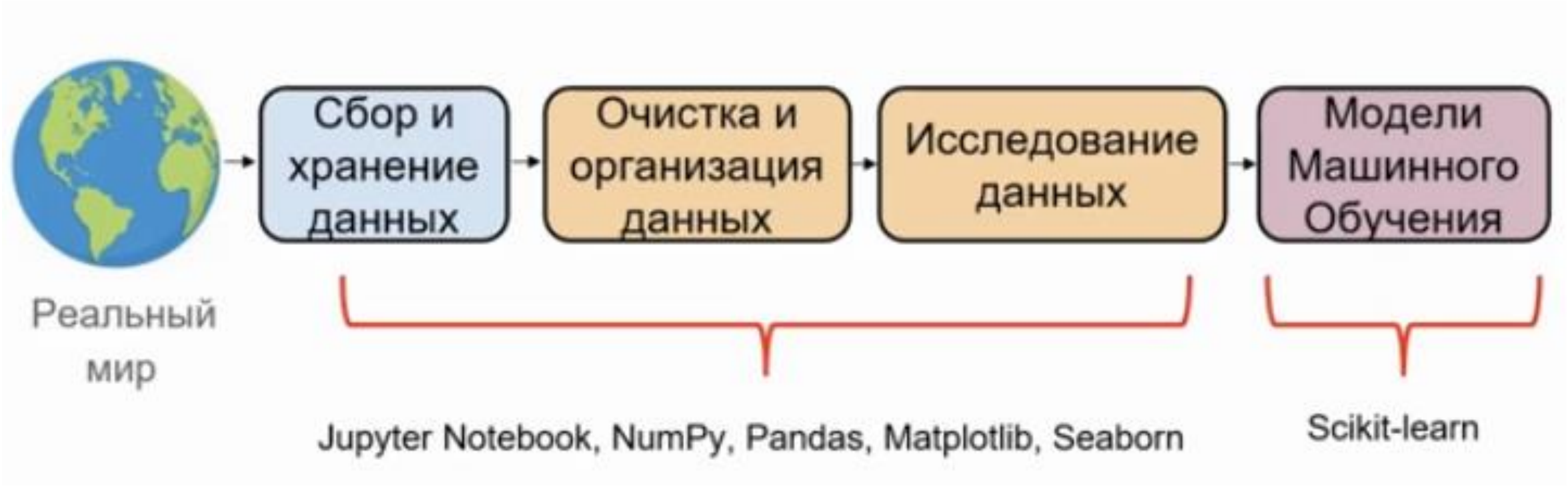
Основные понятия машинного обучения

Что такое машинное обучение:

- Машинное обучение (machine learning) – это изучение статистических компьютерных алгоритмов, которые автоматически улучшаются на данных. Здесь алгоритмы сами выбирают наилучшее решение на основе входных данных.
- Машинное обучение является частью ИИ

Этапы машинного обучения:



Примеры задач, решаемые с помощью машинного обучения:

- Кредитный скоринг
- Риски страхования
- Предсказания цен
- Фильтрация спама
- Сегментация клиентов
- И многое другое

Общий подход к постановке задач по машинному обучению

- Алгоритмы работают на основе «хороших» данных в достаточном количестве
- На основе признаков (features) в наборе данных найти требуемую целевую переменную (label/target)
- Алгоритмы с помощью статистических методов обрабатывают данные, в результате чего выясняют, какие признаки являются важными в данных

Типы алгоритмов машинного обучения

- Обучение с учителем (supervised learning) - используются размеченные исторические данные, на основе которых делается предсказание целевой переменной
- Обучение без учителя (unsupervised learning) – применяется к неразмеченным данным. Модель машинного обучения ищет возможные закономерности в данных и пытается сгруппировать данные в кластеры на основе признаков, не зная целевую переменную.

Типы задач:

- Целевая переменная принимает дискретное (категориальное) значение - задача классификации
- Целевая переменная принимает непрерывное значение – задача регрессии

Определите тип алгоритма машинного обучения (с учителем, без учителя) и тип задачи (классификации или регрессии):

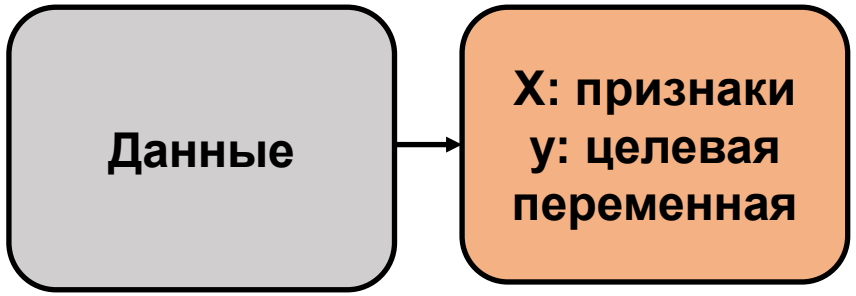
1. Предсказать, сколько фильмов просмотрит пользователь в следующем месяце в онлайн кинотеатре
2. Выяснить, какая из цифр от 0 до 9 нарисована на картинке
3. Поделить пользователей интернет-магазина на группы, чтобы похожие с точки зрения предпочтений пользователи оказались в одной группе.
4. Выбрать продукт из каталога, который можно было бы порекомендовать пользователю к его набранной интернет-корзине.
5. Понять, является ли электронное письмо спамом.

Процесс машинного обучения с учителем

Задача: Предсказать цену продажи для нового дома с известными значениями площади, кол-ва спален и кол-ва санузлов

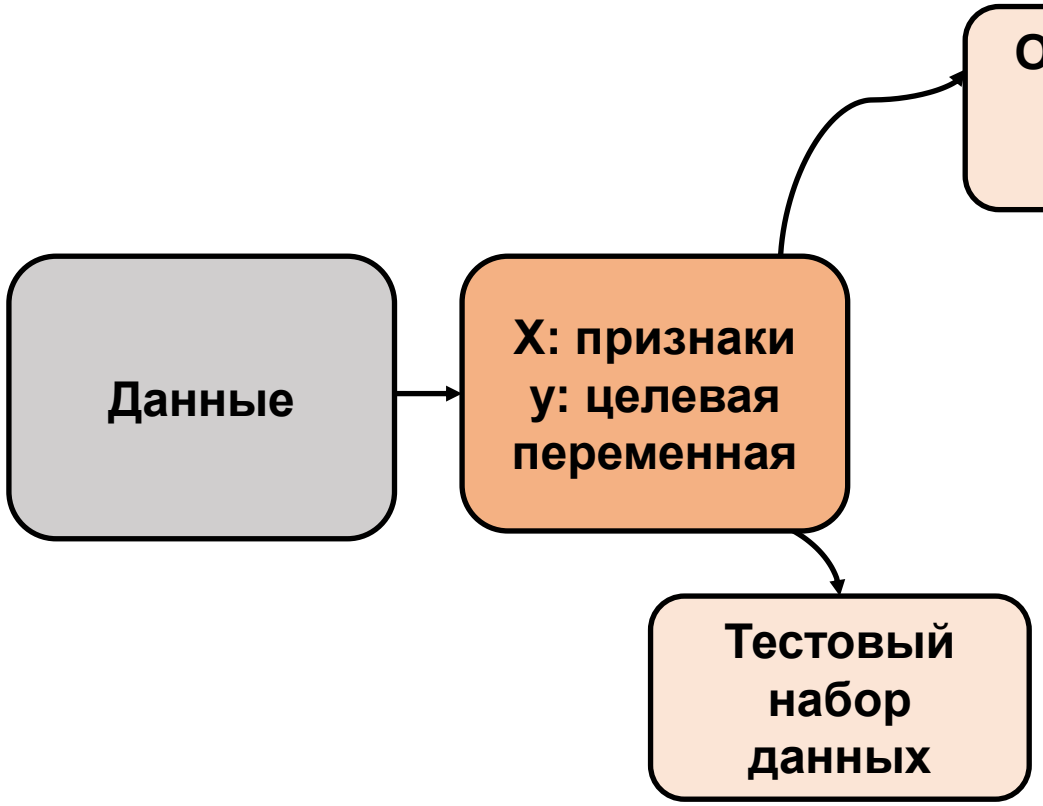
Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	2	2	$550,000

Шаг 1. Разделение данных на признаки и целевую переменную



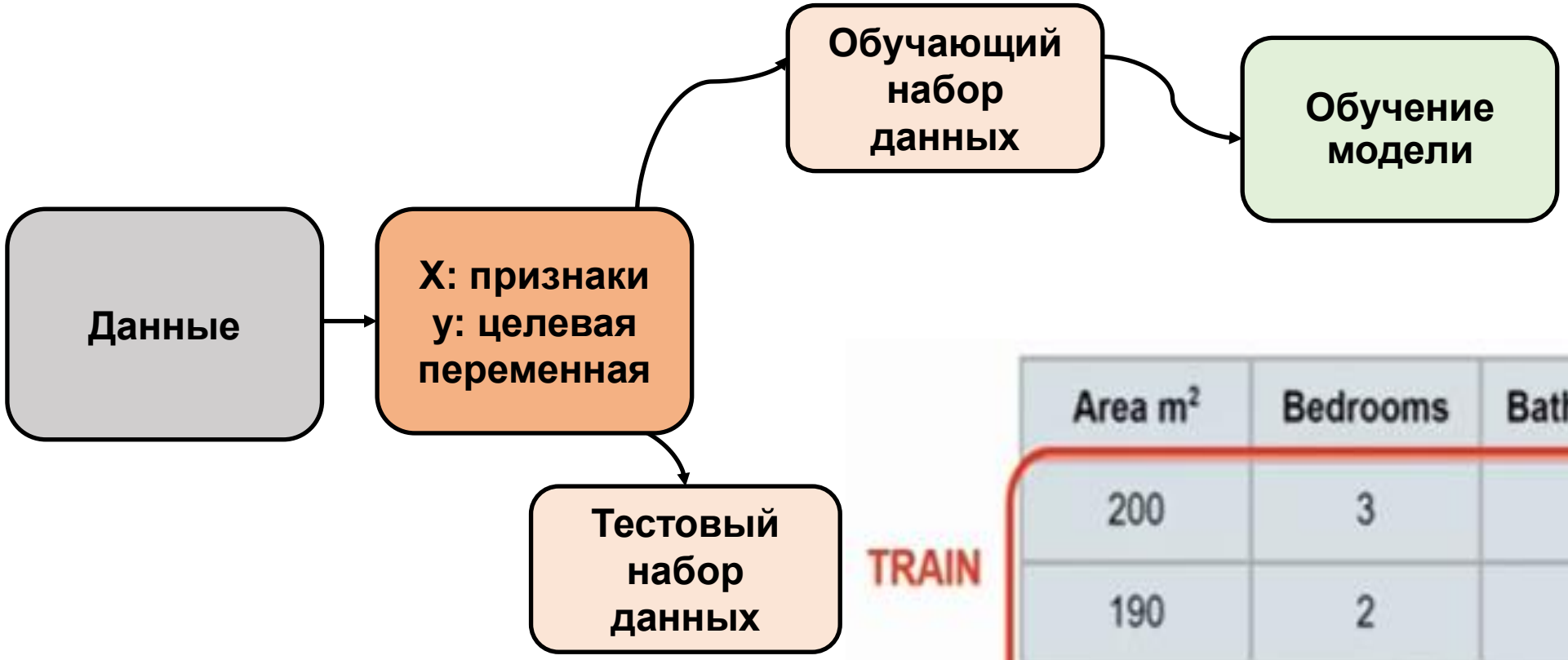
Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	2	2	$550,000

Шаг 2. Разделение признаков и целевой переменной на обучающую и тестовую части



	Area m^2	Bedrooms	Bathrooms	Price	
X TRAIN	200	3	2	$500,000	Y TRAIN
	190	2	1	$450,000	
	230	3	3	$650,000	
X TEST	180	1	1	$400,000	Y TEST
	210	2	2	$550,000	

Шаг 3. Обучение модели только на обучающем наборе данных



TRAIN

Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000

Шаг 4. Предсказание модели для тестового набора данных

Модель видит только признаки X .

