Искусственный Интеллект

Лекция 3: Линейная регрессия. Метрики модели. Регуляризация. Подбор гиперпараметров. Разбиение данных.

Мартынюк Полина Антоновна

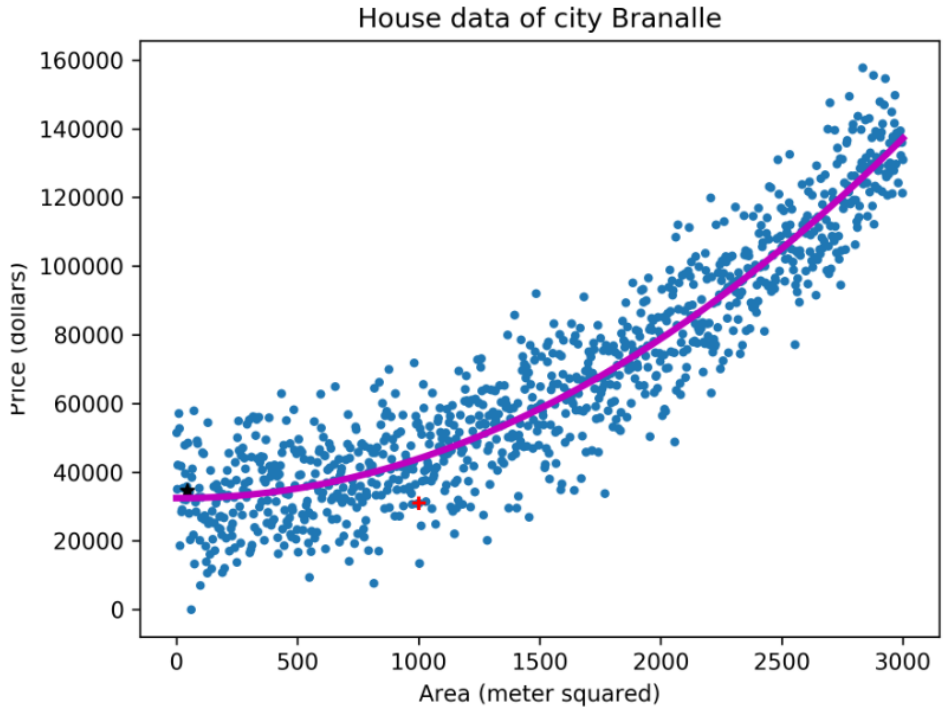
telegram: @PAMartynyuk

email: pa-martynyuk@yandex.ru

В предыдущих сериях…

Регрессия

- **Регрессия – предсказание числового значения**
  по входным данным.



- Функция модели в общем виде:

1) Смещение – *b*

$$y = \sum_{i=1}^{p} (x_i w_i) + b$$

2) Смещение – w_0

$$y = \sum_{i=0}^{p} (x_i w_i)$$

$$x_0 = 1$$

Регрессия: обучение модели

- Функция L (Loss) – **Функция потерь**

$$L(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$\hat{y} = \sum_{i=1}^{p}(x_i \hat{w}_i) + \hat{b}$$

$$L(w_1, \ldots, w_p, b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (w_1 x_1 + \ldots + w_p x_p + b))^2$$

- **Задача обучения – найти такие параметры модели (W, b), при которых значение функции потерь будет минимальным**

$$W, b = argmin(L(w_1, \ldots, w_p, b))$$

Регрессия: градиентный спуск

- Алгоритм градиентного спуска

$$w_1, \ldots, w_p := 0$$
$$b := 0$$
$$for\ i\ in\ range(n\_iter):$$
$$w_1 := w_1 - \alpha \frac{\partial L}{\partial w_1}$$
$$\ldots$$
$$w_p := w_p - \alpha \frac{\partial L}{\partial w_p}$$
$$b := b - \alpha \frac{\partial L}{\partial b}$$

Проход по всем экземплярам данных в обучающей выборке - ***эпоха***

Гиперпараметры

n_iter - число эпох

α - скорость обучения

Регрессия: метрики качества

В общем случае метрика качества не совпадает с функцией потерь, и нужно уметь различать эти понятия:

- **Метрика** — внешний критерий качества, зависящий от предсказанных значений, но не от параметров модели.
- **Функция потерь** — критерий «обучаемости» нашей модели. Возникает, когда мы переходим от построения модели к задаче оптимизации, то есть поиску оптимальных параметров модели.

$$MSE = \frac{1}{n}\sum(\hat{y} - y)^2$$

$$L(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Регрессия: метрики качества

В чем проблема MSE?

$$MSE = \frac{1}{n}\sum(\hat{y} - y)^2$$

Пусть MSE = 100

Какая модель при таком значении работает лучше?

- Если целевое значение *y* лежит в диапазоне [0; 20 000]
- Если целевое значение *y* лежит в диапазоне [0; 200]

Получаемое значение **нельзя рассматривать в отрыве от исходных данных**

Регрессия: метрики качества

Метрика	Плюсы	Минусы
Среднеквадратичная ошибка $$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$	Позволяет накладывать большие штрафы на сильные отклонения, практически обнуляя соответствующие веса признаков	Нестабильна при сильных отклонениях в выборке
Средняя абсолютная ошибка $$MAE = \frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$$	Легко интерпретировать	В отличие от MSE, не штрафует за сильные отклонения
Средняя абсолютная ошибка в процентах $$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{\lvert y_i - \hat{y}_i\rvert}{y_i}$$	Легко интерпретировать	Нельзя использовать, если есть нулевые или близкие к нулю значения, также нет верхнего предела процентной ошибки для слишком больших предсказаний — из-за отклонений в выборке можно получить ошибку в сотни процентов

