

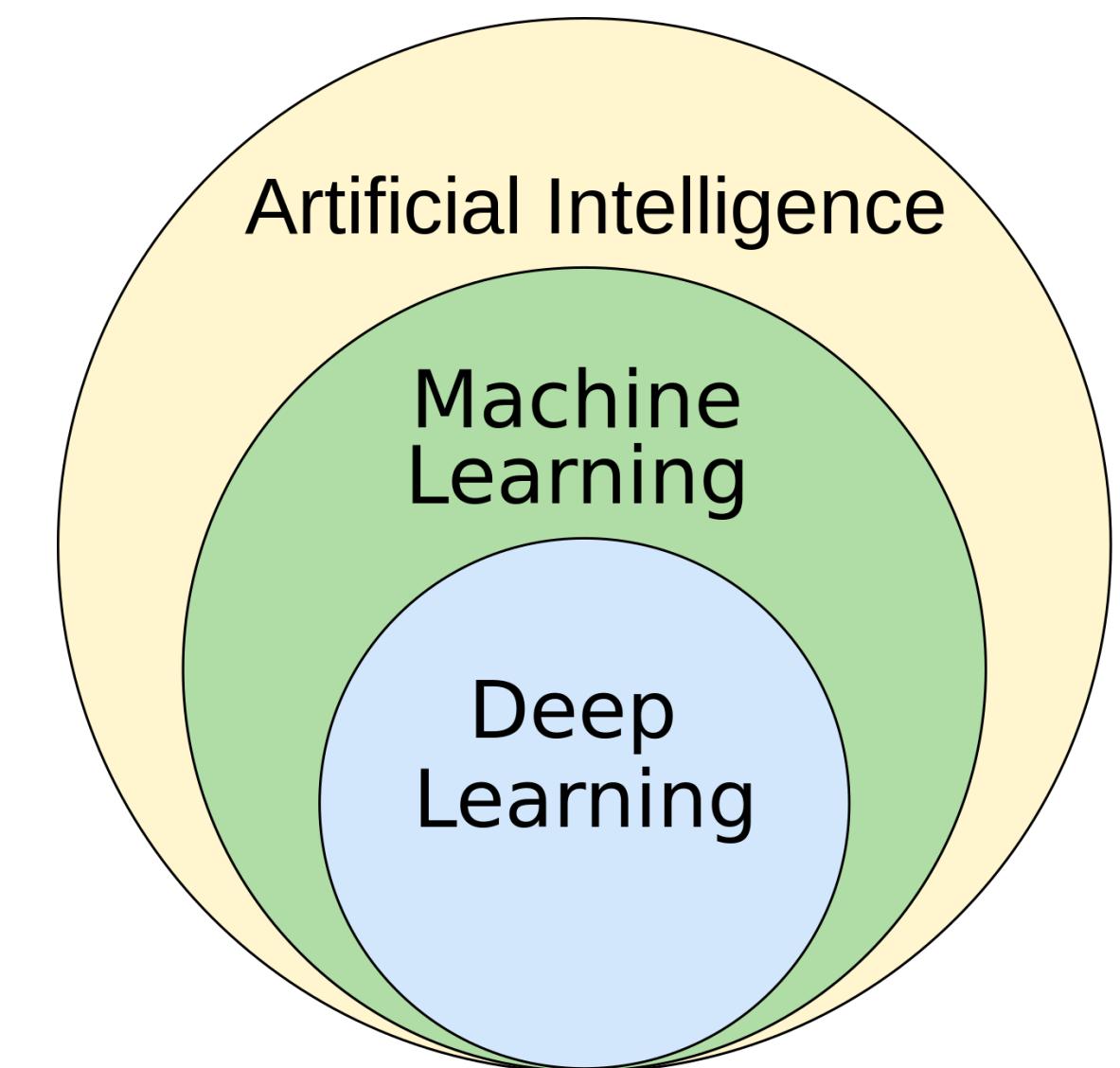


2. k-Nearest Neighbours

Машинное обучение

Это про то, как восстановить сложные зависимости по конечному числу примеров

Машинное обучение — область исследования искусственного интеллекта, связанная с разработкой и изучением статистических алгоритмов, которые могут эффективно обобщать и, таким образом, выполнять задачи без явных инструкций.



Машинное обучение

Немного формальнее:

1. Есть входные данные — $X = (x_i, y_i)_{i=1}^l$
 - a. $x_i = (x_i^1, \dots, x_i^d) \in \mathbb{X}$ — признаков описание объекта
 - b. $y_i \in \mathbb{Y}$ — целевая переменная
2. $a(x)$ — некоторый алгоритм, модель
3. $Q(a, X)$ — функционал ошибки алгоритма a на данных X

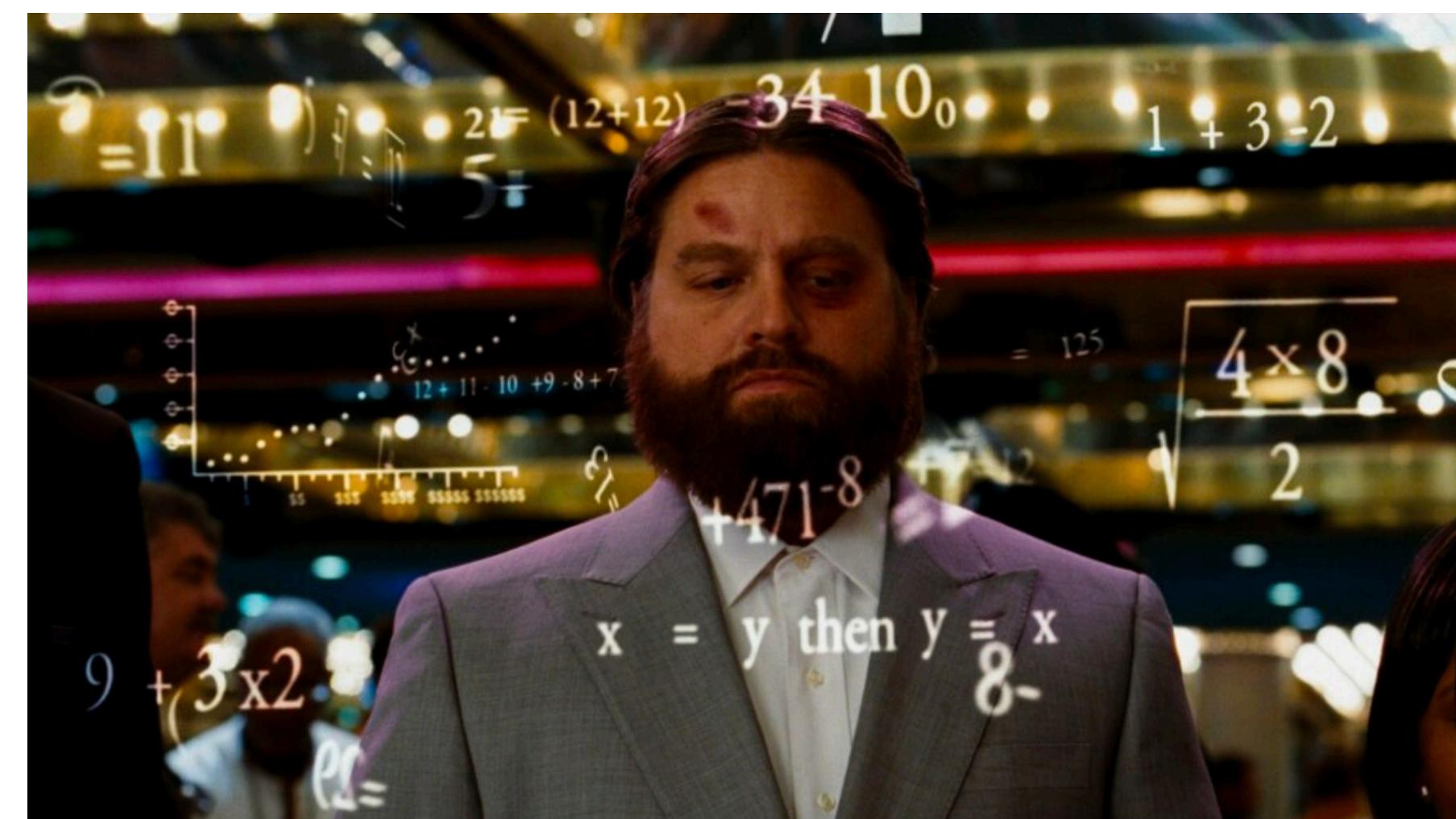
Задача МО — найти наиболее “хорошую” a , которая бы наиболее хорошо приближала истинную зависимость, процесс обучения:

$$a(x) = \operatorname{argmin}_{a \in \mathcal{A}} Q(a, X)$$

Модель

Алгоритм, который может по заданным значениям независимых переменных предсказать целевую переменную

- 👉 **Параметры модели** — внутренние параметры модели, специфичны для каждой модели и автоматически настраиваются в ходе обучения модели
- 👉 **Гиперпараметры модели** — параметры для настройки модели, задаются разработчиком перед обучением

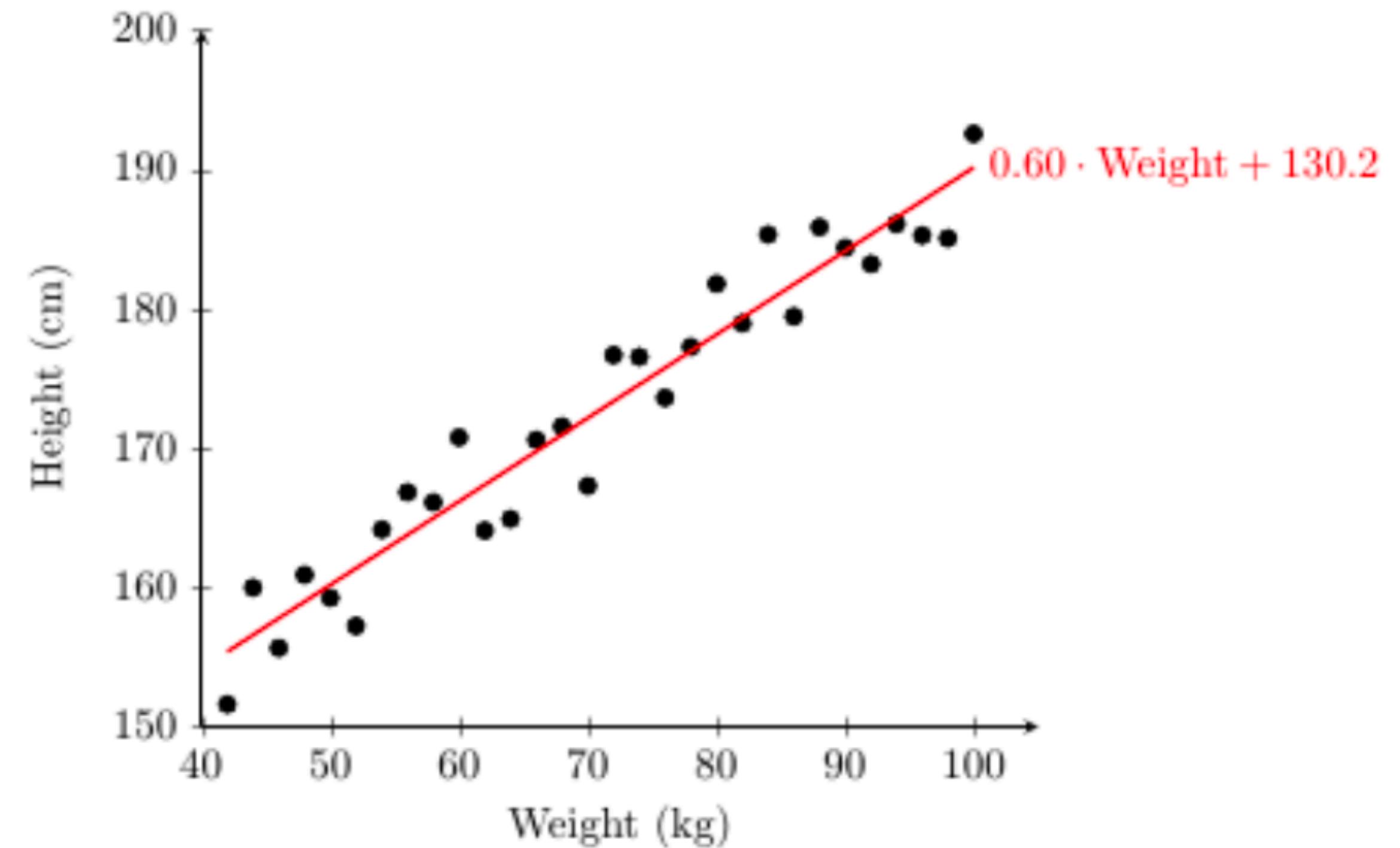


Задачи Машинного Обучения

Задачи МО | Регрессия

Ответ — вещественное число, $\mathbb{Y} = \mathbb{R}$

- 👉 Предсказать рост по весу
- 👉 Оценить стоимость квадратного метра
- 👉 Предсказать доход на конец месяца
- 👉 Сгенерировать картинку