Она не видела еще эти данные и не знает ответы

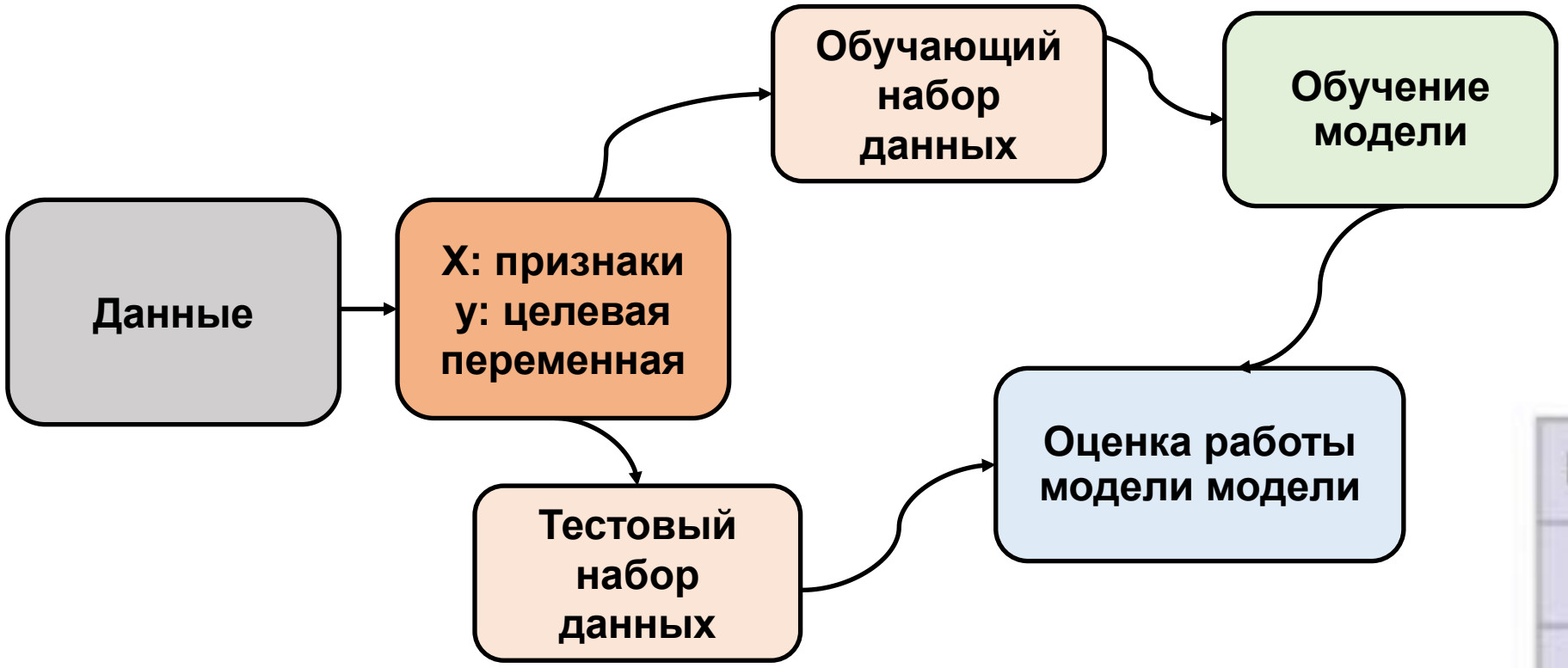
	Area m^2	Bedrooms	Bathrooms
TEST	180	1	1
	210	2	2

Шаг 5. Модель предсказывает значения целевой переменной для тестового набора данных

Predictions	Area m^2	Bedrooms	Bathrooms
$410,000	180	1	1
$540,000	210	2	2

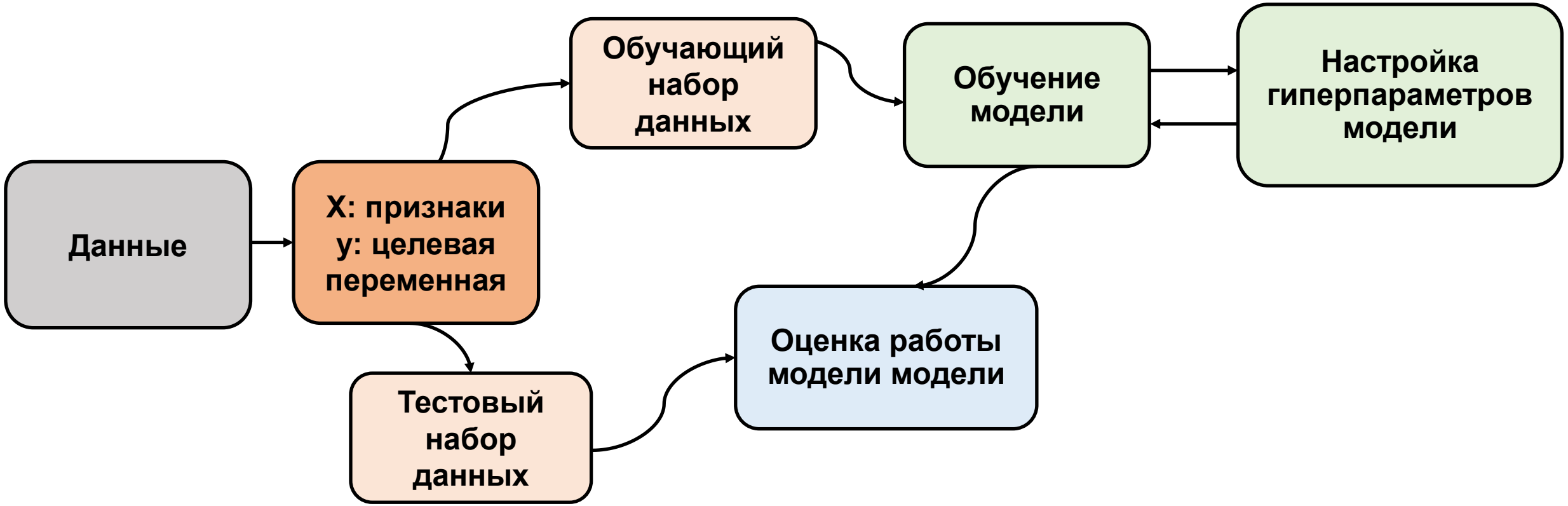
Шаг 6. Оценка работы модели с помощью тестового набора данных

Сравнение предсказаний с истинными значениями целевой переменной

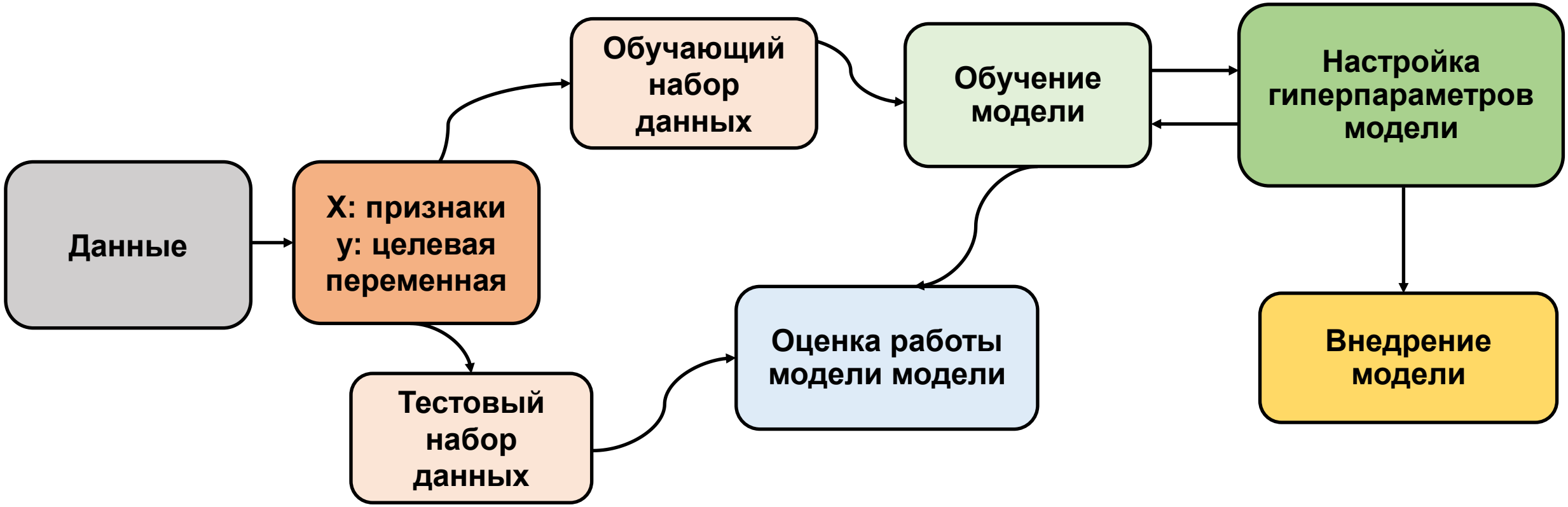


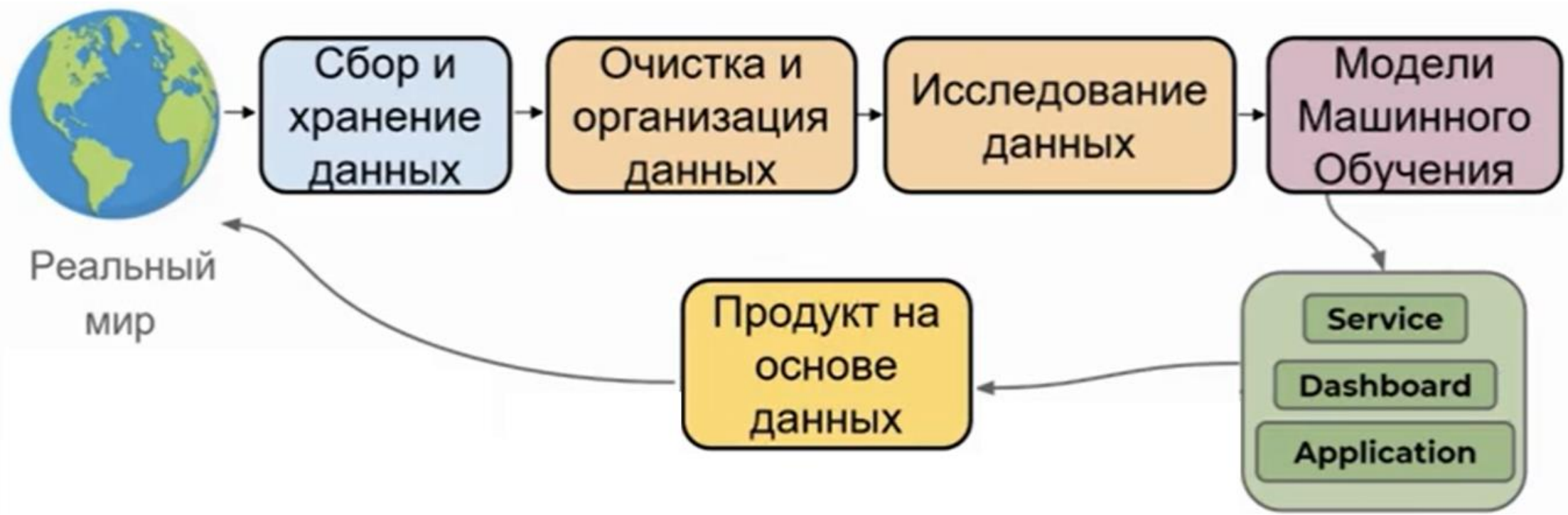
Predictions	Price
$410,000	$400,000
$540,000	$550,000

Шаг 7. В случае неудовлетворительного результата работы модели



Шаг 8. Внедрение модели

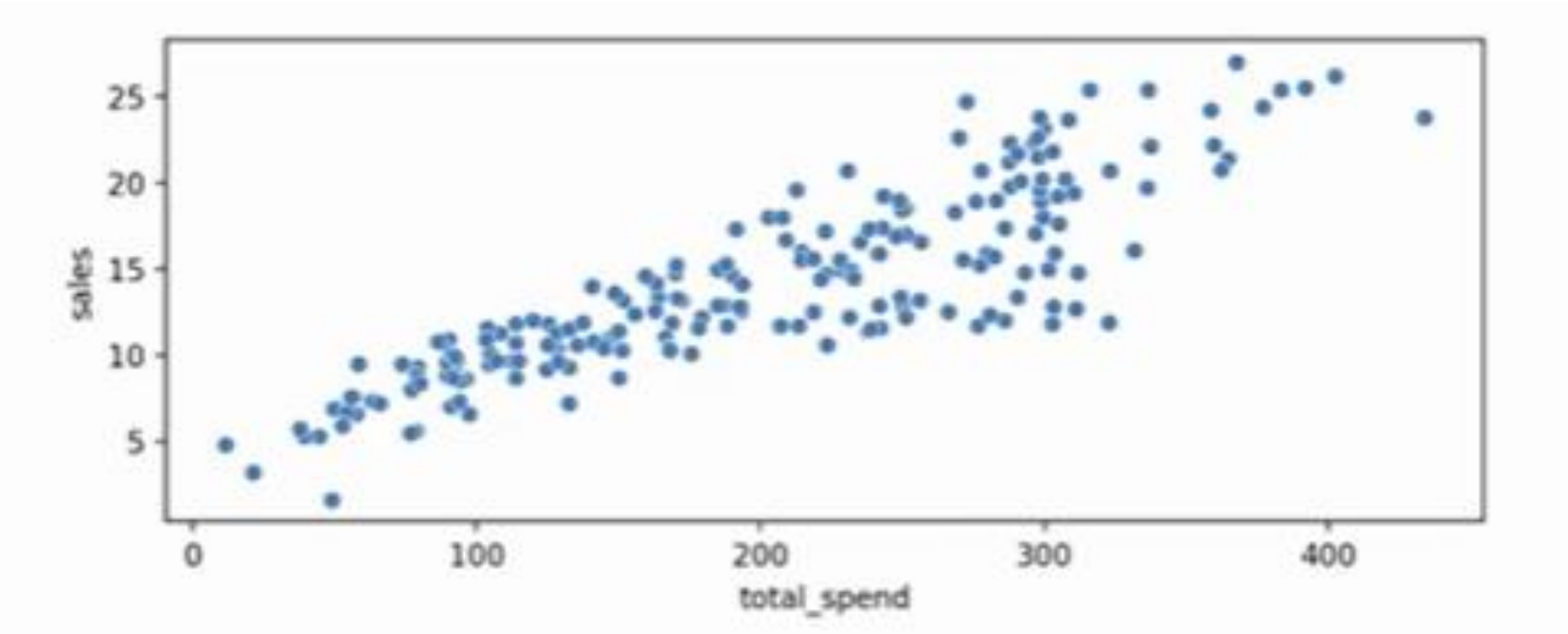



Реальный мир
Сбор и хранение данных
Очистка и организация данных
Исследование данных
Модели Машинного Обучения
Service
Dashboard
Application
Продукт на основе данных