Регрессия: метрики качества

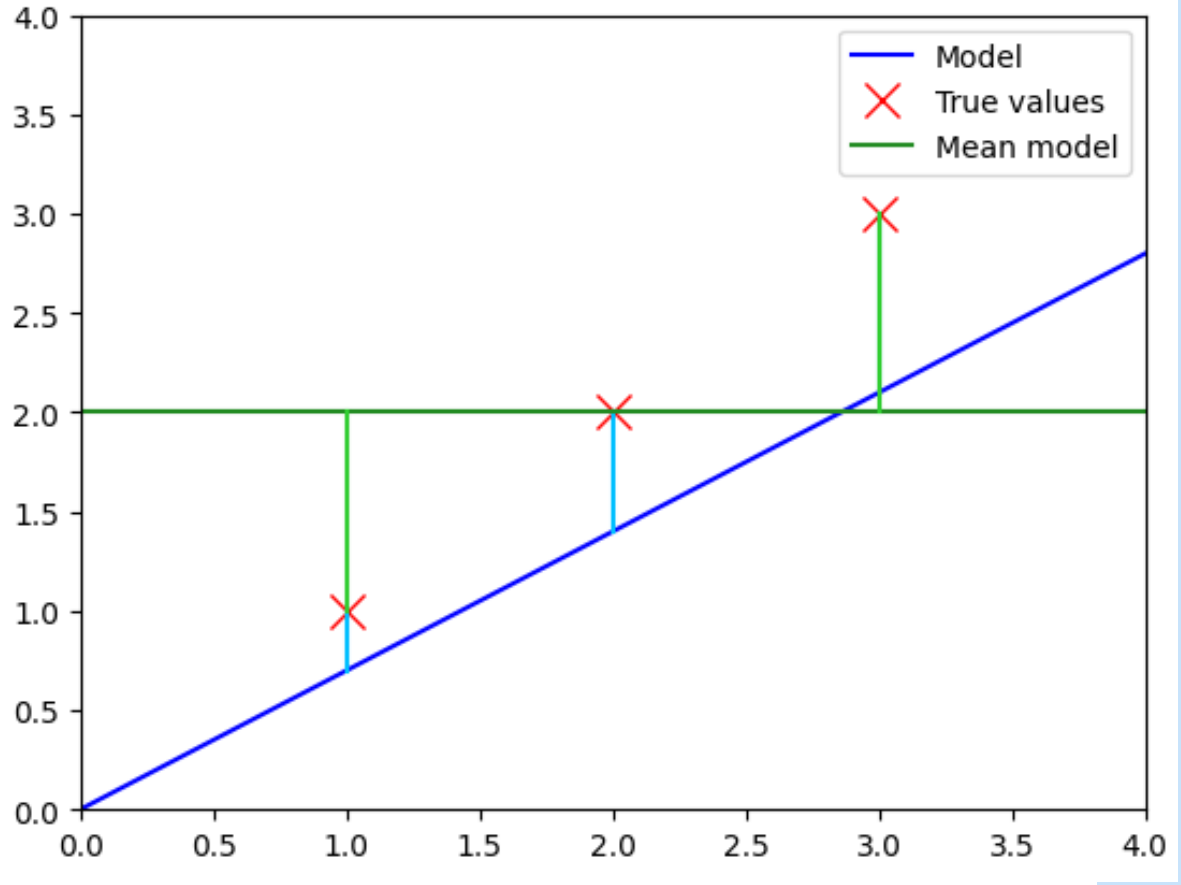
- **R^2** (коэффициент детерминации)

Оцениваемая модель

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

Константная модель, которая всегда предсказывает среднее значение *y*

$$R^2 = 1 - \frac{MSE_{model}}{MSE_{avg}}$$



Регрессия: метрики качества

- Метрика R^2 максимальна при MSE = 0.
- Значение R^2 меньше 0 означает, что рассматриваемая модель *хуже* константой.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

$$R^2 = 1 - \frac{MSE_{model}}{MSE_{avg}}$$

$$E(R^2) \in (-\infty; 1]$$

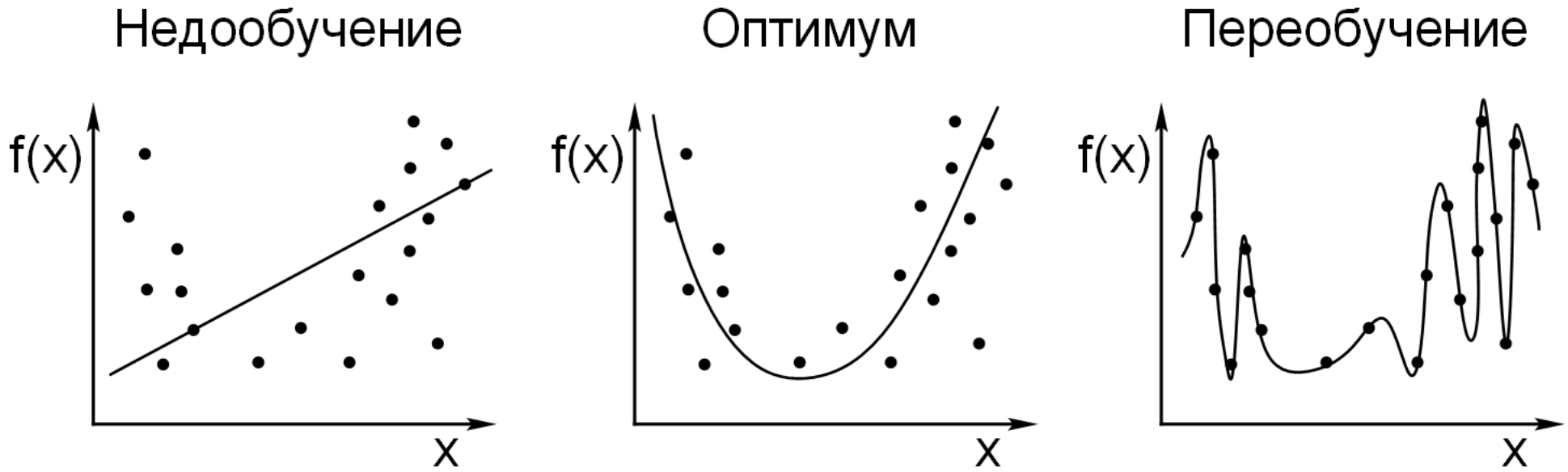
Регрессия: подбор параметров

- **Веса** (W, b) **модель подбирает сама** в процессе обучения на данных
- **Гиперпараметры подбирает разработчик**
  - Число эпох
    n_iter
  - Скорость обучения
    α

$$
\begin{aligned}
&w_1, \ldots, w_p := 0 \\
&b := 0 \\
&for\ i\ in\ range(n\_iter): \\
&\quad w_1 := w_1 - \alpha \frac{\partial L}{\partial w_1} \\
&\quad \ldots \\
&\quad w_p := w_p - \alpha \frac{\partial L}{\partial w_p} \\
&\quad b := b - \alpha \frac{\partial L}{\partial b}
\end{aligned}
$$

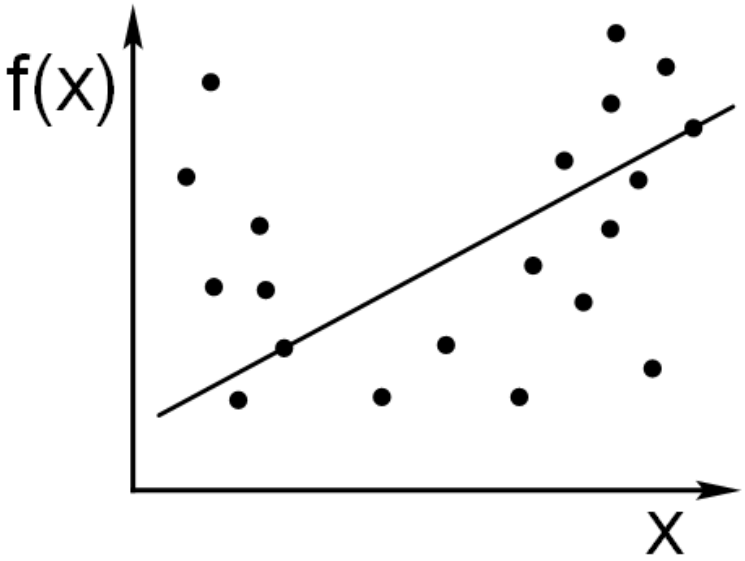
Регуляризация / Regularization

- Главная цель модели - **обобщиться** под все экземпляры данных, которые могут встретиться.
- Модель может **переобучиться**, если слишком хорошо выучит тренировочный датасет. Из-за этого результаты на настоящих данных будут хуже.
- **Регуляризация** не позволяет модели быть слишком "сложной" или слишком "самоуверенной".