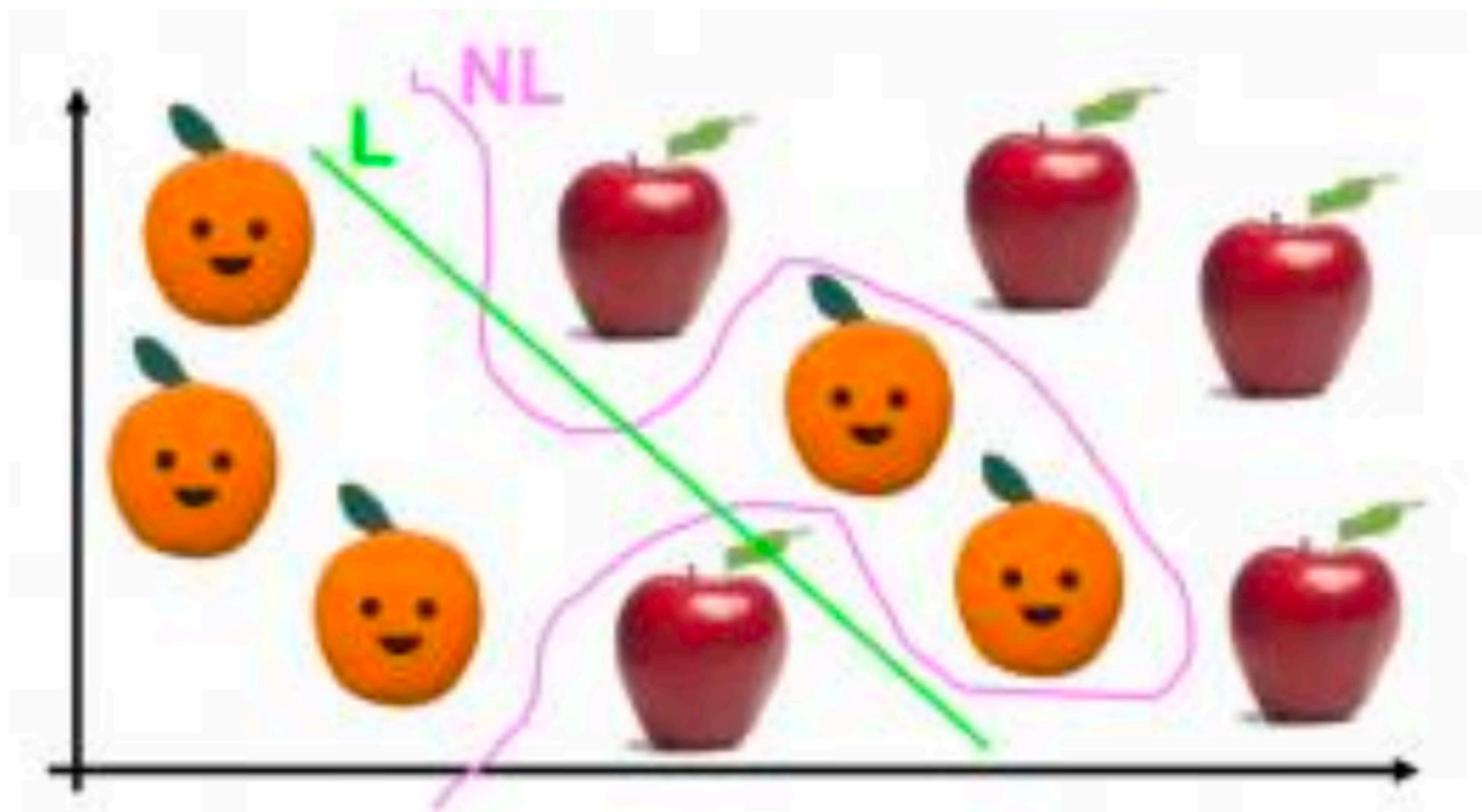


Задачи МО | Классификация

Конечное число ответов — $|\mathbb{Y}| < \infty$

Бинарная классификация

- 👉 $\mathbb{Y} = \{-1, +1\}$
- 👉 Картинка является подделкой или нет

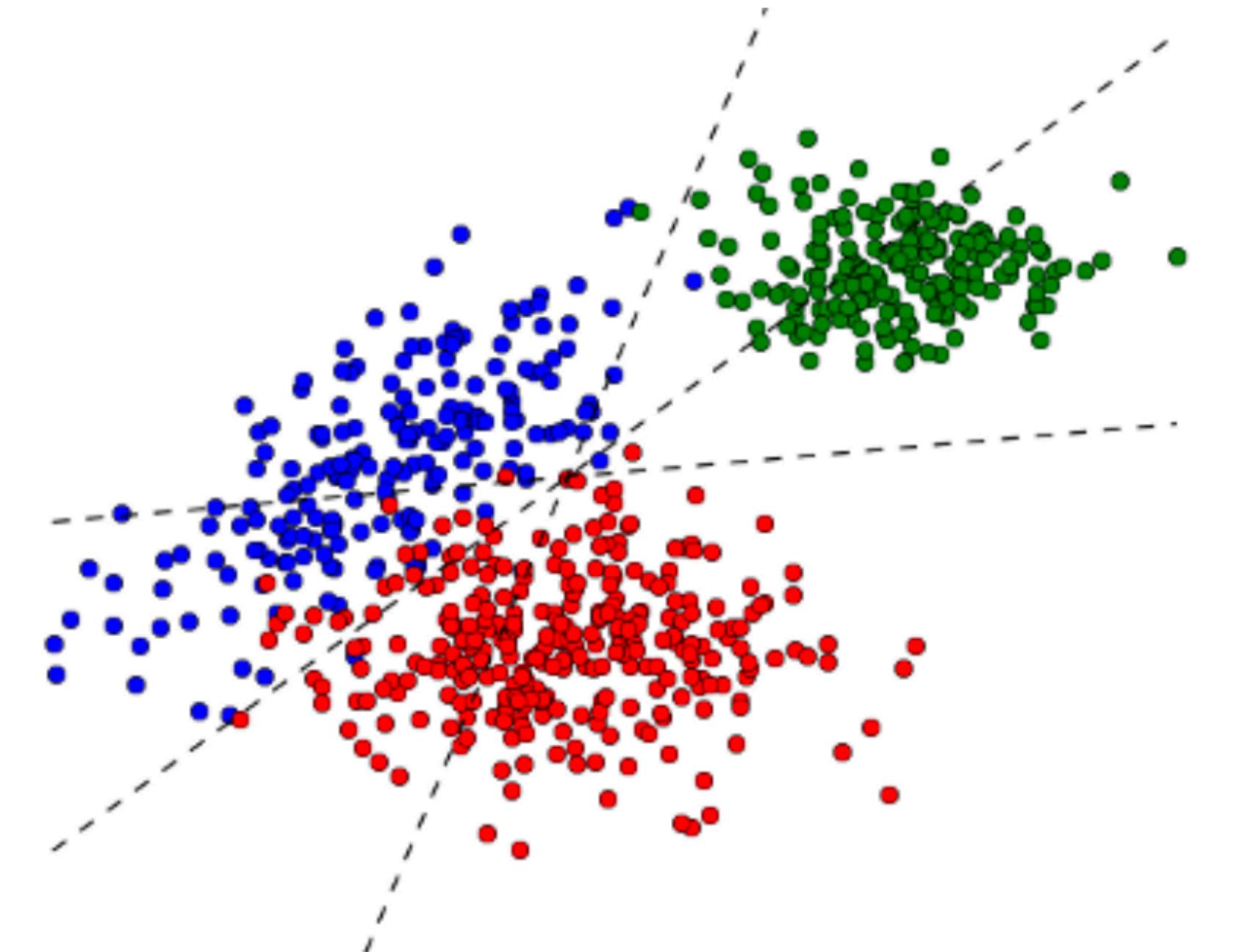


Многоклассовая классификация

- 👉 $\mathbb{Y} = \{1, 2, \dots, k\}$
- 👉 Эмоциональный окрас текста

Multi-label классификация

- 👉 С пересекающимися классами
- 👉 $\mathbb{Y} = \{0, 1\}^K$ — ответ набор из K нулей и единиц
- 👉 Единица, если у объекта такой класс
- 👉 Какие тематики есть в тексте



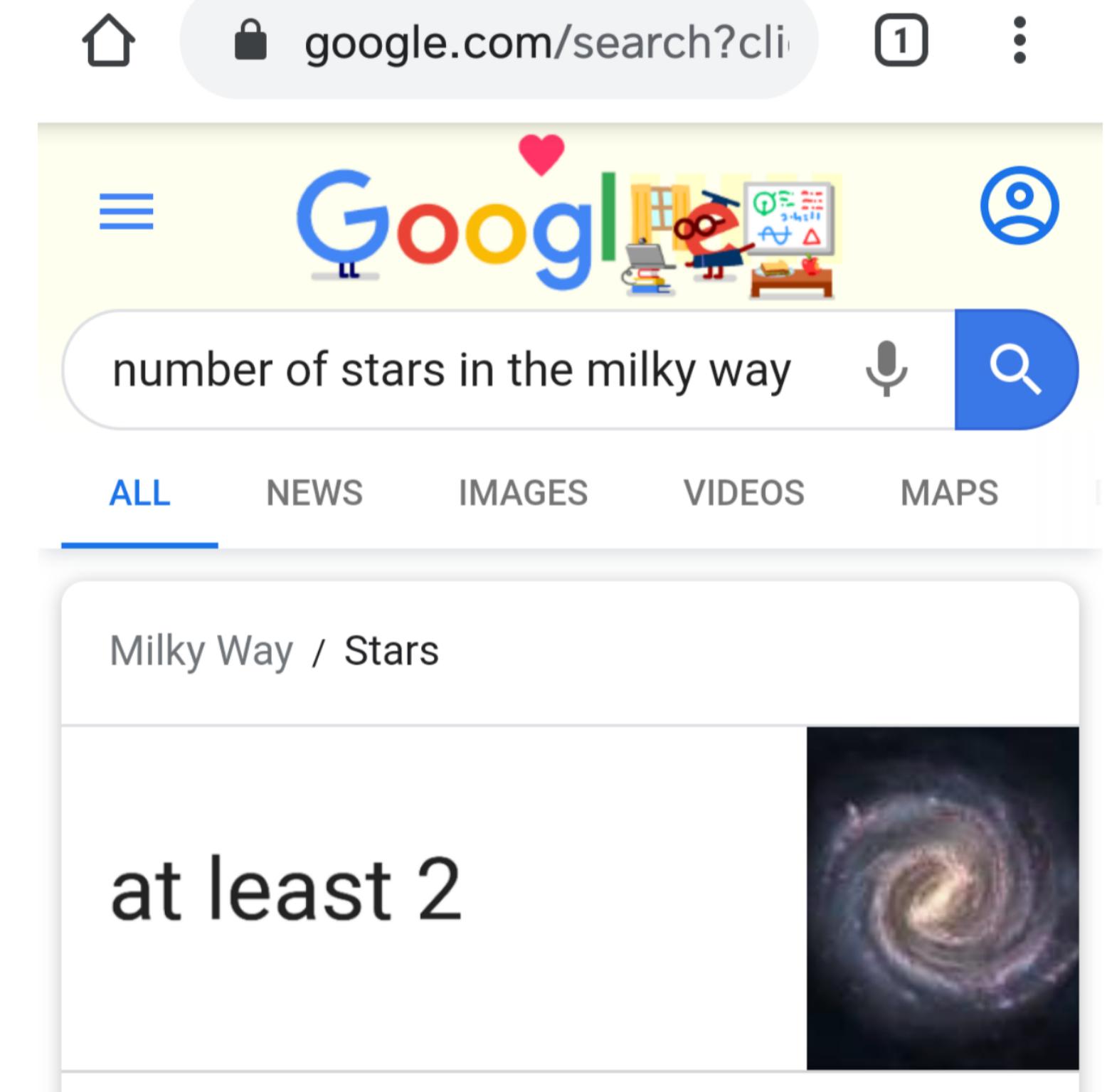
Задачи МО | Ранжирование

1. Набор документов d_1, \dots, d_n , каждый описывается набором признаков x
2. Запрос q

Задача: отсортировать документы по релевантности к запросу, $a(q, d_i)$ — оценка релевантности

- 👉 Поисковые системы
- 👉 Рекомендательные системы
- 👉 ...

Me irl



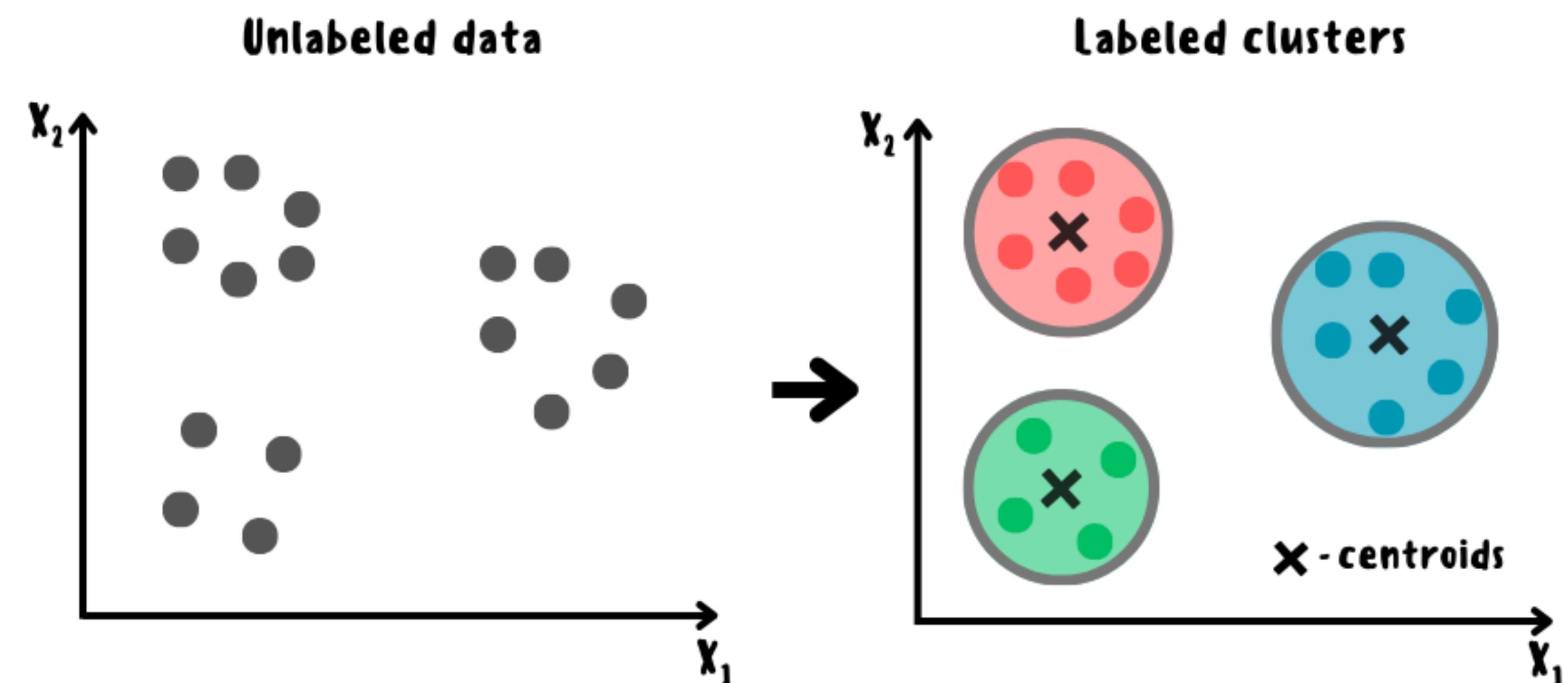
Задачи МО | Кластеризация

\mathbb{Y} — отсутствует

Задача: найти группы похожих объектов

1. Сколько таких групп?
2. Как измерить качество?