Линейная регрессия

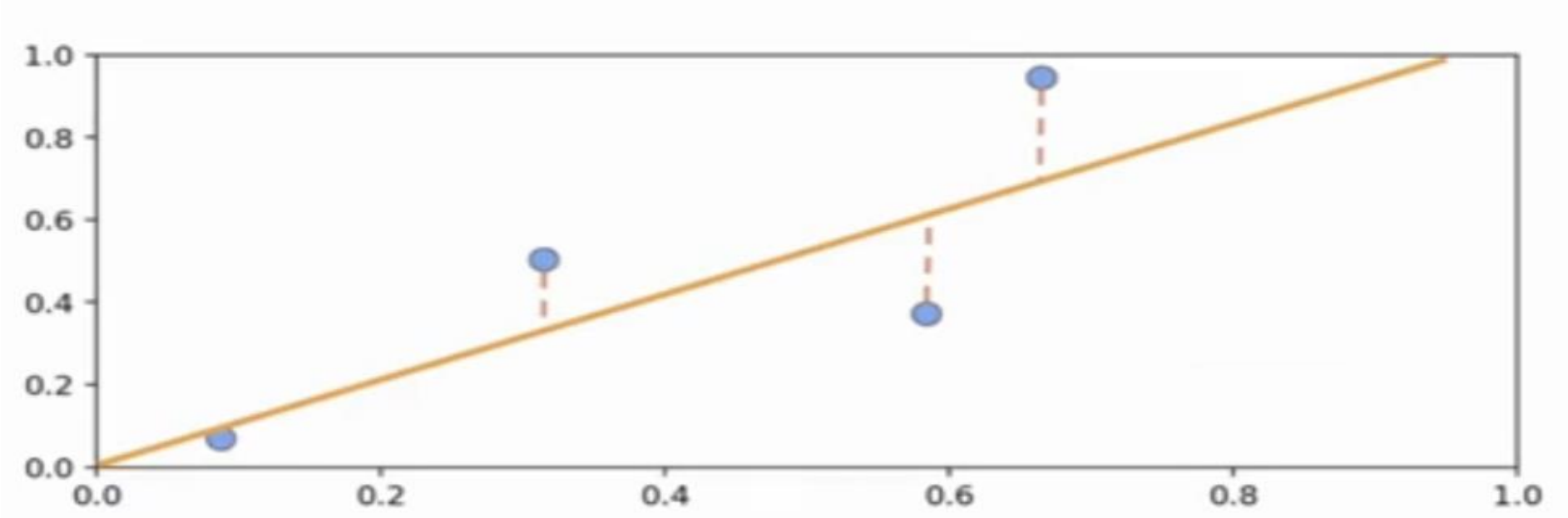
Возникает задача:

как провести линию так, чтобы она наилучшим образом соответствовала этим точкам



Линейная регрессия

Проведя линию, можем измерить расстояния от точек до линии - это будут ошибки (остатки). Остатки могут быть как положительные, так и отрицательные. Необходимо выбрать такую линию, где общее расстояние между точками и линией будет минимальным



Линейная регрессия

- В случае одной переменной x или одного признака уравнение регрессии имеет вид $y = ax+b$

- В случае нескольких переменных $x_1, x_2, x_3, \ldots, x_n$ или нескольких признаков уравнение регрессии имеет вид

  $\hat{y} = w_0+w_1x_1+w_2x_2+\ldots+w_nx_n = w_0+\sum_{i=1}^{n} w_i x_i$

  w_i – параметры модели (веса), i=(0,n)

Необходимо подобрать такие веса w_i, чтобы минимизировать ошибки, т.е. функция суммы квадратов ошибок достигала минимума

Линейная регрессия

$\hat{y}_j - y_j$

$(\hat{y}_j - y_j)^2$

$\sum_{i=1}^{m} (\hat{y}_j - y_j)^2$

$\frac{1}{m} \sum_{i=1}^{m} (\hat{y}_j - y_j)^2$ - среднее значение суммы квадрата ошибок

Линейная регрессия

Функция потерь (стоимостная функция, функционал ошибки):

$$Q(w, X) = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_j - y_j)^2 =$$

$$= \frac{1}{m}\sum_{i=1}^{m}(w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n - \hat{y}_j)^2 \to \min$$

$w_0, w_1, w_2, \ldots, w_n$

Процесс поиска оптимального набора параметров (весов) называется **обучением**

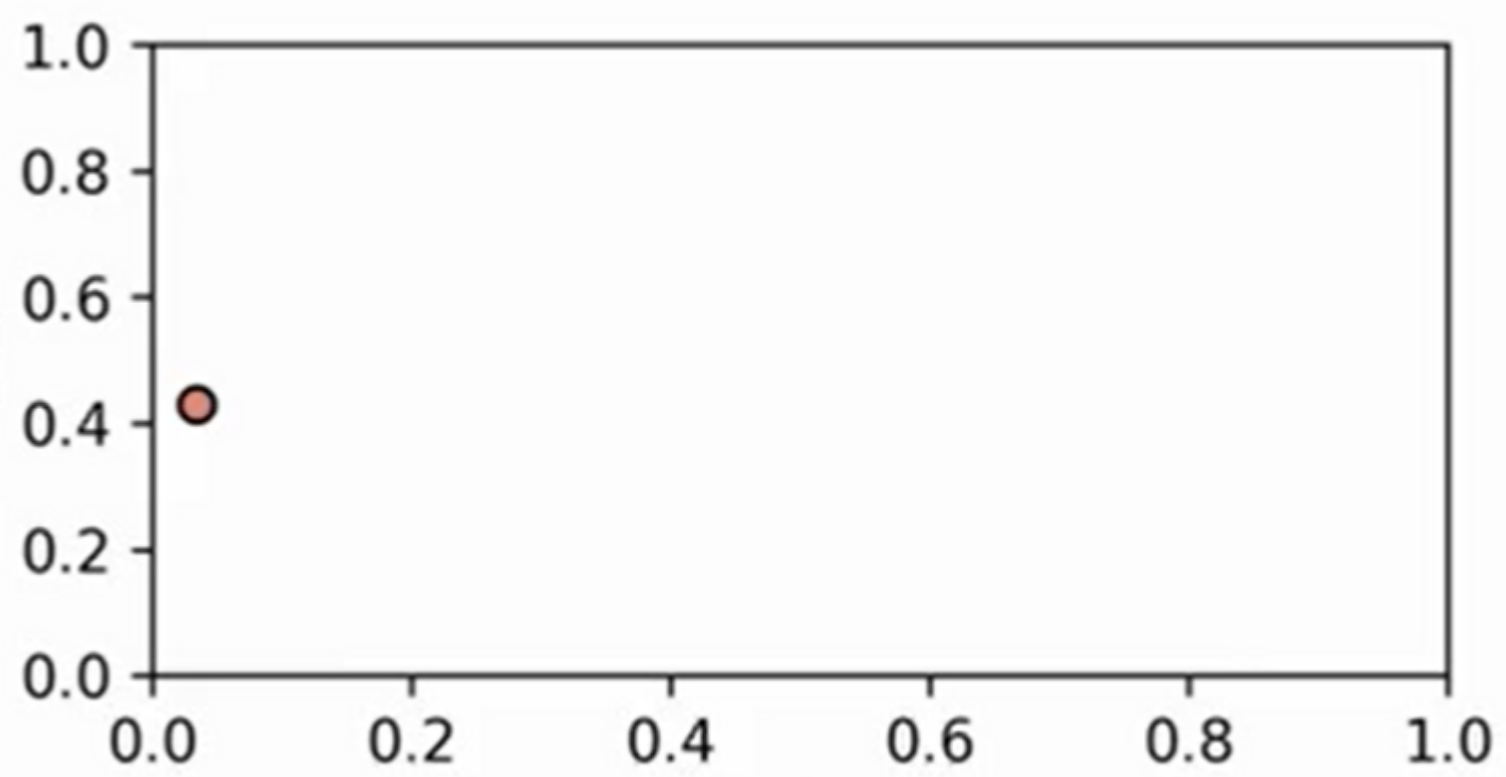
Линейная регрессия

- В случае одной переменной x или одного признака веса w_0, w_1 можно найти методом наименьших квадратов

- В случае нескольких переменных для нахождения w_i используется метод градиентного спуска

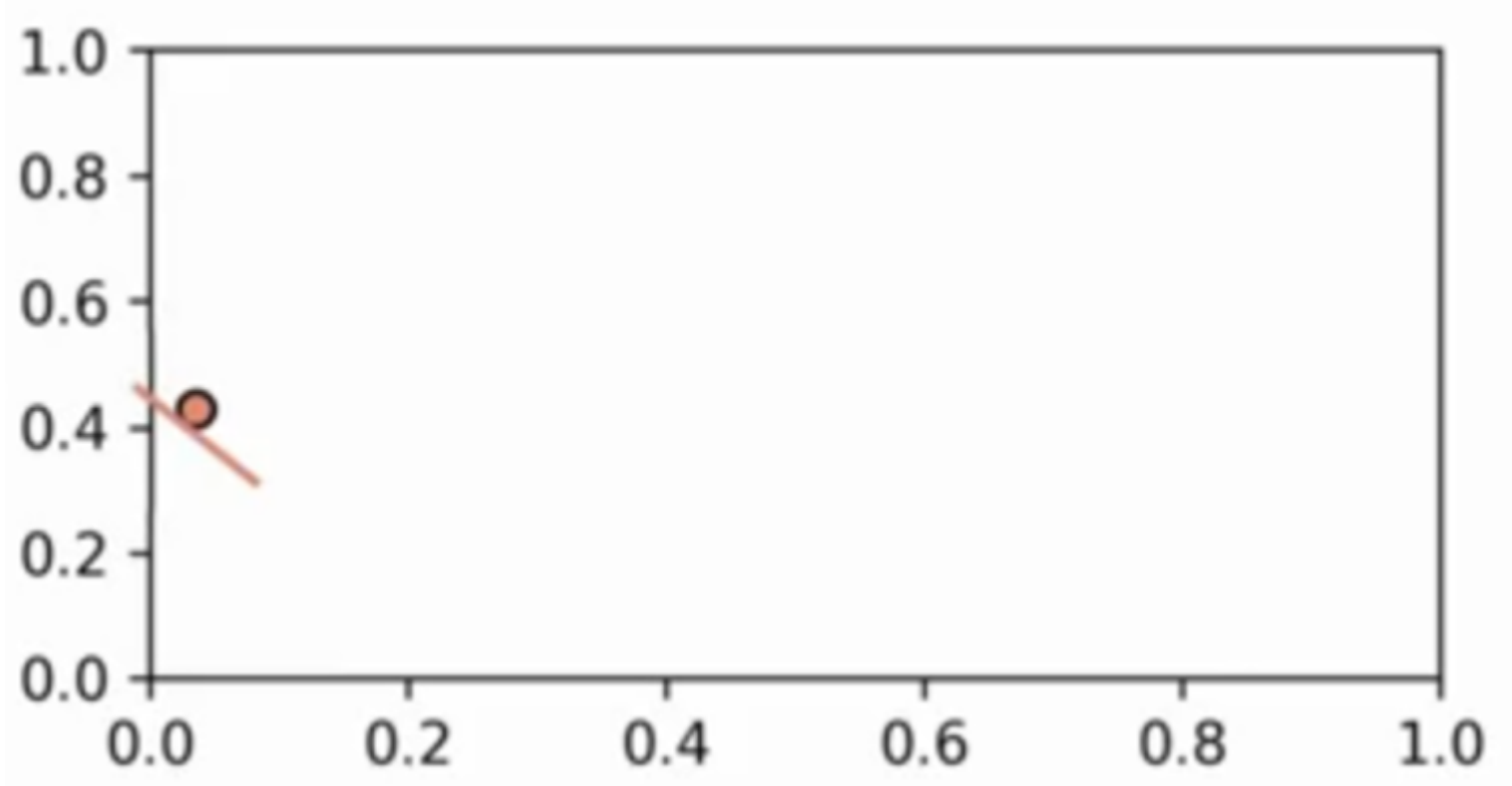
Метод градиентного спуска

1. Находим начальную точку



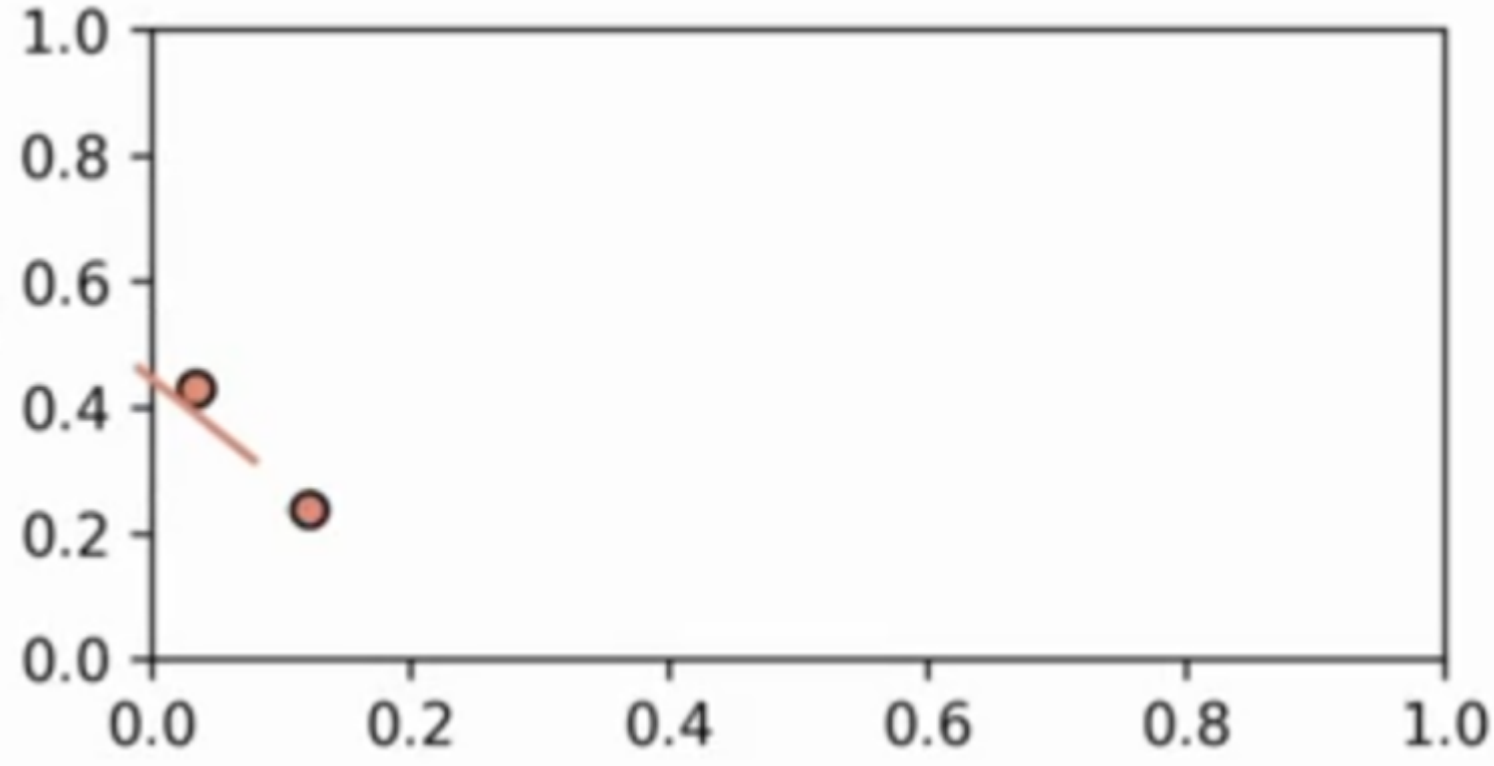
Метод градиентного спуска

2. Находим градиент в этой точке



Метод градиентного спуска

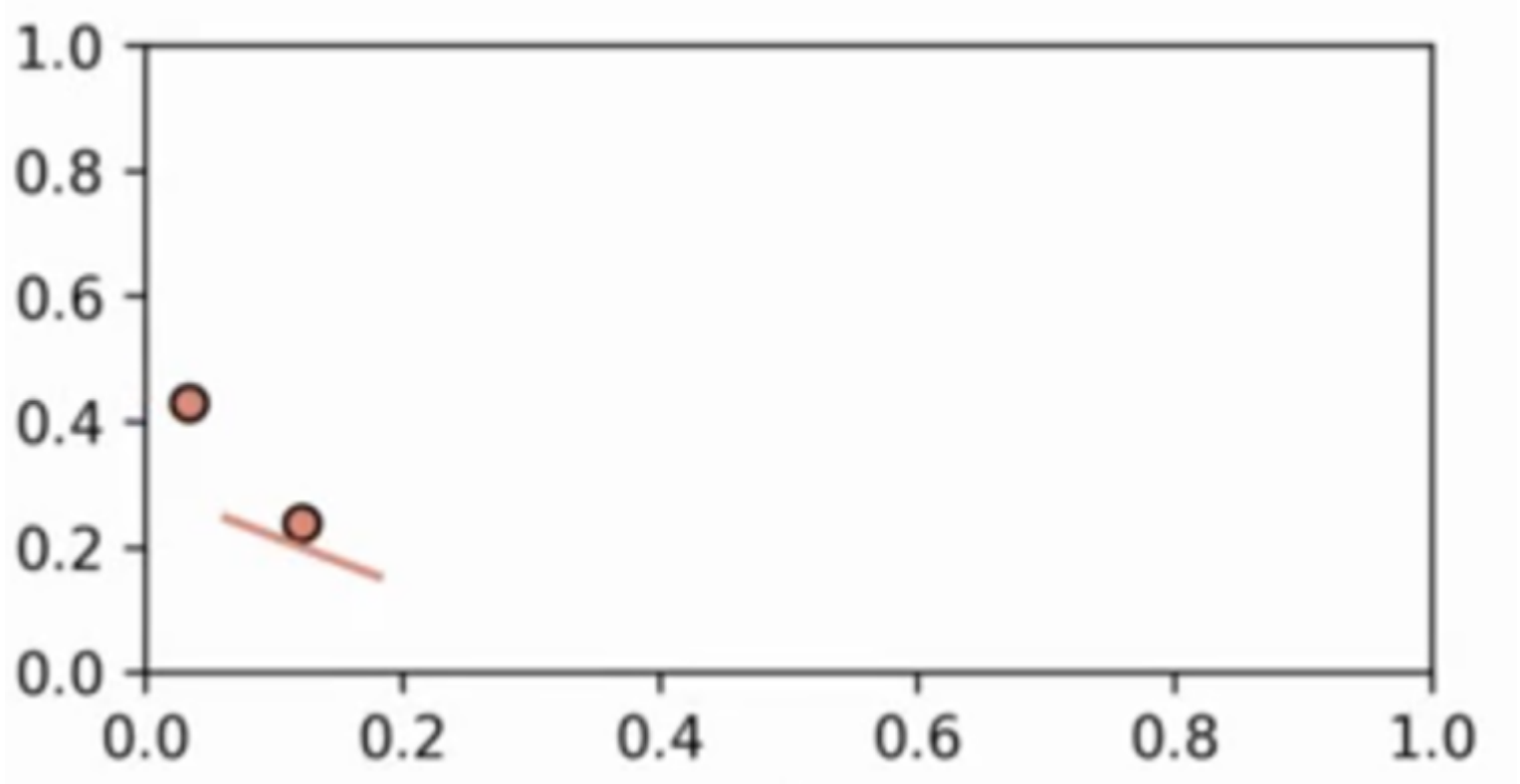
3. В сторону антиградиента делаем шаг, пропорциональный величине градиента



Антиградиент указывает направление наибольшего убывания функции.

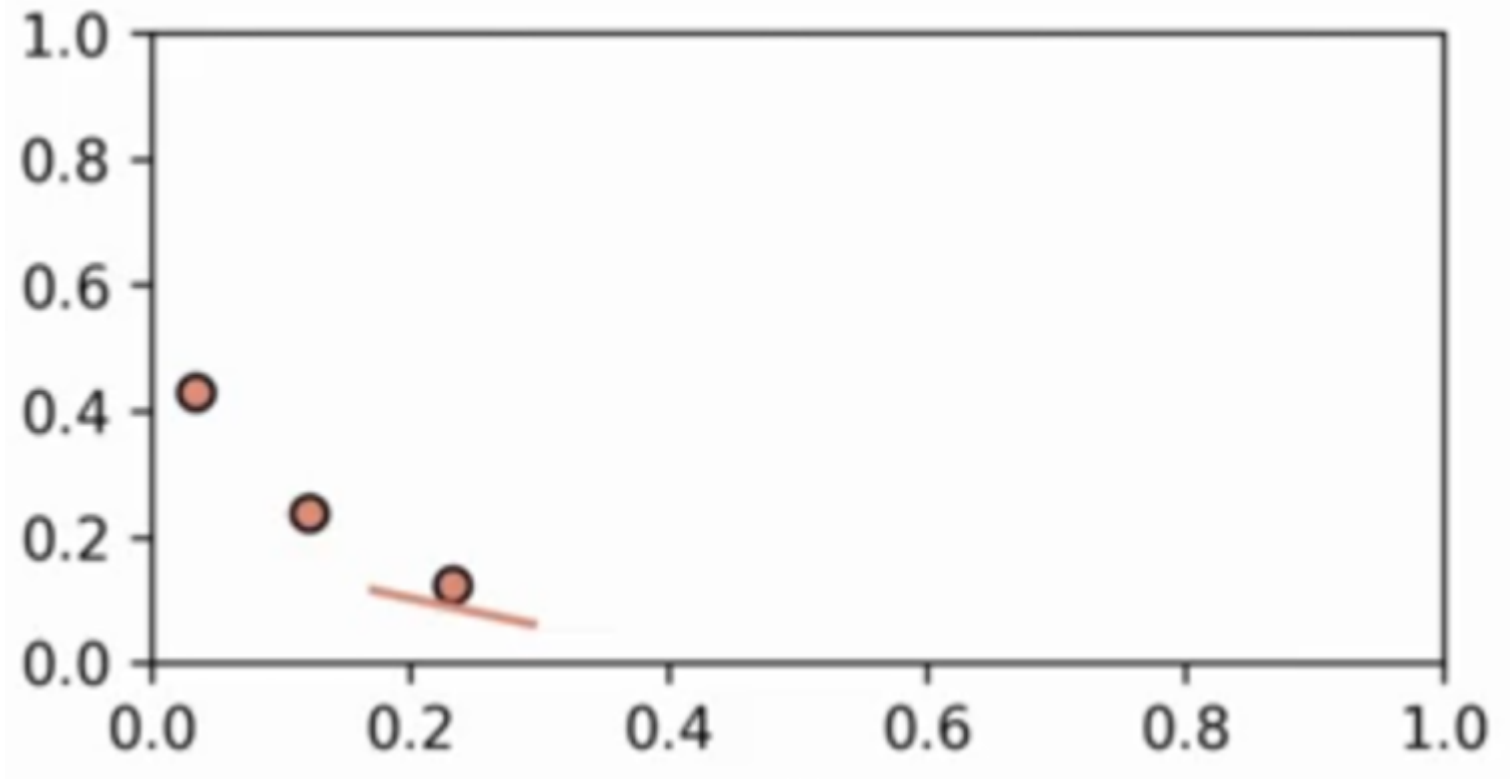
Метод градиентного спуска

4. Повторяем шаги



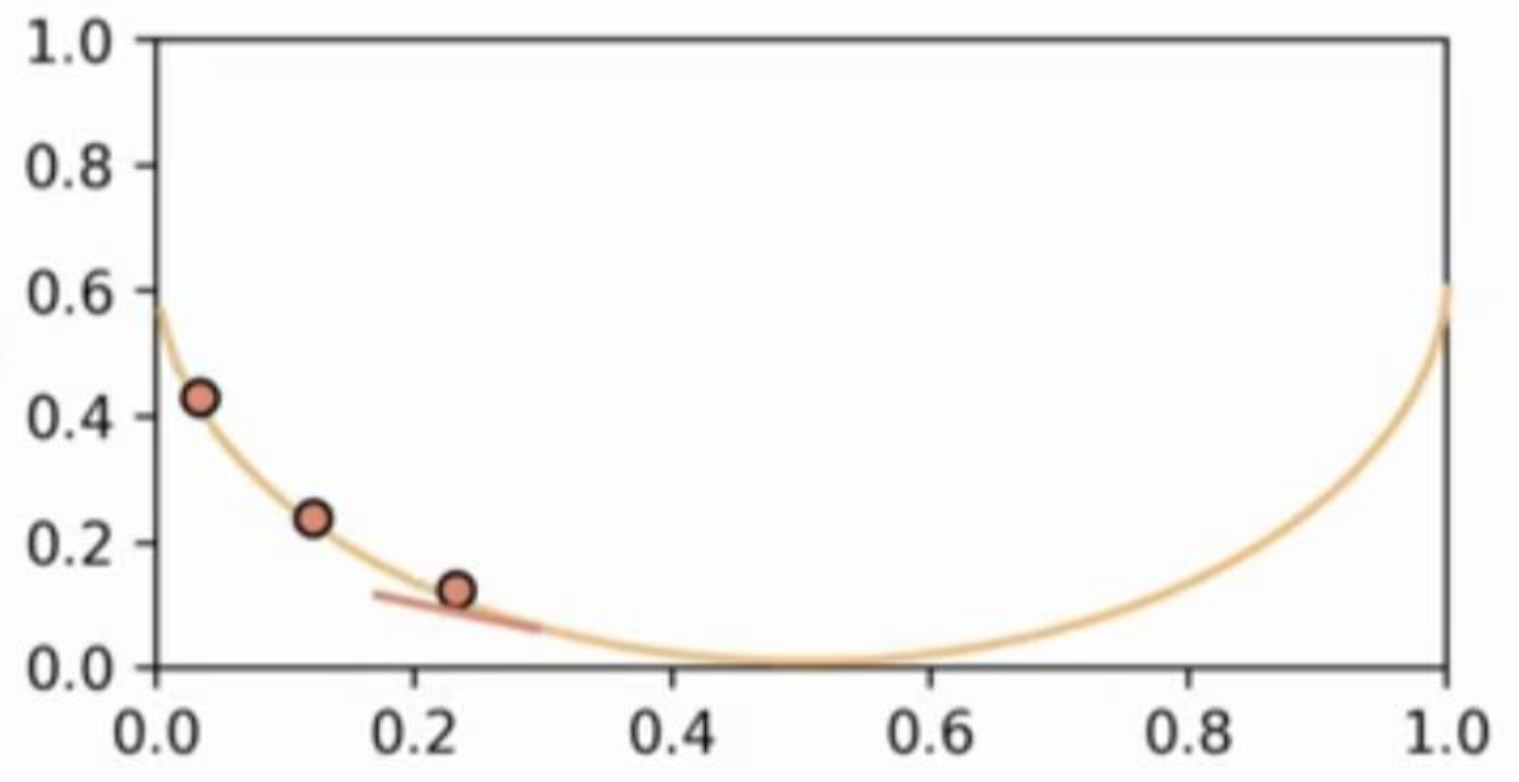
Метод градиентного спуска

5. Повторяем шаги



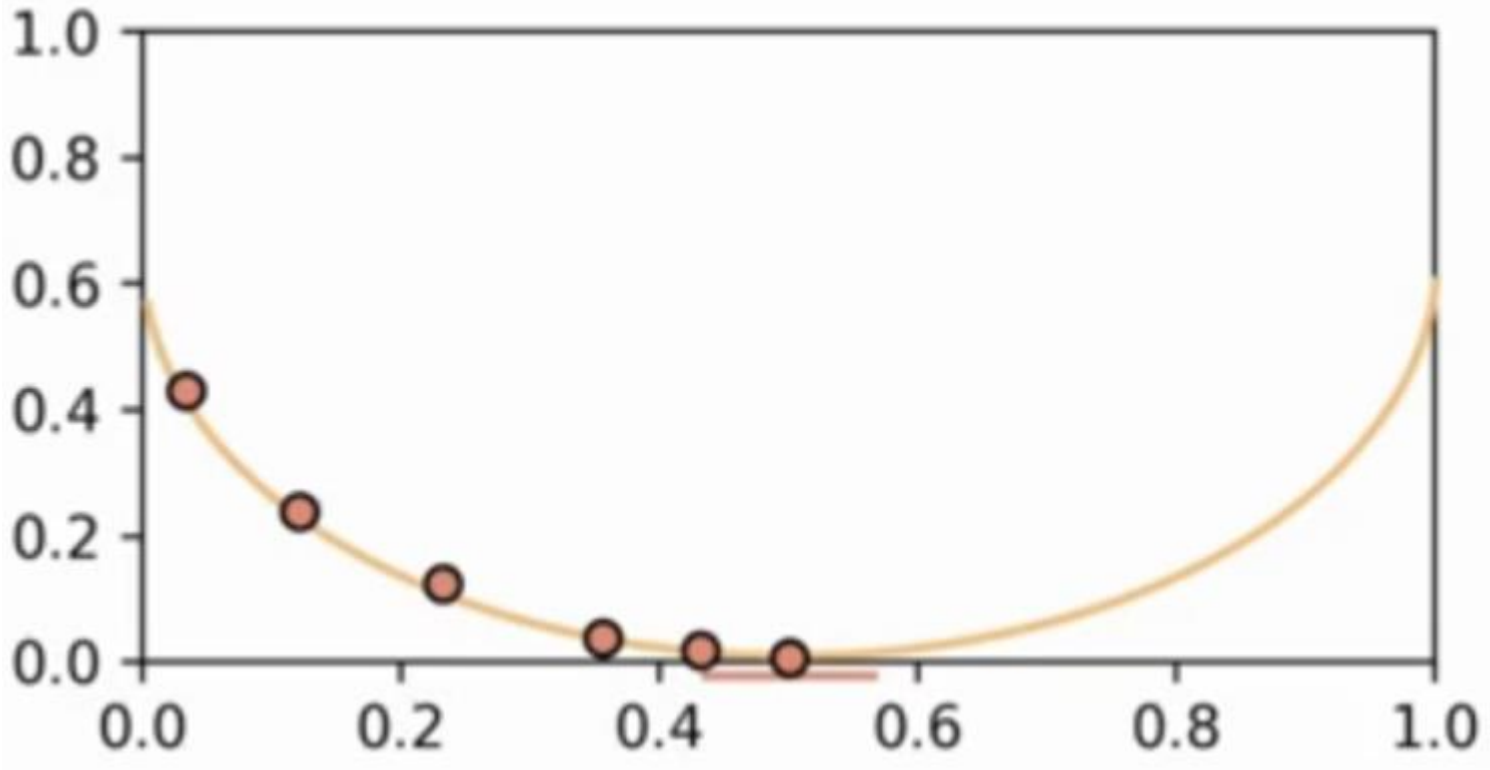
Метод градиентного спуска

6. Повторяем шаги



Метод градиентного спуска

В конце находим минимум



Метод градиентного спуска

В случае нескольких переменных те же самые шаги:

1. Вычисляем в точке градиент (производную)
2. Двигаемся в направлении антиградиента
3. Величина шага пропорциональна градиенту
4. Повторяем шаги, пока не найдем минимум функции

$$x_k = x_{k-1} - \mu \nabla f(x_{k-1})$$

Библиотека Scikit-Learn

Scikit-Learn – библиотека со многими алгоритмами машинного обучения

Методы импорта, обучения, использования алгоритмов выполняются одинаково для разных моделей

Scikit-Learn содержит много полезных методов, включая функции разбиения данных на обучающий и тестовый наборы, функции кросс-валидации, а также метрики оценки модели.

Библиотека Scikit-Learn

Установка библиотеки:

pip install scikit-learn

Разбиение на обучающую и тестовую выборки

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

Библиотека Scikit-Learn

Импортирование модели из библиотеки:

from sklearn.model_family import ModelAlgo

Создание модели:

model = ModelAlgo(parametr1, parametr2)

Обучение модели:

model.fit(X_train, y_train)

Предсказание модели:

y_pred=model.predict(X_test)

Метрики для регрессии:

Средняя абсолютная ошибка – Mean Absolute Error (MAE)

- усредняет абсолютные значения ошибок:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Для некоторых точек, находящихся далеко от линии регрессии, модель плохо предсказывает

Среднеквадратическая ошибка - Mean Squared Error (MSE) - усредняет квадраты ошибок

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

большие ошибки «наказываются» сильнее;

неудобство – единицы измерения в квадрате

Среднеквадратическое отклонение - Root Mean Squared Error (RMSE) – квадратный корень от среднеквадратической ошибки

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

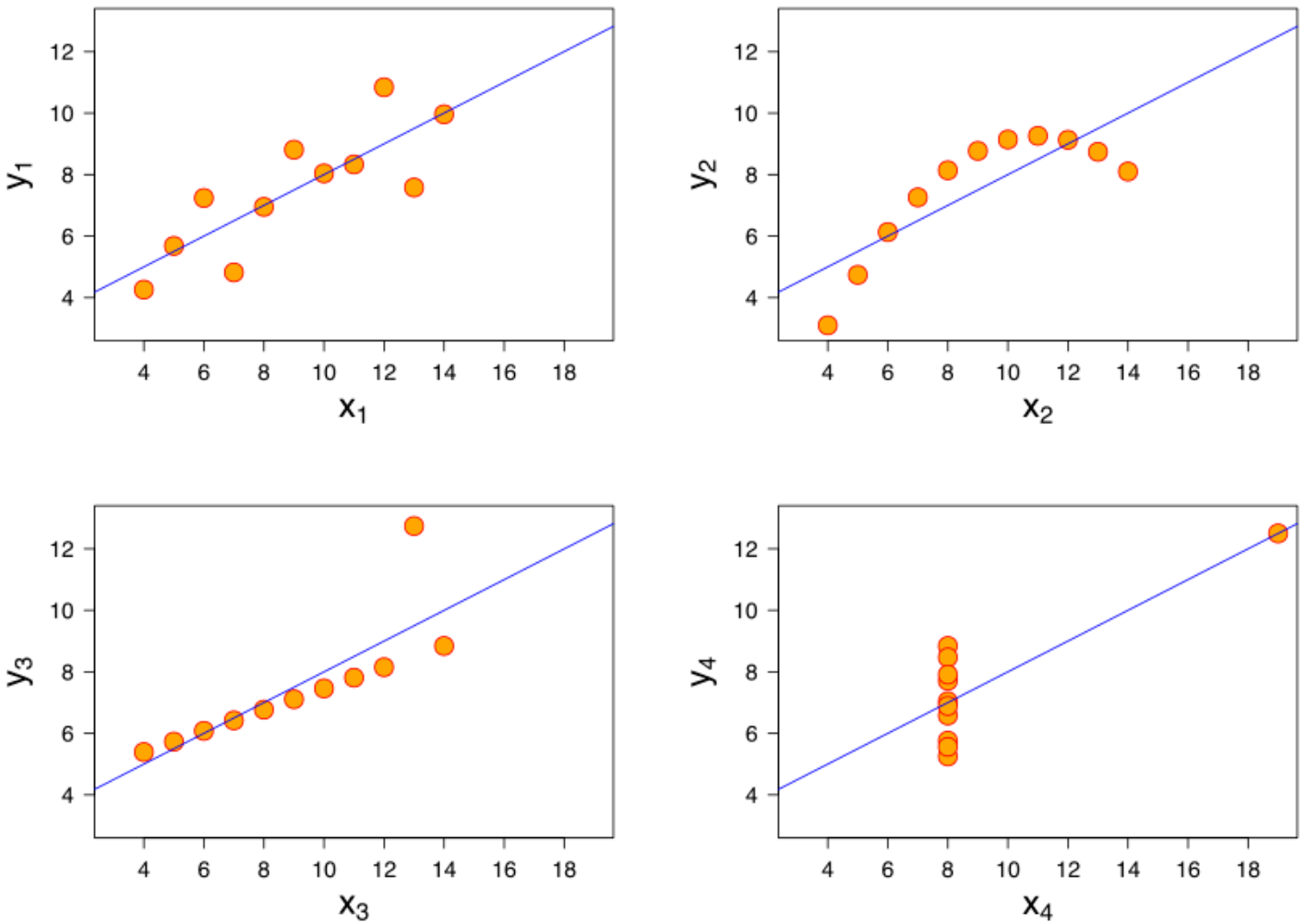
- единицы измерения те же, что и у целевой переменной;
- «наказываются» большие ошибки.

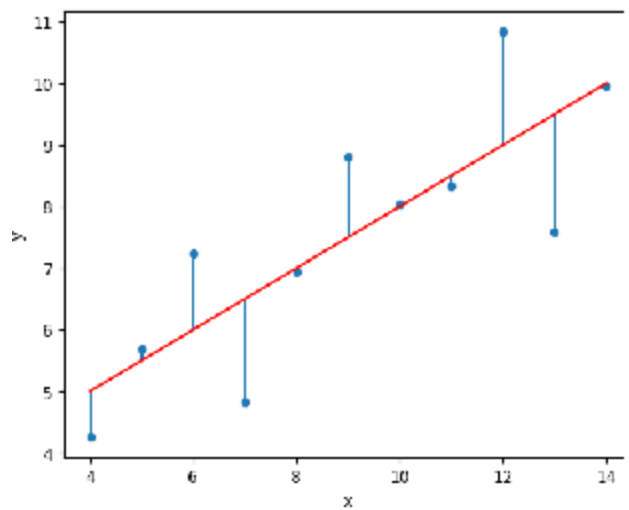
```
from sklearn.metrics import error_metric
error=error_metric(y_test, y_pred)
```

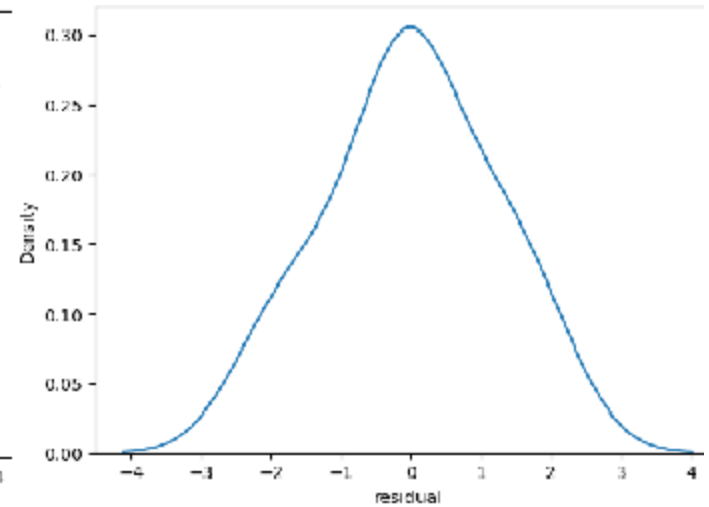
- Для линейной регрессии имеет смысл смотреть не только на метрики, но и на остатки.

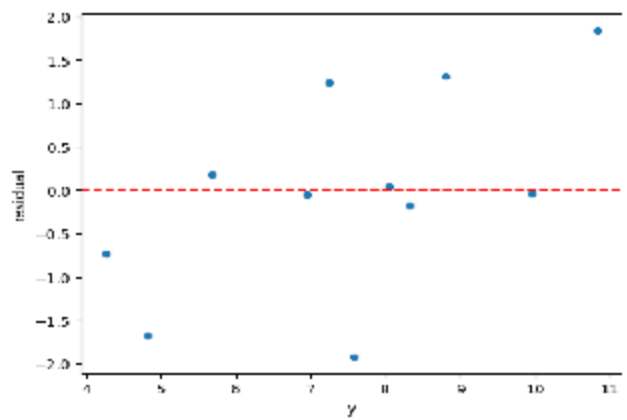
- Линейная регрессия подходит не для каждого набора точек
- Анализ остатков позволяет понять, насколько линейная регрессия подходит для набора точек

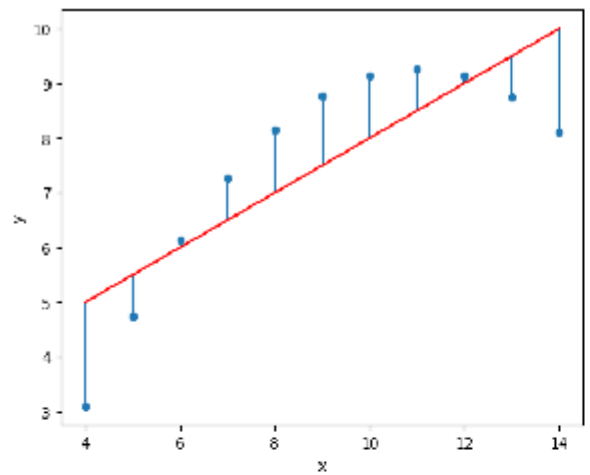
Квартет Энскомба

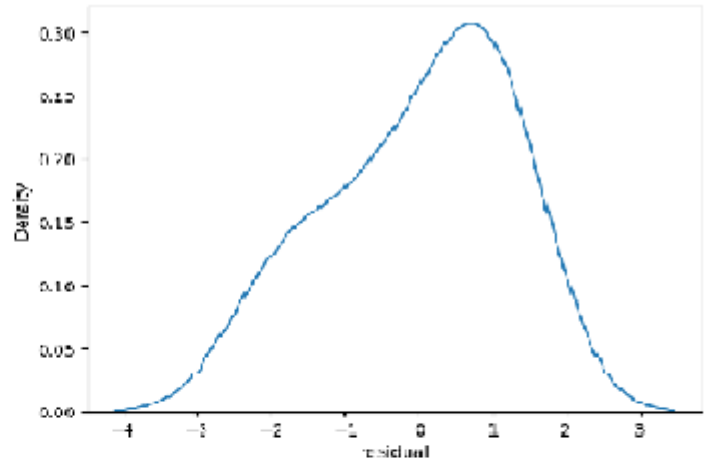


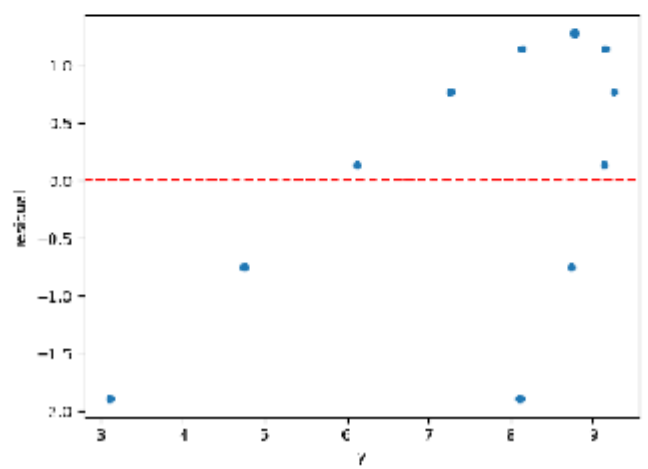

y
x

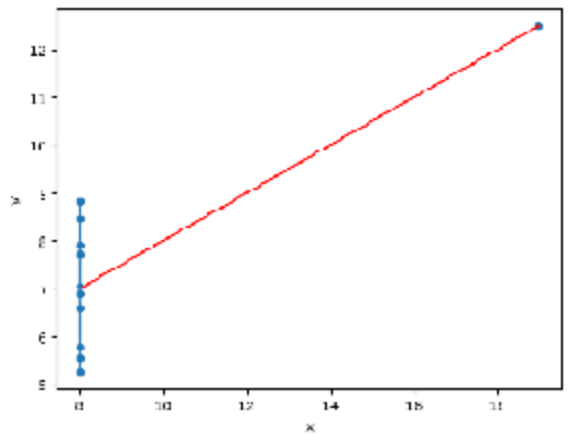

Density
residual

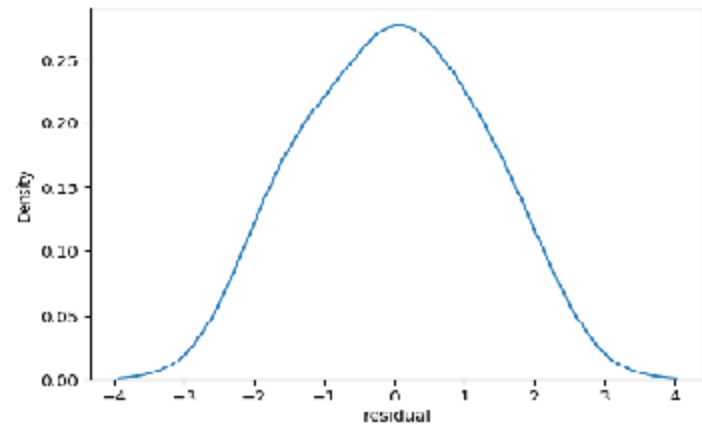

residual
y

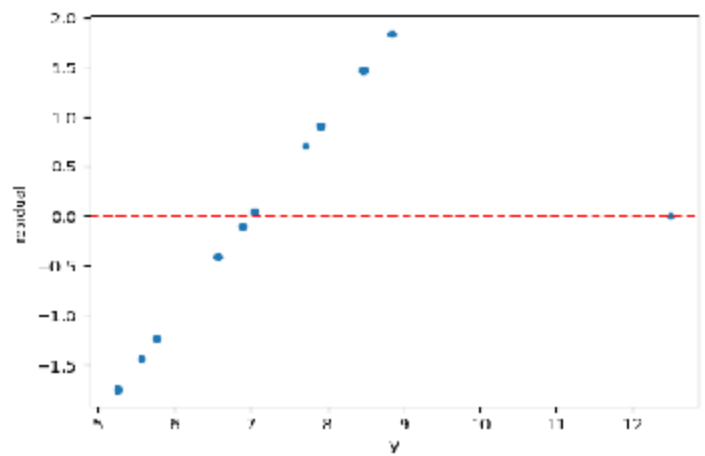

y
x


Density
0.30
0.25
0.20
0.15
0.10
0.05
0.00


residual
y


x


Density
0.25
0.20
0.15
0.10
0.05
0.00
−4
−3
−2
−1
0
1
2
3
4
residual


residual
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
5
6
7
8
9
10
11
12
y

Полиномиальная регрессия

Один из способов улучшения линейной модели - применение для признаков полиномов более высокой степени

При поиске коэффициентов wi в линейном уравнении можно попробовать брать не только сами признаки, но и более высокие степени этих признаков.

Также между признаками возможны зависимости, т.е. один признак важен тогда, когда важен и другой (синергия)

Вопрос: как учесть это в модели?

- Простейший способ – добавить произведение двух признаков в дополнение каждому из признаков в отдельности.
- Это можно сделать в Scikit-Learn с помощью методов preprocessing.
- В этой библиотеке есть много полезных методов для обработки данных перед обучением модели.
- Например, PolynomialFeatures – автоматически создает полиномы более высоких порядков, а также произведения комбинаций признаков.

- Примеры слагаемых, которые создаются:
- Постоянное число 1 (все признаки равны 0)
- Возведение признака в степень
- Произведение пар признаков в различных возможных комбинациях

В случае 2-х переменных x_1, x_2

1, x_1, x_2, x_1^2, x_2^2, $x_1 * x_2$ - итого становится 6 признаков

Для всех 6 признаков ищется w_i и они могут найти больше сигналов в данных

Нет 100% гарантии, что модель улучшится

Надо пробовать

Недообучение и переобучение модели

Переобучение:

Модель слишком точно повторяет шумы и неточности в данных.

Это часто приводит к малым ошибкам на обучающем наборе данных, но к большим ошибкам на тестовых/проверочных данных

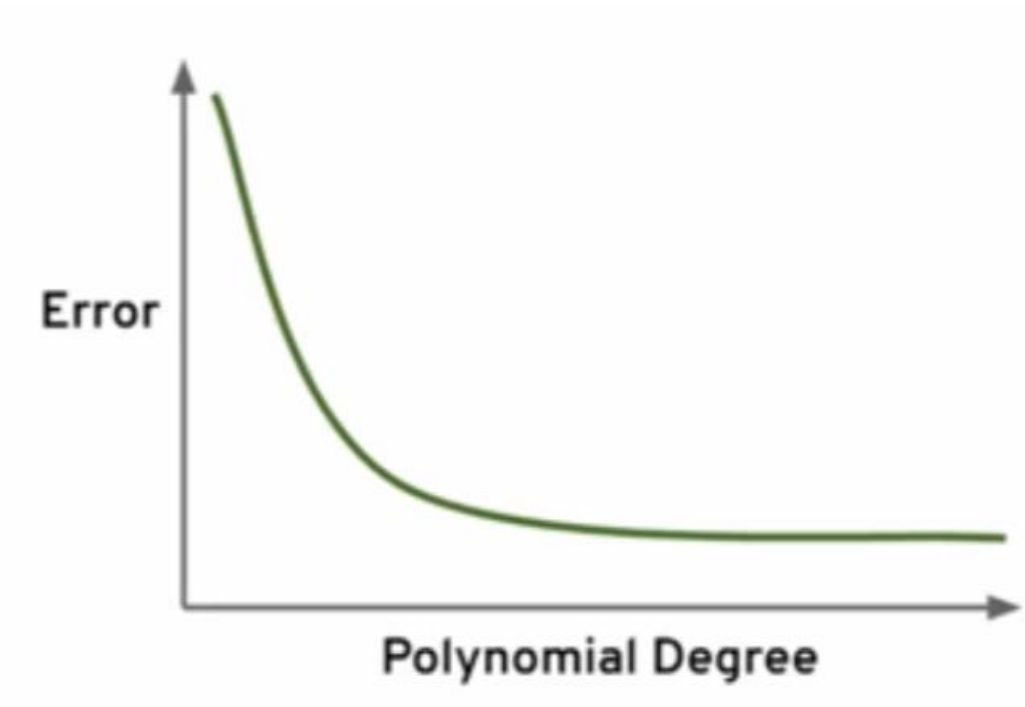
Недообучение:

Показывает слабые результаты и на обучающем, и на тестовом наборах данных.

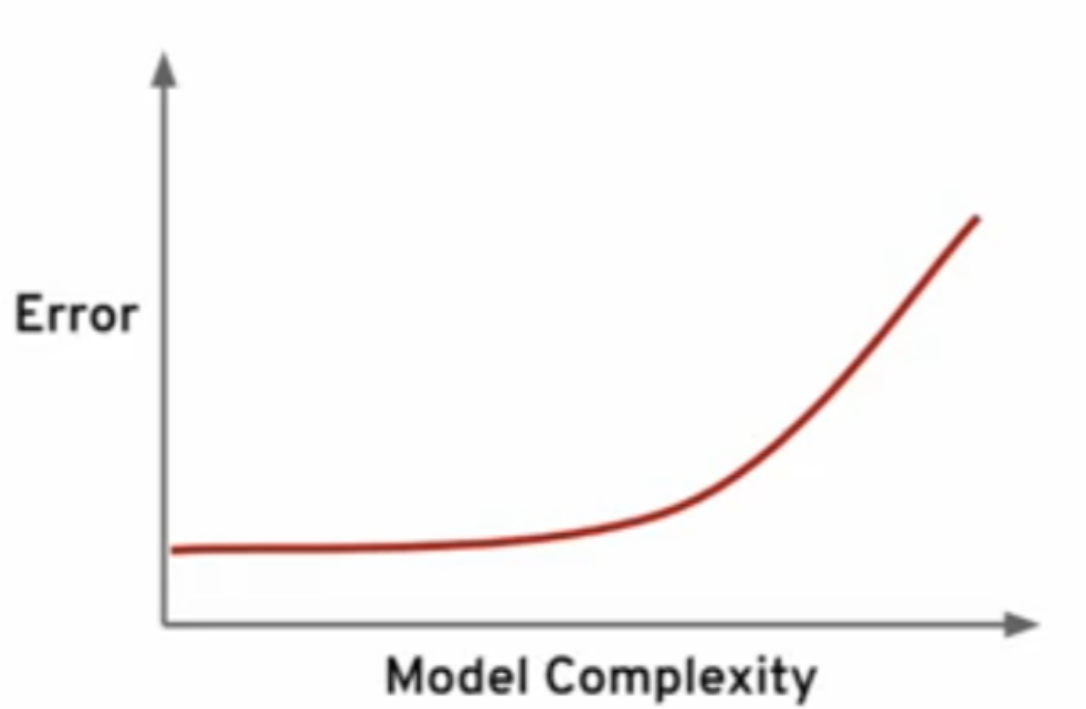
Недообученность модели говорит о том, что выбранная модель очень простая и нужно выбрать более сложную модель

Связь между сложностью и ошибкой модели

Хорошая модель



Плохая модель

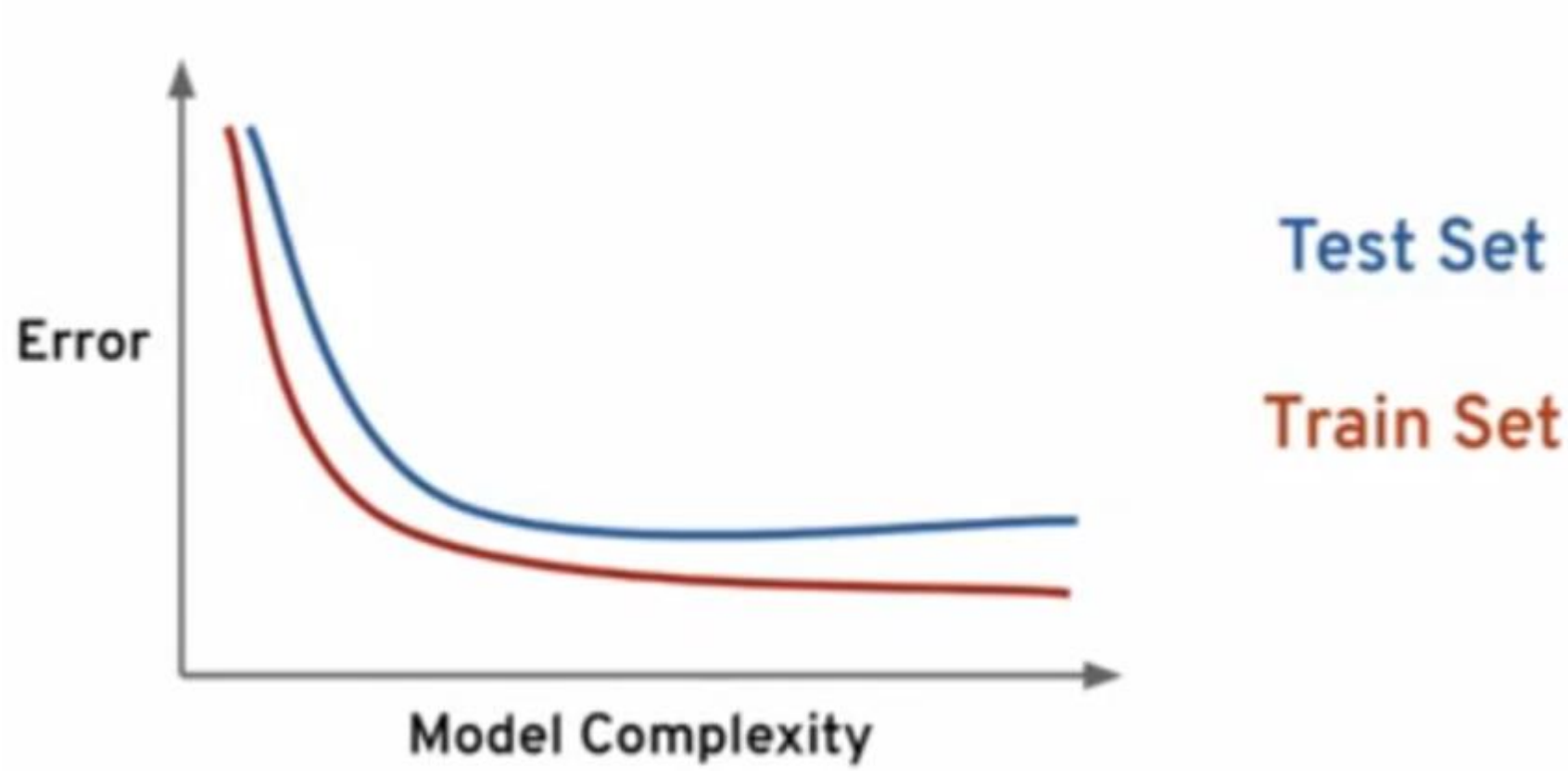


Переобучение сложнее заметить, потому что модель показывает хорошие результаты на обучающем наборе данных и выглядит все так, что она работает хорошо

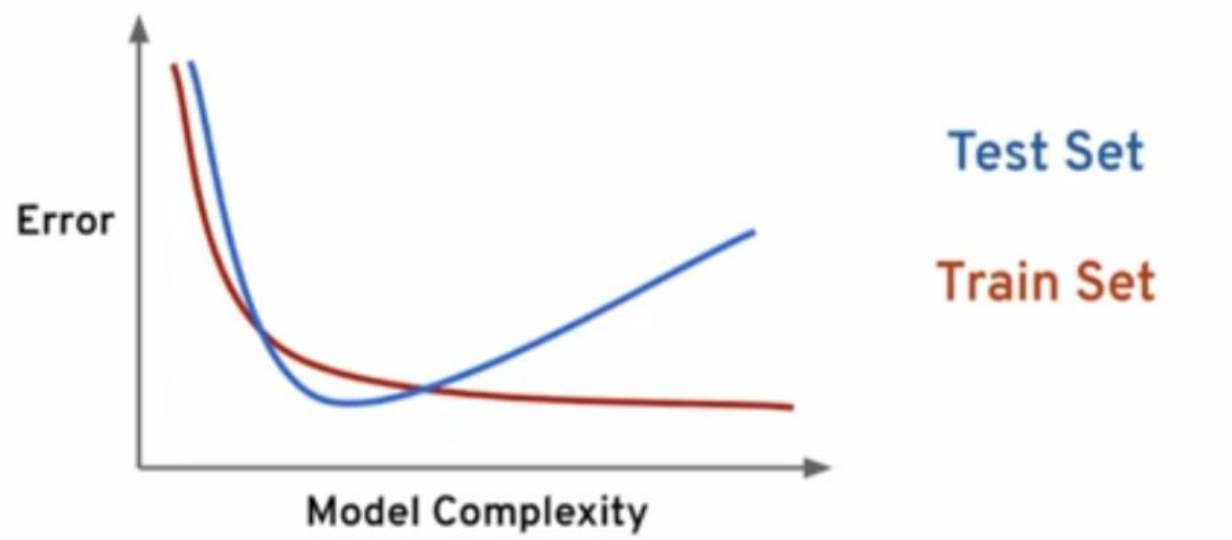
Проблема:

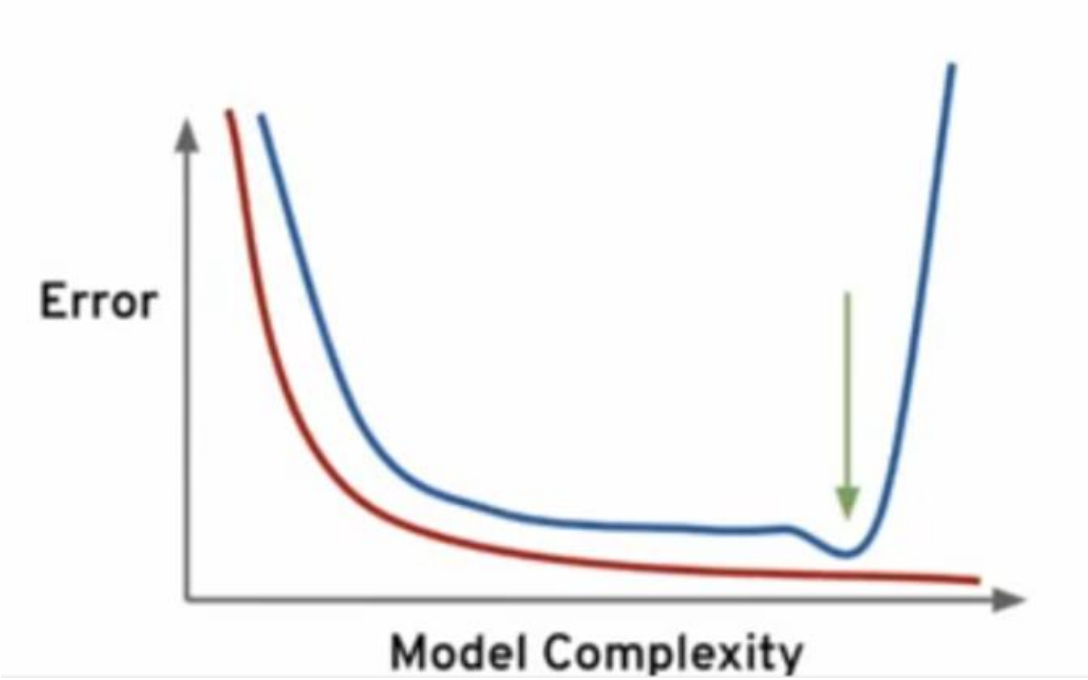
Как найти баланс между недообученностью и переобученностью модели

Нужно смотреть результаты на обучающем наборе данных, сравнивая их с результатами на тестовом наборе данных.



Переобученность модели





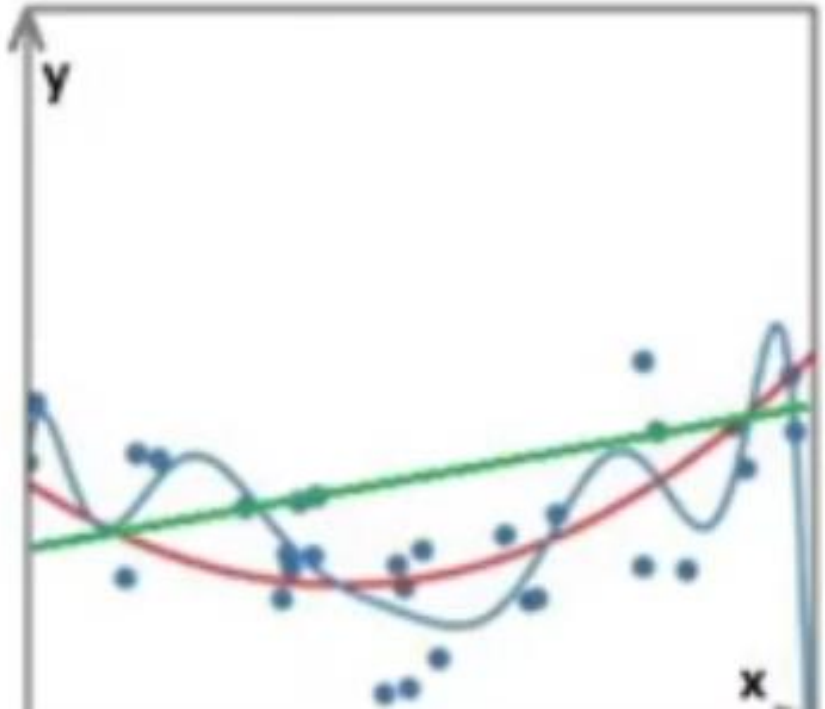
- При выборе сложности модели, а также оценке работы модели, необходимо изучать ошибки модели как на обучающем, так и на тестовом наборах данных.
- Для полиномиальной регрессии – это степень полинома, но другие алгоритмы могут иметь свои гиперпараметры, определяющие сложность модели.

Пусть в задаче регрессии мы использовали:

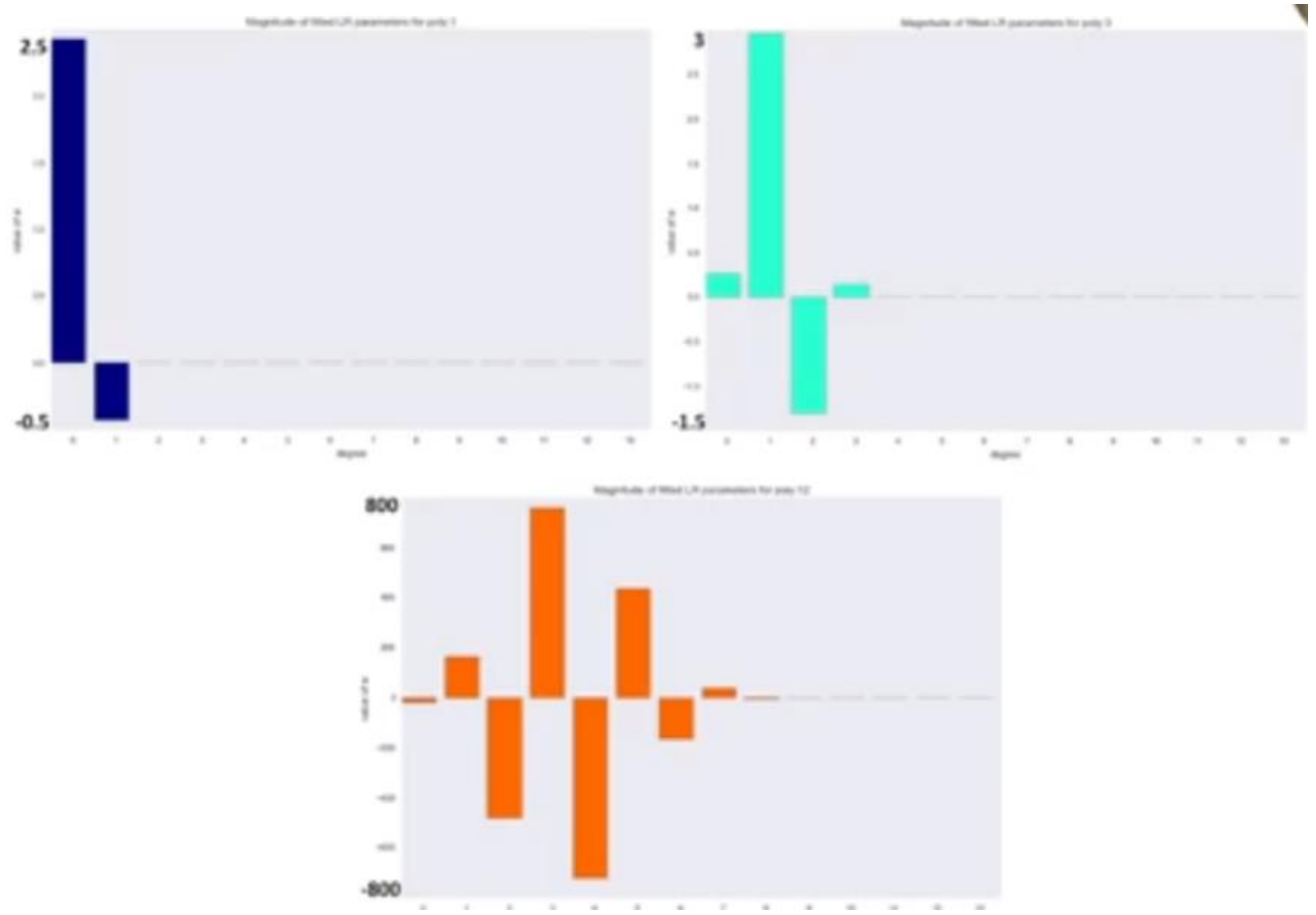
А) линейную модель с одним признаком x

Б) линейную модель с тремя признаками x_1, x_2, x_3

В) линейную модель с 12 признаками: x_1, x_2, x_3, …, x_{12}



Сравнение весов



Большие значения весов w_i являются признаком переобученности

Чтобы снизить переобучение, необходимо уменьшить веса.

Решение проблемы: регуляризация

Раньше минимизировали функцию потерь $Q(w, X)$

К этой функции добавим слагаемое $\alpha * R(w)$:

$$Q(w, X) + \alpha * R(w) \rightarrow \min$$

$R(w)$ – регуляризатор (штрафует большие веса у модели)

При минимизации новой функции веса будут уменьшаться

Наиболее используемые регуляризаторы:

L1 – регуляризатор: R(w) = $\|w\|_1 = \sum_{i=1}^{n} |w_i|$

L2 – регуляризатор: R(w) = $\|w\|_2 = \sum_{i=1}^{n} w_i^2$

ElasticNet: R(w) = $\|w\|_1 + \|w\|_2 = \sum_{i=1}^{n} |w_i| + \sum_{i=1}^{n} w_i^2$

Какой из регуляризаторов сработает лучше, неизвестно

L1 регуляризация

Не все признаки в задаче могут быть нужны

- Некоторые признаки могут не иметь отношения к задаче, т.е. они не нужны
- Если есть ограничения на скорость получения предсказаний, то чем меньше признаков, тем быстрее
- Если признаков больше, чем объектов, то решение задачи будет неоднозначным

В таких случаях можно не использовать некоторые признаки

В результате обучения модели с L_1-регуляризатором происходит зануление некоторых маленьких весов, т.е. отбор признаков

L1

- В результате обучения модели с L_1-регуляризатором происходит зануление некоторых маленьких весов, т.е. отбор признаков

- Модели, в которых часть весов равна 0, называются разреженными

ElasticNet

ElasticNet: R(w) = $\|w\|_1 + \|w\|_2 = \sum_{i=1}^{n}|w_i| + \sum_{i=1}^{n} w_i^2$

$Q(w, X) + \alpha 1 * \sum_{i=1}^{n}|w_i| + \alpha 2 * \sum_{i=1}^{n} w_i^2 \to \min$

или

$Q(w, X) + \alpha\left(\frac{1-\lambda}{2} * \sum_{i=1}^{n} w_i^2 + \lambda * \sum_{i=1}^{n}|w_i|\right) \to \min \qquad \lambda = (0;1)$

Масштабирование признаков

- Масштабирование признаков ускоряет сходимость итераций для тех алгоритмов, которые зависят от масштаба признаков
- Если признаки имеют разный масштаб, то одни веса могут обновляться быстрее других, поскольку сами значения признаков участвуют в обновлении весов

- Если бы все признаки имели одинаковый масштаб значений, тогда сходимость итераций будет происходить с одинаковой скоростью для каждого из коэффициентов
- Сведение разных признаков к единой шкале позволяет сравнивать признаки, имеющие разные единицы измерения, следовательно можно сравнивать разные коэффициенты модели между собой
- Для некоторых алгоритмов масштабирование – необходимое условие

2 способа масштабирования

1) стандартизация – данные должны получить среднее значение μ=0 и среднеквадратическое отклонение σ = 1

$$X_{changed} = \frac{x-}{\sigma}$$

2) нормализация – данные должны оказаться в диапазоне от 0 до 1

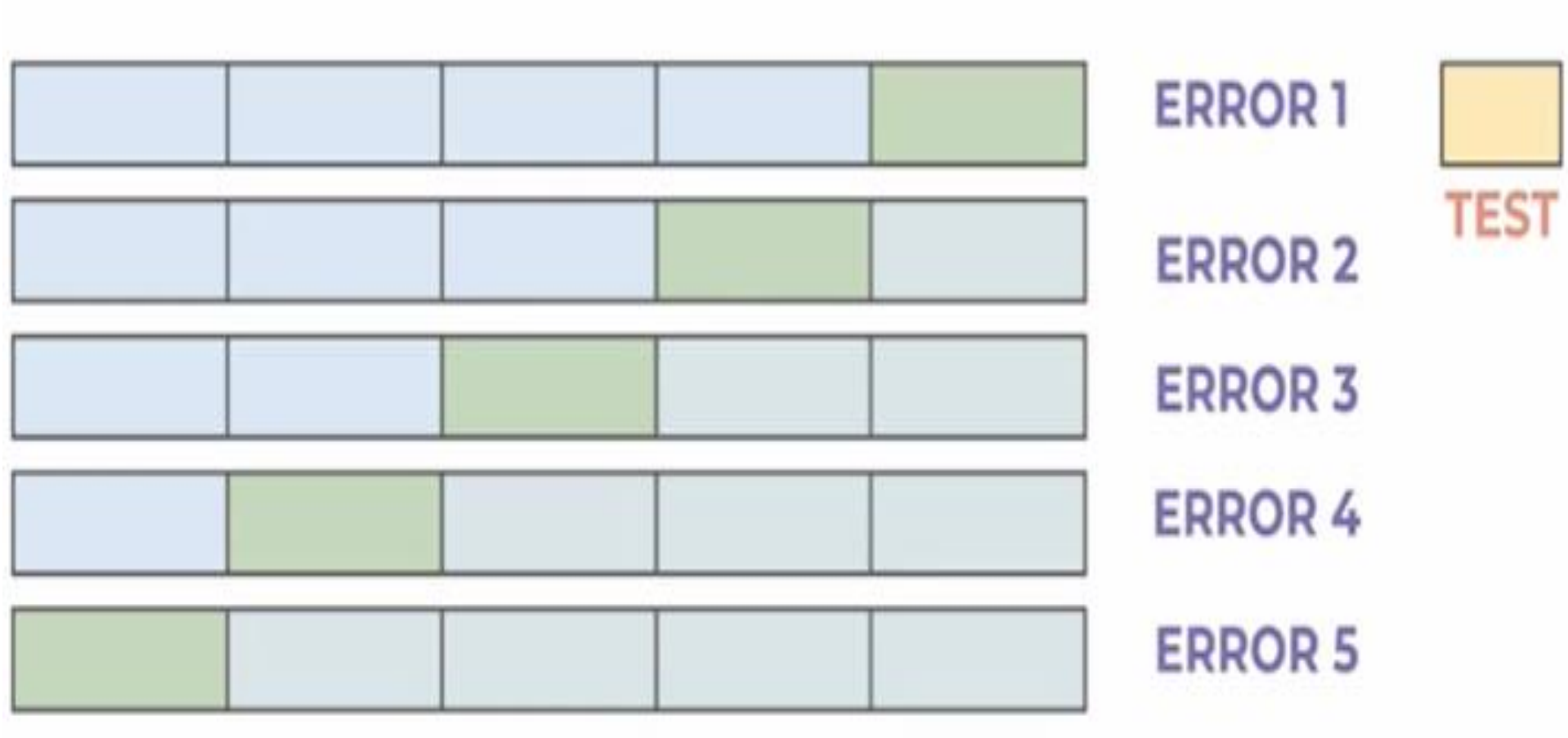
$$X_{changed} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Масштабирование в sklearn

- Метод ***.fit()*** проводит предварительную работу, вычисляет необходимые значения. Применяется только для обучающего набора данных
- Метод ***.transform()*** масштабирует данные и возвращает новую версию данных
- Целевую переменную не масштабируем

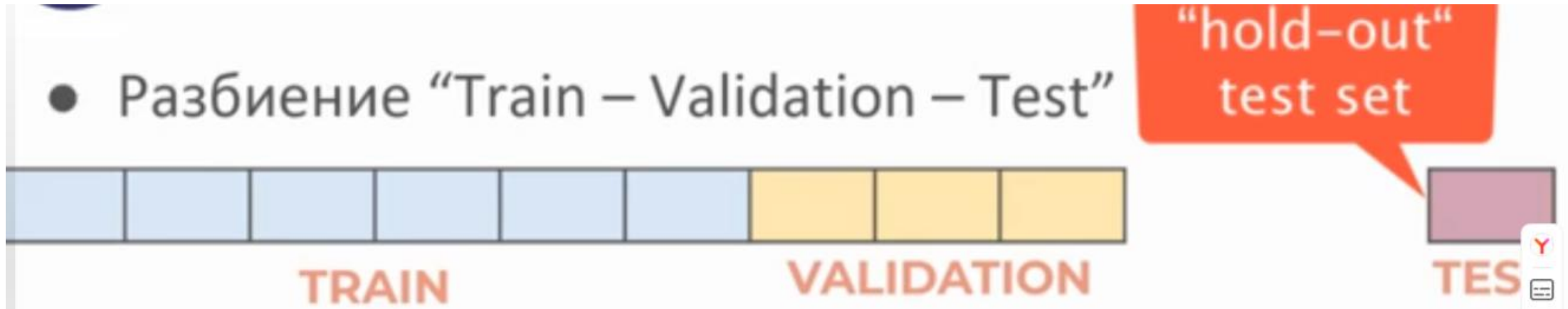
Кросс-валидация

- Можно ли
- обучать модель на всем наборе данных
- Проверять модель на всем наборе данных


ERROR 1
ERROR 2
ERROR 3
ERROR 4
ERROR 5
TEST

- Метод cross_val_score() позволяет сделать это автоматически.
- На вход подается модель и обучющий набор данных
- Это позволяет применить кросс-валидацию в К шагов для любой модели
- Функция cross_validate() позволяет посмотреть различные метрики кросс-валидации, а также понять, сколько времени заняли процессы обучения и проверки

- Разбиение “Train – Validation – Test”



- Обучение на данных Train
- Проверка и выбор гиперпараметров на Validation