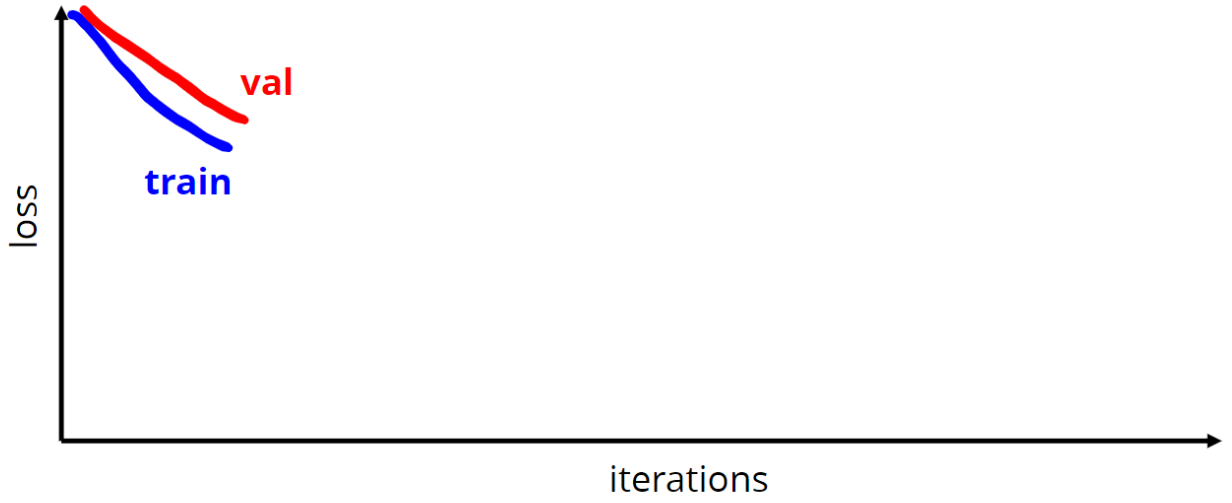


Регуляризация / Regularization

Недообучение

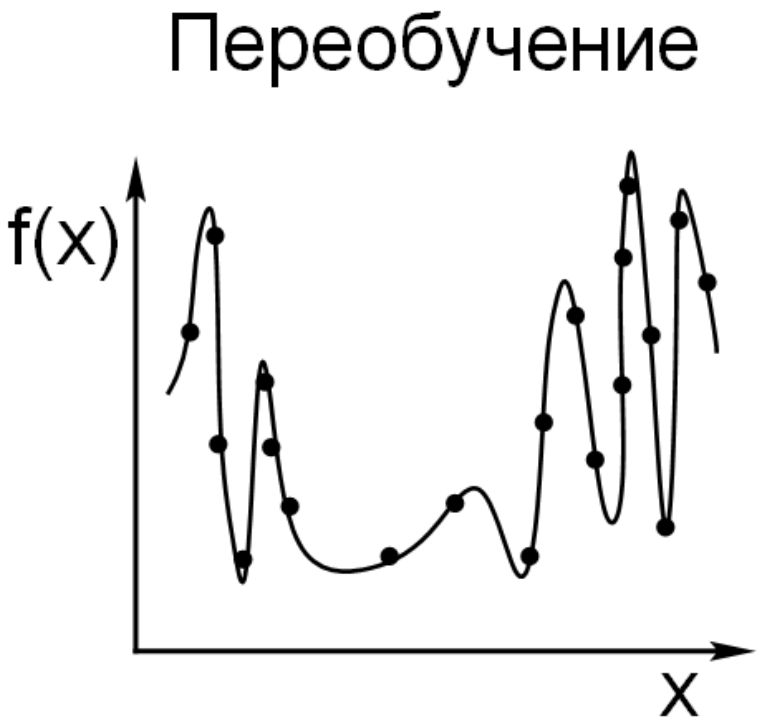


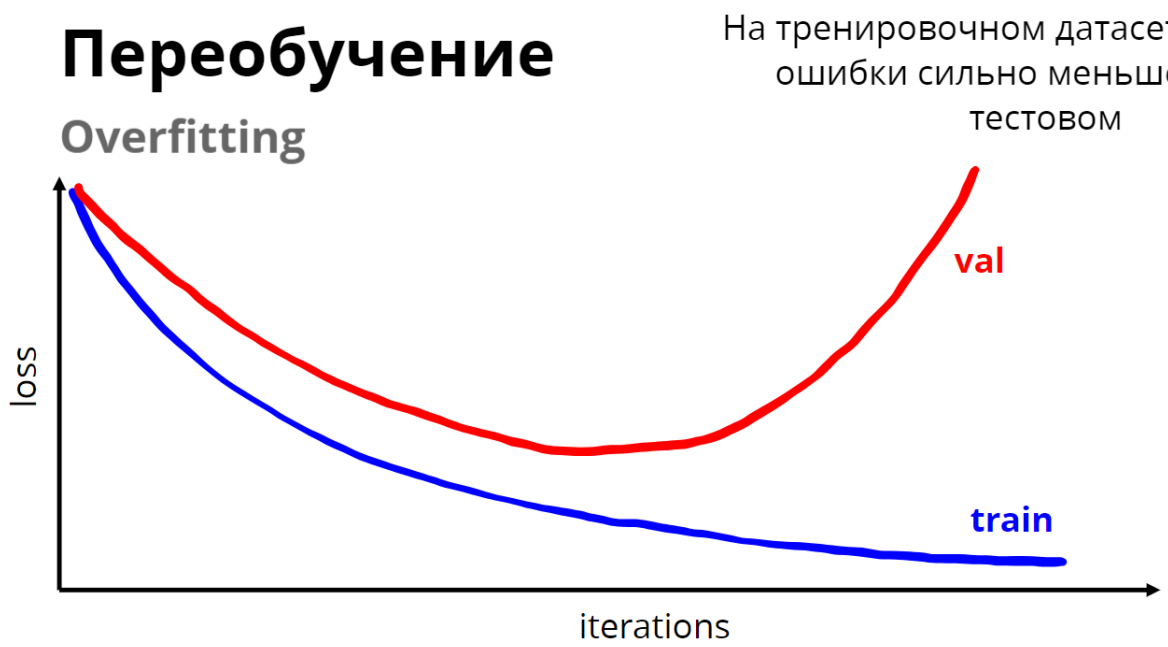
Недообучение



И на тренировочном, и на тестовом датасете значение ошибки велико

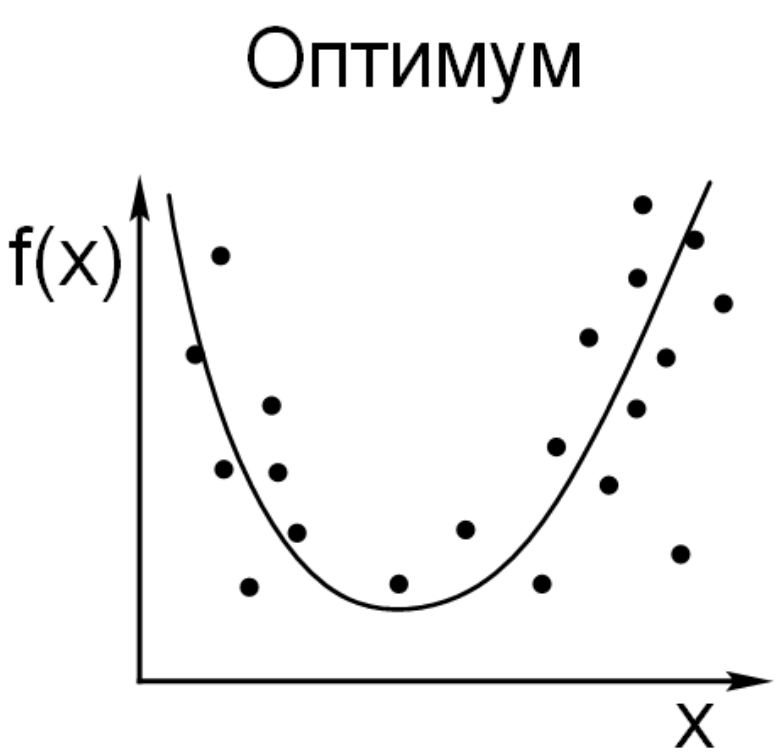
Регуляризация / Regularization





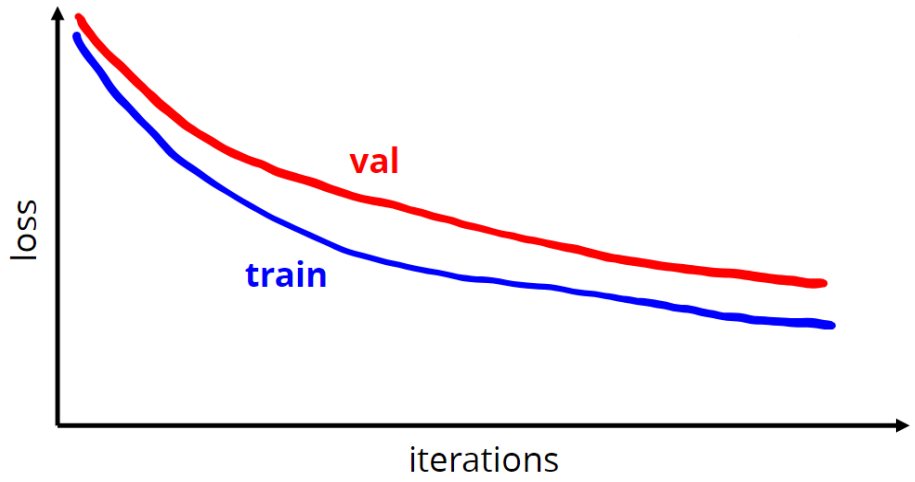
На тренировочном датасете значение ошибки сильно меньше, чем на тестовом

Регуляризация / Regularization



Нормальный вариант

Ошибка на тестовых данных не сильно отличается от ошибки на тестовых данных



Регуляризация / Regularization

- **Из-за чего возникает переобучение?**

Все Марии покупают Волгу?

Имя	Доход	Марка машины
Мария	70 000	Волга
Алексей	150 000	BMW
Михаил	100 000	Dodge
Анастасия	200 000	Renault
Мария	140 000	Волга
Colin McRae	300 000	Subaru
Василий	45 000	Volkswagen
Мария	80 000	Волга
Сергей	100 000	Renault

Регуляризация / Regularization

- **Проблемы с весами**