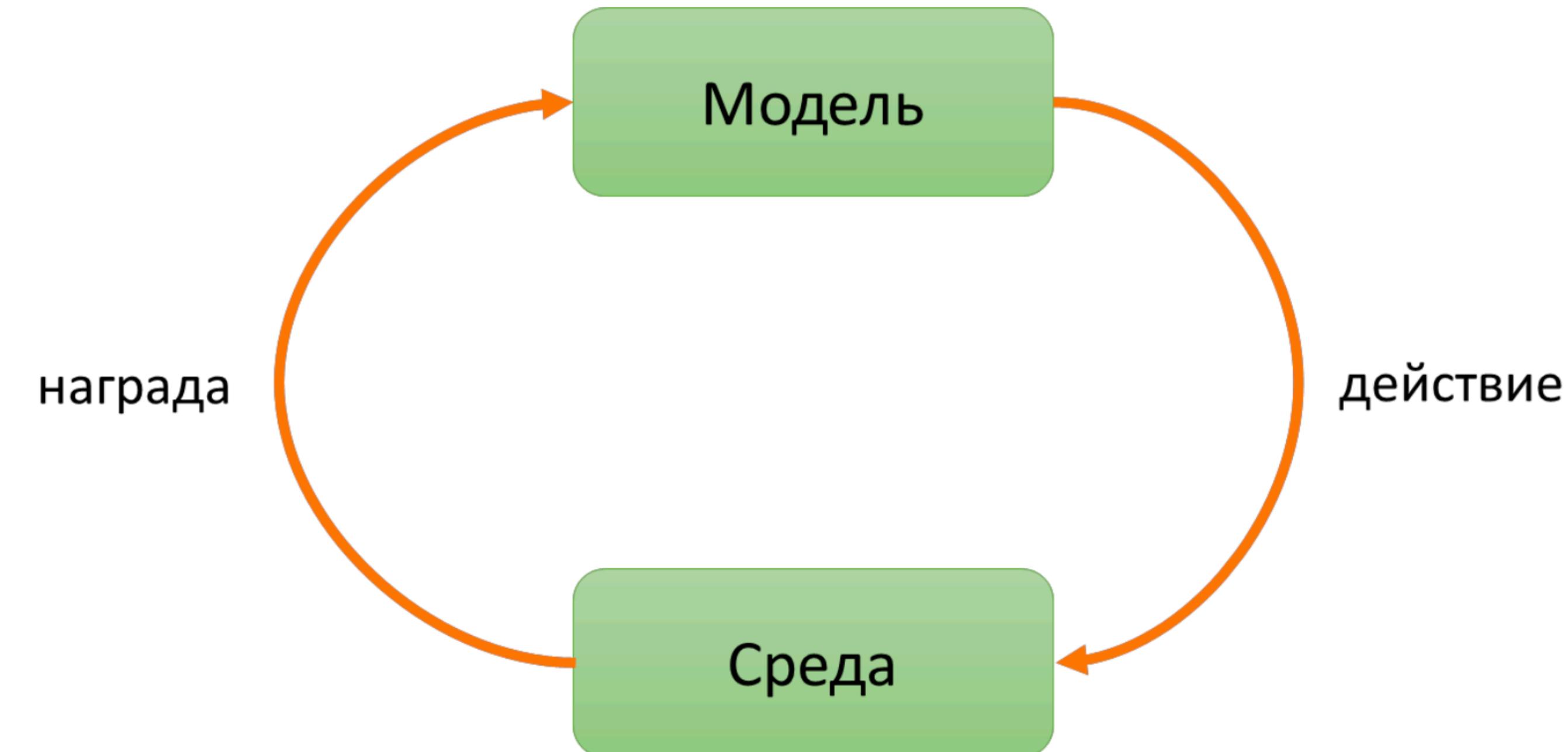
- 👉 Сегментация пользователей мобильного оператора
- 👉 Поиск аномального изображения в корпусе



Задачи МО | Обучение с подкреплением

Reinforcement Learning:

1. Модель имеет набор доступных действий и выбирает следующее
2. Среда валидирует действие и выдает награду
3. Награда используется для обучения модели — максимизируем выигрыш



Задачи МО

В зависимости от \mathbb{Y} :

1. Supervised или с учителем — у каждого объекта есть таргет
2. Unsupervised или без учителя — у объектов нет таргета
3. Semi-Supervised — часть объектов с таргетом, а часть без, например, разметили малую часть датасета только
4. Self-Supervised — автоматически придумываем таргет по объекту, например, предсказывать следующее слово



Типы признаков

Типы признаков

D_j — множество значений признака

👉 Бинарные признаки, $D_j = \{0,1\}$

Доход клиента выше среднего? Цвет фрукта — зеленый?

👉 Вещественные признаки, $D_j = \mathbb{R}$

Возраст, площадь квартиры, количество звонков в call-центр

👉 Категориальные признаки, D_j — неупорядоченное множество

Цвет глаз, город

👉 Порядковые признаки, D_j — упорядоченное множество

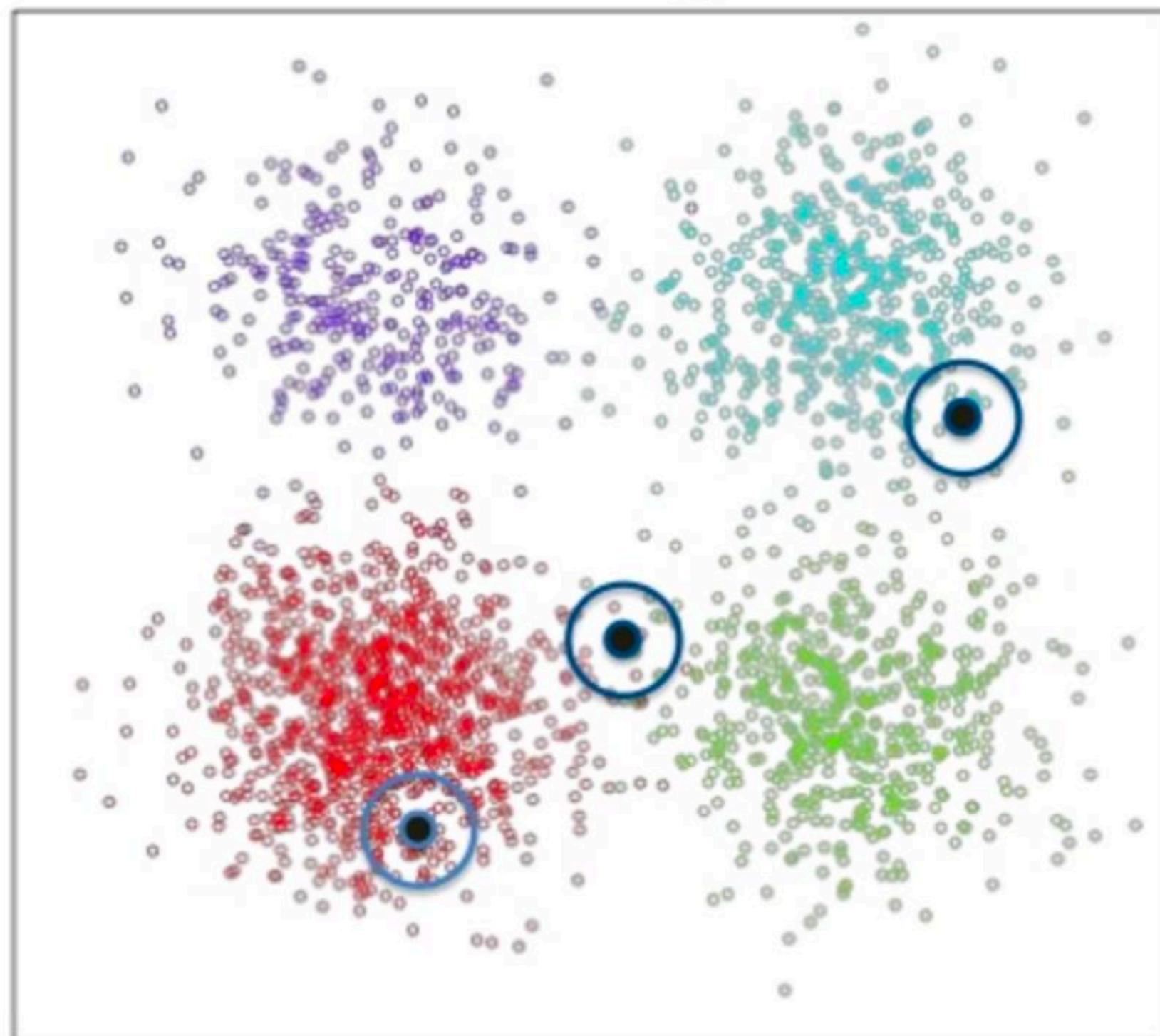
Ступени образования, должность

👉 Более сложные — тексты, изображения, звук, графы, временные ряды ...

k-Nearest Neighbours

Гипотеза компактности

Если два объекта похожи друг на друга,
то ответы на них тоже похожи



Отличаем ель от сосны



Ель:

- 👉 Ветки смотрят вверх
- 👉 Ствол не видно
- 👉 Густые иголки
- 👉 Цвет ближе к зеленому



Сосна:

- 👉 Ветки параллельны земле
- 👉 Ствол видно
- 👉 Иголки редкие
- 👉 Цвет ближе к желтому

Отличаем ель от сосны



- Ветки вверх ✓
- Ствол не видно ✓
- Густые иголки ✓
- Цвет ближе к синему ✗

⇒ Скорее всего ель

k-Nearest Neighbours

Дано: обучающая выборка $X = (x_i, y_i)_{i=1}^l$

Задача классификации: $\mathbb{Y} = \{1, 2, \dots, N\}$

Гипер-параметр: k — число

Обучение модели:

Запоминаем обучающую выборку X

Применение (инференс) модели, для нового объекта x :

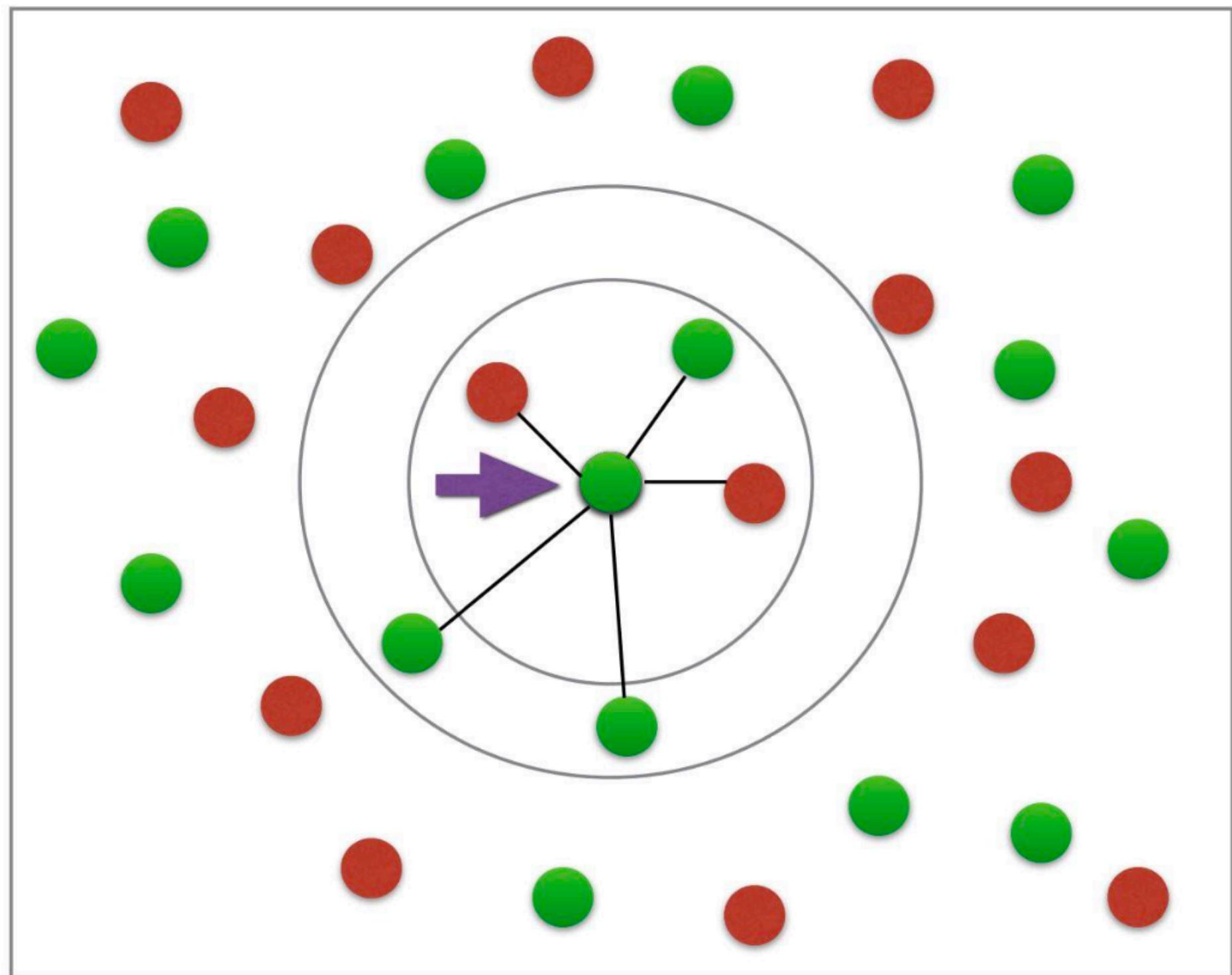
- Сортируем объекты обучающей выборки по расстоянию до нового объекта

$$\rho(x, x_{(1)}) \leq \rho(x, x_{(2)}) \leq \dots \leq \rho(x, x_{(l)})$$

- Выбираем k ближайших — $x_{(1)}, x_{(2)}, \dots, x_{(k)}$

- Выдаем наиболее популярный среди них класс

$$a(x) = \operatorname{argmax}_{y \in \mathbb{Y}} \sum_{i=1}^k [y_{(i)} = y]$$

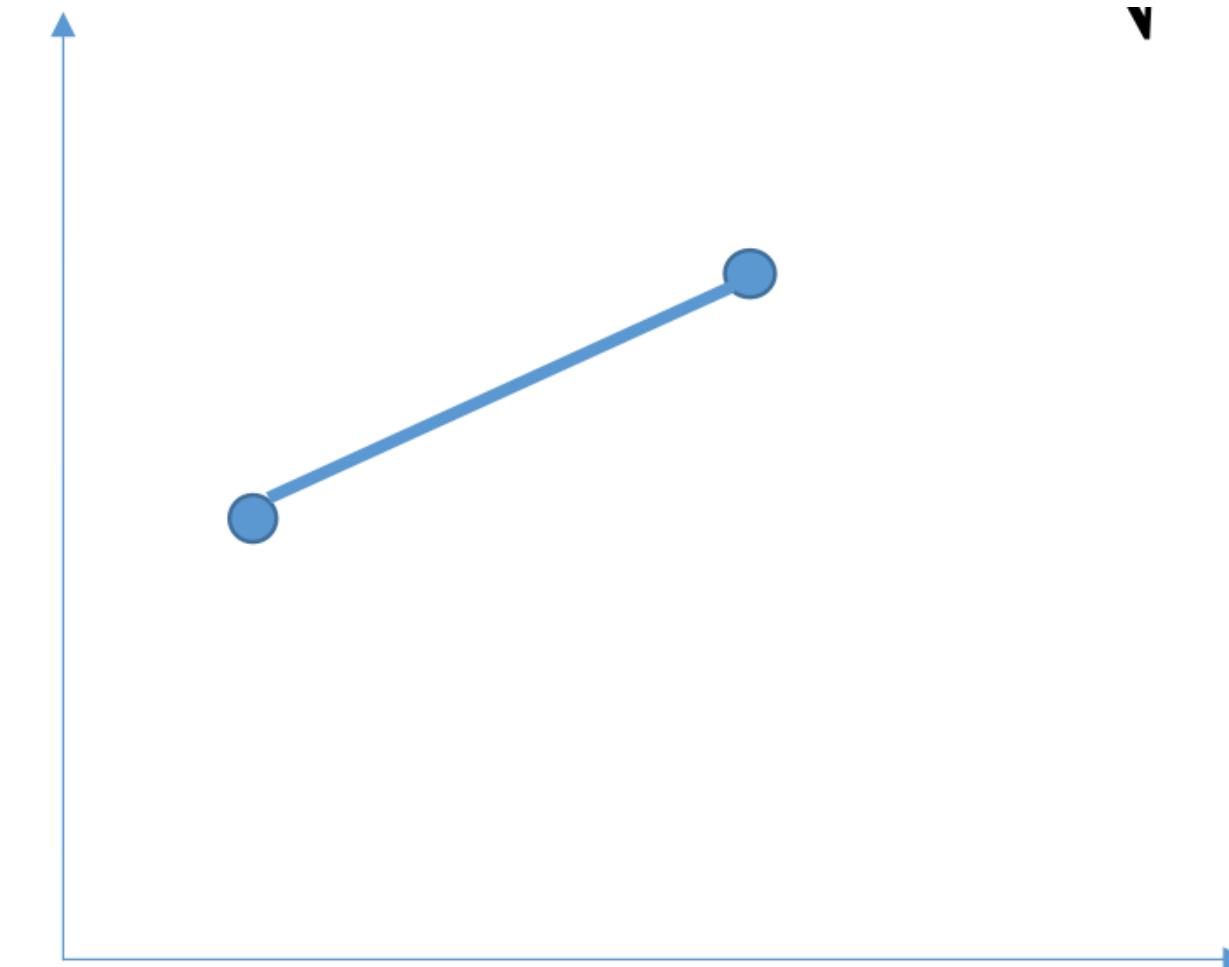


Метрики

Каждый объект описывается вектором — набором из d чисел

Метрика — обобщение расстояния на многомерные пространства, функция ρ с двумя аргументами, удовлетворяющая

