```

```
# другой способ
for name in cols:
    data[name + '_vc'] = data[name].map(data.groupby(name).size())
```

Недостатки Count Encoding

- коллизии (несколько категорий — один код)
  добавляем шум (не всегда это проблема)
- **проблема шума (мелкие категории)**
  мелкие категории объединяем в одну, и новые
- **проблема новых категорий**
- **проблема утечки информации**
  честно: без заглядывания в будущее
  это не страшная утечка (допустима в соревнованиях)

7. По значениям ДРУГОГО категориального признака

id	city	sex	income	city_svd	sex_svd
0	Moscow	M	110	-0.66	-0.79
1	London	F	200	-0.66	-0.62
2	London	M	130	-0.66	-0.79
3	Paris	M	120	-0.37	-0.79
4	Moscow	F	180	-0.66	-0.62

```
from numpy.linalg import svd
```

```
def code_factor(data, cat_feature, cat_feature2):
    """
    кодирование на основе другого категориального
    """
     ct = pd.crosstab(data[cat_feature], data[cat_feature2])
     u, _, _ = svd(ct.values)
     coder = dict(zip(ct.index, u[:,0]))
     return (data[cat_feature].map(coder))
```

Категориальные признаки

Перед кодированием создается новая таблица — количество одновременного появления категорий двух признаков

city\sex	F	M
London	1	1
Moscow	1	1
Paris	0	1

```
pd.crosstab(data['city'], data['sex'])
```

Можем кодировать одним из столбцов этой таблицы, а можем воспользоваться SVD разложением и кодировать первым столбцом разложения или несколькими столбцами разложения

Категориальные признаки

```
data['city_svd'] = code_factor(data, 'city', 'sex')
```

```
data['sex_svd'] = code_factor(data, 'sex', 'city')
```

$$X = \|x_{ij}\| = U\Lambda V^T$$

$$\Lambda' = \mathrm{diag}(\lambda_1, \ldots, \lambda_r, 0, \ldots, 0)$$

$$X' = U\Lambda' V^T = \sum_{i=1}^{r} \lambda_i u_i v_i$$

где x_{ij} — количество раз, когда i-я категория встречается с j-й

Дьяконов А. Г. Методы решения задач классификации с категориальными признаками // Прикладная математика и информатика. Труды факультета Вычислительной математики и кибернетики МГУ имени М.В. Ломоносова. 2014. № 46. С. 103–127.

8. Хэш-кодирование

- средство против сильно разреженных данных
- могут быть коллизии (можно выполнять разные хэш-кодирования)

id	city	sex	income	city_0	city_1	sex_0	sex_1
0	Moscow	M	110	1.0	-1.0	1.0	0.0
1	London	F	200	-2.0	2.0	-1.0	0.0
2	London	M	130	-2.0	2.0	1.0	0.0
3	Paris	M	120	1.0	-2.0	1.0	0.0
4	Moscow	F	180	1.0	-1.0	-1.0	0.0

```
from sklearn.feature_extraction import FeatureHasher
```

```
fh = FeatureHasher(n_features=2, input_type='string')
```

```
for name in cols:
    tmp = fh.fit_transform(data[name]).toarray()
    tmp = pd.DataFrame(tmp, columns=[name + '_' + str(i) for i in range(tmp.shape[1])])
    data = pd.concat([data, tmp], axis=1)
```

9. По значению целевого — Target Encoding

- Mean Target Encoding
- Std Target Encoding
- ...

id	city	sex	income	city_mt	sex_mt	target
0	Moscow	M	110	0.5	0.67	1
1	London	F	200	0.5	0.50	1
2	London	M	130	0.5	0.67	0
3	Paris	M	120	1.0	0.67	1
4	Moscow	F	180	0.5	0.50	0

- это форма стекинга — строим прогноз по одному (кодируемому) признаку
- подходит для любых алгоритмов (если правильно сделана)

Target Encoding

Почему хорошая идея —
упорядочивание категорий исходя из смысла задачи

Многие кодировки «случайны»,
а кодирование по значению целевого «логично»

Про mean target encoding

Кодирование средним (от программы Кантора)

Замечание: я бы скептически относился к разделу результаты

Можно по разному применять TE во время обучения: в тупую, по кросс-валидации, LOO, со вложенной кросс-валидацией. Это сильно влияет на результат
Плюс другие типы энкодинга иногда имеет смысл использовать параллельно друг с другом, чтобы получить более информативные «эмбеддинги» категорий
Ещё в зависимости от энкодинга могут разные гиперпараметры алгоритма получаться
С TE по кросс-валидации можно регуляризацию применять. Если OHE, то с глубиной и фичефракшен
Для лейбл энкодинга может глубина больше потребоваться

Поэтому такое сравнение может быть не очень корректно

Категориальные признаки

Главная проблема: подглядываем в целевую переменную, а это ведет к переобучению
Пример:неадекватная кодировка мелких категорий + слияние этих категорий
Нельзя допустить утечки значений целевого!

Способы борьбы:

- **кодирование по отложенной выборке**
  — сокращаем выборку для обучения
  Хороший пример: кодирование по куску испорченных данных
- **k-fold-кодировка**
  Идея: не использовать метку объекта при кодировании
  разбиваем на фолды и кодируем
  при LOO-кодировании проблемы утечки остаются
- **кодирование по случайным подвыборкам**
- **кодирование по предыдущим объектам (CatBoost)**

Почему плохо LOO-кодировать

id	sex	target	sex_LOO
0	M	1	0.25
1	M	1	0.25
2	M	0	0.5
3	M	0	0.5
4	M	0	0.5

```
d1 = data.groupby('sex')['target'].sum()
d2 = data.groupby('sex').size()
```

```
data['sex_loo'] = (data['sex'].map(d1) - data['target']) / data['sex'].map(d2)
```

В явном виде утечка...

Ещё и упорядочивание неестественное (1 получила меньшее значение)!

Категориальные признаки: сглаживание

добавляем среднее значение целевого с весом

$$\text{mean} = \frac{k_1 + \alpha \frac{m_1}{m}}{k + \alpha}$$

где k_1 — число 1 в категории, k — число всех объектов в категории, m_1 — число 1 в выборке, α — параметр

+ борьба с редкими категориями на них оценка ненадёжна

можно использовать и другие поправки на основе средних

тонкий вопрос: как кодировать на обучении, как на тесте обычно на тесте пересчитывают (по всему обучению)

Категориальные признаки: добавление шума

Чаще мультипликативный шум