  Переобучение происходит, если модель слишком сильно полагается только на один параметр, то есть вес одного признака “затмевает” остальные

$$w_{\text{имя}} x_{\text{имя}} + w_{\text{доход}} x_{\text{доход}} + b = y$$

$$w_{\text{имя}} \to \infty$$

$$w_{\text{доход}} \to 0$$

Регуляризация / Regularization

- **L2 регуляризация**
- **Ridge**

- Свободный член (смещение b) **не входит** в компонент регуляризации!

Процесс обучения:

$$w_1 := 0$$
$$w_2 := 0$$
$$b := 0$$
$$for\ i\ in\ range(n\_iter):$$
$$w_1 := w_1 - \alpha\frac{\partial L}{\partial w_1} - 2\lambda w_1$$
$$w_2 := w_2 - \alpha\frac{\partial L}{\partial w_1} - 2\lambda w_2$$
$$b := b - \alpha\frac{\partial L}{\partial b}$$

$$L_{ridge}(y, \hat{y}) = L(y, \hat{y}) + \lambda \sum_{i=1}^{p} {w_i}^2$$

Регуляризация / Regularization

- **L1 регуляризация**
- **Lasso (Least Absolute Shrinkage and Selection Operator)**

- Свободный член (смещение b) **не входит** в компонент регуляризации!