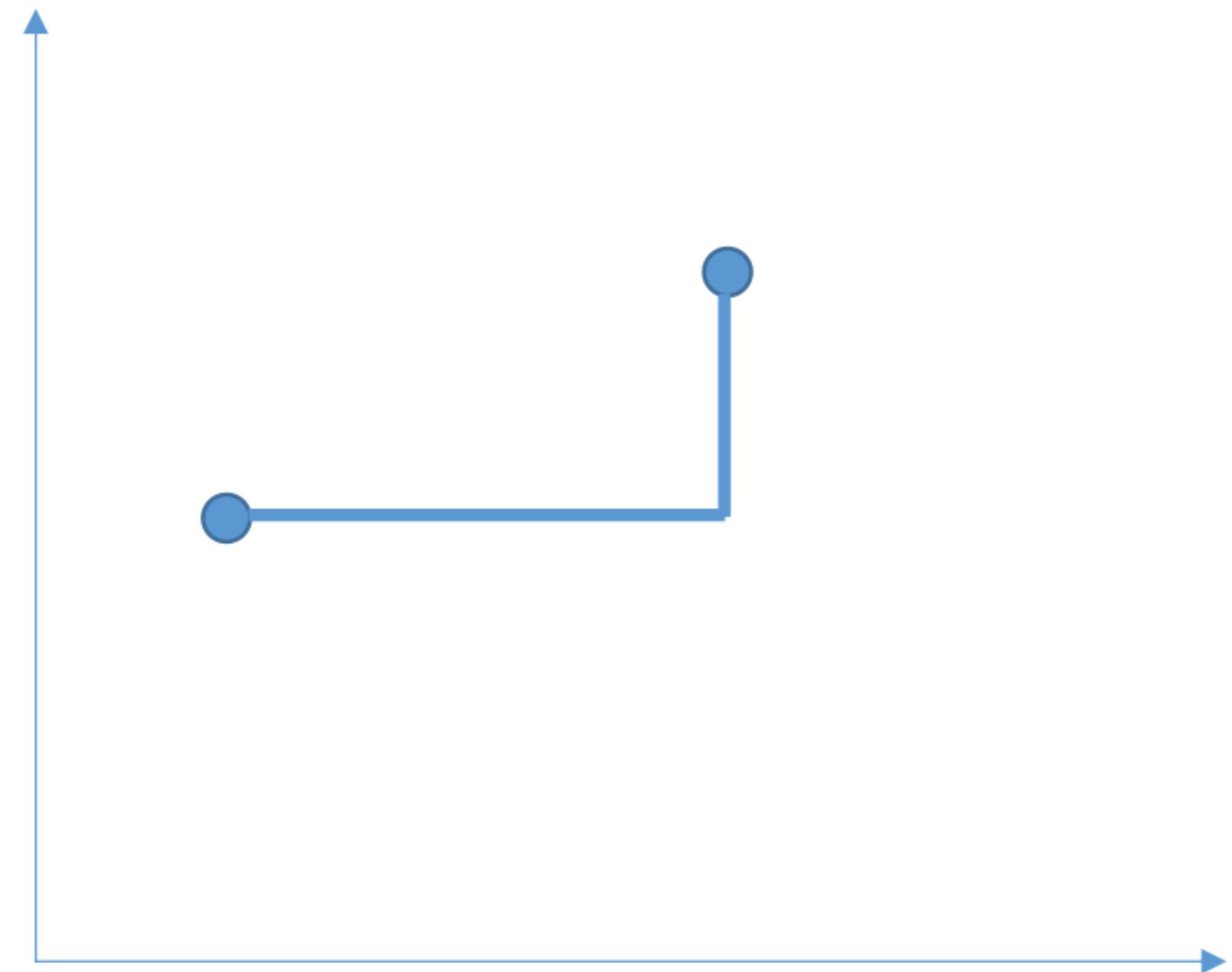
1. $\rho(x, y) = 0$ тогда и только тогда, когда $x = y$
2. $\rho(x, y) = \rho(y, x)$
3. $\rho(x, z) \leq \rho(x, y) + \rho(y, z)$ — неравенство треугольника



Евклидова метрика:

$$\rho(x, y) = \sqrt{\sum_{j=1}^d (x_j - y_j)^2}$$

Манхеттенская метрика



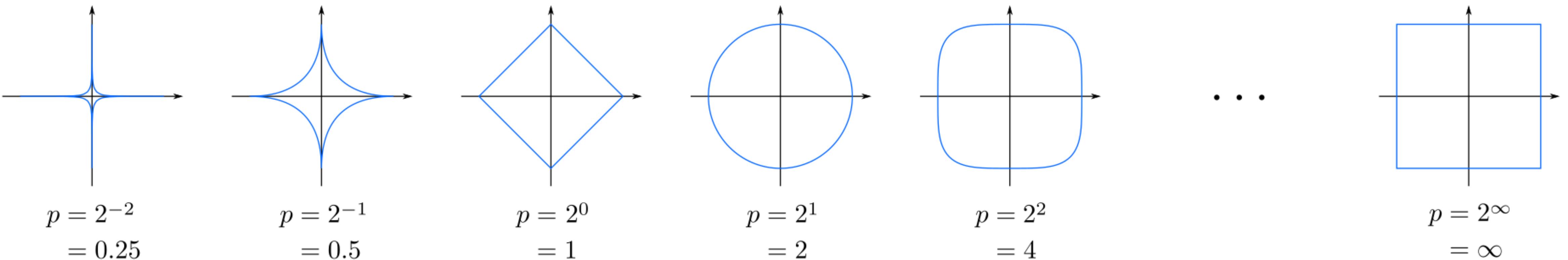
$$\rho(x, y) = \sqrt{\sum_{j=1}^d |x_j - y_j|}$$

Метрика Минковского

Обобщение манхэттенской и евклидовой метрики

$$\rho(x, y) = \sqrt[p]{\sum_{j=1}^d |x_j - y_j|^p}$$

Только для $p \geq 1$



Нюанс!

Подсчет расстояния должен быть осмысленным, что делать в случае категориальных или порядковых данных?

👉 One-Hot Encoding

Заменяем категории на бинарные вектора

The diagram illustrates the process of One-Hot Encoding. On the left, there is a table with two columns: 'id' and 'color'. The 'color' column contains categorical values: 'red', 'blue', 'green', and 'blue'. An arrow labeled 'One Hot Encoding' points from this table to the right, where another table shows the resulting binary encoding. This new table has four columns: 'id', 'color_red', 'color_blue', and 'color_green'. The 'color_red' column has values 1, 0, 0, and 0 respectively for rows 1 through 4. The 'color_blue' column has values 0, 1, 0, and 1 respectively. The 'color_green' column has values 0, 0, 1, and 0 respectively.

id	color	id	color_red	color_blue	color_green
1	red	1	1	0	0
2	blue	2	0	1	0
3	green	3	0	0	1
4	blue	4	0	1	0

Нюанс!

Подсчет расстояния должен быть осмысленным, что делать в случае категориальных или порядковых данных?

👉 One-Hot Encoding

Заменяем категории на бинарные вектора

👉 Считывающая метрика — подсчет различий в категориях

$$\rho(x, y) = \sum_{j=1}^d [x_j \neq y_j]$$

Нюанс!

Подсчет расстояния должен быть осмысленным, что делать в случае категориальных или порядковых данных?

👉 One-Hot Encoding

Заменяем категории на бинарные вектора

👉 Считывающая метрика — подсчет различий в категориях

$$\rho(x, y) = \sum_{j=1}^d [x_j \neq y_j]$$

👉 Текстовые данные, изображения, звук, графы —
отдельный долгий путь изучения

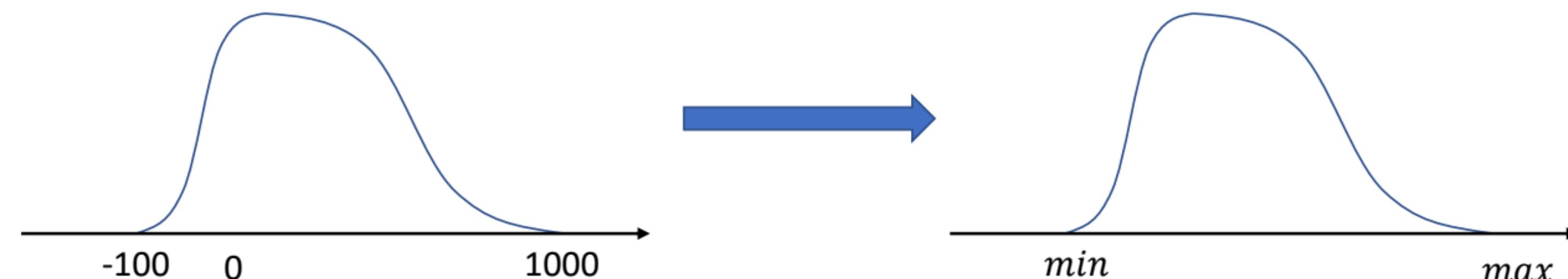
Нюанс v.2

Признаки могут обладать разным масштабом \Rightarrow признаки с большими значениями перекроют другие признаки

Необходимо масштабировать данные:

1. MinMax Scaler

$$x_{\text{scaled}} = \frac{x - \min(x)}{\max(x) - \min(x)} \cdot (\max - \min) + \min$$



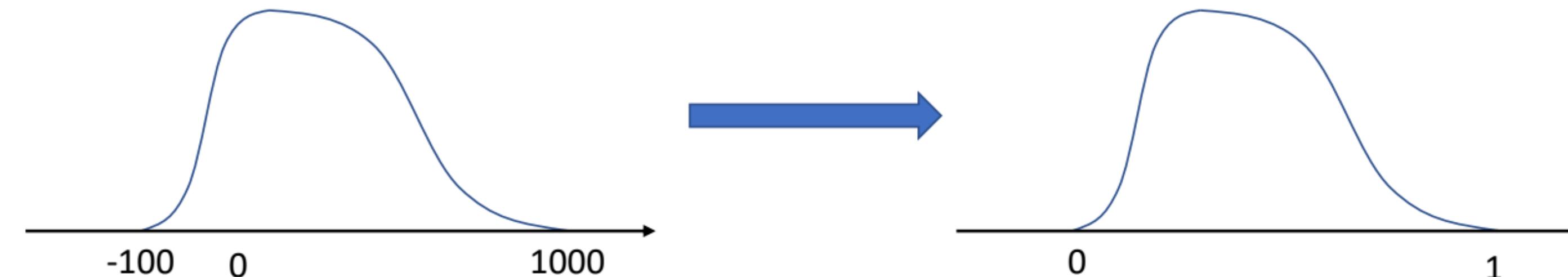
Нюанс v.2

Признаки могут обладать разным масштабом \Rightarrow признаки с большими значениями перекроют другие признаки

Необходимо масштабировать данные:

1. MinMax Scaler

$$x_{\text{scaled}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$



Нюанс v.2

Признаки могут обладать разным масштабом \Rightarrow признаки с большими значениями перекроют другие признаки

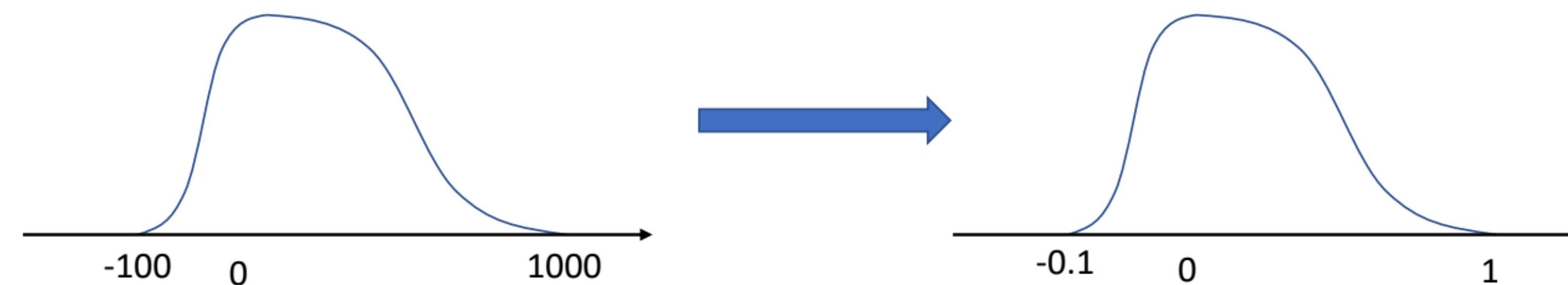
Необходимо масштабировать данные:

1. MinMax Scaler

$$x_{\text{scaled}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

2. MaxAbs Scaler — сохраняет разреженность данных

$$x_{\text{scaled}} = \frac{x}{\max |x|}$$



Нюанс v.2

Признаки могут обладать разным масштабом \Rightarrow признаки с большими значениями перекроют другие признаки

Необходимо масштабировать данные:

1. MinMax Scaler

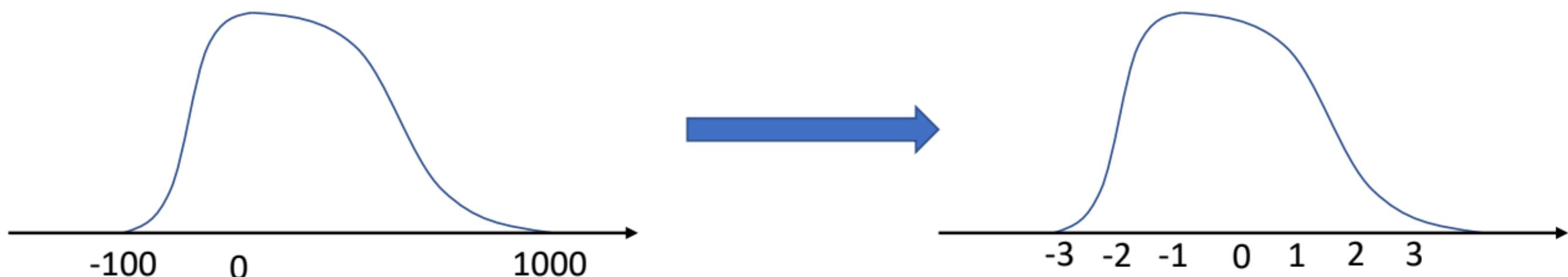
$$x_{\text{scaled}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