```
def add_noise(series, noise_level):
    return series * (1 + noise_level * np.random.randn(len(series)))
```

Кодирование по предыдущим объектам (CatBoost)

- есть в CatBoost (там аналогичный приём и для вычисления градиента)
- одна категория в обучении кодируется по-разному, а на контроле фиксировано
- нет гиперпараметров (хотя напрашивается ввести)
- можно использовать разные порядки, т.е. при построении разных деревьев случайно сортировать порядки, индуцированные признаками

id	sex	sex_cb	target
0	M	NaN	1
1	F	NaN	1
2	M	1	0
3	M	0.5	1
4	F	1	0

```
gb = data.groupby(name)
data[name + '_cb'] = (gb['target'].cumsum() - data['target']) / gb.cumcount()
```

Target Encoding

Примеры кодирования по другим статистикам для бинарного целевого вектора:

$$(y_1, \ldots, y_k),$$
$$k_1 = y_1 + \ldots y_k, \;\; k_0 = k - k_1$$

$$\text{mean} = \frac{k_1}{k} \qquad \text{diff} = k_1 - k_0$$

$$\text{difflog} = \frac{\log k_1}{\log k_0} \qquad \text{normdiff} = \frac{k_1 - k_0}{k}$$

Использование статистик напрямую не связанных со значением целевого признака (например, дисперсию) уменьшает утечку!

На практике хороша смесь подходов!

10. Экспертное кодирование

Если признак порядковый, то есть естественная нумерация
Пример: кодирование названий географических регионов

11. Вложение категориальных признаков в маломерное пространство (Category Embedding)

пример: транзакции упорядочены во времени, категория покупки — слово, естественным образом получаем текст $\Rightarrow$ word2vec

Категориальные признаки: пропуски

- создание отдельной категории («нет значения»)
- игнорирование пропусков
  (например, в dummy-кодировке сопоставить им нулевую строку, т.е. соответствующие объекты не принадлежат ни одной категории)
- стандартные методы обработки пропусков
  (при плотном кодировании, например, по целевому вектору)

Статья на towardsdatascience.com

Кодирование mean target Tinkoff Data Science Challenge

Категориальные признаки: пример кода

https://contrib.scikit-learn.org/category_encoders/

https://www.kaggle.com/waydeherman/tutorial-categorical-encoding

https://www.kaggle.com/ogrellier/python-target-encoding-for-categorical-features

https://www.kaggle.com/vprokopev/mean-likelihood-encodings-a-comprehensive-study

https://www.kaggle.com/mlisovyi/9-ways-to-treat-categorical-features-updated

https://github.com/DenisVorotyntsev/CategoricalEncodingBenchmark

Вещественные признаки

были способы генерации признаков (в обработке данных):

- дискретизация (binning)
- деформация (функция над признаком)
- сглаживание
- нормировка (специальный вид деформации)
- функции над несколькими, например, сумма значений

Новые признаки

- суммы групп признаков
- мономы (например, $8x_1^4 x_2^{-3} x_3^{-0.5}$)
- расстояние до какого-то выделенного объекта (например, абсолютно здоровый человек)

	f	g	f^2	fg	g^2
0	1.0	0.0	1.0	0.0	0.0
1	2.0	2.0	4.0	4.0	4.0
2	3.0	1.0	9.0	3.0	1.0
3	4.0	1.0	16.0	4.0	1.0

```
from sklearn.preprocessing import PolynomialFeatures
```

```
pf = PolynomialFeatures(degree=2, interaction_only=False, include_bias=False)
```

```
data = pf.fit_transform(data)
```

Вещественные признаки

как искать взаимодействия

- использование предметной области
- перебор операций из словаря (будет нужна процедура отбора хороших признаков)
  например, генерации признаков для сигналов
- анализ взаимодействий признаков в моделях
  например, последовательные сплиты в деревьях
  в нейросетях генерация признаков — автокодировщики

Пример генерации с помощью моделей

Генерация бинарного признакового пространства с помощью RF

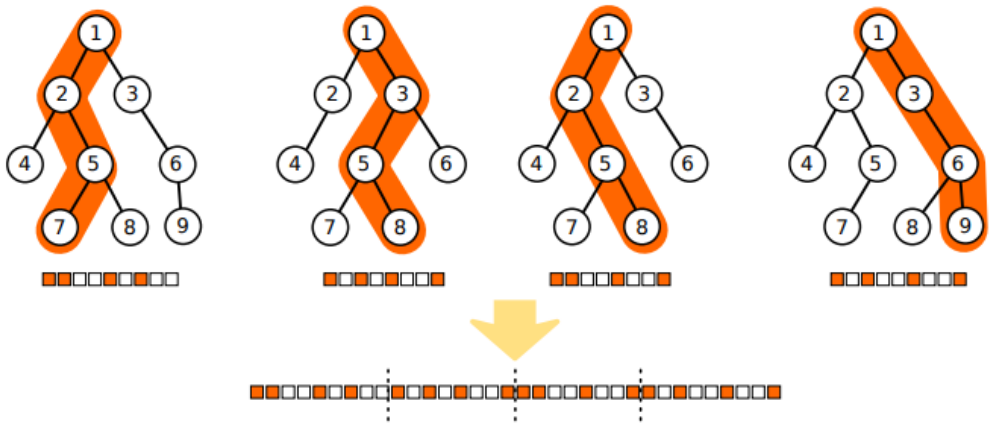


Figure 1. Example of feature induction. A single instance is classified in the leaf node by a random forest of 4 decision trees. Each decision node receive a unique identifier. If a test is satisfied in a node then the corresponding bit is asserted. Finally the encodings for each tree are combined by concatenation (or more generally by hashing the features ID onto a smaller dimensional space).

Celine Vens, Fabrizio Costa. Random Forest Based Feature Induction

https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.454.7215&rep=rep1&type=pdf

Временные признаки

date_time	deg_C	relative_humidity	absolute_humidity
10Mar10 – 1Jan11	1.3 – 46.1	8.9 – 90.8	0.2 – 2.23
2010-03-10 23:00:00	11.2	56.8	0.7848
2010-03-11 00:00:00	10.7	55.7	0.7603
2010-03-11 01:00:00	10.3	57.0	0.7702
2010-03-11 02:00:00	10.1	62.7	0.7648
2010-03-11 03:00:00	10.5	59.6	0.7517

Преобразование признаков

```
data[name] = pd.to_datetime(data[name], errors='coerce')
```

м.б. преобразование в вещественный признак

Временные признаки

Характеристика момента времени

- час, минута, секунда
- время суток
- день, день недели, день года
- неделя, месяц
- время года, год
- праздник / выходной
- особый день (первый понедельник месяца, начало Олимпиады и т.п)

Временные признаки

	date	date_day	date_dayofweak	date_dayofyear	date_month
0	2017-12-10	10	6	344	12
1	2016-11-13	13	6	318	11
2	2008-01-01	1	1	1	1
3	2017-05-06	6	5	126	5

```
data[name + '_day'] = data[name].dt.day
```

```
data[name + '_dayofweek'] = data[name].dt.dayofweek
```

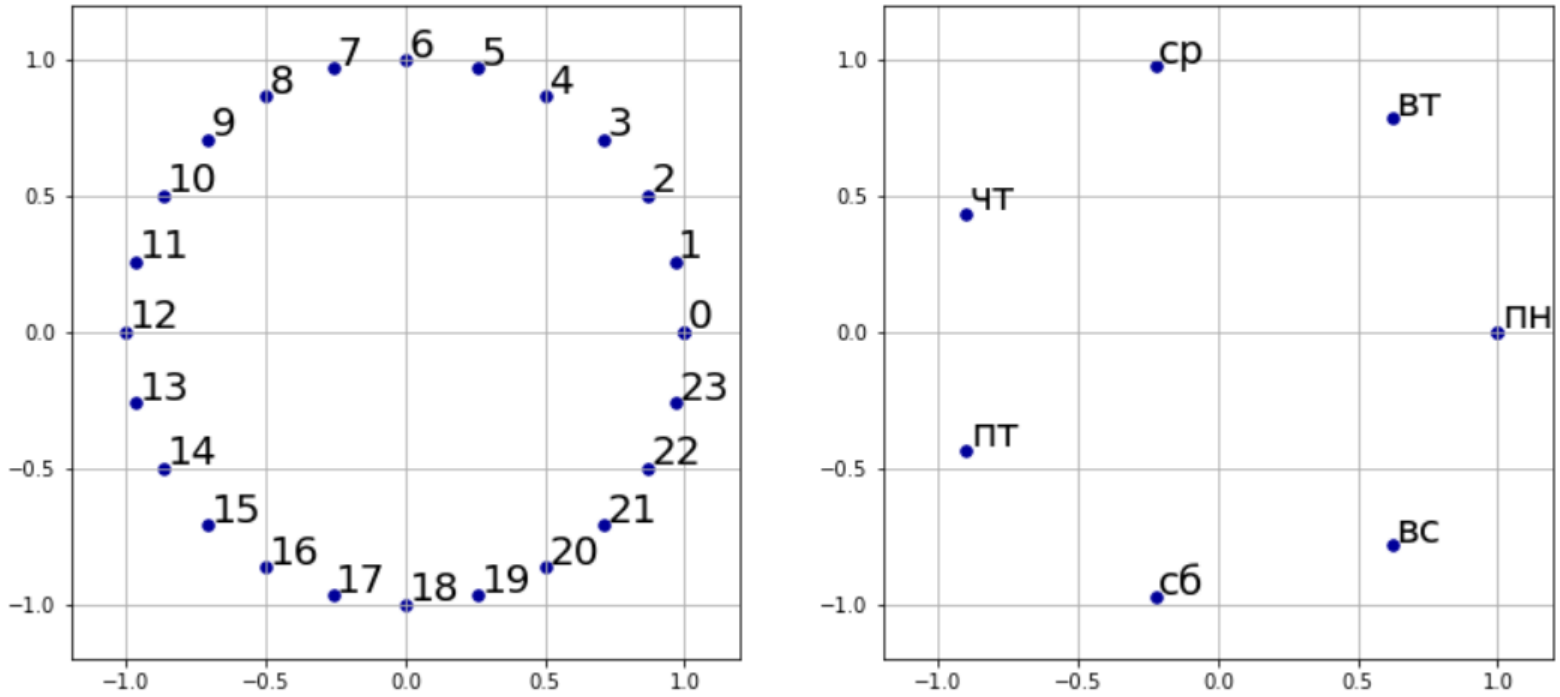
```
data[name + '_dayofyear'] = data[name].dt.dayofyear
```

```
data[name + '_month'] = data[name].dt.month
```

https://www.kaggle.com/maximkazantsev/tps-07-21-eda-catboost
https://pythonru.com/primery/kak-ispolzovat-modul-datetime-v-python

https://www.dataquest.io/blog/python-datetime-tutorial/

Кодирование циклических признаков




```
t = np.linspace(0, 2*np.pi, 8)
x = np.cos(t)
y = np.sin(t)
```

https://towardsdatascience.com/feature-engineering-time-3934038e0dbe

https://ianlondon.github.io/blog/encoding-cyclical-features-24hour-time/

Временные признаки

Взаимодействие пары временных признаков

- разница времён
- проверка на совпадения: один ли день недели/год и т.п.
- близость к дедлайну ($T_{\max} - T$)
- возраст ($T - T_{\text{день рождения}}$)
- сколько после/до праздника / большой покупки / регистрации

\+ нормировать на некоторые шаблоны (max разность)

Использование для других признаков

- устаревание в весовых схемах (больший вес новым объектам)
- что за последний промежуток T (операции по карте за последний месяц)
- формирование разбиения обучение / тест

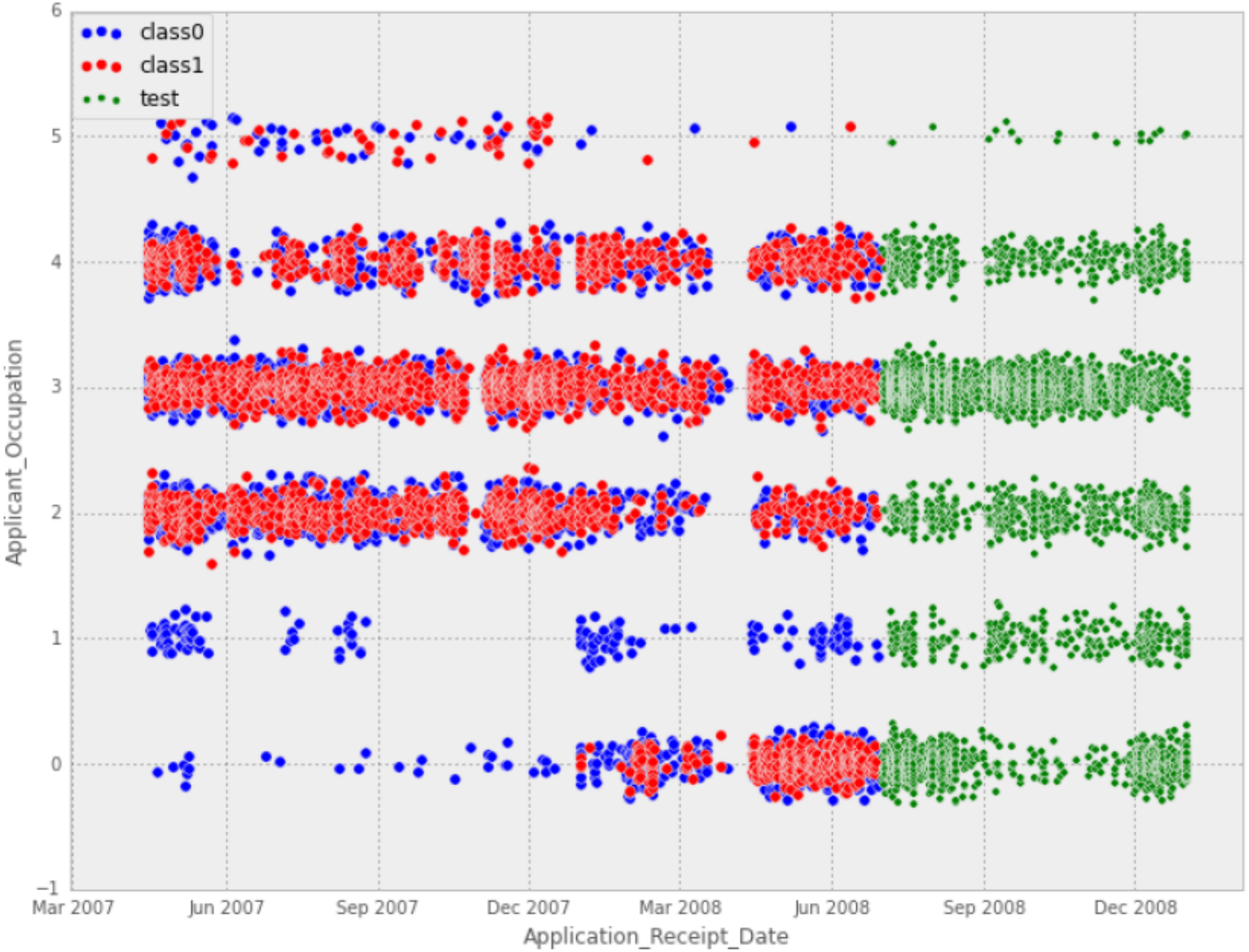
Временные признаки

Использовать для генерации других признаков (продолжение)

- сколько транзакций в этот день
- число транзакций клиента в день / число всех транзакций
- какой по счёту закрыл сделку перед концом торгов

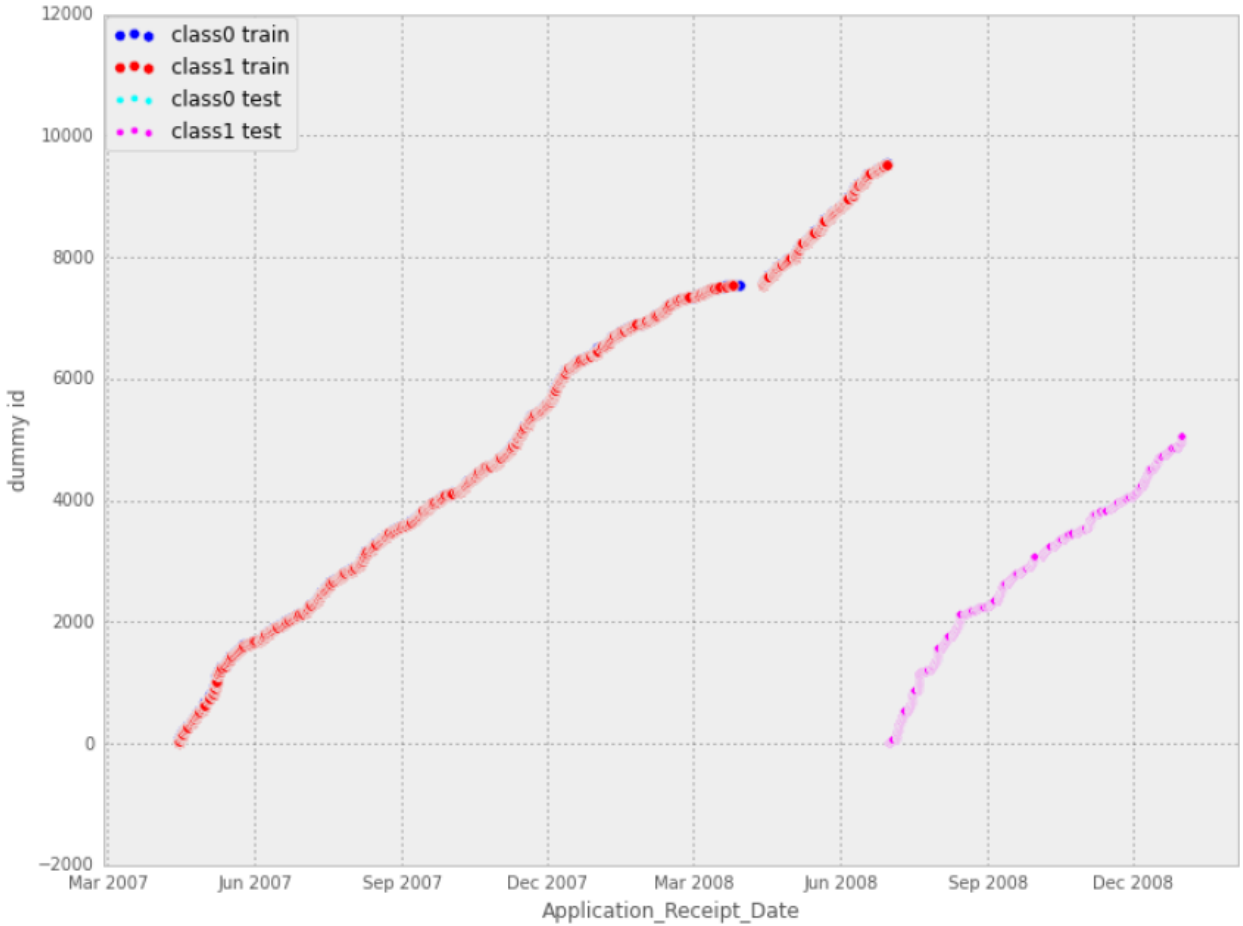
Можно смотреть на стабильность признаков!
Как меняются распределения признаков / значение целевого

Это позволяет:
– выявить стабильные признаки (data shift, target shift)
– правильно сформировать разбиения обучение / тест



Приём: по старой истории кодировать признаки, по новой обучать!

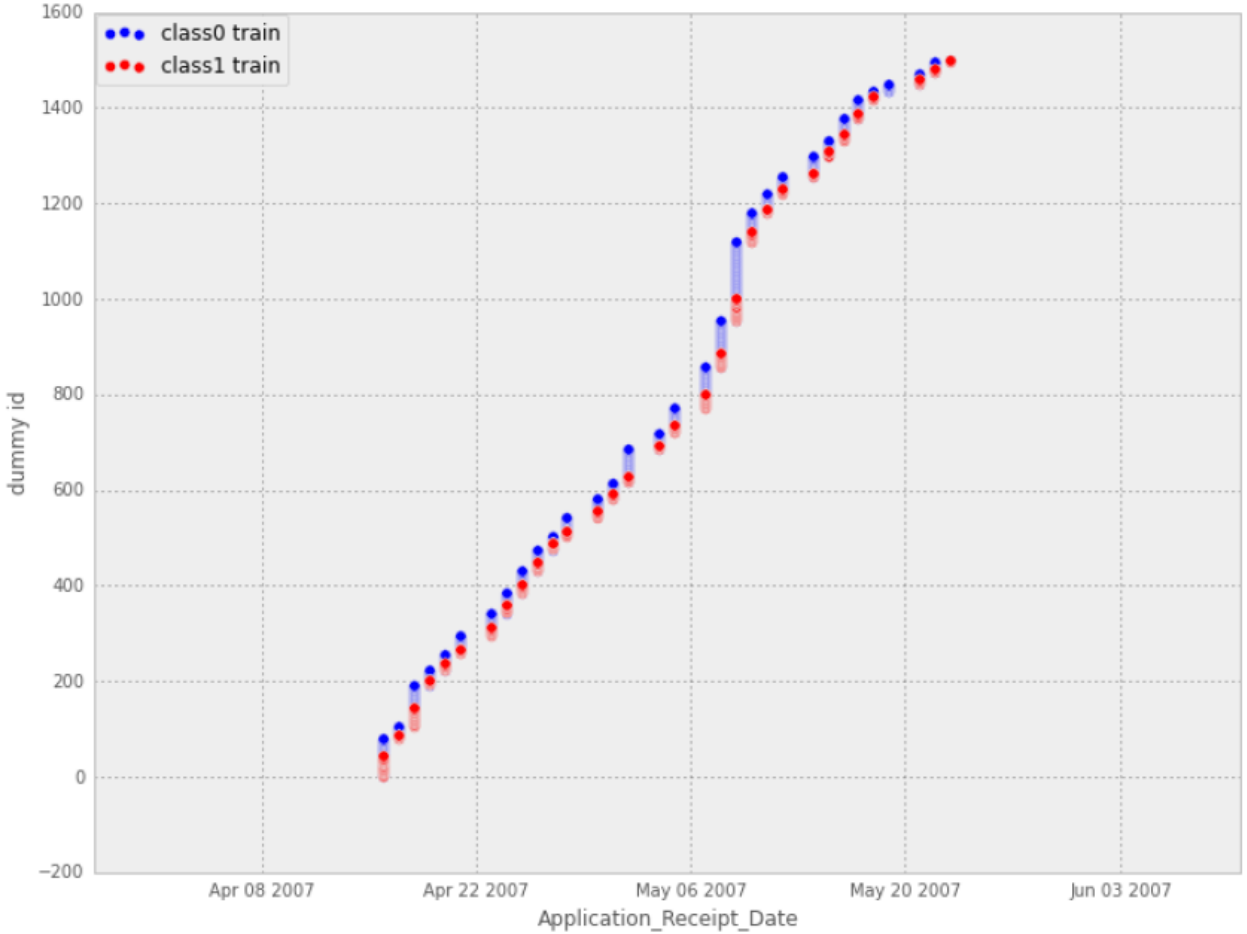
Визуализация id (номер в таблице) — время



Позволяет много чего выявить.

Здесь — разрыв, разную скорость заполнения данными

Временные признаки



Увеличив предыдущий рисунок, можно увидеть «утечку» — в течении дня class 1 получал больший id.

Временные признаки

Географические (пространственные) признаки: Spatial Variables

— отражают локализацию в пространстве

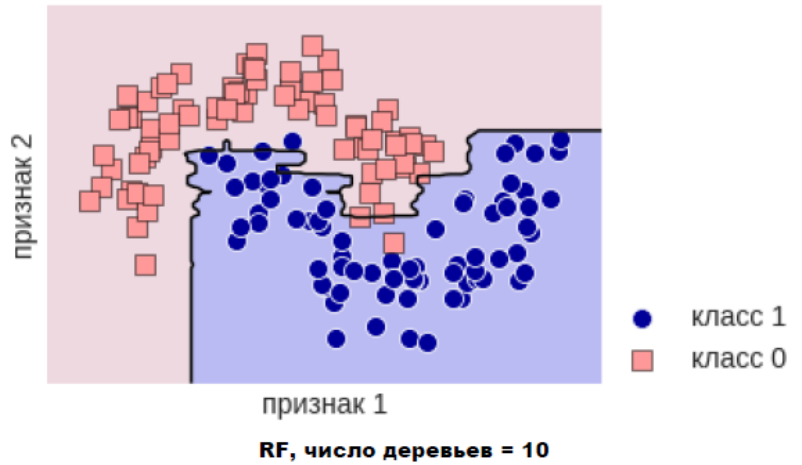
- координаты
- города
- страны
- адреса
- траектории / скорости перемещения и т.п.

Можно сконвертировать в координаты

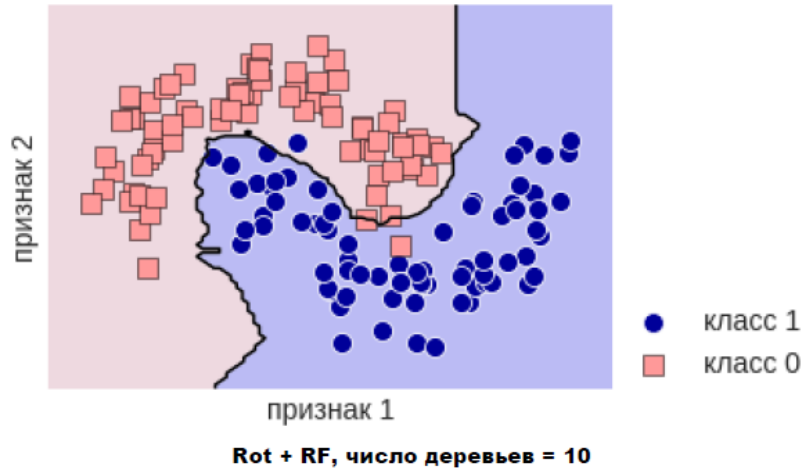
Географические (пространственные) признаки

- **Проекции на разные оси**
  чтобы реализовывались более сложные «поверхности разделения»
- **Кластеризация**
  чтобы выделить отдельные регионы
- **Идентификация, привязка, характеристики окрестности**
  в случае точных координат:
  - где находится объект
  - какие объекты также рядом (плотность объектов)
  - что ещё рядом
- **Анализ траекторий**
  если изменение координат во времени
- **Деанонимизация данных**
  часто география «вскрывает» местоположение
- **Исследование странностей**
  телепортации, слишком частые координаты и т.п.

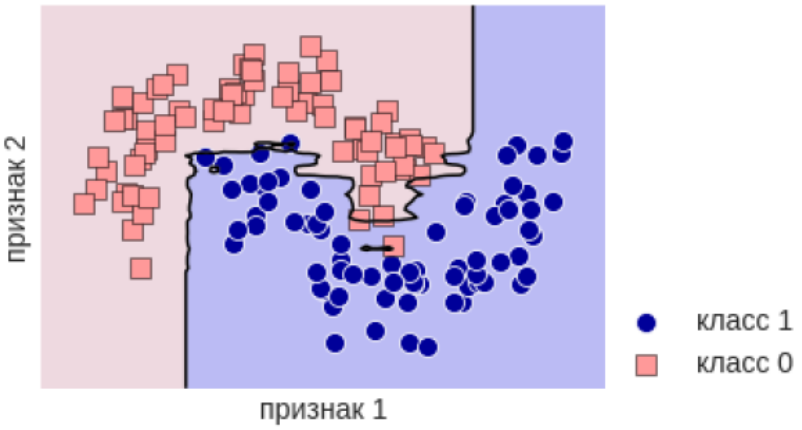
Проекции на разные оси



RF, число деревьев = 10



Rot + RF, число деревьев = 10



RF, число деревьев = 1000

Rot + RF, число деревьев = 1000

Временные признаки

Генерация признаков

- **расстояния до**
  - **объектов** (дорога, АЗС, метро, банк, школа, остановка, магазин, город и т.п.)
  - **вычисленных объектов** (самого дорогого дома, скопления людей)
  - **границ**
  - **кластеров**
- **использовать для генерации других признаков**

  средняя цена квартиры в районе, число школ в районе, плотность населения

Временные признаки

Пример: анализ трафика и конверсии (процент клиентов, которые совершили целевое действие) в различных точках продаж

Временные признаки

Данные заказчика

- **статистика посещений**
  - визиты
  - покупки/конверсия
  - ...
- **данные магазина**
  - площадь
  - персонал
  - категория
  - ...

Наши данные (анализ окрестности)

- наличие остановок / метро
- конкуренты
- где находится магазин
- численность населения

Другое

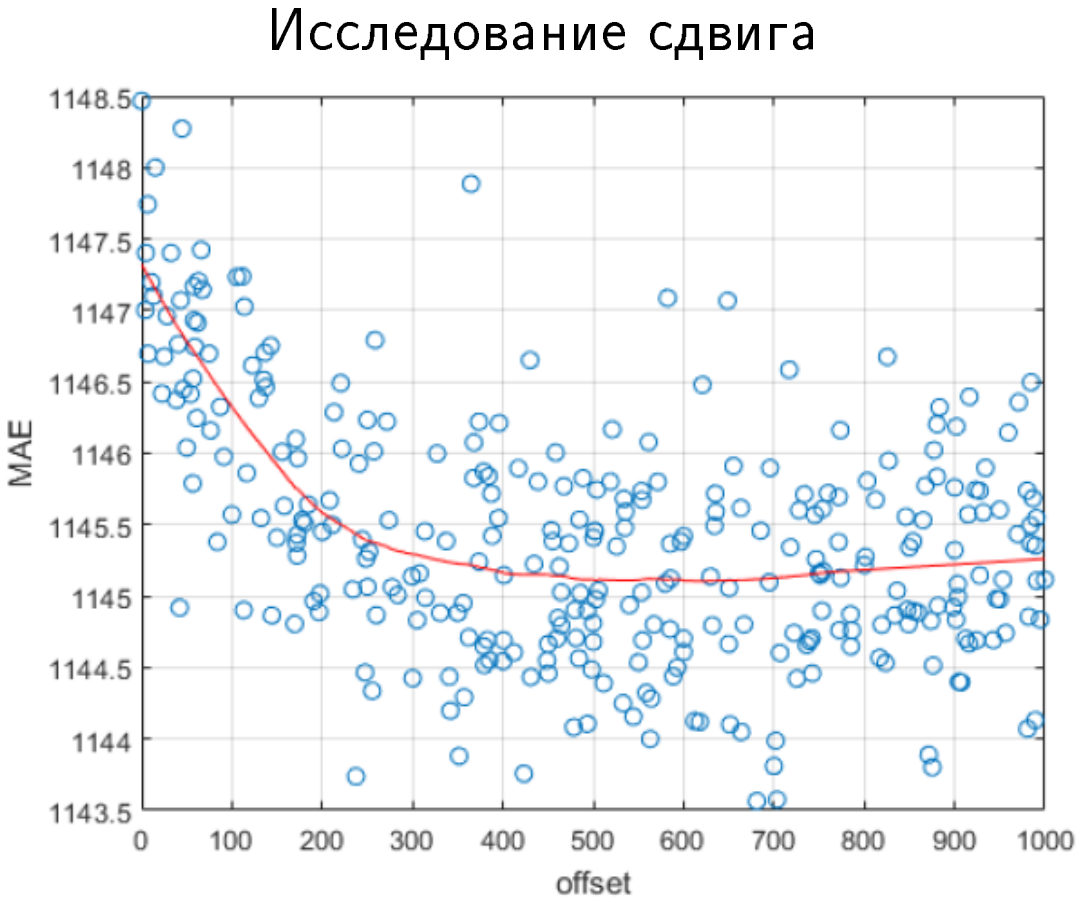
Финансовые признаки (деньги), количество и т.п.

- абсолютные суммы $\rightarrow$ относительные
  стоимость, площадь $\rightarrow$ стоимость кв. м.
- остатки от деления / округления

	price	f_p
0	12.50	0.50
1	0.99	0.99
2	10.00	0.00
3	18.01	0.01

  позволяет отделить точные от приблизительных
- статистики транзакций

Генерация целевого признака



Преобразование целевого признака $\log(y + \text{offset})$

Другие примеры при генерации признаков

Использование контекста

бинарные «первый/последний этаж», «балкон» и т. п.
LTV (loan to value) — отношение суммы кредита к стоимости жилья

Признак «значение плотности признака»

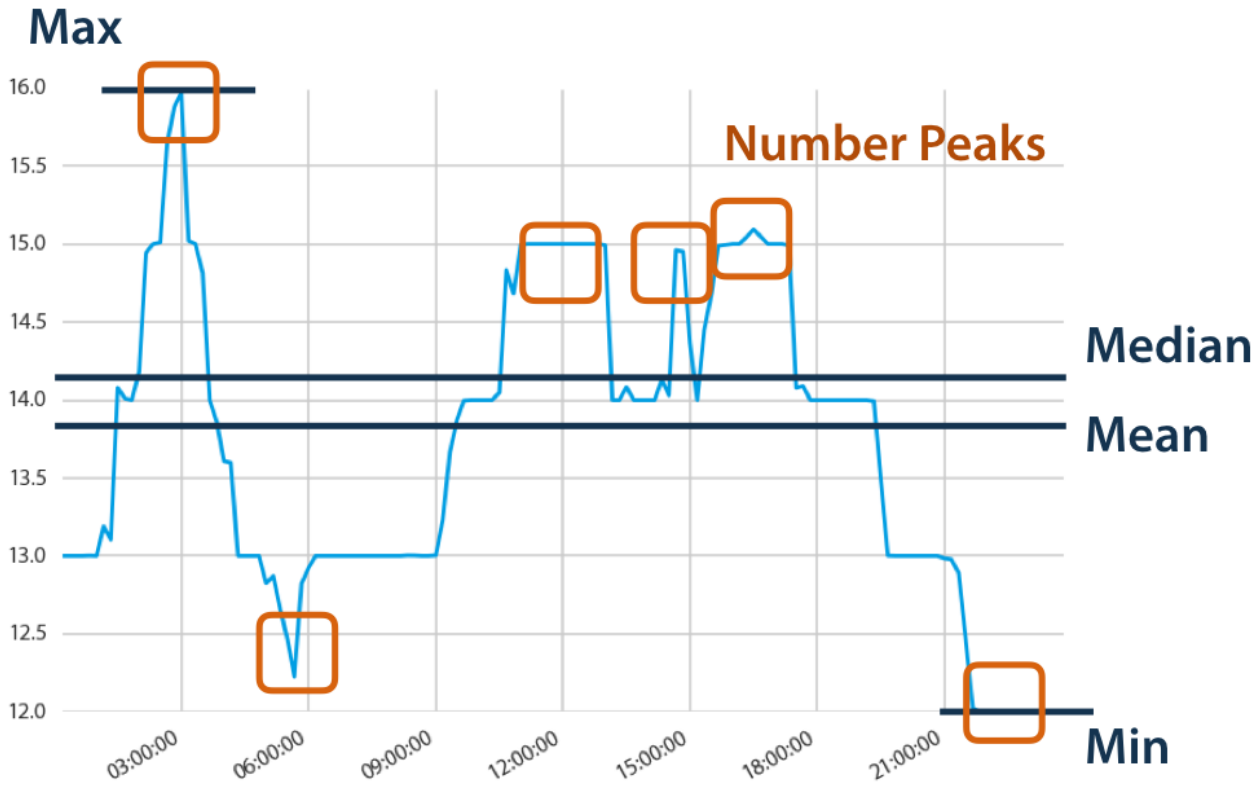
строим непараметрическую оценку плотности и новый признак — значение оценки на объекте $\widehat{f}(x)$

Текстовые признаки

длина предложения, количество существительных, знаки препинания и т. д.

Обработка текстов — отдельная тема

Признаки для временных рядов



а также выраженность тренда, линейность, сила сезонной компоненты и т. д.

Анализ временных рядов — отдельная тема

Pseudo labelling

Pseudo labelling (псевдо-маркировка)

Pseudo labelling является методом с использованием неразмеченных (в некоторых случаях тестовых) данных вместе с размеченными (обучающей выборкой) для создания более совершенных алгоритмов.

Основная идея заимствована из semi-supervised learning https://www.researchgate.net/publication/280581078

Идея в чем-то напоминает стекинг, но реализована по-другому.

Чаще применяется когда размеченных данных мало. Не достаточность данных для обучения видно на кривой обучения, которая показывает как быстро растет точность по мере увеличения объёма обучающей выборки.

Pseudo labelling

Как можно использовать данные с метками и без совместно?

Для этого мы можем построить двухэтапную процедуру:

- на этапе 1 мы обучаем модель только на данных train и прогнозируем для неразмеченных (например, test)
- на этапе 2 мы используем прогнозы из этапа 1 как «псевдо» метки для неразмеченных данных, объединяем данные train+test и заново обучаем модель для этого объединённого набора данных.

Эту процедуру можно повторять много раз до тех пор, пока точность продолжает расти.

Всегда ли это работает?

Есть несколько важных моментов, которые нужно учитывать

- Прежде всего, **модель из этапа 1 должна быть достаточно хорошей**, так как мы создаем прогнозы для тестовых данных с помощью модели и используем их в качестве псевдо-меток
- Во-вторых, **объединённый набор данных для этапа 2 должен быть сбалансирован** как 70% на 30% или, в самом крайнем случае, 50% на 50%, если мы говорим о реальных метках и псевдо-метках соответственно. Это поможет вашей модели по-прежнему учиться на реальных метках и настраиваться с помощью псевдо-меток. Если псевдо-часть будет слишком большой, модель примет её как настоящие метки и обучится на ошибках из прогнозов этапа 1

Pseudo labelling

- В-третьих, во время обучения на уровне второго этапа необходимо **вычислять функционал качества обучения только по реальным меткам**. Например, можно проделать это так: указать веса для объектов — 1.01 для реальных объектов и 0.99 для объектов с псевдо-метками, а когда придет время вычислять метрику, берем только объекты с весом, равным 1.01
- Если это возможно, **модели 1-го и 2-го этапа должны быть разными**. В этом случае модель, которую вы обучаете на 2-м уровне, научится чему-то новому у модели на 1-м уровне. Модели из того же семейства тоже делают свою работу, но обычно дают меньшее улучшение
- При неоднократном применении процедуры важно сохранять обучающей набор в неизменном виде. От шага к шагу должен меняться только прогноз для неразмеченной части данных

Pseudo labelling

Ещё несколько замечаний.

Для получения псевдо-меток можно использовать чужой алгоритм, даже тот про который ничего неизвестно кроме возможности построения прогнозов

Число повторений двухэтапной процедуры до выхода на плато как правило невелико

В соревнованиях это практически всегда позволяет улучшить качество алгоритма

В бизнес-приложениях бывает полезнее найти те примеры, на которых алгоритм выдаёт неуверенные предсказания, и вручную доразметить именно их

Pseudo labelling

В бизнес-задачах неразмеченные данные могут быть не только тестовыми, но и данными, на которые у вас нет бюджета или времени для маркировки. В крупных компаниях это довольно распространённая ситуация, например, вы использовали только часть своей клиентской базы для обучения и построения модели на ней впоследствии. В этой ситуации нет меток для всех клиентов.

Так же смотри Semi-Supervised Learning и Self-Supervised Learning

https://github.com/yassouali/awesome-semi-supervised-learning

https://youtu.be/GBgns_U9mmI

https://github.com/jason718/awesome-self-supervised-learning

Другие приемы создание признаков:

- смотреть ошибки алгоритма на очевидных примерах https://habr.com/ru/post/419885/ главы 14-19
- AutoFE https://www.kaggle.com/liananapalkova/automated-feature-engineering-for-titanic-dataset
- AutoML

Книги

- http://www.feat.engineering/index.html
- Alice Zheng, Amanda Casari. Feature Engineering for Machine Learning, Principles and Techniques for Data Scientists // O'Reilly Media, 2018.
- Feature Engineering for Machine Learning and Data Analytics. Edited by Guozhu Dong and Huan Liu. CRC Press, 2018.
- Max Kuhn, Kjell Johnson. Feature Engineering and Selection: A Practical Approach for Predictive Models. CRC Press, 2020.
- Soledad Galli. Python Feature Engineering Cookbook. Packt Publishing, 2020.
- Pablo Duboue. The Art of Feature Engineering: Essentials for Machine Learning. Cambridge University Press, 2020.
- Курс «How to Win a Data Science Competition: Learn from Top Kagglers» https://ru.coursera.org/learn/competitive-data-science

Полезные ссылки

https://github.com/alteryx/featuretool — автоматическая генерация признаков из нескольких

https://arxiv.org/abs/1904.12857 — генерация пересечений категориальных признаков

Серия постов Understanding Feature Engineering
https://towardsdatascience.com/understanding-feature-engineering-part-1-continuous-numeric-data-da4e47099a7b

https://nagornyy.me/courses/data-science/feature_engineering/

https://blog.dominodatalab.com/manual-feature-engineering/

https://bloomberg.github.io/foml/#lecture-12-feature-extraction

https://mlcourse.ai/articles/topic6-features/

https://github.com/dipanjanS/practical-machine-learning-with-python/tree/master/notebooks/Ch04_Feature_Engineering_and_Selection

Заключение

Примеры

- Задача с 3 основными признаками
  https://youtu.be/FPAT3YkY7EE
  пара решений с большим числом новых признаков
  https://github.com/RaevskyDN/wildfire-ai
  https://github.com/VovaForbes/WildFire
- Categorical Encoding Benchmark
- Определение вероятности невозврата кредита
  Постановка задачи https://dyakonov.org/2017/12/25/
  Хороший код решения (3-е место)
  https://github.com/Dyakonov/sascompetitions
  1-е место (идея решения) https://youtu.be/FahVtbxsEzA