Процесс обучения:

$$w_1 := 0$$
$$w_2 := 0$$
$$b := 0$$
$$for\ i\ in\ range(n\_iter):$$
$$w_1 := w_1 - \alpha\frac{\partial L}{\partial w_1} - \lambda\ sign(w_1)$$
$$w_2 := w_2 - \alpha\frac{\partial L}{\partial w_2} - \lambda\ sign(w_2)$$
$$b := b - \alpha\frac{\partial L}{\partial b}$$

$$sign(x) = \begin{cases} 1, & \text{если } x > 0; \\ 0, & \text{если } x = 0; \\ -1, & \text{если } x < 0. \end{cases}$$

$$L_{lasso}(y, \hat{y}) = L(y, \hat{y}) + \lambda \sum_{i=1}^{p} |w_i|$$

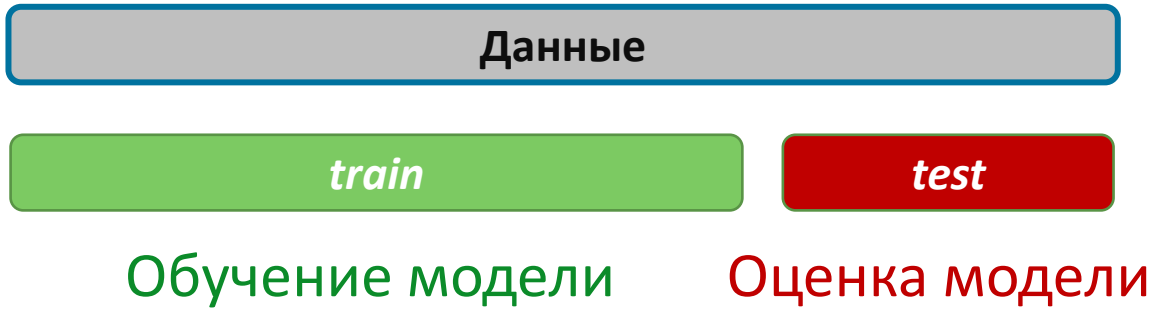
Регуляризация / Regularization

- **Параметр регуляризации** λ (lambda)
- **В sklearn используется обратный параметр** $C = \frac{1}{\lambda}$

L2 / Ridge	L1 / Lasso
Неразреженный результат	Разреженный результат
Нет отбора фич	Есть отбор фич

Разбиение данных: Hold-out

Метод **hold-out** представляет из себя простое разделение на train и test:




```
import numpy as np
from sklearn.model_selection import train_test_split

X, y = np.arange(1000).reshape((500, 2)), np.arange(500)

X_train, X_test, y_train, y_test = train_test_split(
        X,
        y,
        test_size=0.2,
        random_state=42
)
```

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

Разбиение данных: Hold-out

```
# np.arange(n) возвращает равномерно распределенные значения в пределах заданного интервала.
print(np.arange(3))
'''
Output: [1 2 3]
'''
```

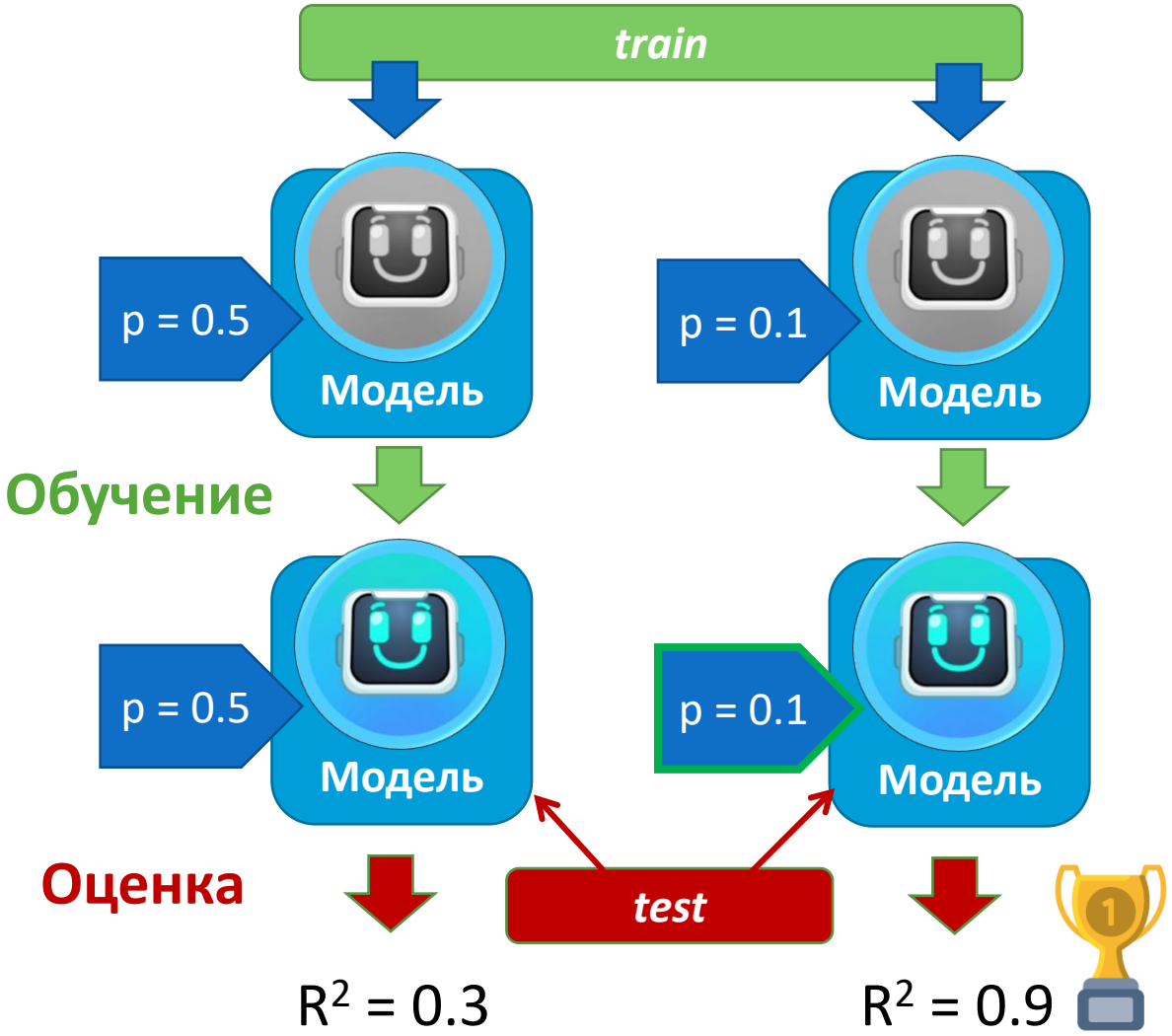
https://numpy.org/doc/stable/reference/generated/numpy.arange.html

```
# array.reshape((n, m)) придает массиву новую форму, не изменяя его данные.
a = np.arange(6).reshape((3, 2))
print(a)
'''
Output:
[[0 1]
 [2 3]
 [4 5]]
'''
```

https://numpy.org/doc/stable/reference/generated/numpy.reshape.html#numpy-reshape