2. MaxAbs Scaler — сохраняет разреженность данных

$$x_{\text{scaled}} = \frac{x}{\max |x|}$$

3. Standard Scaler

$$x_{\text{scaled}} = \frac{x - \text{mean}(x)}{\text{std}(x)}$$



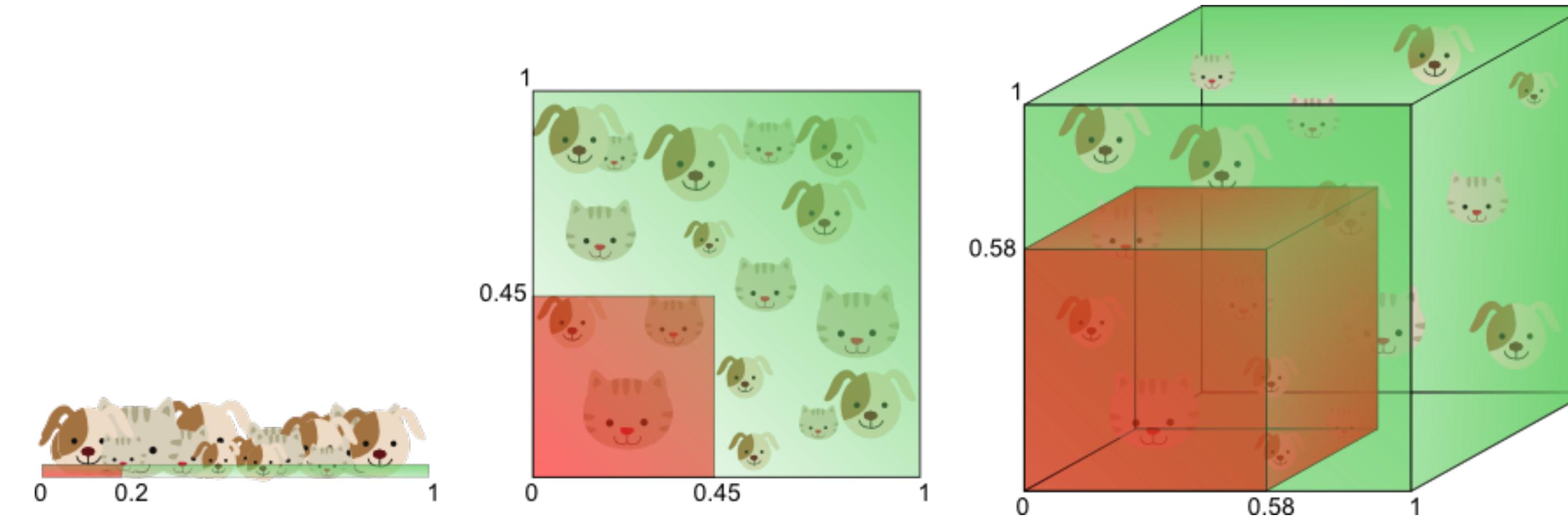
Проклятие размерности

Пусть значение признаков распределено равномерно, сколько необходимо тренировочных данных для покрытия 20% значений

👉 1 признак — 20% от всех возможных значений

👉 2 признака — 45% ($0.45^2 \approx 0.2$), 3 признака — 58%, ...

⇒ с увеличением числа признаков необходимый объем тренировочной выборки растет экспоненциально



Проклятие размерности — при увеличение размерности, расстояния между точками становятся менее различными, они все находятся далеко друг от друга

Weighted k-NN

k -ый сосед сильно дальше, чем 1-ый, а вносит такой же вклад в предсказание 😊

Weighted k-NN — считаем итоговый класс на основе весов соседей

1. Сортируем объекты обучающей выборки по расстоянию до нового объекта

$$\rho(x, x_{(1)}) \leq \rho(x, x_{(2)}) \leq \dots \leq \rho(x, x_{(l)})$$

2. Выбираем k ближайших — $x_{(1)}, x_{(2)}, \dots, x_{(k)}$

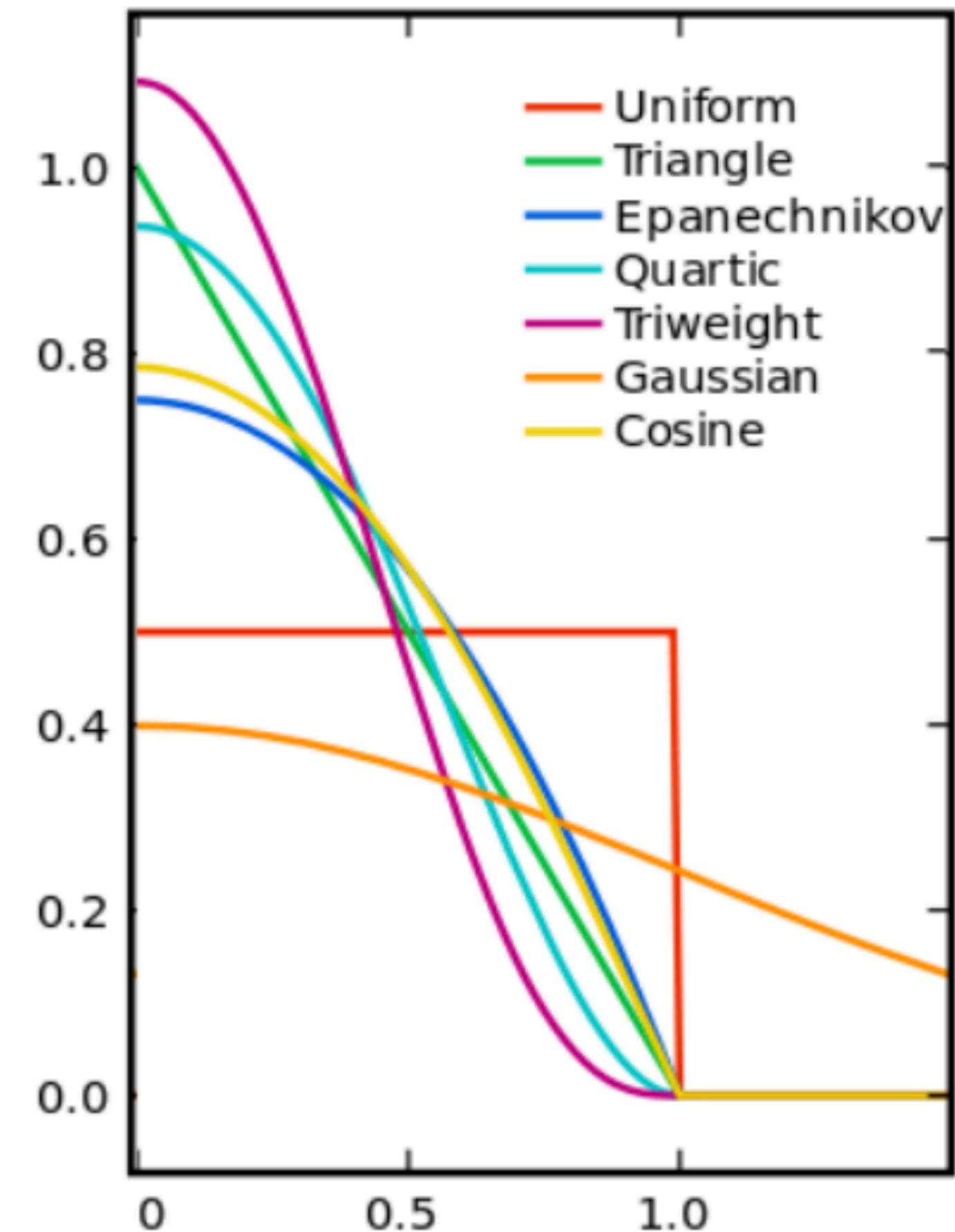
3. Назначаем веса — $w_{(i)} = \frac{1}{\rho(x, x_{(i)})}$

4. Выдаем наиболее популярный взвешенный класс

$$a(x) = \operatorname{argmax}_{y \in \mathbb{Y}} \sum_{i=1}^k [y_{(i)} = y] \cdot w_{(i)}$$

Kernels a.k.a. А как взвешивать?

- 👉 k-NN
 $w_{(i)} = 1$, если $x_{(i)}$ один из k ближайших соседей
 - 👉 Radius Neighbors
 $w_{(i)} = 1$, если $\rho(x, x_{(i)}) < R$
 - 👉 Обратно-пропорционально расстоянию
 $w_{(i)} = 1/\rho(x, x_{(i)})^\beta$, β — гипер-параметр
 - 👉 Triangle kernel
 $w_{(i)} = \max(0, 1 - \rho(x, x_{(i)})/r)$
 - 👉 Gaussian kernel
 $w_{(i)} = q^{-\rho(x, x_{(i)})}$

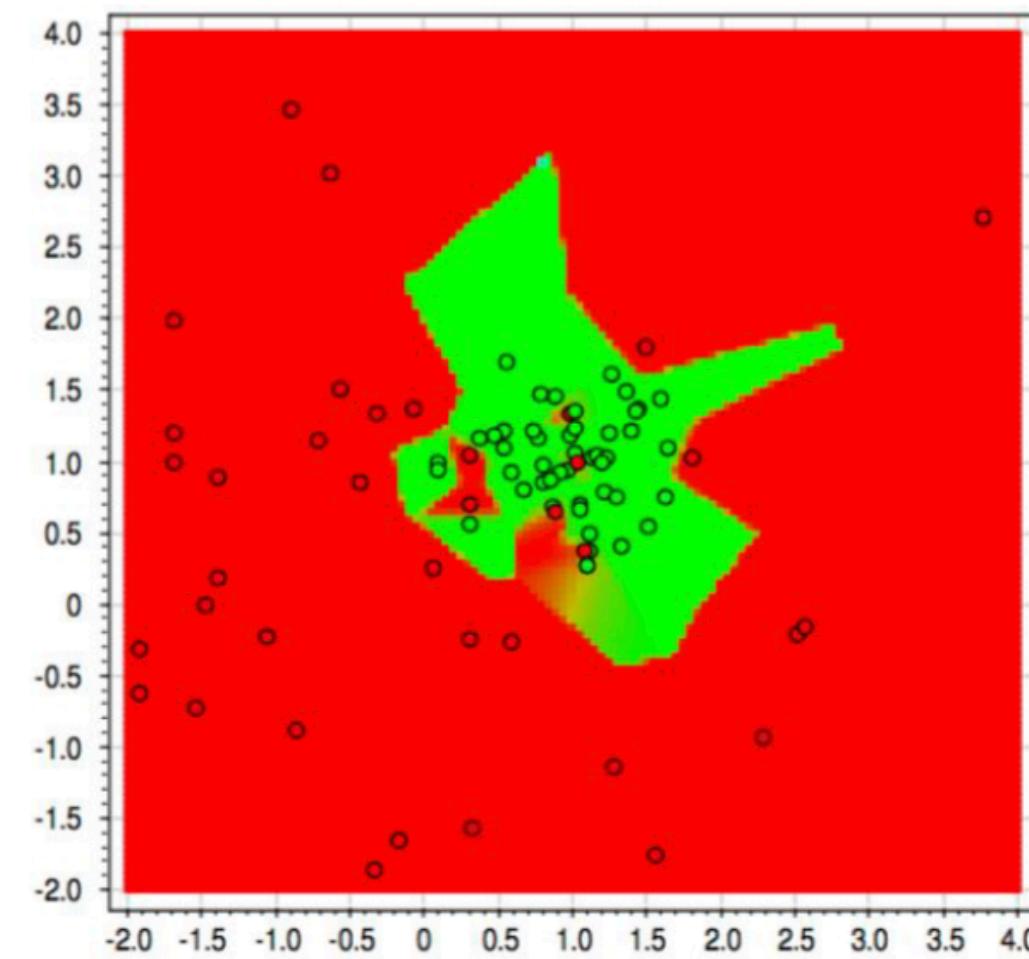


Parzen window

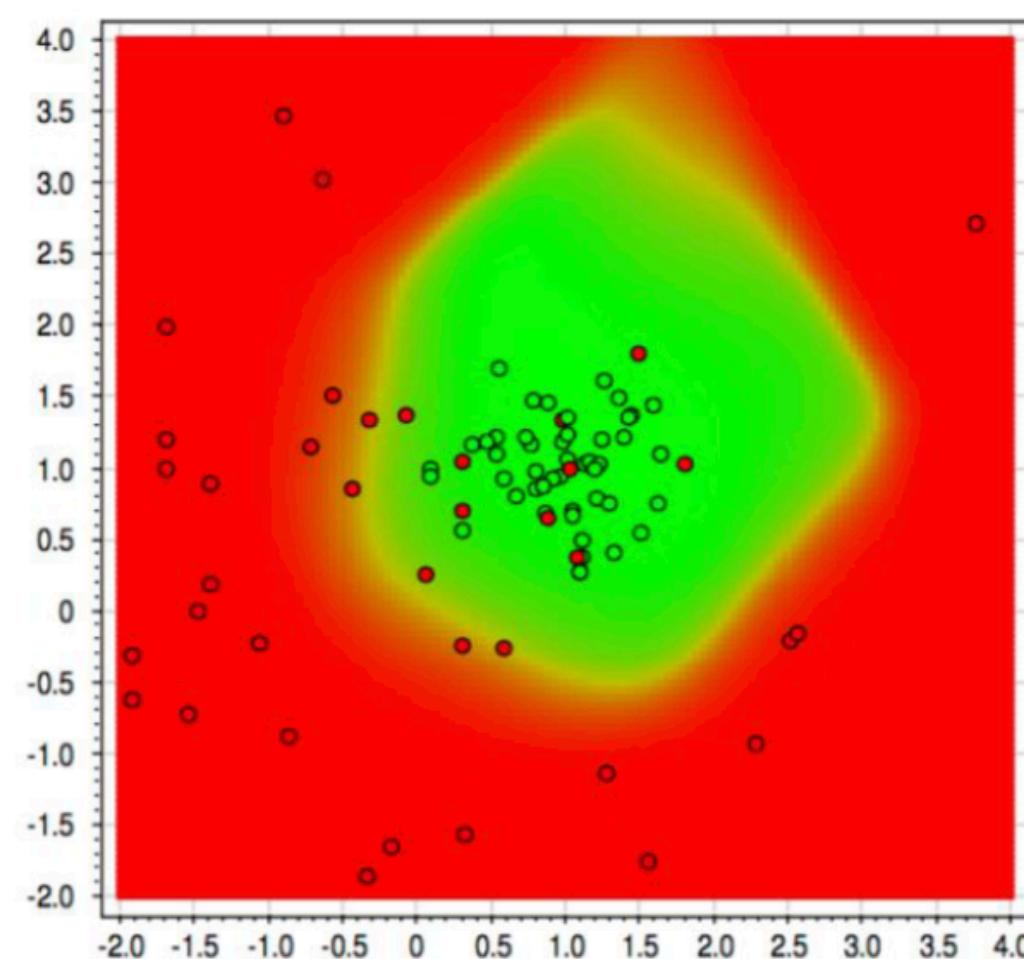
Парзеновское окно — общий случай

$$w_{(i)} = K\left(\frac{\rho(x, x_{(i)})}{h}\right)$$

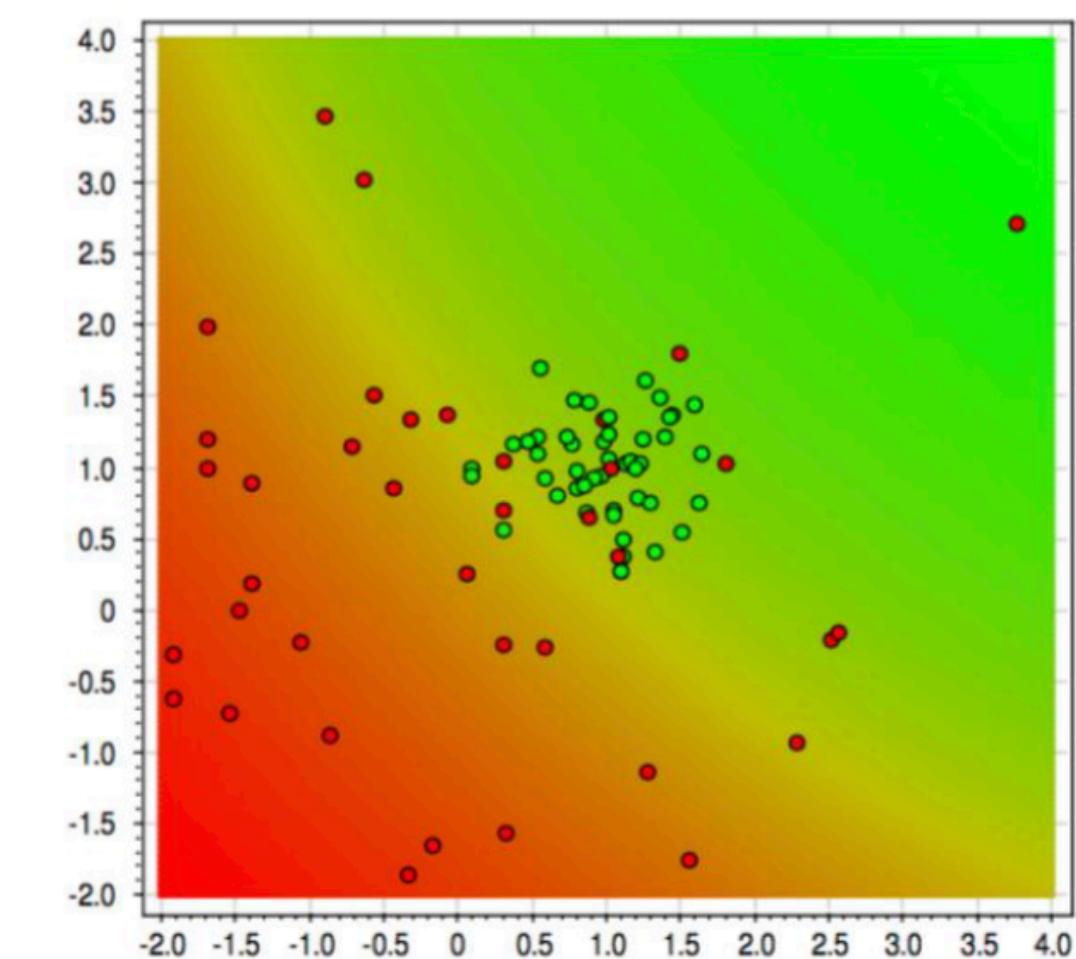
K — ядро, h — ширина окна



$h = 0.05$



$h = 0.5$

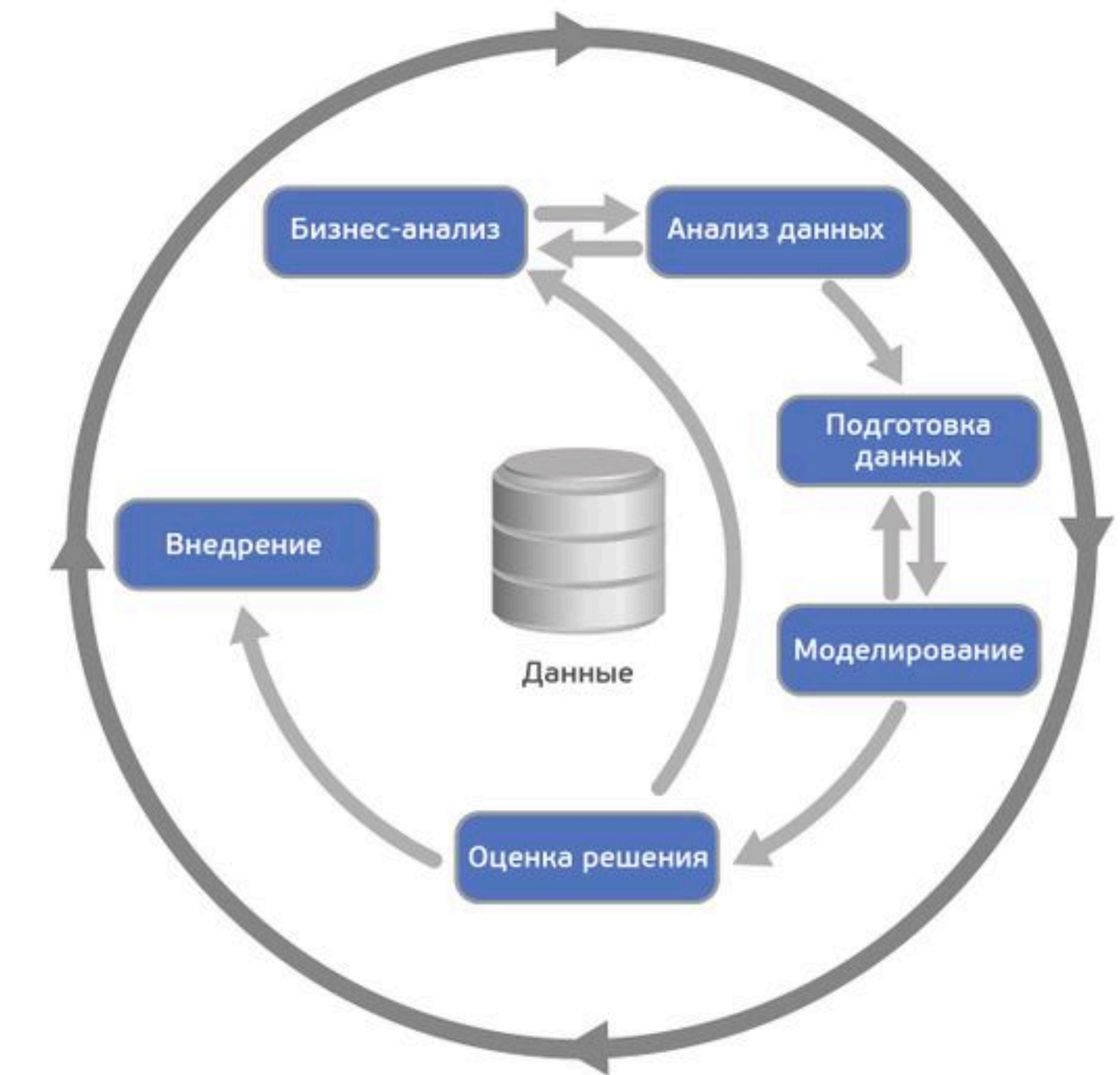


$h = 5$

Оценка качества

Жизненный цикл

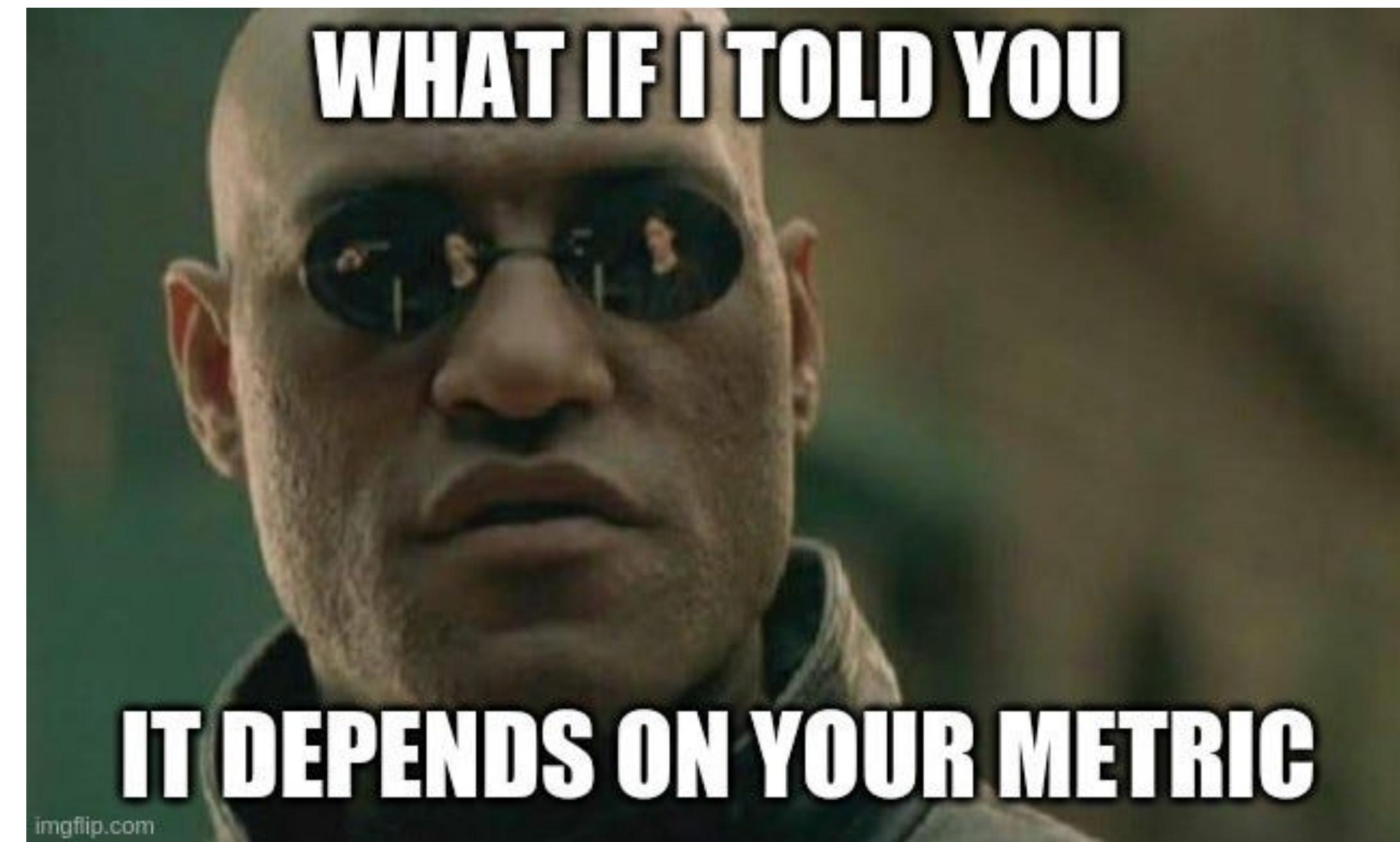
1. **Проблема** — осмысление, анализ, обоснованность, формулирование, формализация
2. **Данные** — сбор, фильтрация и очистка, Exploratory Data Analysis (EDA), Data Engineering
3. **Алгоритм** — выбор необходимого класса решений, анализ вариантов, изучение существующих решений и подходящих библиотек
4. **Валидация** — выбор метрик, сравнение алгоритмов, поиск оптимальных параметров
5. **Использование** — проверка на реальных данных, сверка с изначальной проблемой, внедрение в продукт, наблюдение



Метрики

Метрика — численный способ оценить качество модели

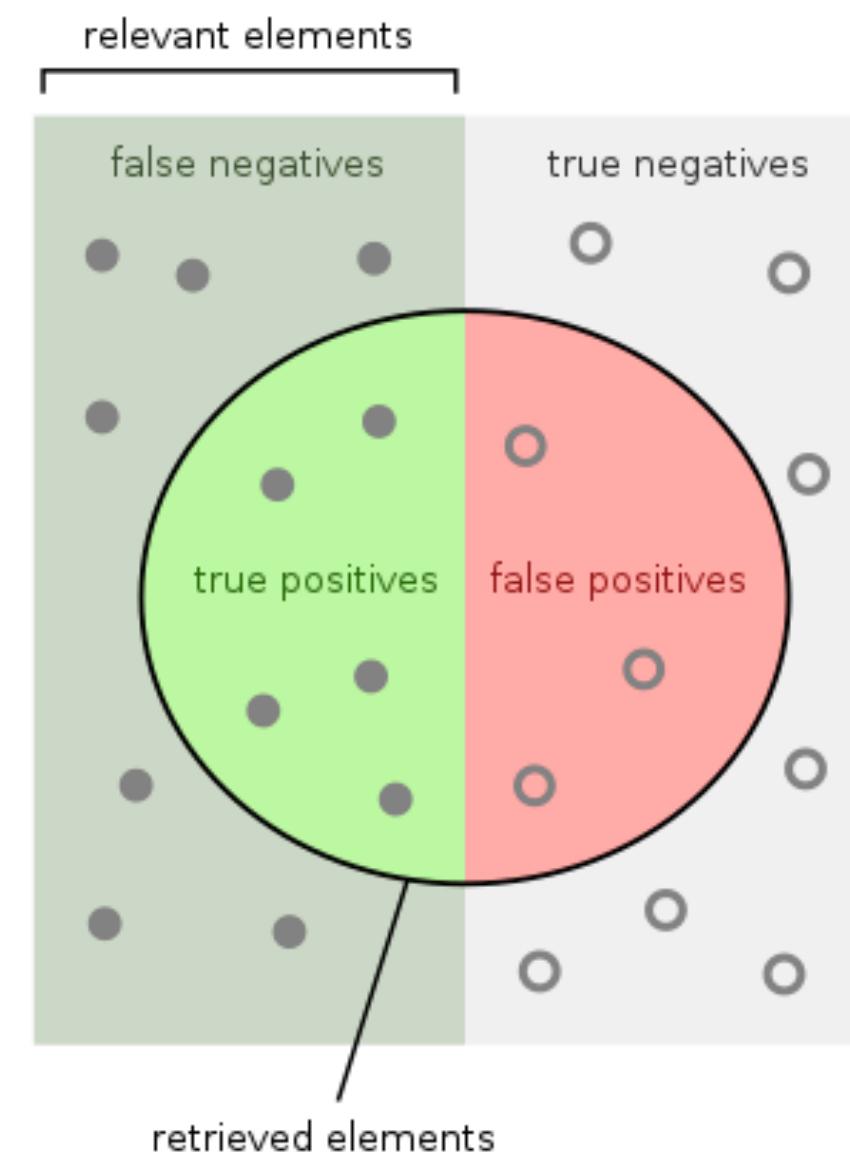
- 👉 Сколько раз правильно классифицировали подделку
 - 👉 Среднее отклонение предсказанной цены дома от истинной
 - 👉 “Плотность” кластеров
-  Выбор метрики определяется задачей и доменом