Регрессия: подбор гиперпараметров

1. Подбор гиперпараметров



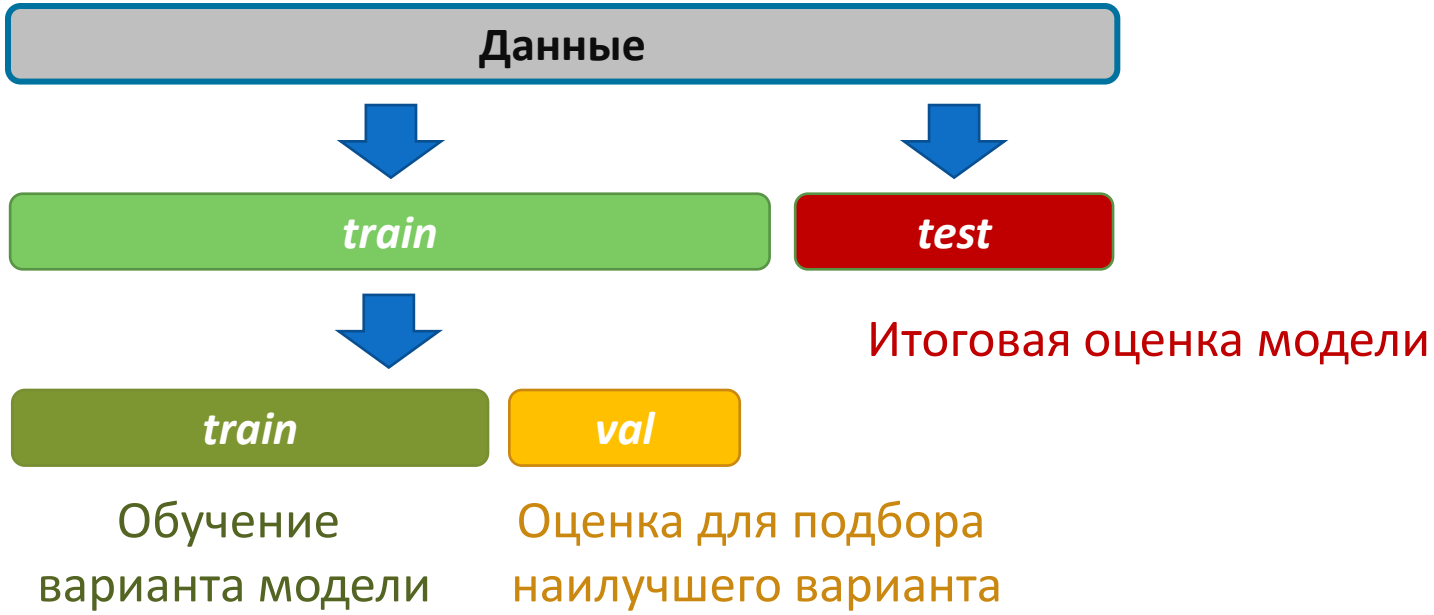
2. Оценка итоговой модели

Регрессия: подбор гиперпараметров

Тестовое множество не должно использоваться в процессе обучения, поскольку это нарушает чистоту обучения(в модель через подобранные гиперпараметры просачивается информация о тестовой выборке)

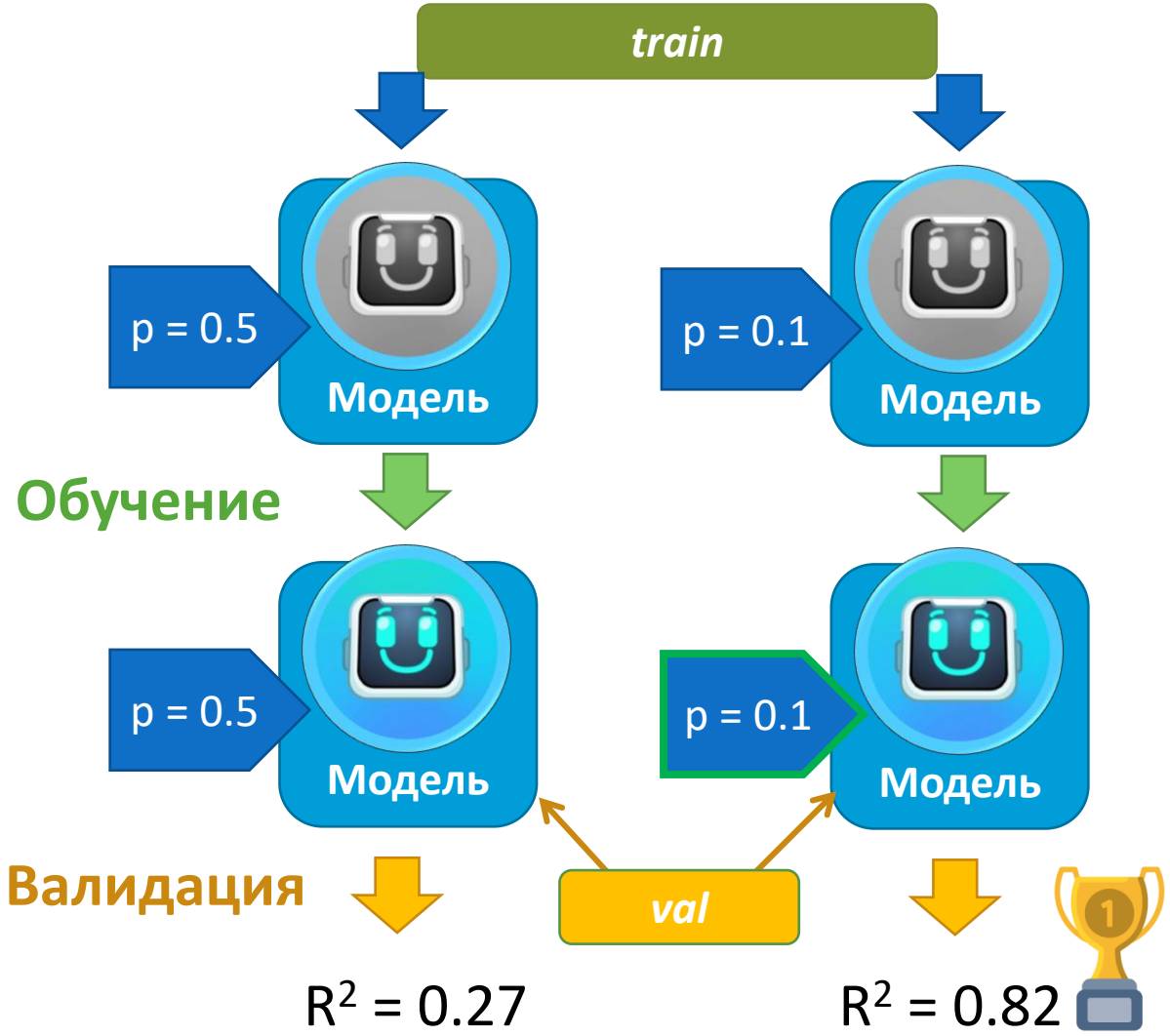
Как решить проблему?

- Выделить свой test в рамках train

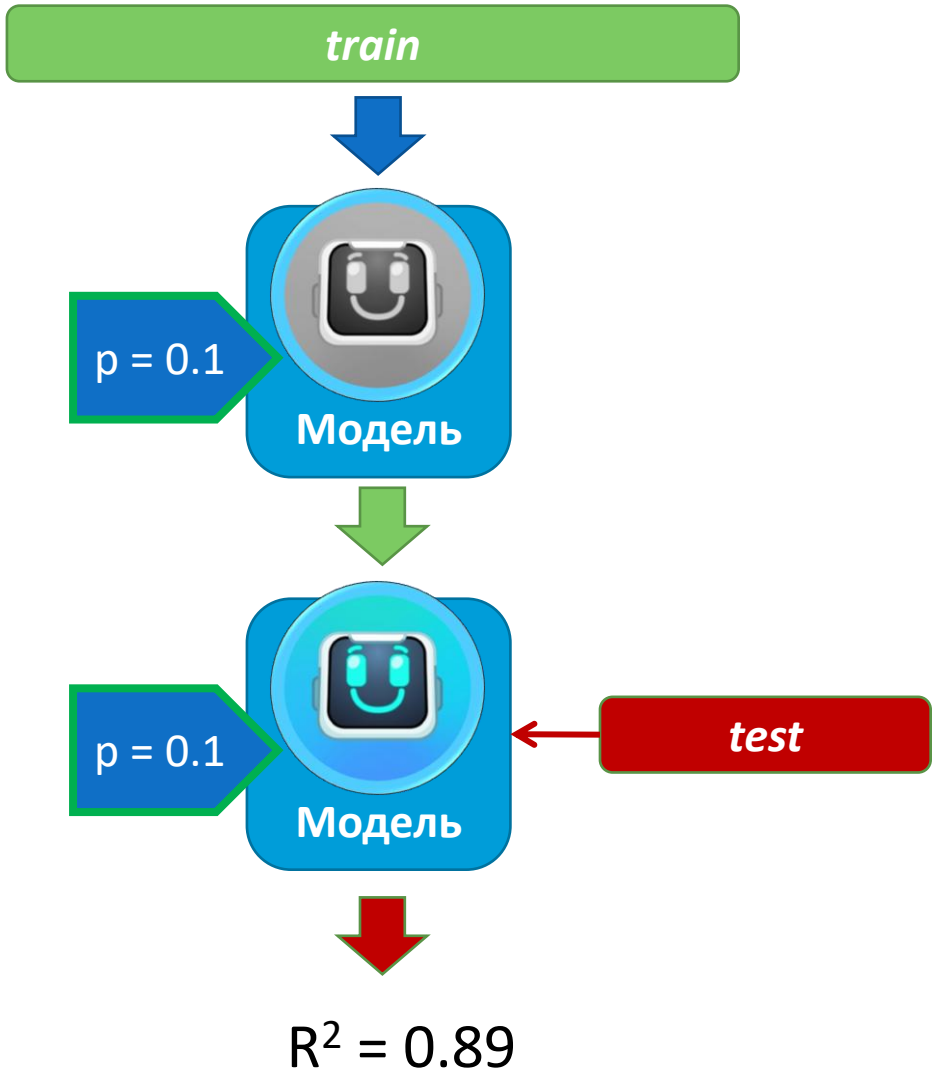


Регрессия: подбор гиперпараметров

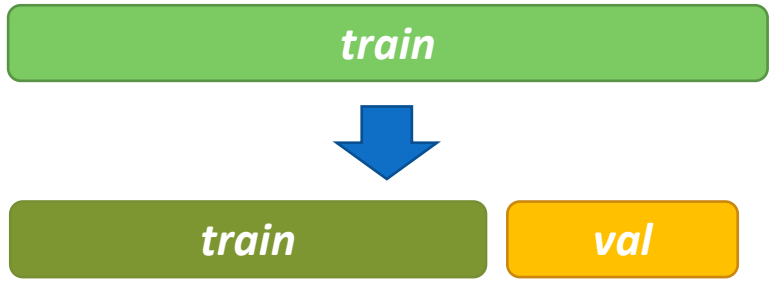
1. Подбор гиперпараметров



2. Оценка итоговой модели

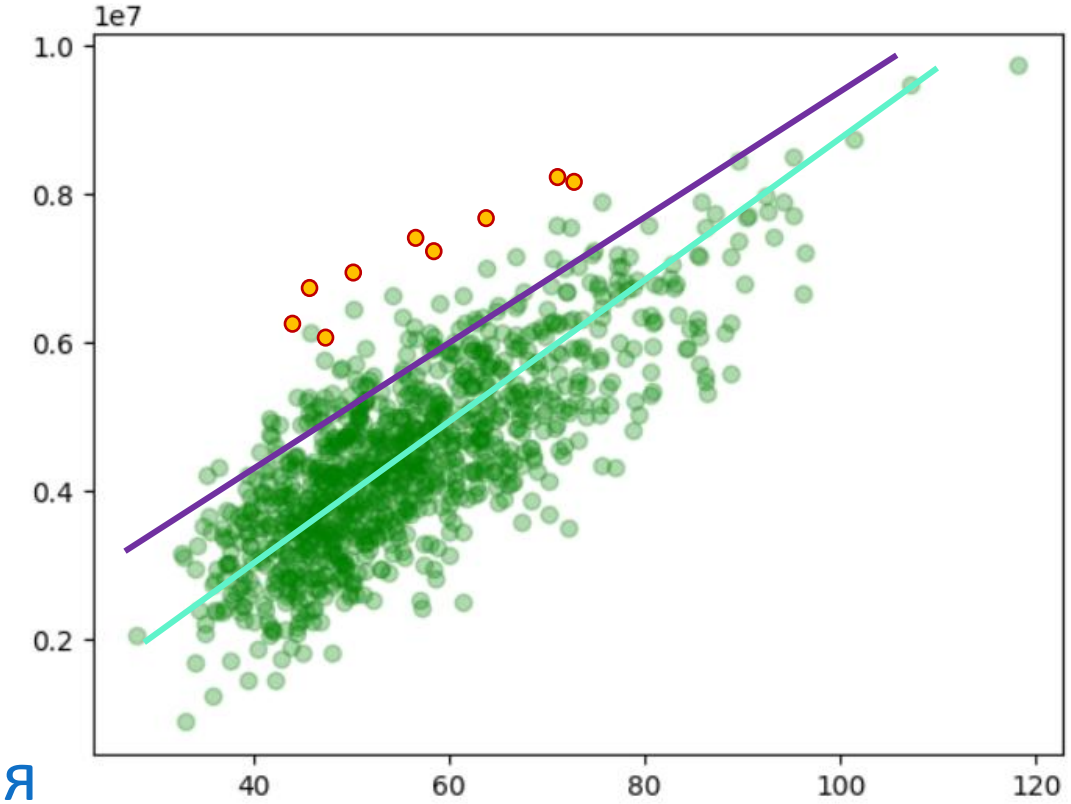


Разбиение данных



Обучение варианта модели

Оценка для подбора наилучшего варианта



Что может пойти не так?

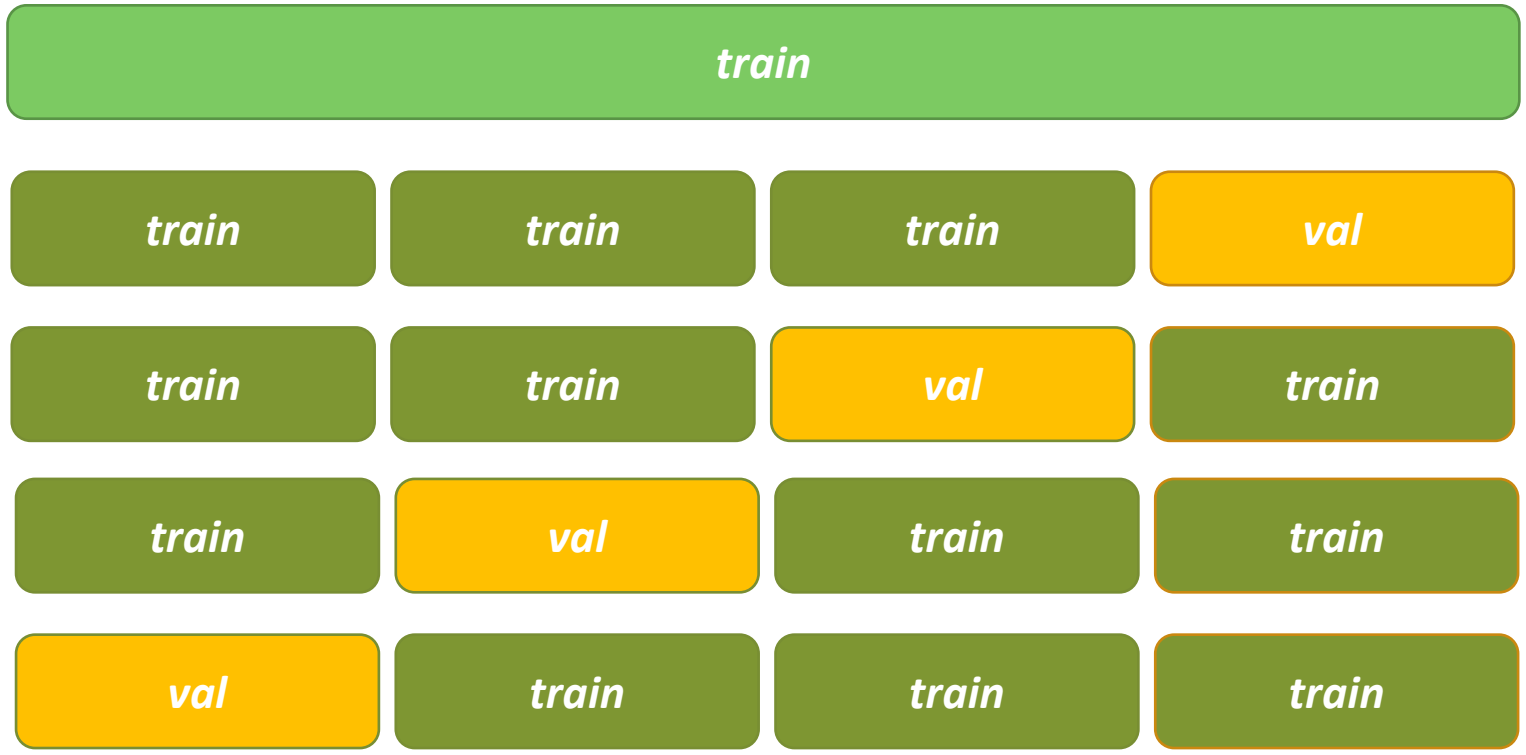
- Неравномерное распределение примеров данных

Решение – перекрёстная валидация

Разбиение данных: скользящий контроль или перекрёстная валидация

Cross validation (k-Fold) – перекрёстная валидация

- Разбиваем данные на k фолдов (k folds) и производим k итераций обучения и оценки модели. На каждом этапе получаем свою оценку модели. Итоговая оценка вычисляется как среднее оценок.



$$MSE = \frac{1}{n}\sum_{i=1}^{k} MSE_i$$

MSE_1

MSE_2

MSE_3

MSE_4

Оценка модели

Обучение модели

Разбиение данных: скользящий контроль или перекрёстная валидация

Cross validation – перекрёстная валидация

```
import numpy as np
from sklearn.model_selection import KFold

X = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]])
y = np.array([1, 2, 3, 4])
kf = KFold(n_splits=2)

for train_index, test_index in kf.split(X):
        print("TRAIN:", train_index, "TEST:", test_index)
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]

' ' '
result:
TRAIN: [2 3] TEST: [0 1]
TRAIN: [0 1] TEST: [2 3]
' ' '
```

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

Разбиение данных: скользящий контроль или перекрёстная валидация

Leave-one-out (LOO) — частный случай метода k-Fold: в нём каждый фолд состоит ровно из одного экземпляра данных.

```
import numpy as np
from sklearn.model_selection import LeaveOneOut

X = np.array([[1, 2], [3, 4], [5, 6]])
y = np.array([1, 2, 3])
loo = LeaveOneOut()

for train_index, test_index in loo.split(X):
        print("TRAIN:", train_index, "TEST:", test_index)
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]
' ' '
result:
TRAIN: [1 2] TEST: [0]
TRAIN: [0 2] TEST: [1]
TRAIN: [0 1] TEST: [2]
' ' '
```

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.LeaveOneOut.html

Разбиение данных: скользящий контроль или перекрёстная валидация

Стратифицированные фолды (Stratified k-Fold) - каждый фолд содержит примерно такое же соотношение классов, как и всё исходное множество. Такой подход может потребоваться в случае, например, очень несбалансированного соотношения классов

Разбиение данных: скользящий контроль или перекрёстная валидация

Стратифицированные фолды (Stratified k-Fold)

```
import numpy as np
from sklearn.model_selection import StratifiedKFold

X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])
y = np.array([0, 0, 1, 1])
skf = StratifiedKFold(n_splits=2)

for train_index, test_index in skf.split(X, y):
        print("TRAIN:", train_index, "TEST:", test_index)
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]

' ' '
result:
TRAIN: [1 3] TEST: [0 2]
TRAIN: [0 2] TEST: [1 3]
' ' '
```

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

Спасибо за внимание!