Метрики

Если не знаете, что выбрать:

- 👉 F1-score для задачи классификации, гармоническое среднее между *precision* — сколько из предсказанных положительных действительно положительные *recall* — сколько положительных было предсказано среди всех положительных
- 👉 R^2 -score для задачи регрессии — насколько лучше “рандомного решения”
 $R^2 \in (-\infty; 1]$ — 0 для среднего предсказания, 1 для идеального



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

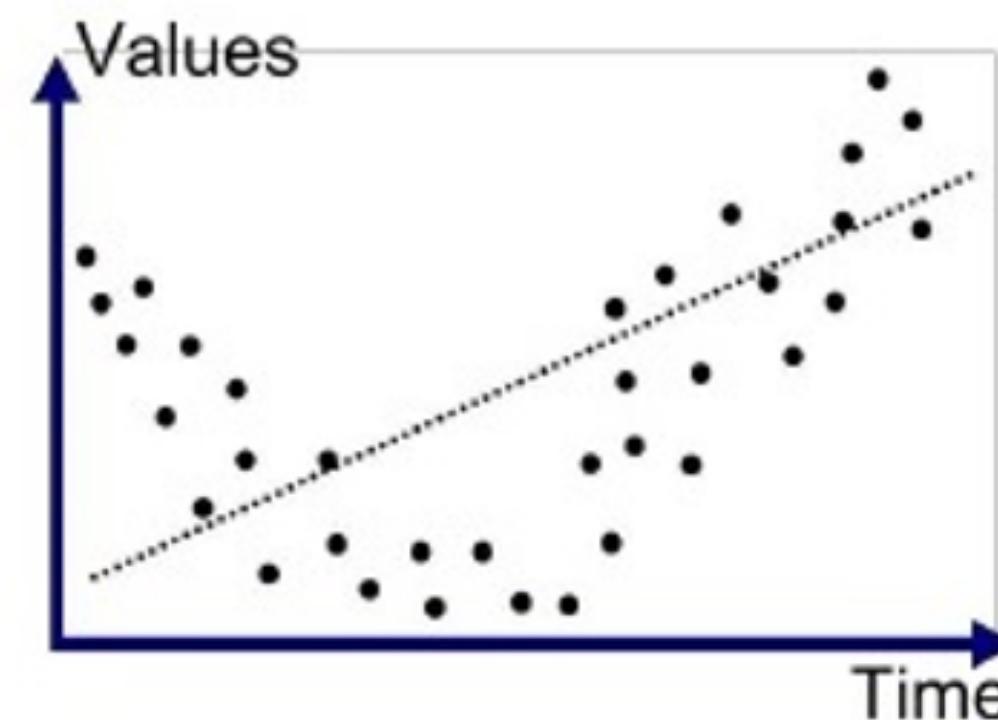
Переобучение

Идеальный мир — хорошие результаты на любых данных

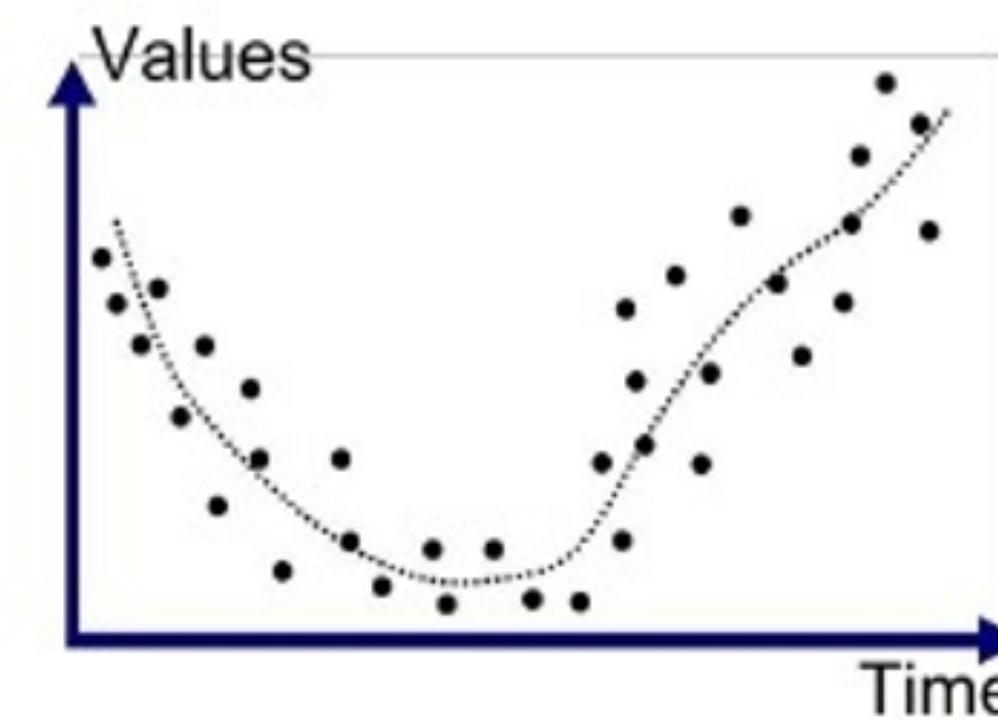
👉 **Переобучение** (overfitting) — модель хорошо научилась работать с обучающей выборке,

но в реальности всё плохо

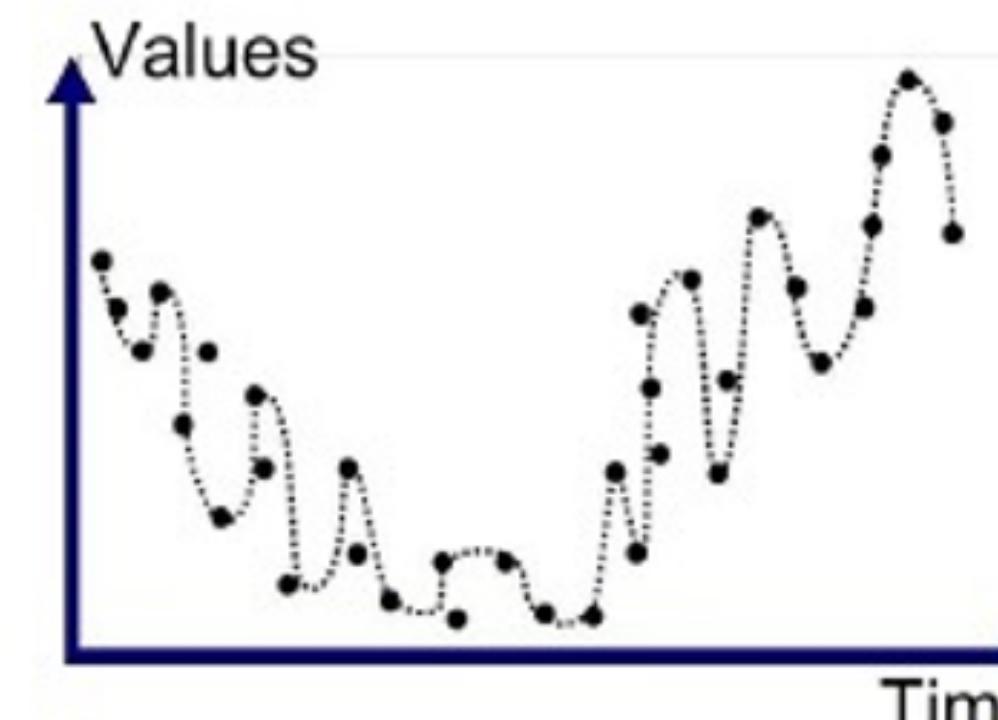
👉 **Недообучение** (underfitting) — модель даже с обучающей выборкой не справляется



Underfitted



Good Fit/Robust



Overfitted

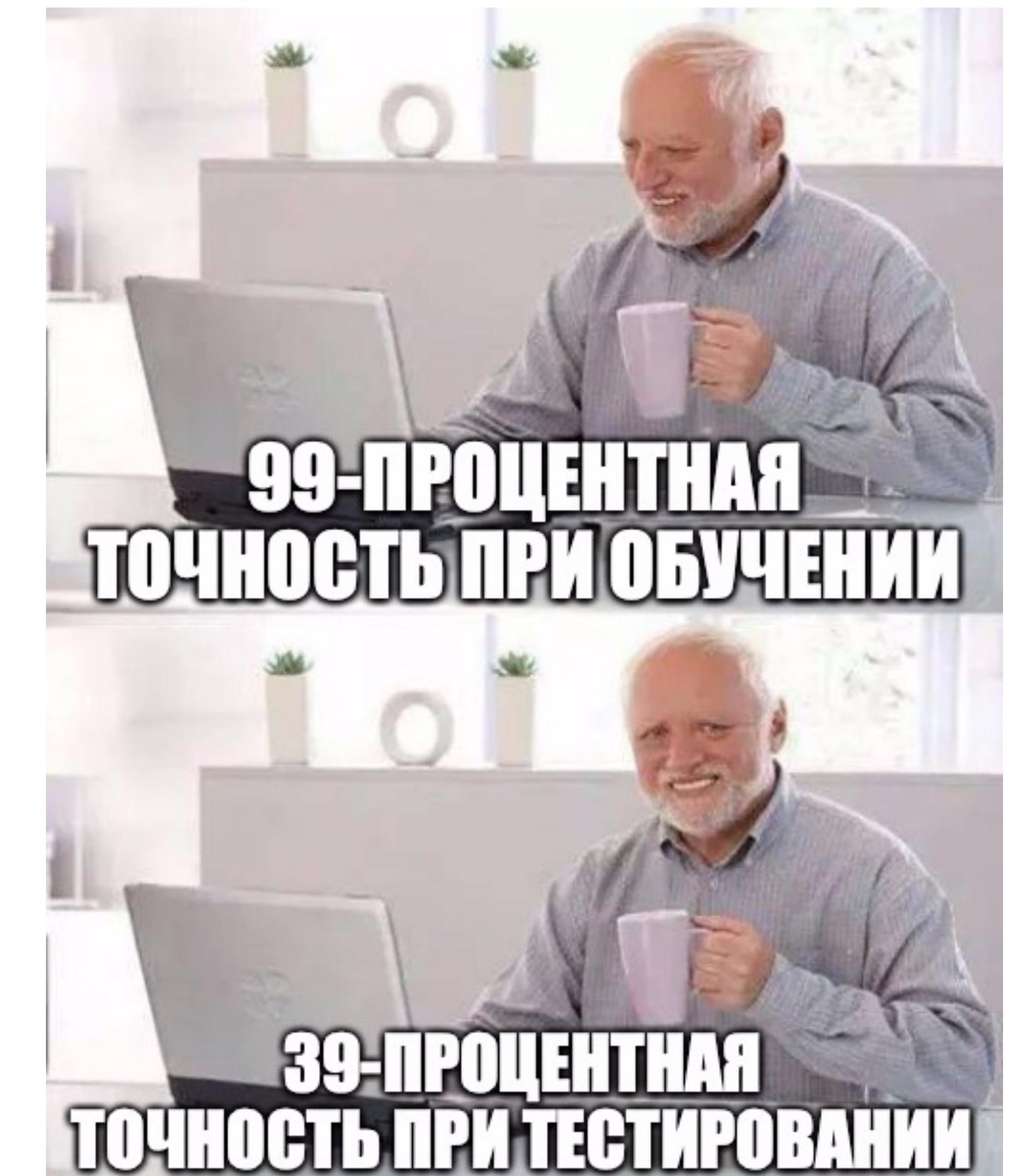
Валидация

Разделим датасет на тренировочную и тестовую части

- 👉 Тренировочные данные используются для обучения модели
- 👉 Тестовые данные модель “не видит”, оцениваем на них качество модели

Выбираем модель и настраиваем гиперпараметры на основе качества на тестовой части

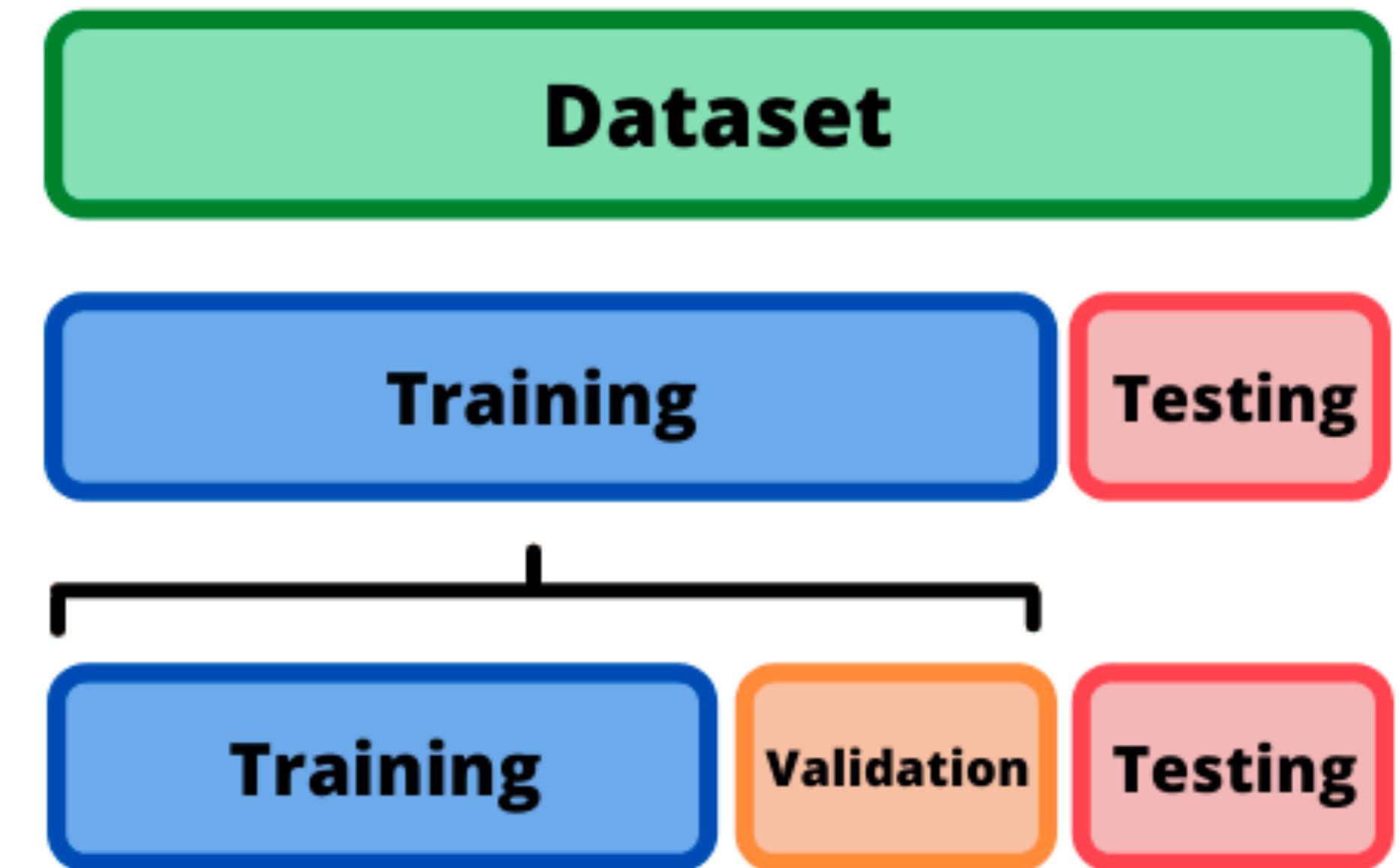
Q: Что произойдет?



Валидация

Необходимо еще выделить валидационную выборку

- 👉 Обучение модели происходит на тренировочной выборке
- 👉 Различные гипер-параметры сравниваются по валидационной выборке
- 👉 Конечное качество определяется тестовой выборкой



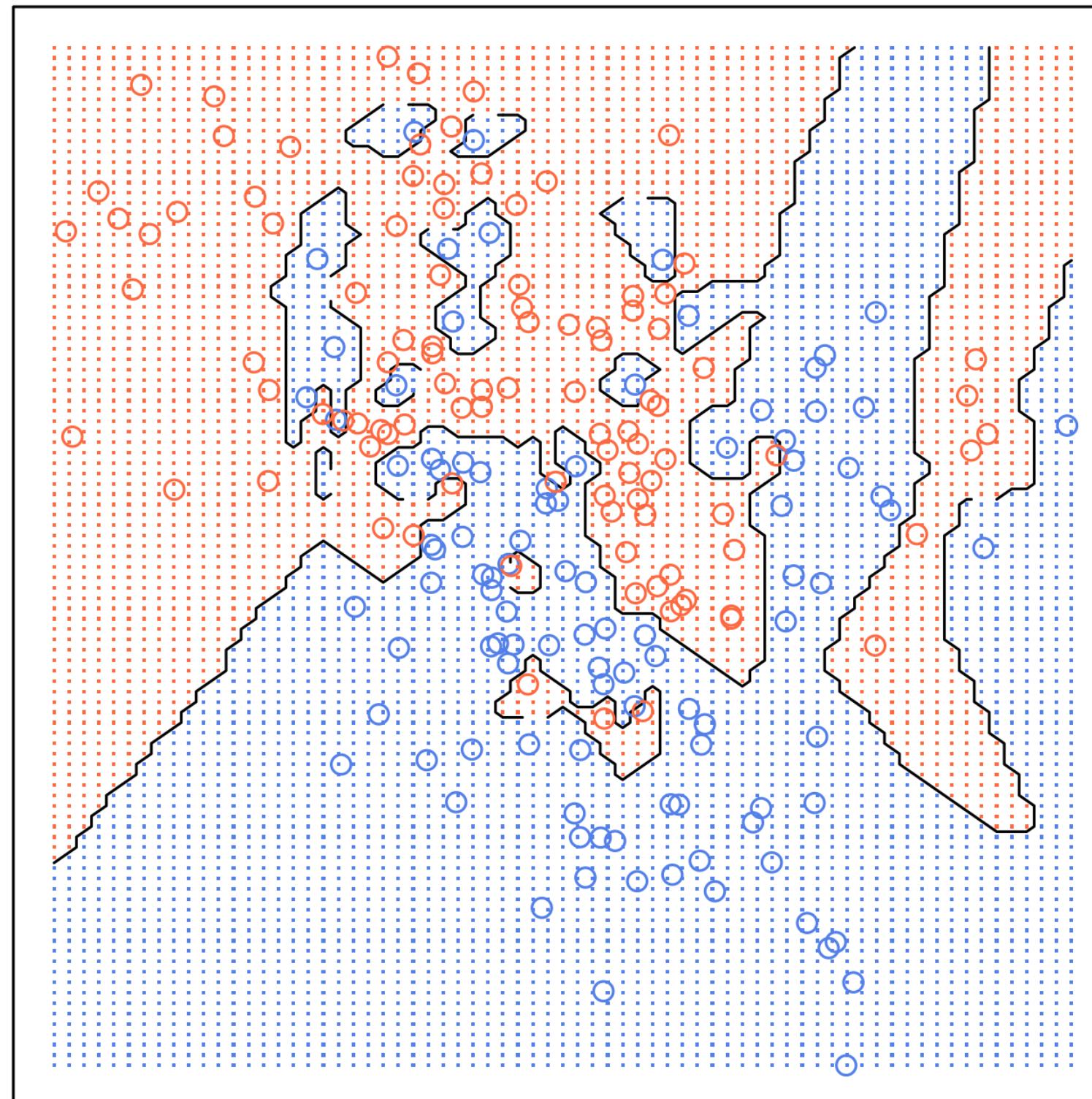
```
>>> from sklearn.model_selection import train_test_split  
>>> split_kwargs = {"test_size": 0.15, "shuffle": True, "random_state": 7}  
>>> _X, X_test, _y, y_test = train_test_split(X, y, **split_kwargs)  
>>> X_train, X_val, y_train, y_val = train_test_split(_X, _y, **split_kwargs)
```

k-NN — ищем k

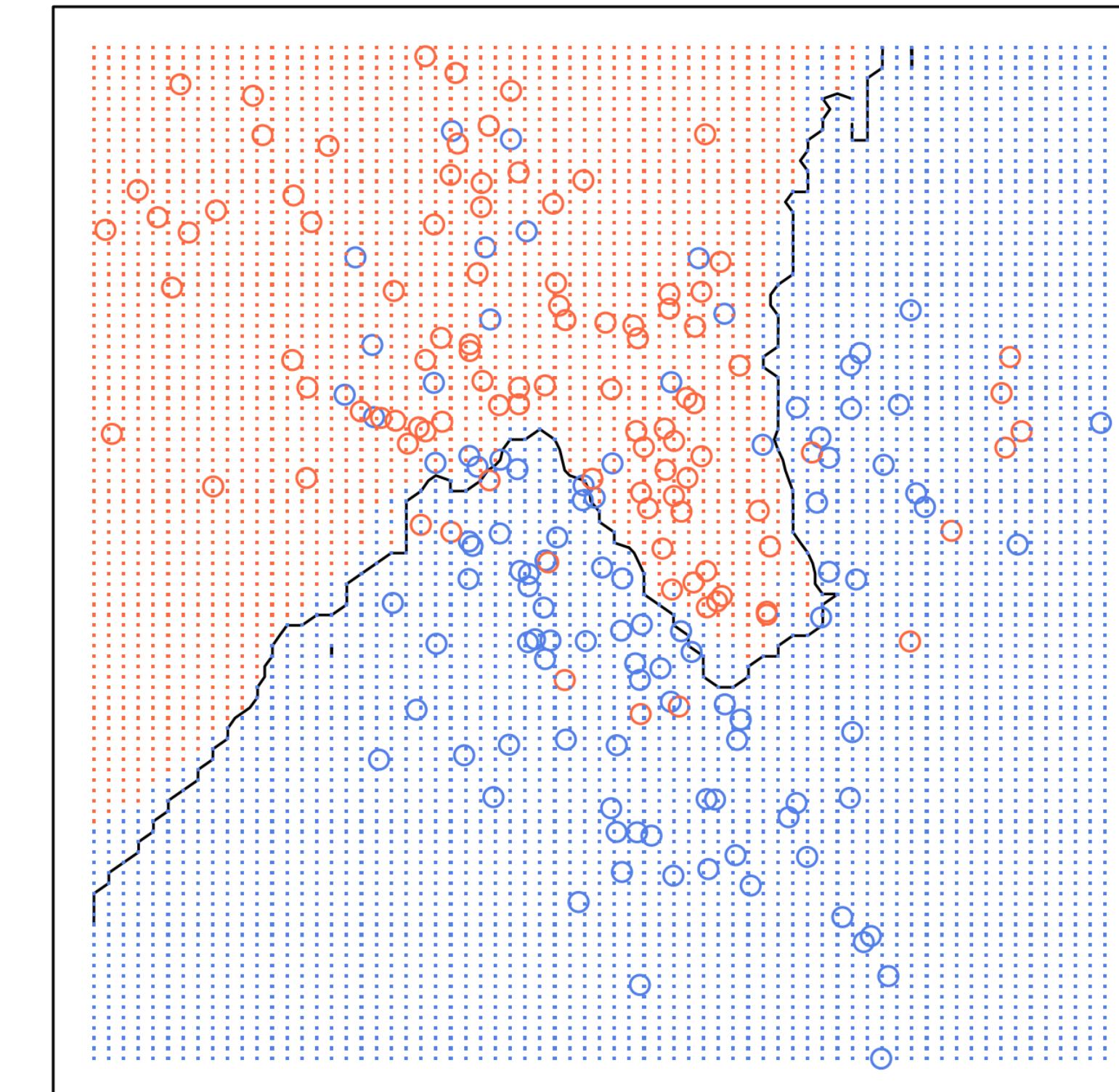
k — это гипер-параметр, подбираем его через train-val выборки

Хорошее начало — $k = \sqrt{l}$

1-nearest neighbours



20-nearest neighbours

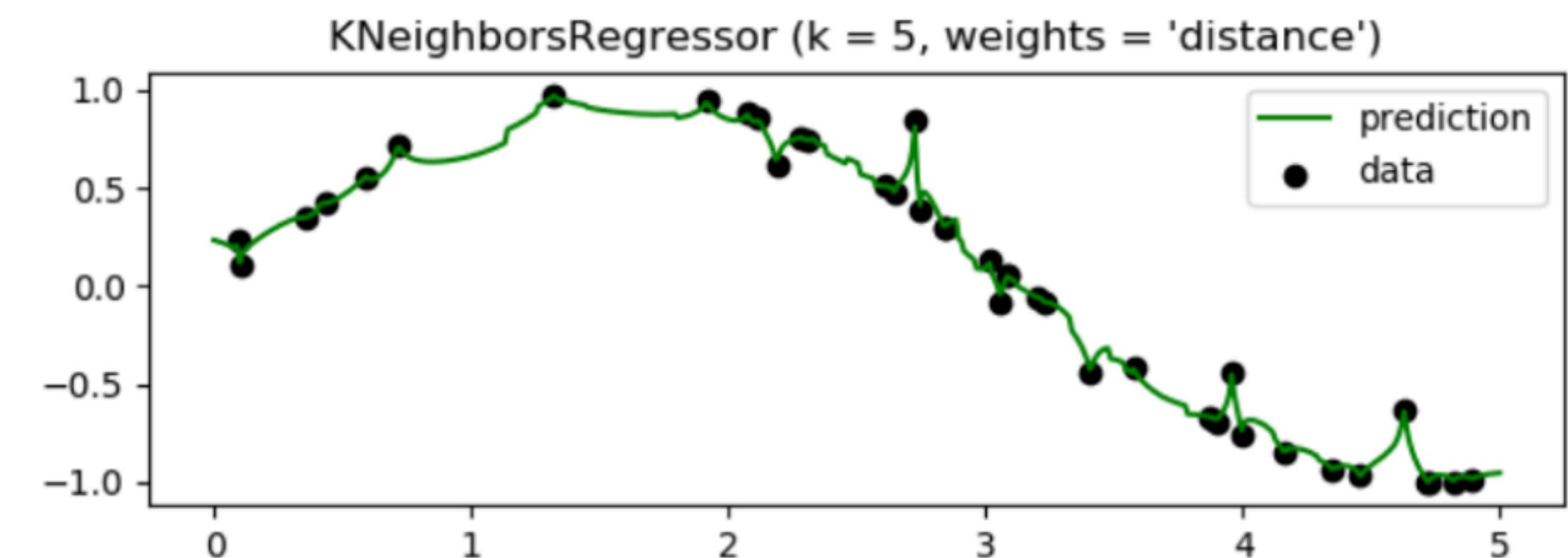
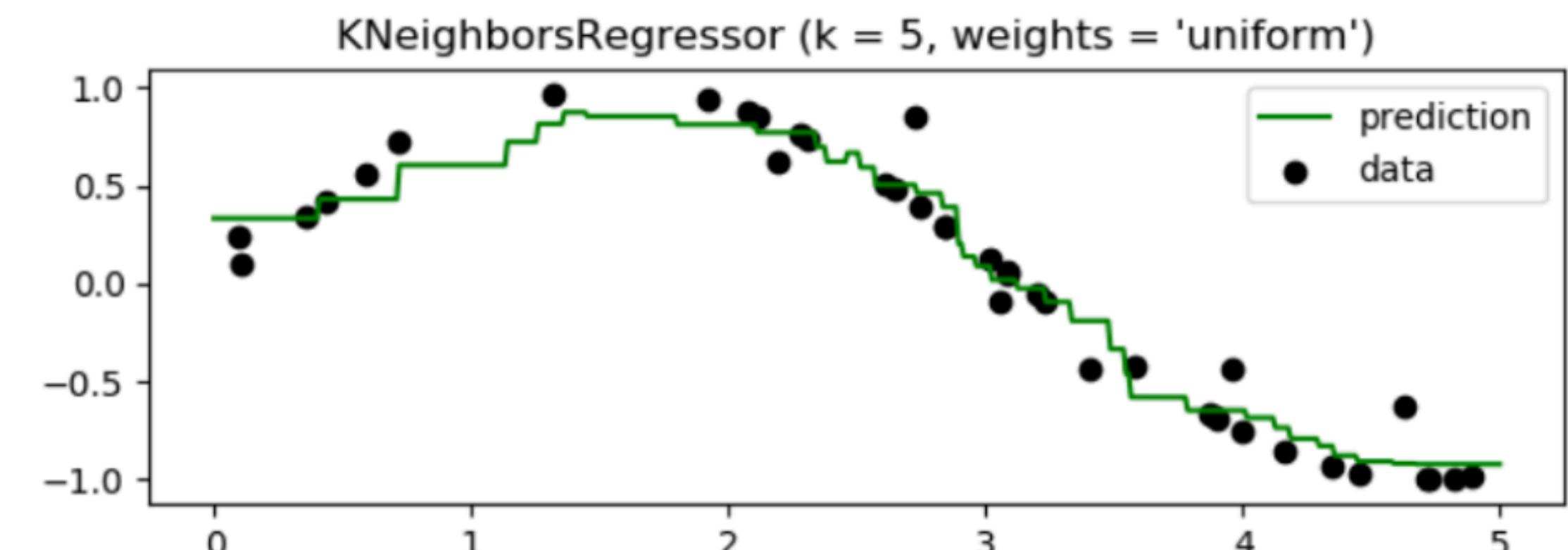


k-NN | Регрессия

Для задачи регрессии, все аналогично, кроме вычисления ответа:

Для ответа усредняем веса, возможно, с добавлением весов

$$a(x) = \frac{\sum_{i=1}^k w_{(i)} y_{(i)}}{\sum_{i=1}^k w_{(i)}}$$



Summary

1. Сформулировали задачу МО и разобрали ее возможные виды — регрессия, классификация, кластеризация, ...
2. Обсудили признаки и как с ними работать — бинарные, численные, категориальные, порядковые, ...
3. Познакомились с первым алгоритмом, KNN — если объекты похожи, то и ответы на них похожи
 - a. Прост в обучении
 - b. Мало гипер-параметров
 - c. Но надо хранить в памяти всю выборку
 - d. Может долго работать — поиск соседей
4. Разобрались как правильно настраивать алгоритм, разбивая данные на части и вычисляя заданную метрику

Me: *uses machine learning*

Machine: *learns*

Me:

