

# Занятие 1

Введение в ML.

# Машинное обучение(Machine Learning)

Что такое обучение?

# Машинное обучение(Machine Learning)

**Обучение** - приобретение необходимой функциональности  
посредством опыта

**Обучение на примерах**

Учимся ходить

Делаем шаг - получилось/нет

Учим названия животных

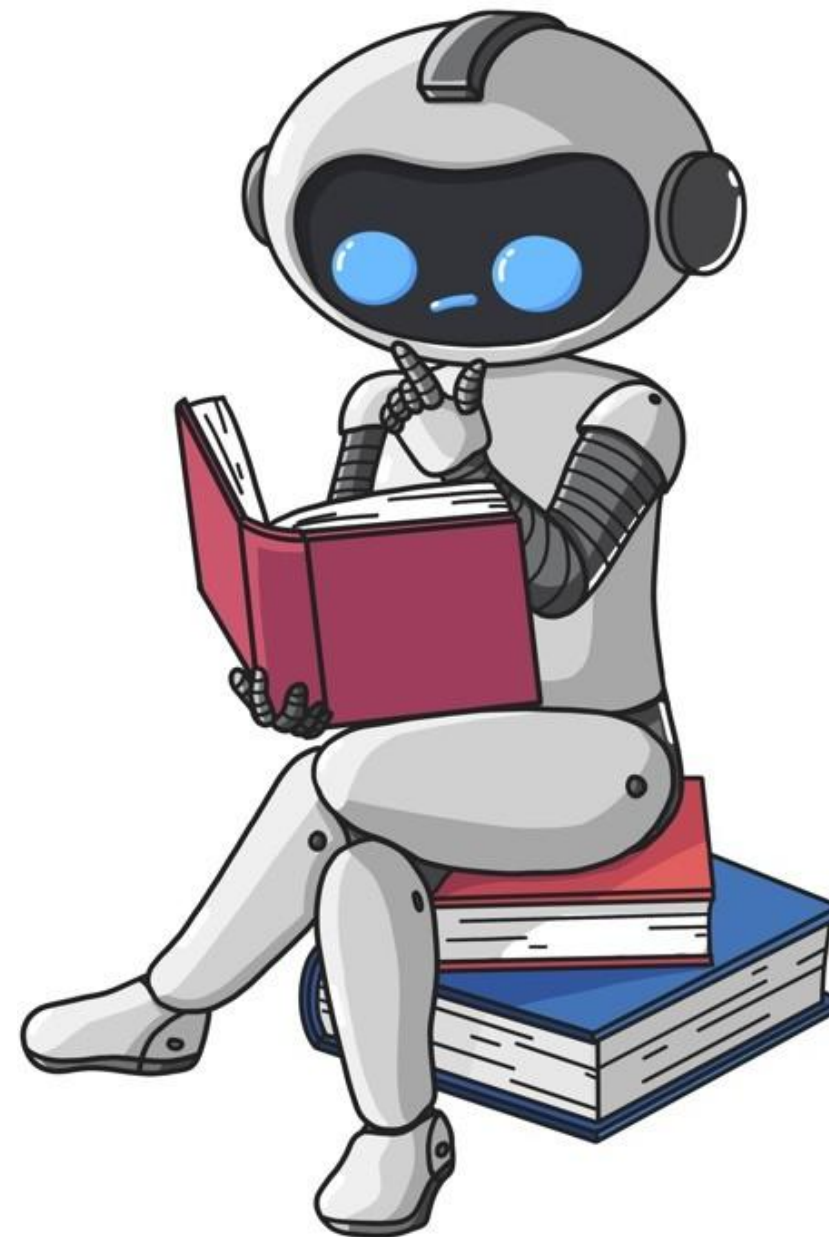
Показывают и называют

**Обучение по определениям**

в школе дают определения

# Машинное обучение

**Машинное обучение** - процесс, в результате которого машина способна показывать поведение, которое в нее не было явно запрограммировано

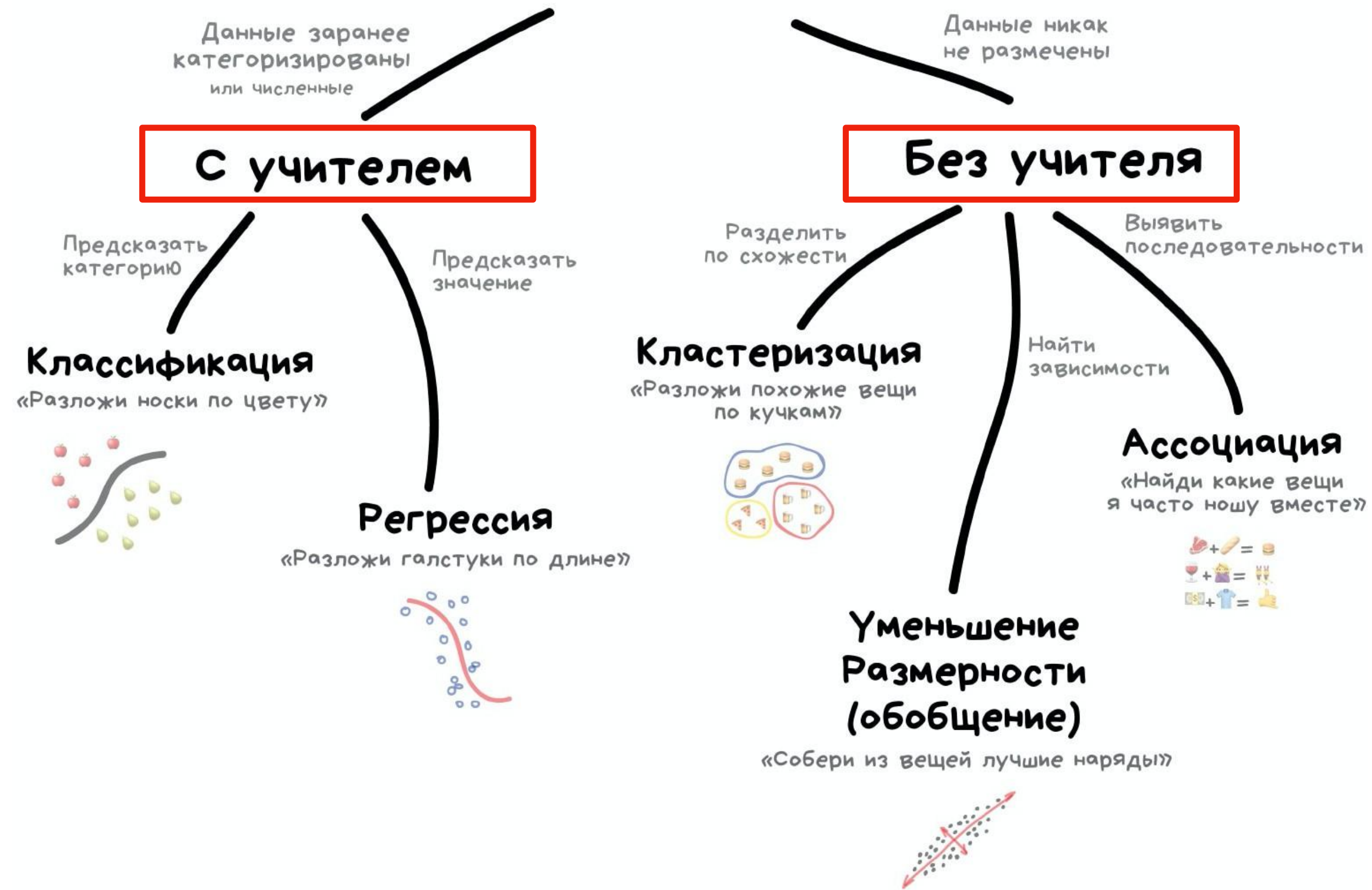


# Постановка задачи классического ML

# Классическое Обучение

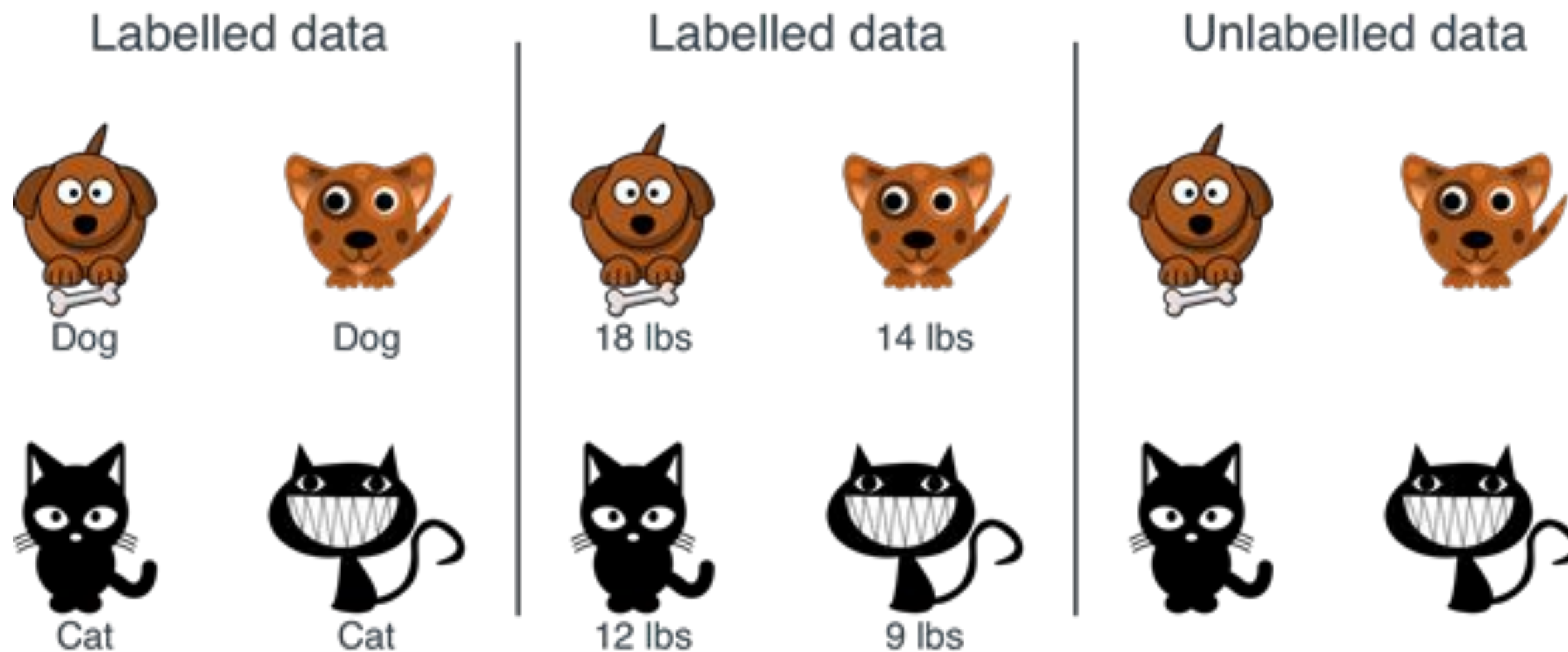


# Классическое Обучение





# Размеченные (labelled) vs неразмеченные (unlabelled) данные





# Размеченные (labelled) vs неразмеченные (unlabelled) данные

## Размеченные

ID	Clump	UnifSize	UnifShape	MargAdh	SingEpiSize	BareNuc	BlandChrom	NormNucl	Mit	Class
1000025	5	1	1	1	2	1	3	1	1	benign
1002945	5	4	4	5	7	10	3	2	1	benign
1015425	3	1	1	1	2	2	3	1	1	malignant
1016277	6	8	8	1	3	4	3	7	1	benign
1017023	4	1	1	3	2	1	3	1	1	benign
1017122	8	10	10	8	7	10		7	1	malignant
1018099	1	1	1	1	2	10	3	1	1	benign
1018561	2	1	2	H	2	1	3	1	1	benign
1033078	2	1	1	1	2	1	1	1	5	benign
1033078	4	2	1	1	2	1	2	1	1	benign

labels

## Неразмеченные

Customer Id	Age	Edu	Years Employed	Income	Card Debt	Other Debt	Address	DebtIncomeRatio
1	41	2	6	19	0.124	1.073	NBA001	6.3
2	47	1	26	100	4.582	8.218	NBA021	12.8
3	33	2	10	57	6.111	5.802	NBA013	20.9
4	29	2	4	19	0.681	0.516	NBA009	6.3
5	47	1	31	253	9.308	8.908	NBA008	7.2
6	40	1	23	81	0.998	7.831	NBA016	10.9
7	38	2	4	56	0.442	0.454	NBA013	1.6
8	42	3	0	64	0.279	3.945	NBA009	6.6
9	26	1	5	18	0.575	2.215	NBA006	15.5
10	47	3	23	115	0.653	3.947	NBA011	4
11	44	3	8	88	0.285	5.083	NBA010	6.1
12	34	2	9	40	0.374	0.266	NBA003	1.6

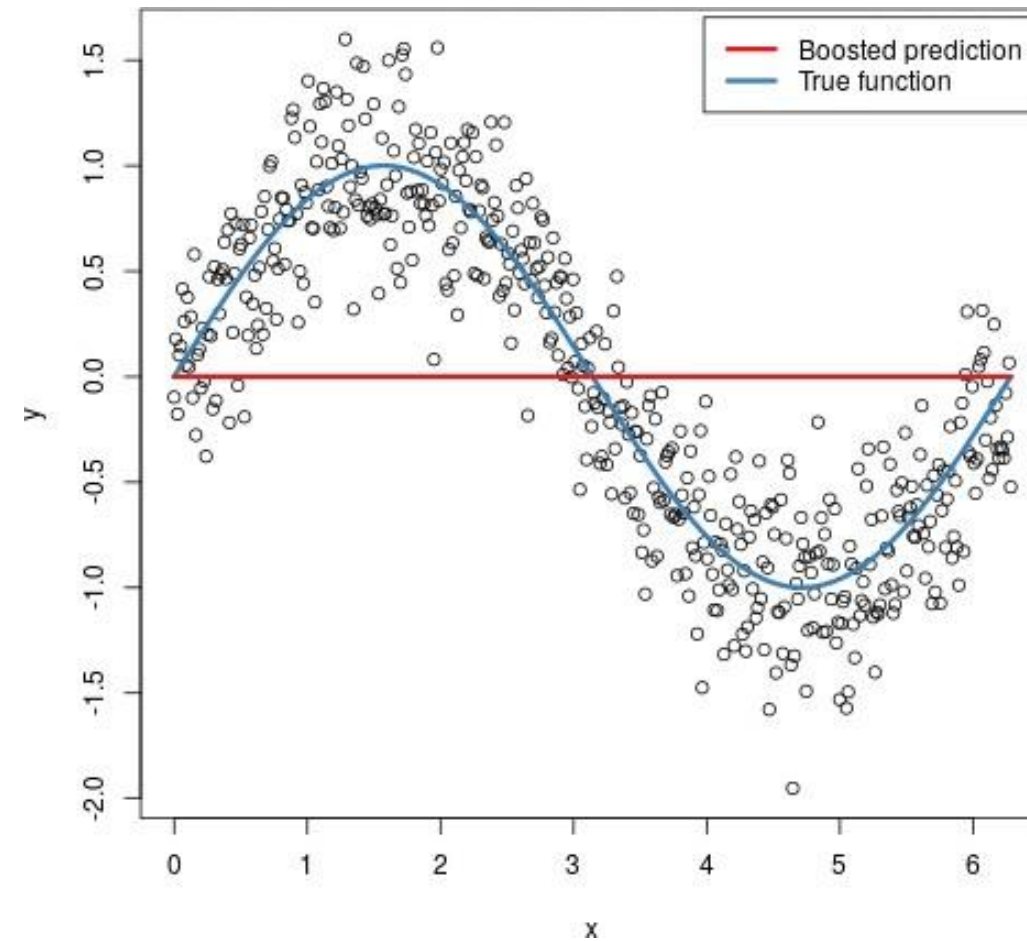
unlabeled

Сегодня мы обсудим механику процесса обучения на примере трёх главных задач классического ML — регрессии, классификации, кластеризации.

А также подробно разберем несколько алгоритмов кластеризации.

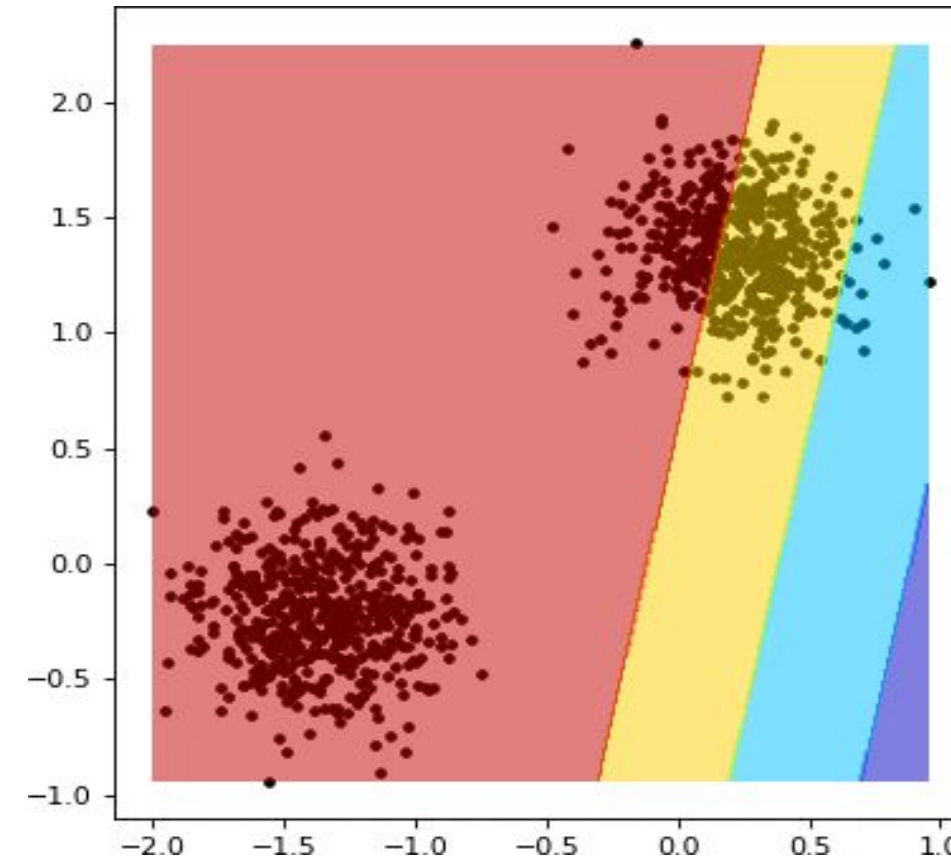
# С учителем

## Регрессия



$Y \subseteq \mathbb{R}^n$ . Нужно восстановить  
обычную функциональную  
зависимость  $f: X \rightarrow Y$

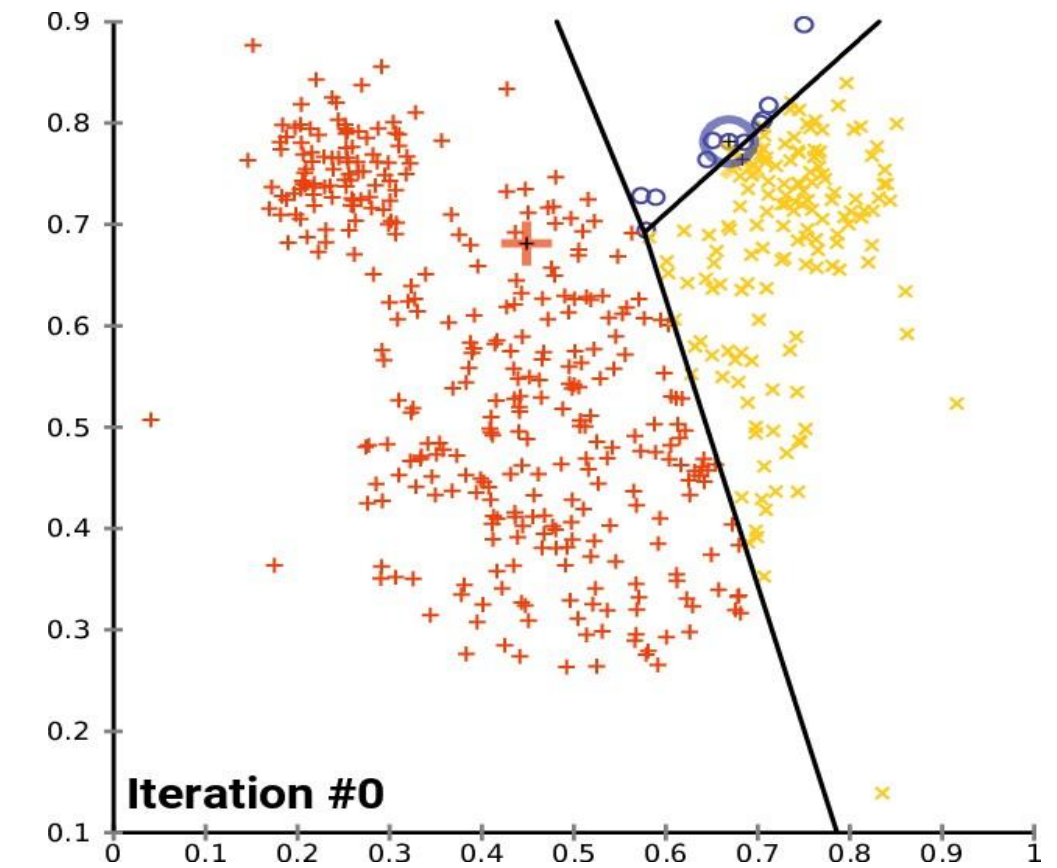
## Классификация



$Y \subseteq [0,1]^n$ . Нужно предсказать  
распределение вероятностей на  
возможных вариантах

# Без учителя

## Кластеризация

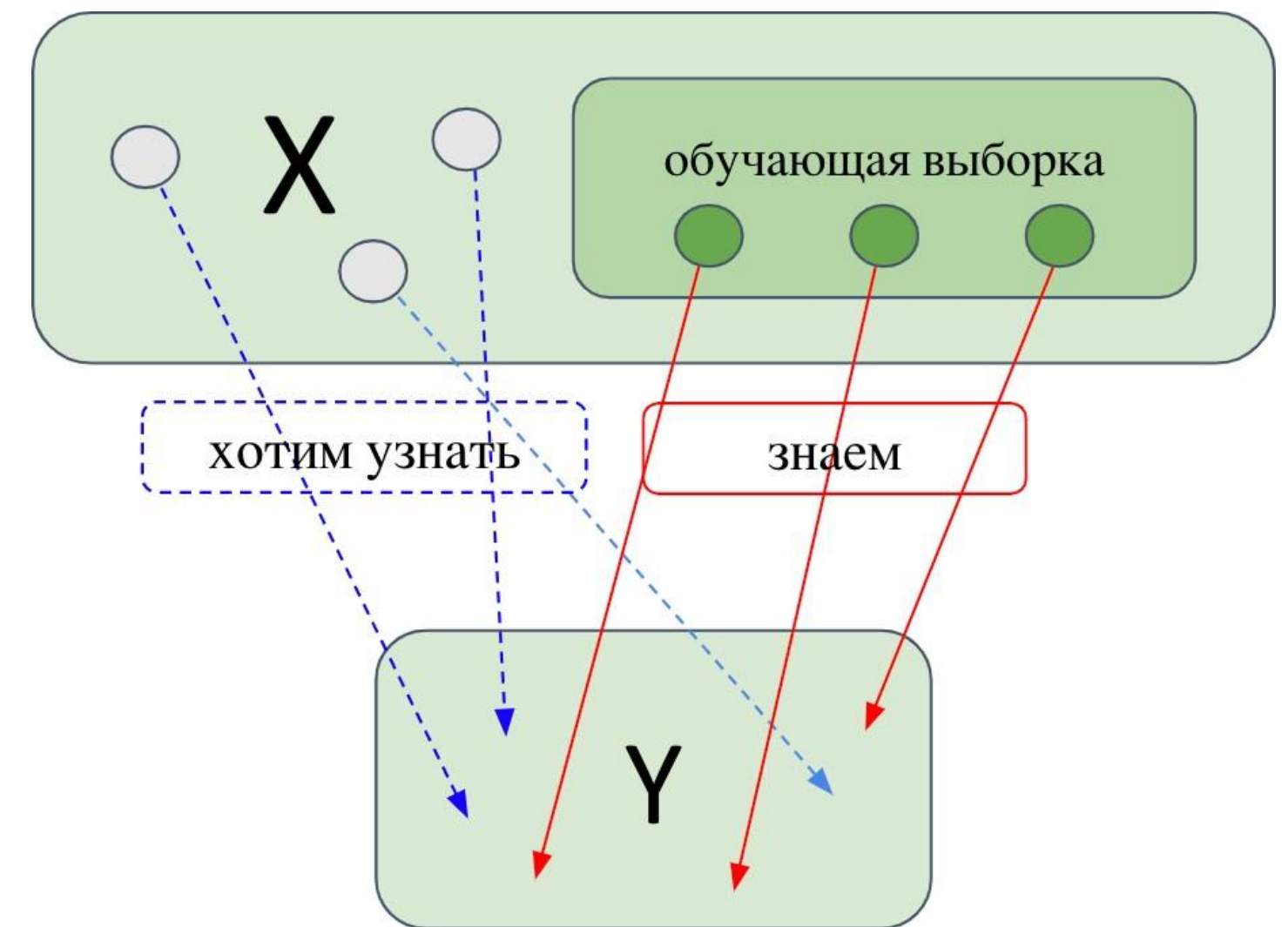


Нужно предъявить такие классы  
эквивалентности, чтобы объекты  
одного класса были более похожи друг  
на друга, чем объекты разных классов



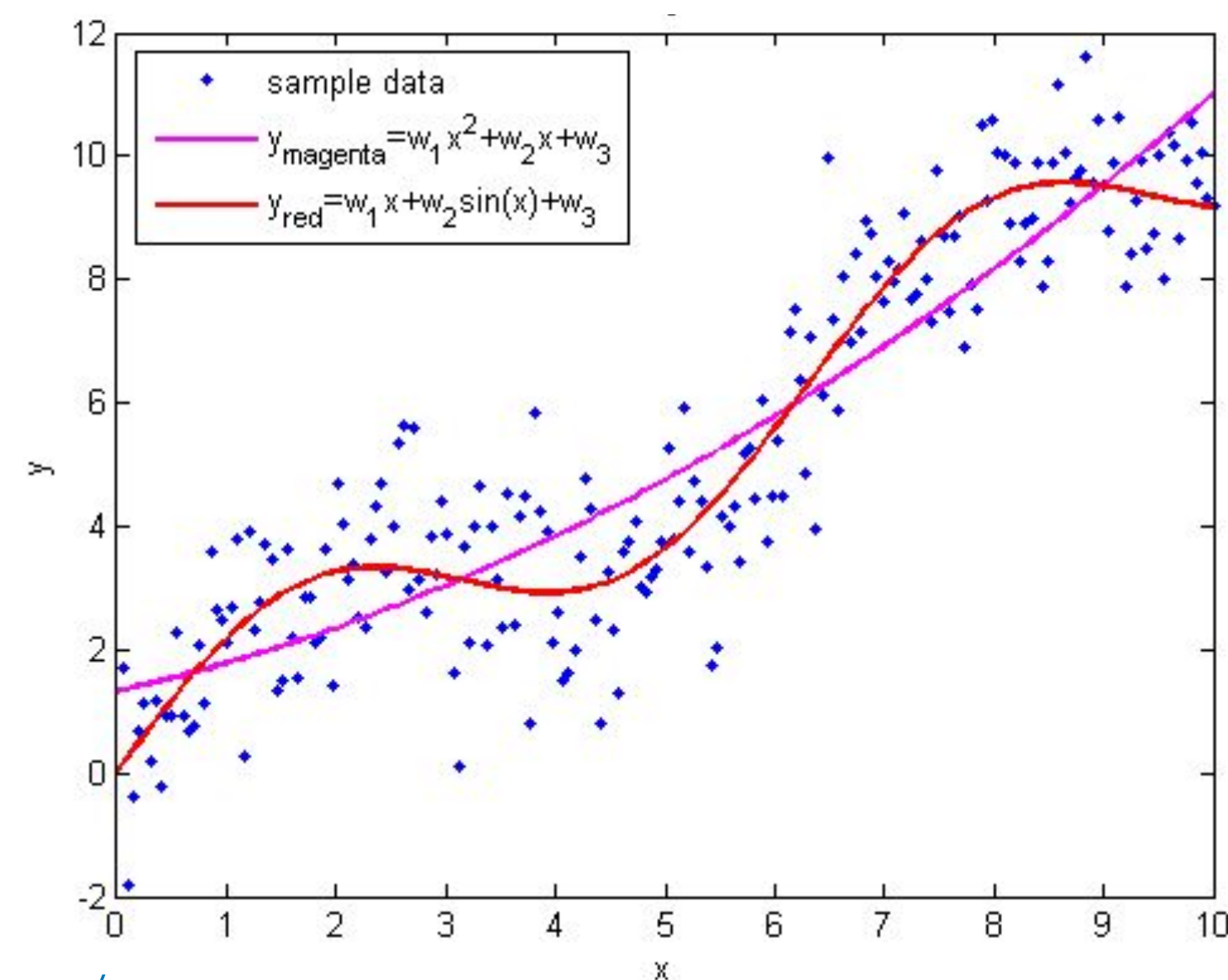
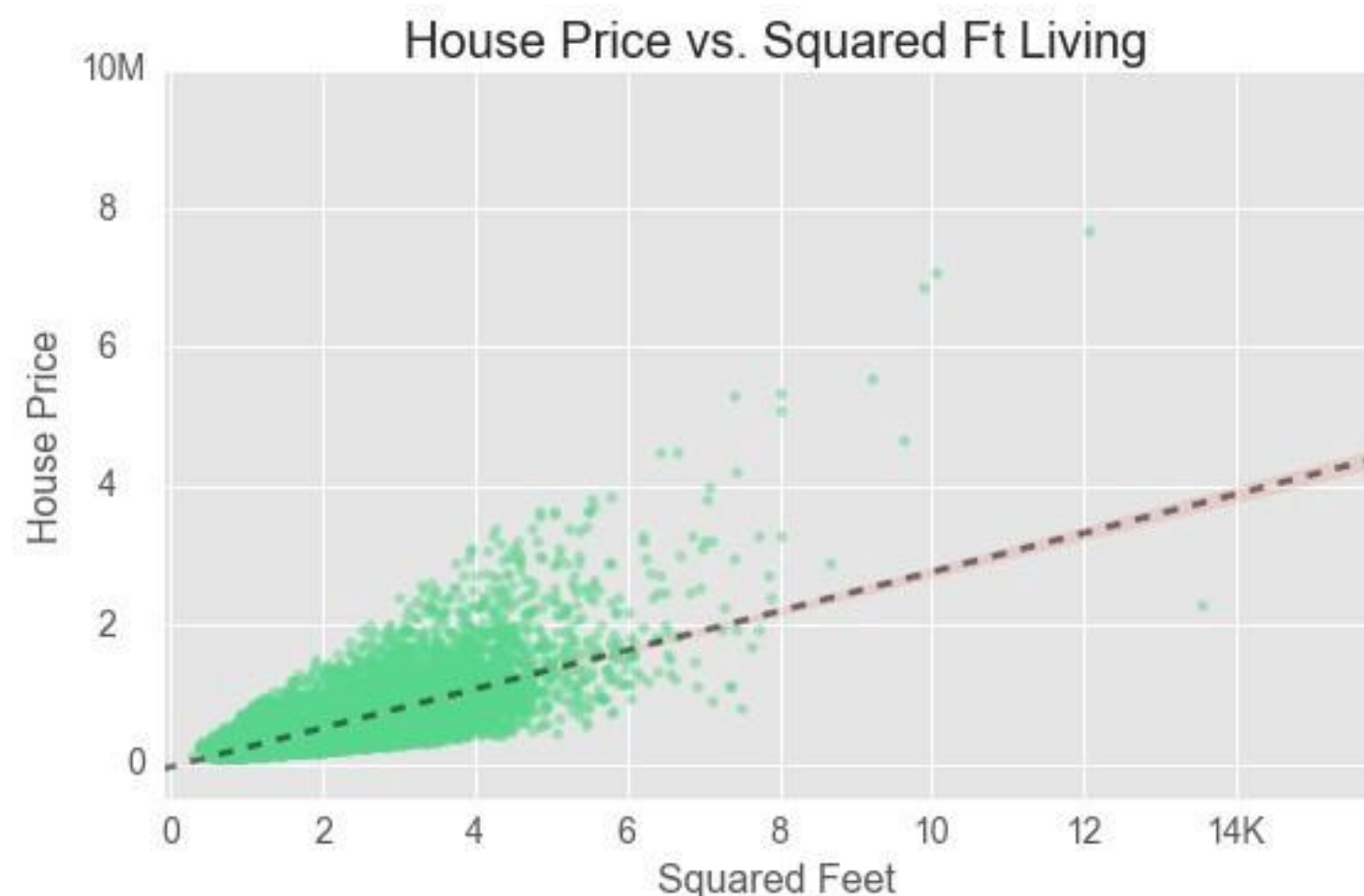
# Постановка задачи (semi-)supervised learning (обучение с учителем)

- $X$  — множество объектов в пространстве признаков
- $Y$  — область значений целевой функции
- $f: X \rightarrow Y$  — неизвестная закономерность  
(даже может иметь стохастическую природу!)
- **Дано:** Обучающая выборка вида  $\{(X_i, y_i)\}_{i=1}^n$
- **Цель:** Максимально точно приблизить  $f$  на всём!



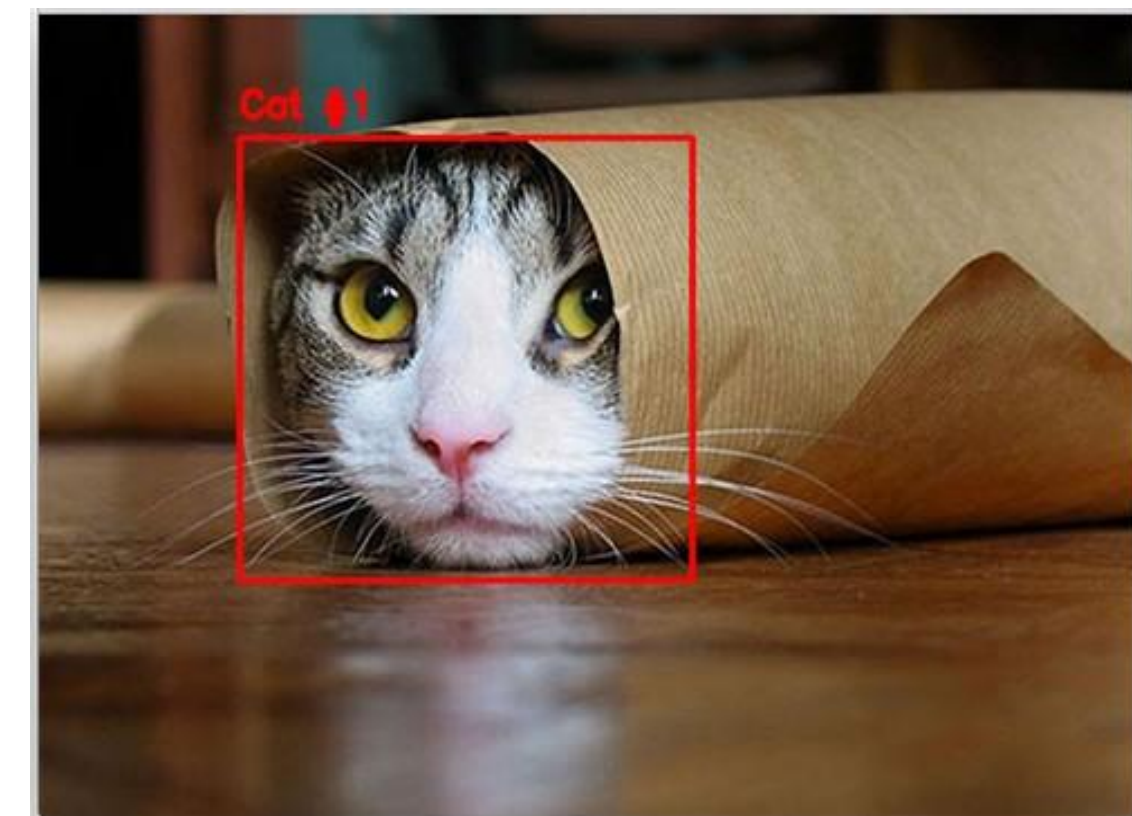
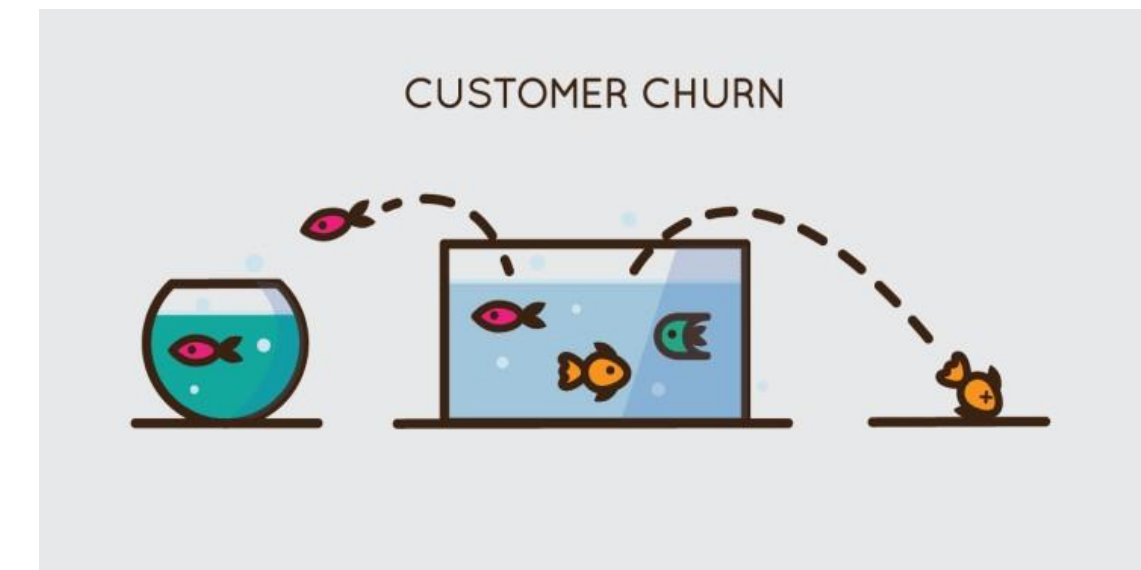
# Примеры задач регрессии

- **Предсказание стоимости жилья** для риэлторской компании
- **Предсказание времени доставки**
- **Предсказание спроса на такси** в конкретном районе в конкретный час завтрашнего дня.



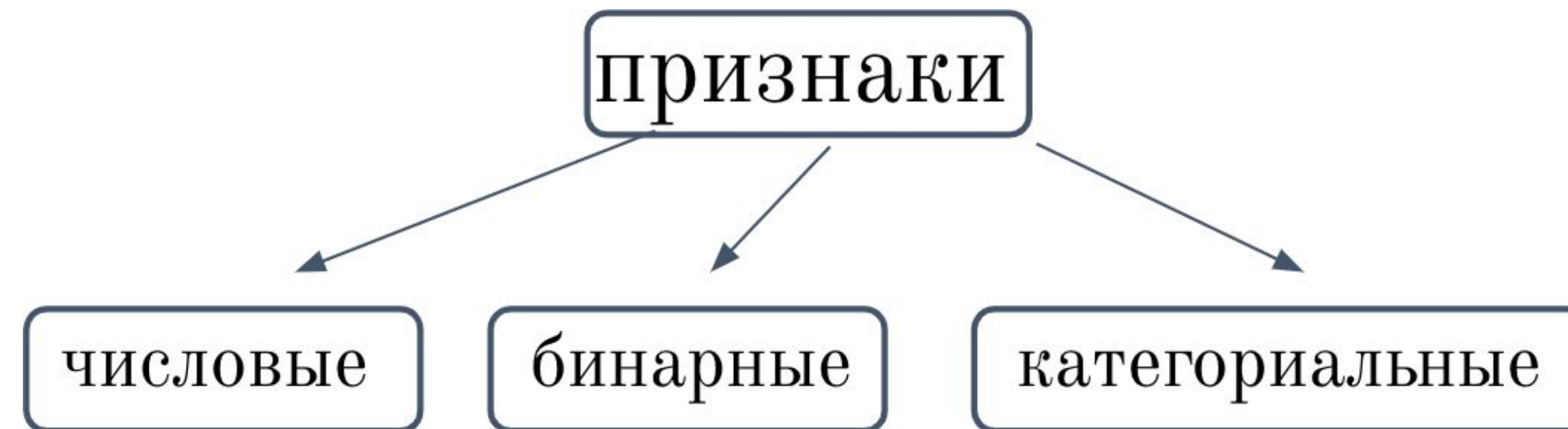
# Примеры задач классификации

- **Предсказание оттока клиентов / сотрудников** на основе их поведения.
- **Ранжирование товаров по вероятности покупки их пользователем** (подход к реализации рекомендательной системы)
- **Классификация клеток ткани** на здоровые и опухолевые
- **Детекция объектов на фото.**





# Устройство пространства признаков



{Москва, Казань, СПб, Нижний Новгород}

- One-hot encoding:  
 $x_i = \text{Казани} \rightarrow 0100$   
 $x_j = \text{Москва} \rightarrow 1000$
- Label encoding:  
 $x_i = \text{Казани} \rightarrow 2$   
 $x_j = \text{Москва} \rightarrow 1$



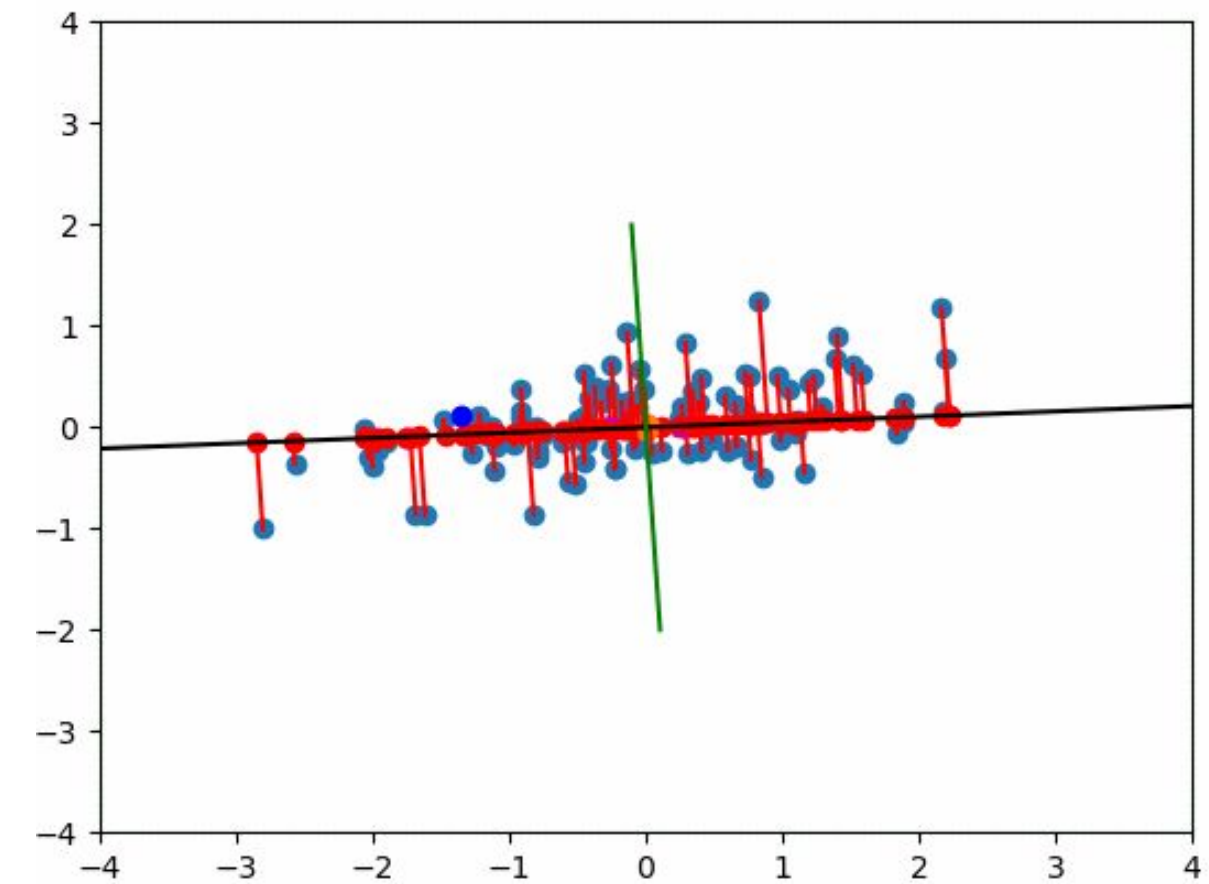
Часто приходится переводить в  
числовые при помощи

- One-hot encoding;
- Label encoding;
- Mean encoding;
- etc.



## Пока ошибка уменьшается

- Сделать предсказания
- Посчитать функцию ошибок
- Обновить параметры модели так, чтобы минимизировать функцию ошибок

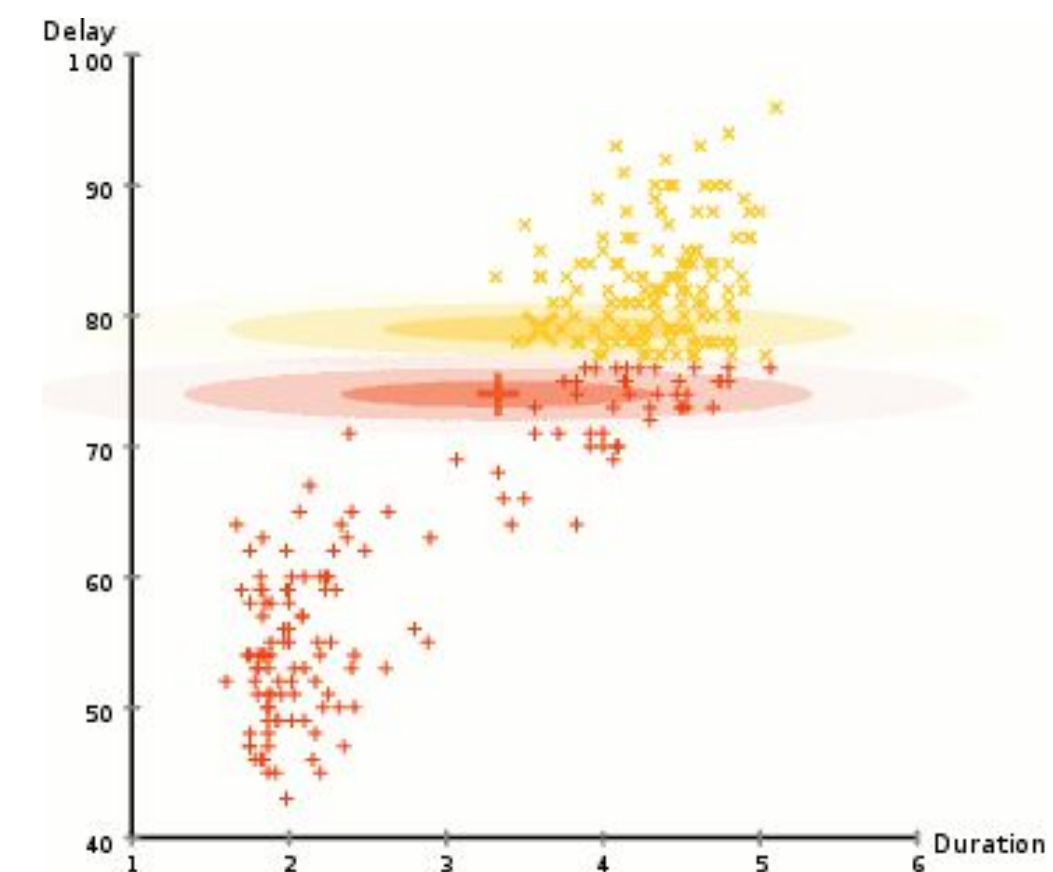
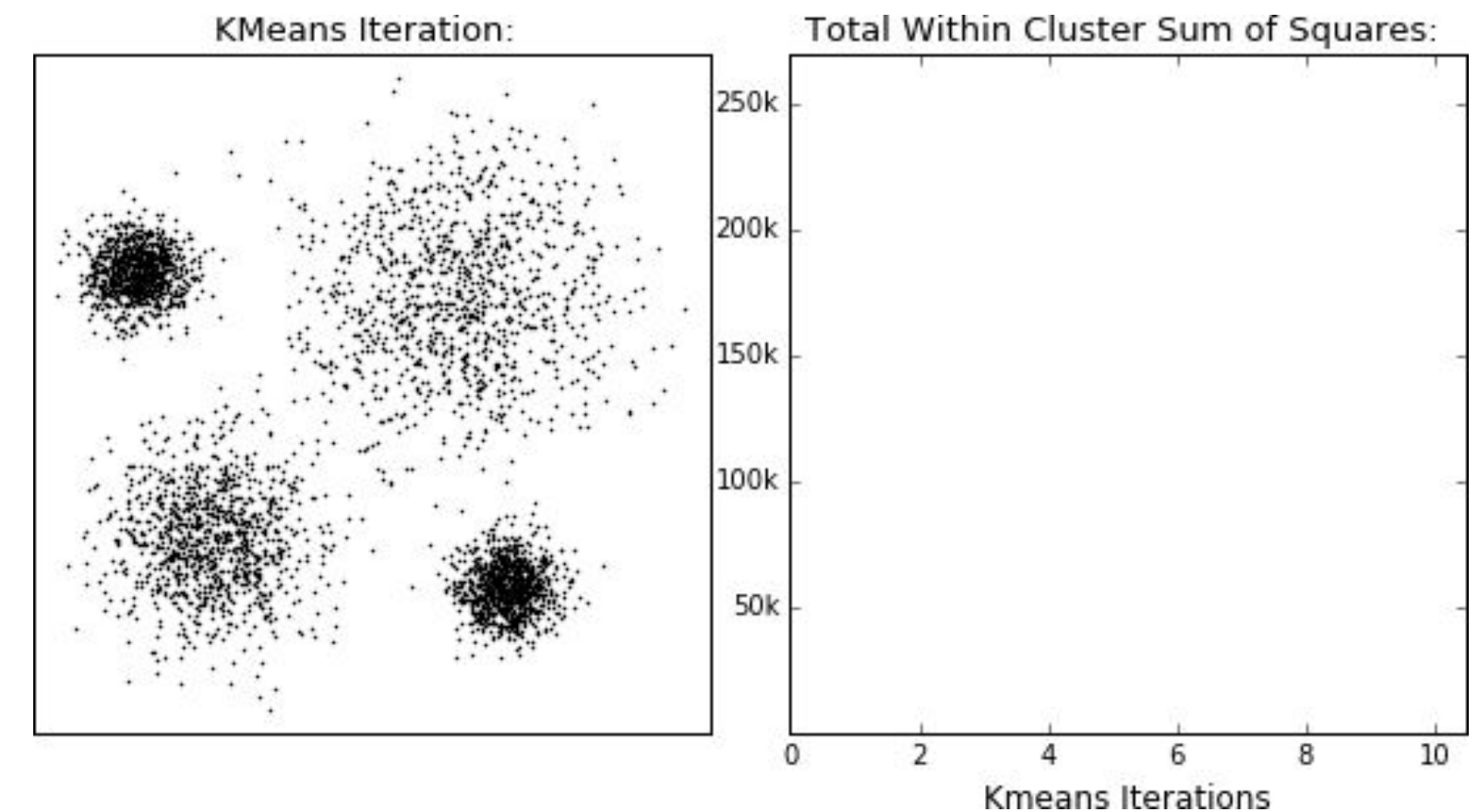


Среднеквадратичная ошибка — среднее длин красных отрезков. Алгоритм ищет оптимальное положение прямой.

# Unsupervised learning (обучение без учителя)

В классических задачах **unsupervised learning** есть  $X$ , но нет обучающей выборки (т.е. мы не знаем правильные ответы).

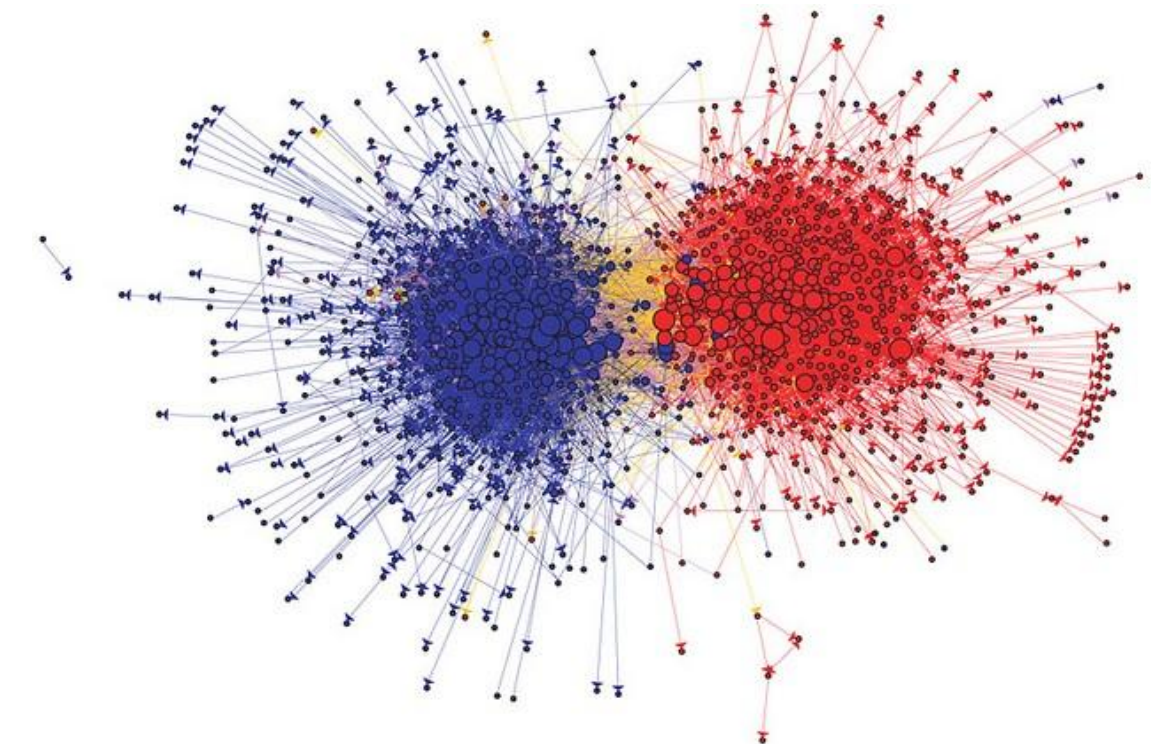
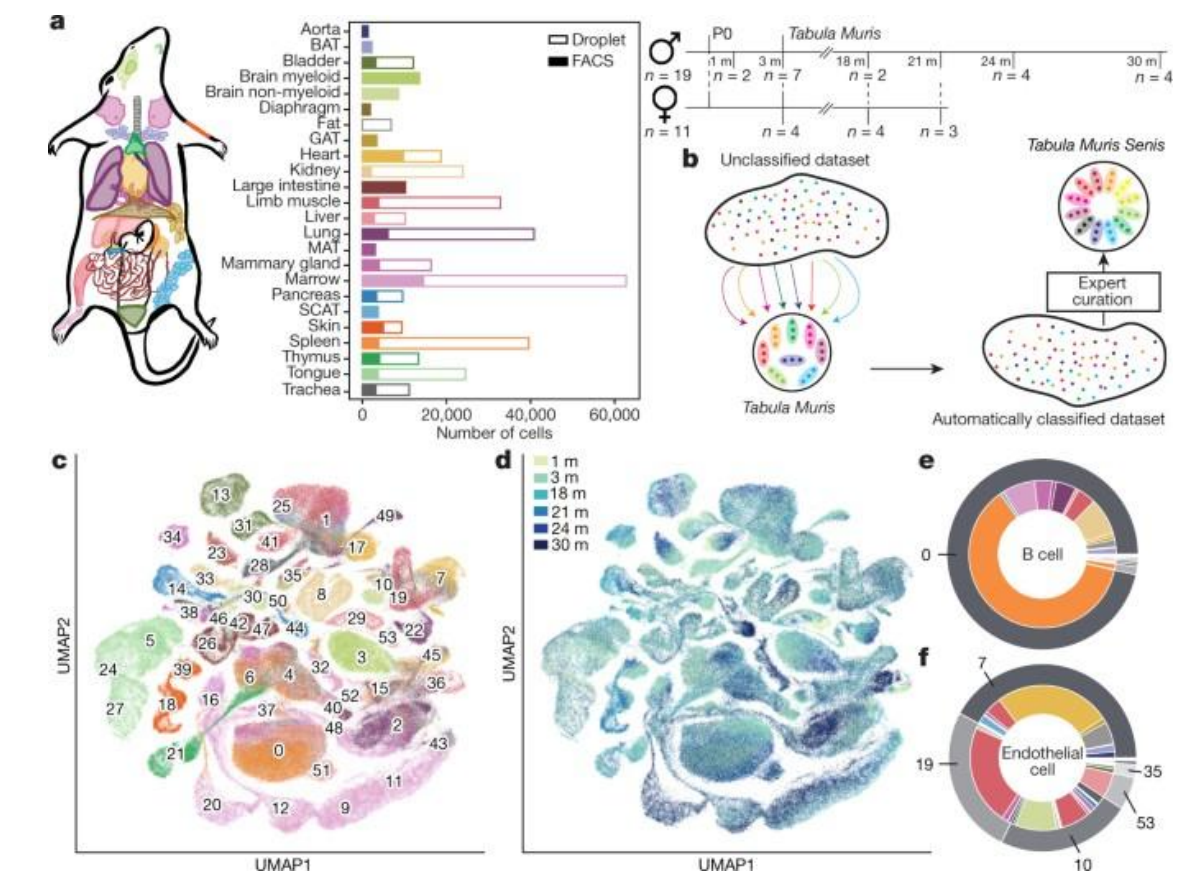
В таких задачах обычно **минимизируют "энтропию" системы**: ищут наиболее удачную расстановку меток





# Примеры задач кластеризации

- **Сегментация аудитории** для таргетирования рекламы
- **Идентификация типов клеток** в образце данных секвенирования
- **Поиск сообществ в социальном графе** (из соцсети или из инсайдерской информации о структуре организации)
- **Задача разделения смеси распределений**

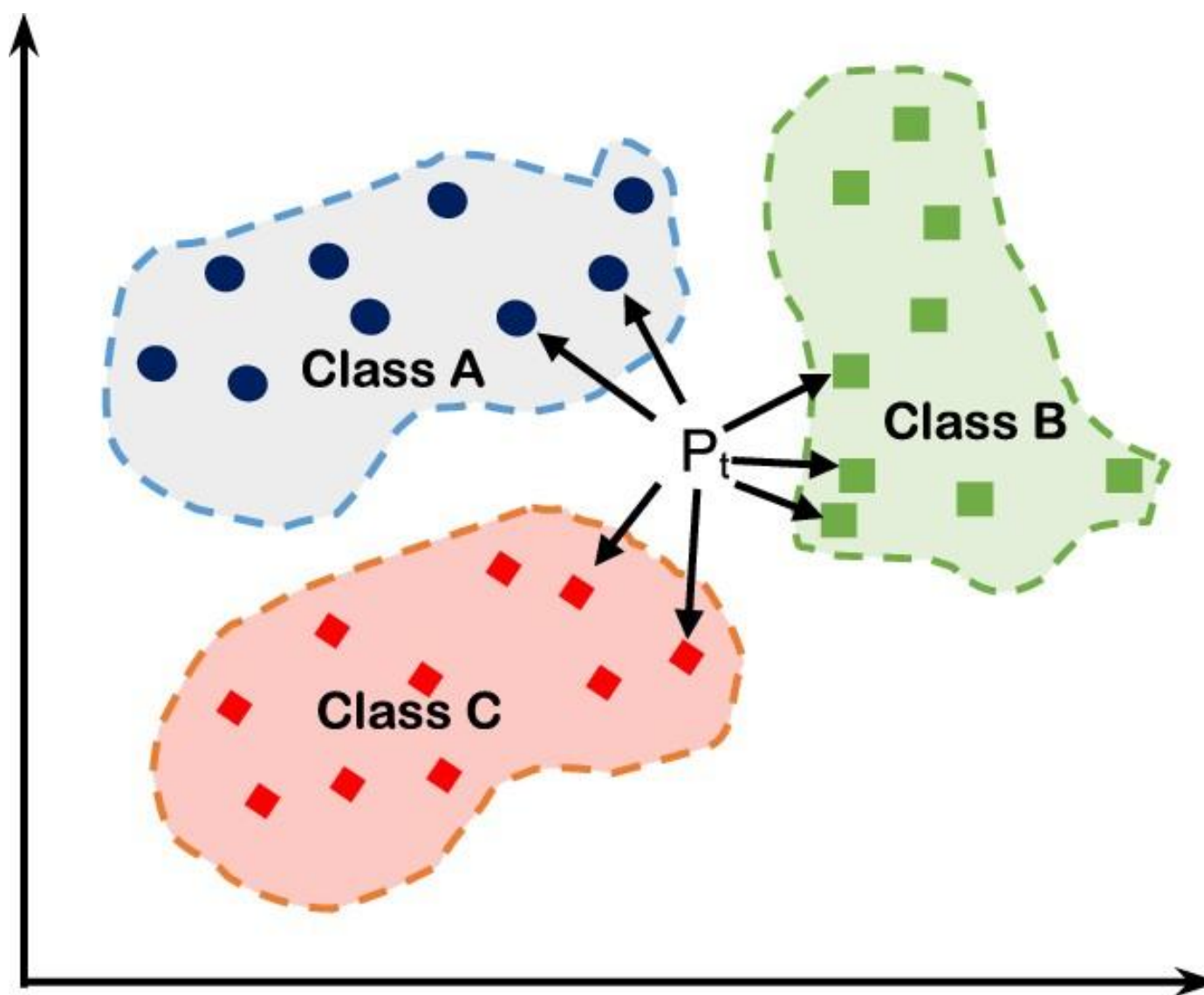


# Простейшие метрические алгоритмы

kNN, k-Means, DBSCAN

# Метрические алгоритмы

**Метрический** классификатор (англ. similarity-based classifier) — **алгоритм** классификации, основанный на вычислении оценок сходства между объектами.



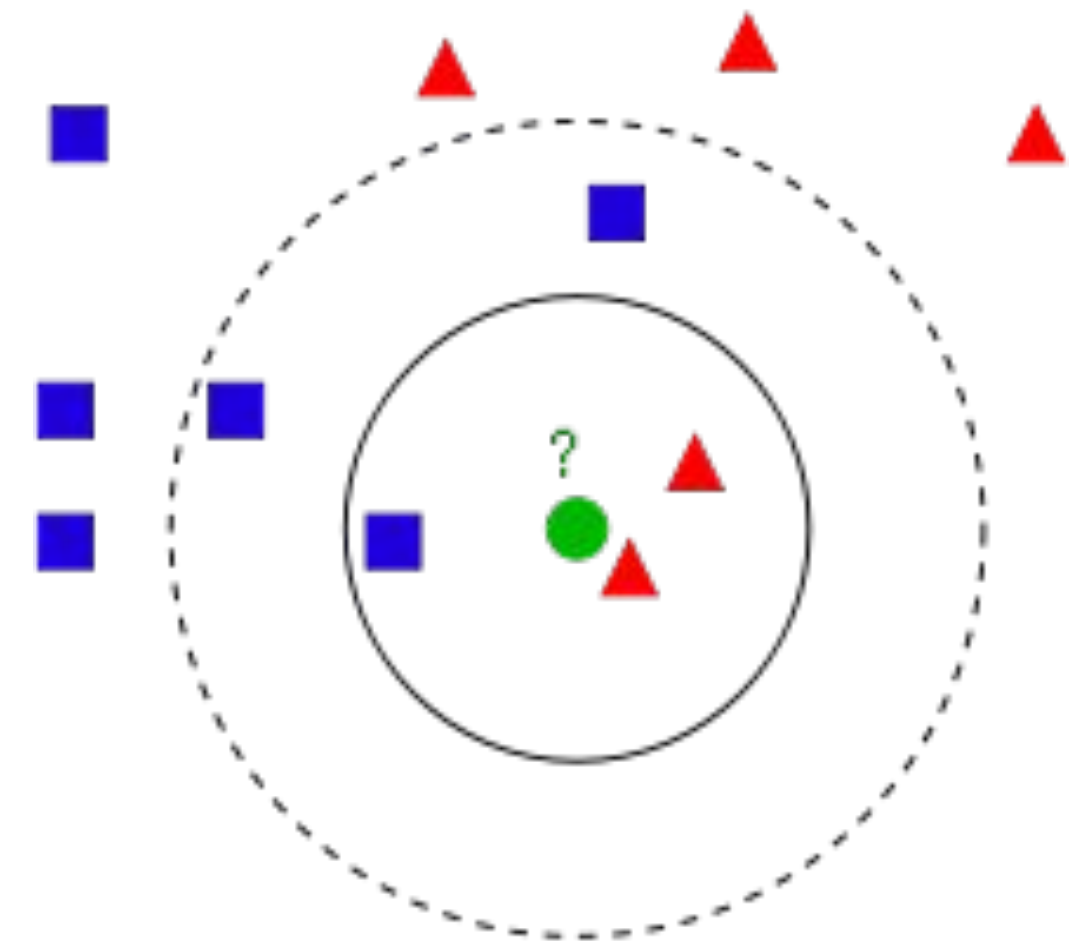


# K Nearest Neighbors (KNN)

## К ближайших соседей

Метод К ближайших соседей

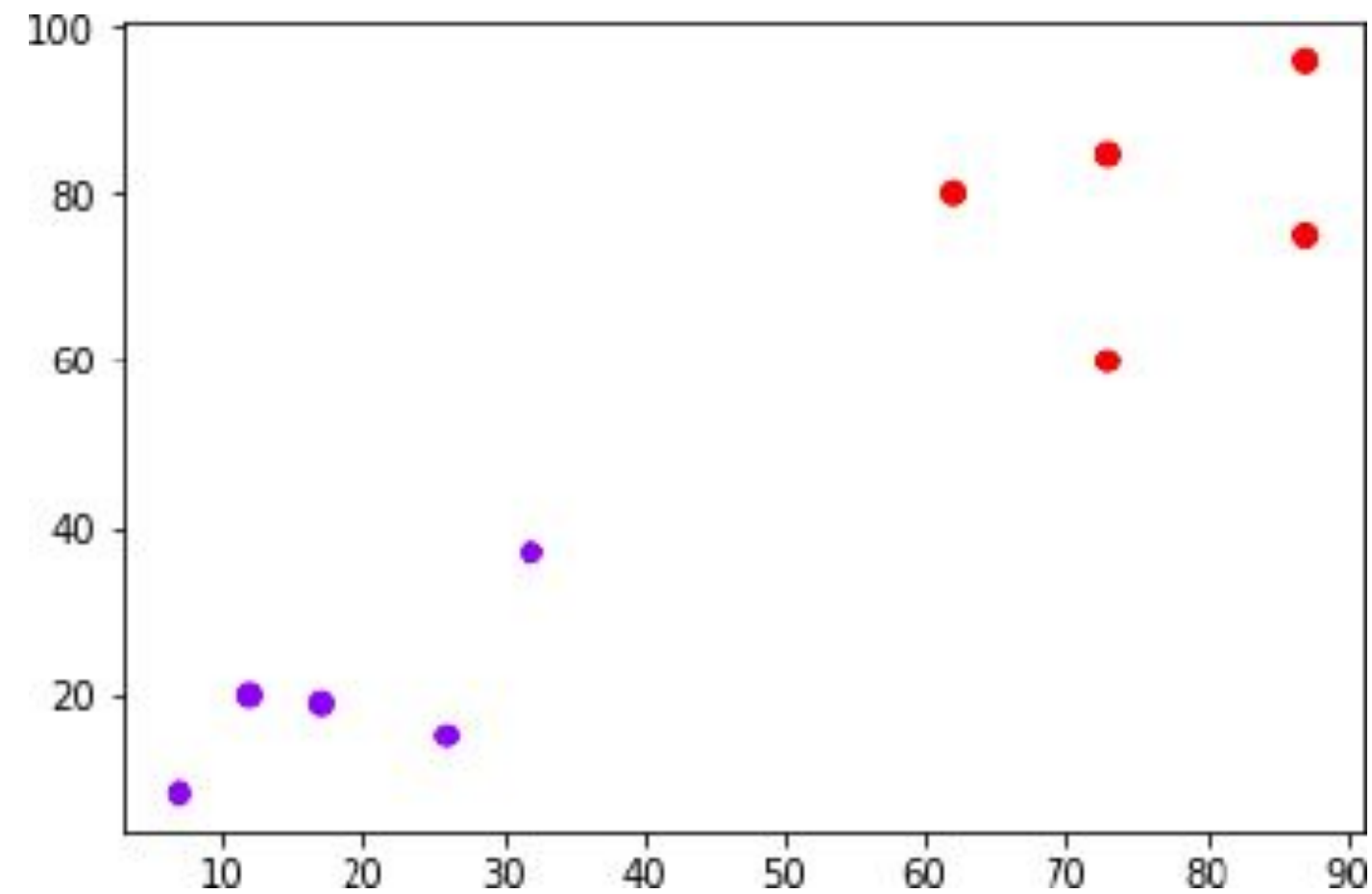
- На вход подается вектор - признаковое описание какого-то объекта
- Находится К ближайших к нему векторов, для которых ответ известен
- Ответ для новой точки выбирается с помощью
  - Усреднения в случае регрессии
  - Голосования в случае классификации
- Возможно также усреднение/голосование с весами



KNN - пример «ленивого», а также  
*непараметрического* алгоритма.

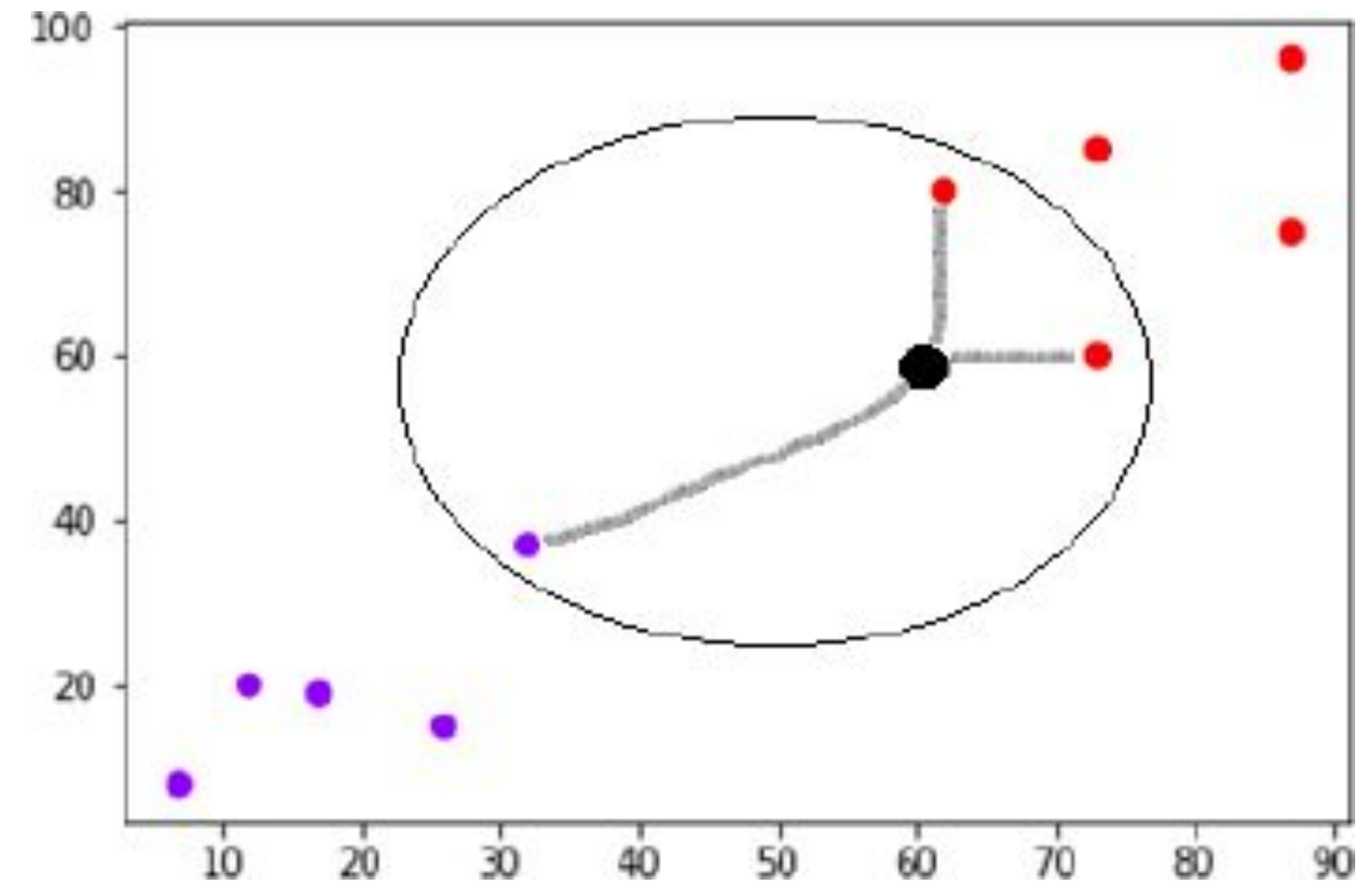
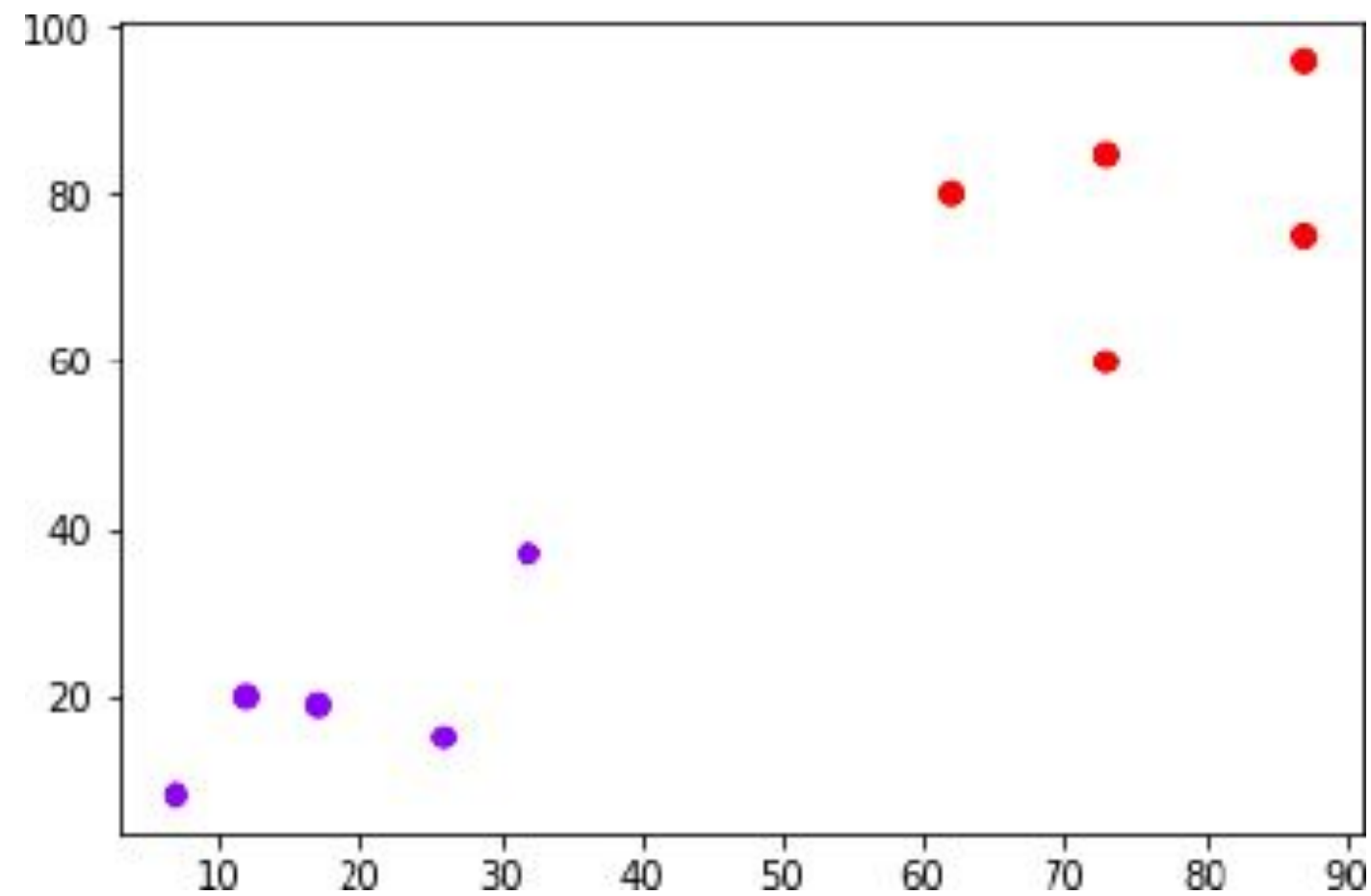
# К Nearest Neighbors (KNN)

## К ближайших соседей



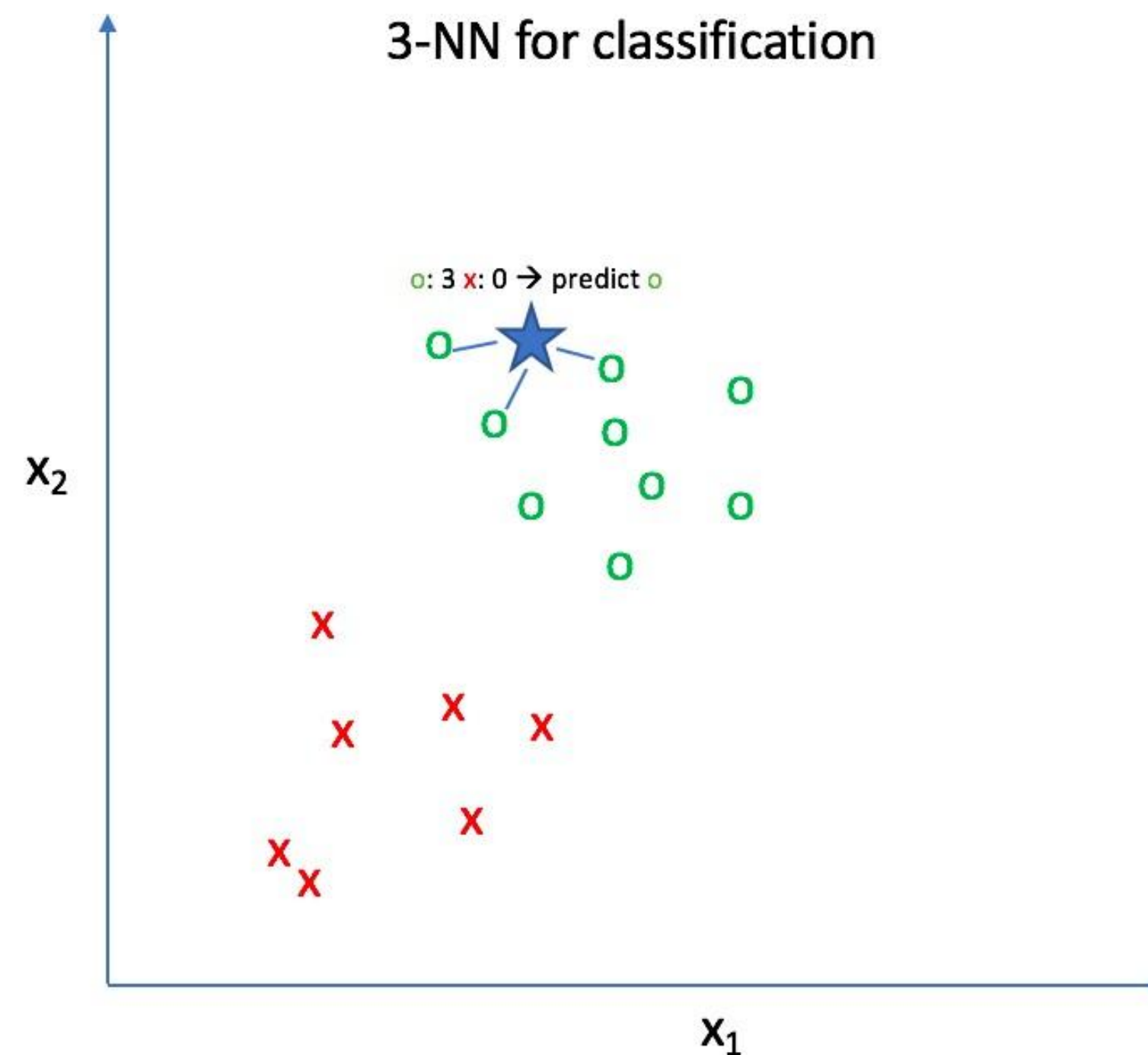
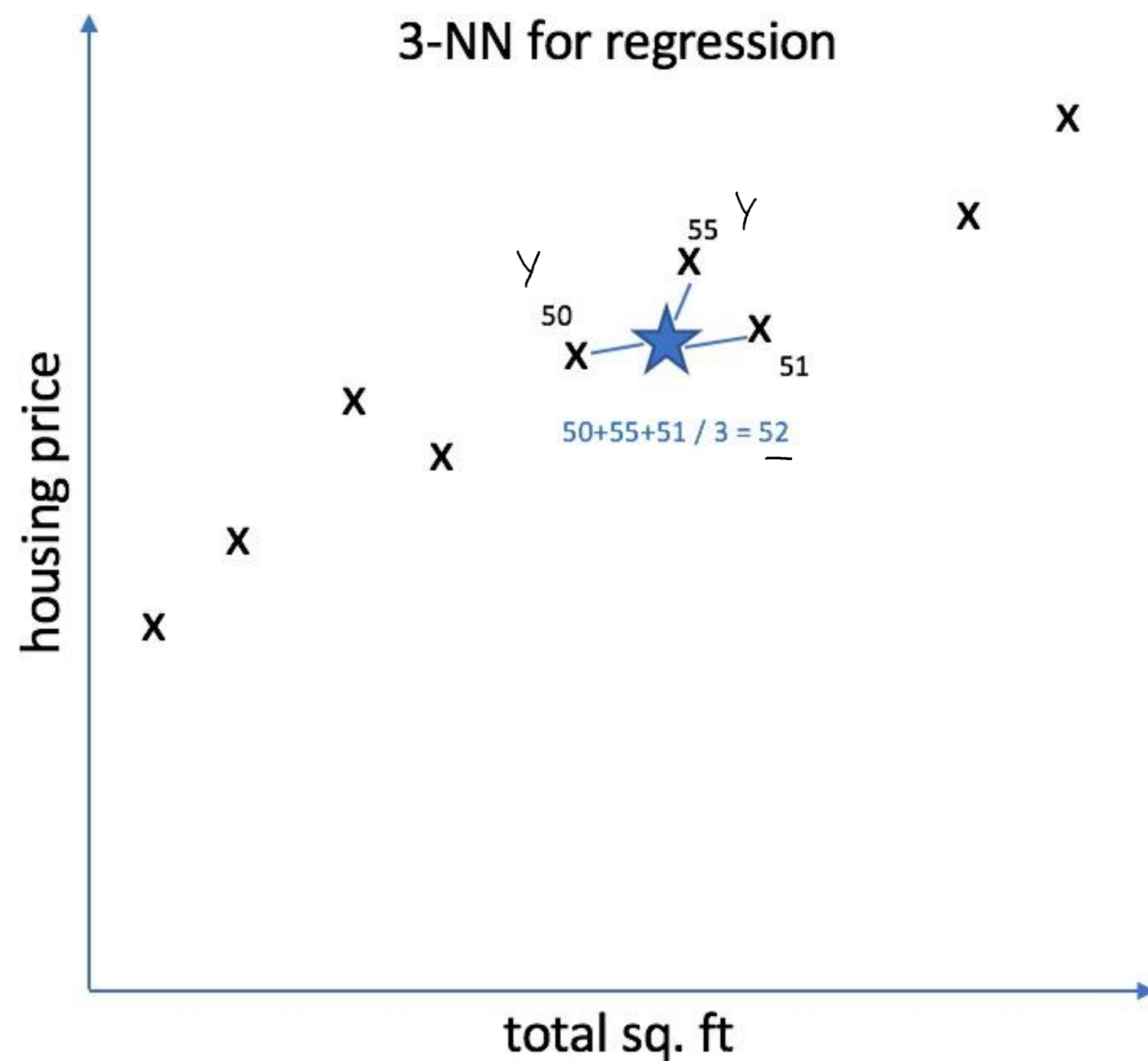
# К Nearest Neighbors (KNN)

## К ближайших соседей



# K Nearest Neighbors (KNN)

## К ближайших соседей



*K* - внешний параметр. Он подбирается так, чтобы модель работала как можно лучше.  
Результат предсказания для некоторых точек может зависеть от *K*.

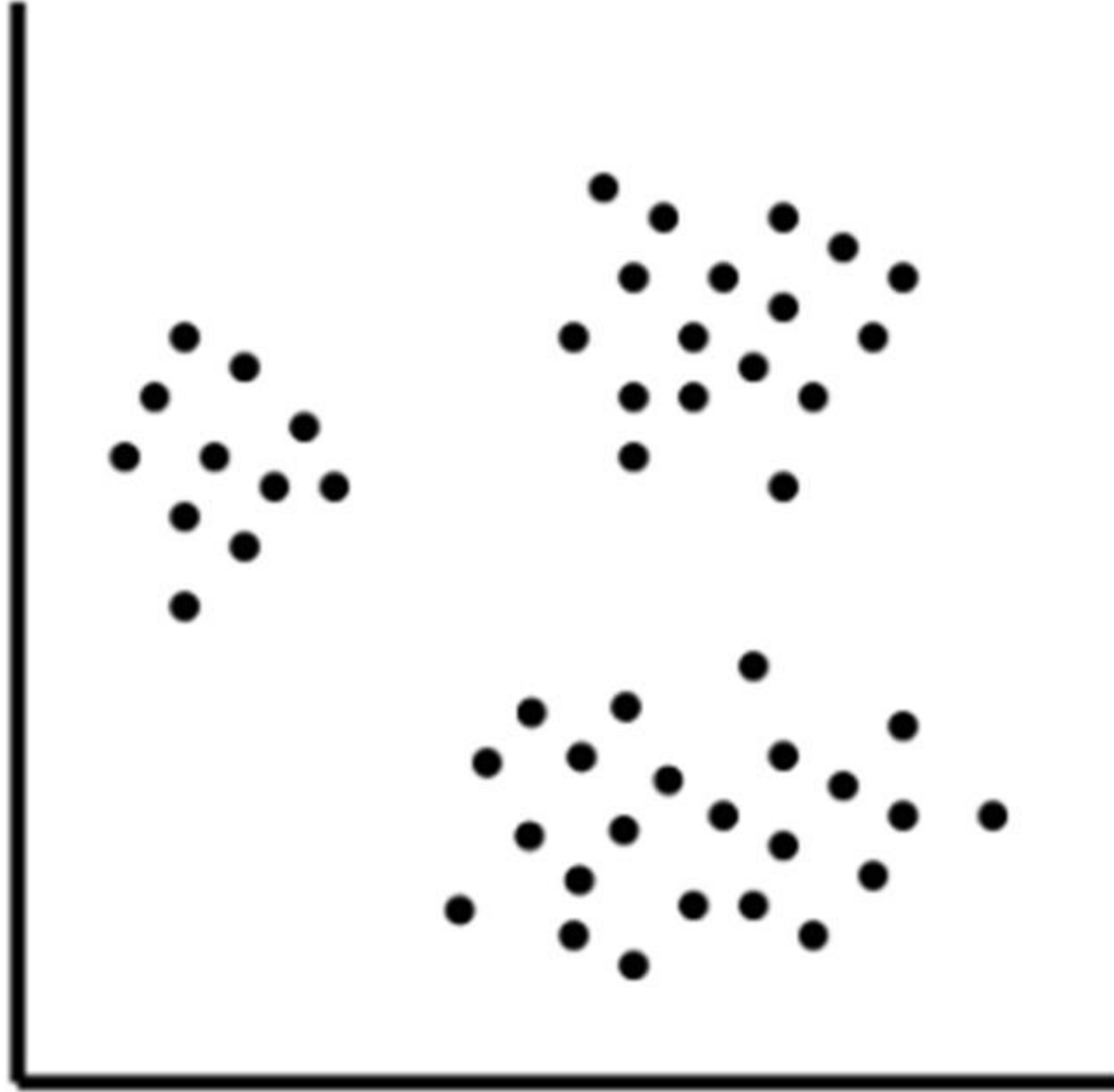
Сейчас мы опустим подробности реализации kNN, нам важно было понять интуицию, которая стояла за этим алгоритмом.

Эта интуиция пригодится нам во всех метрических алгоритмах!

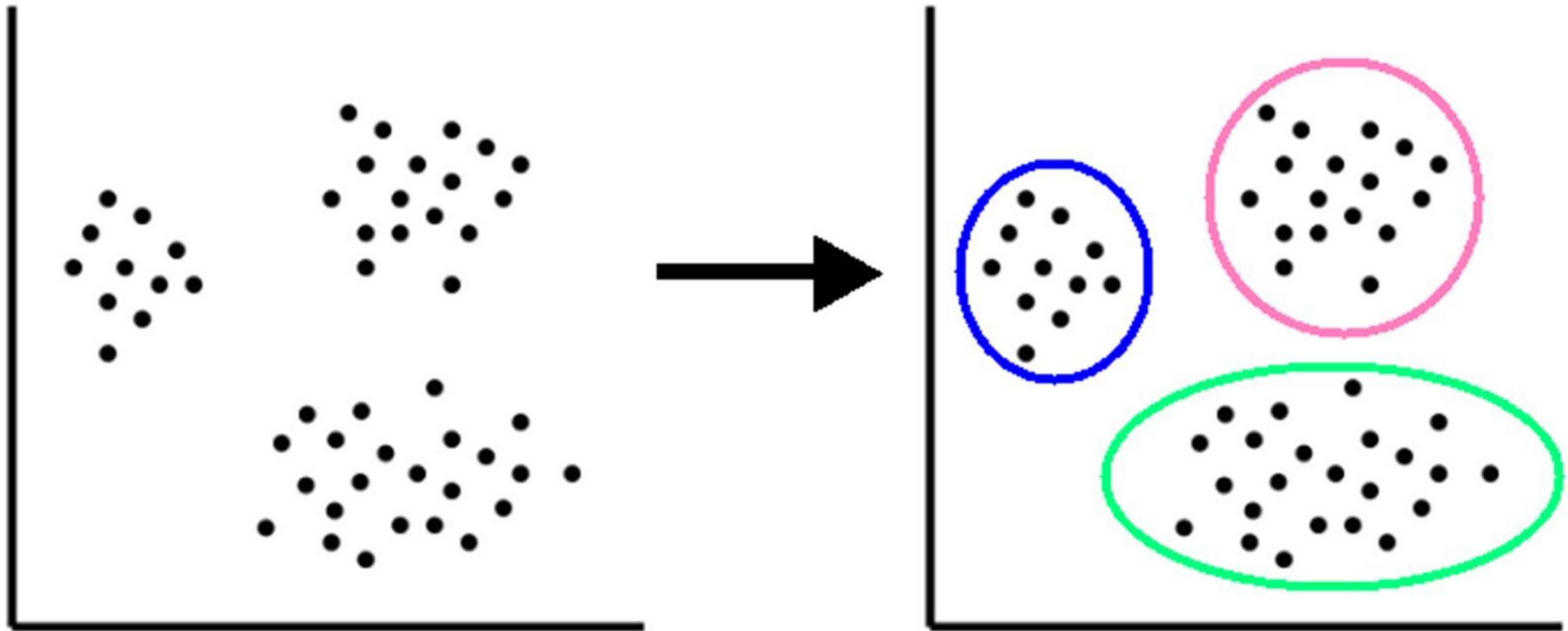


# Unsupervised learning. Задачи кластеризации

# Основная идея задач кластеризации

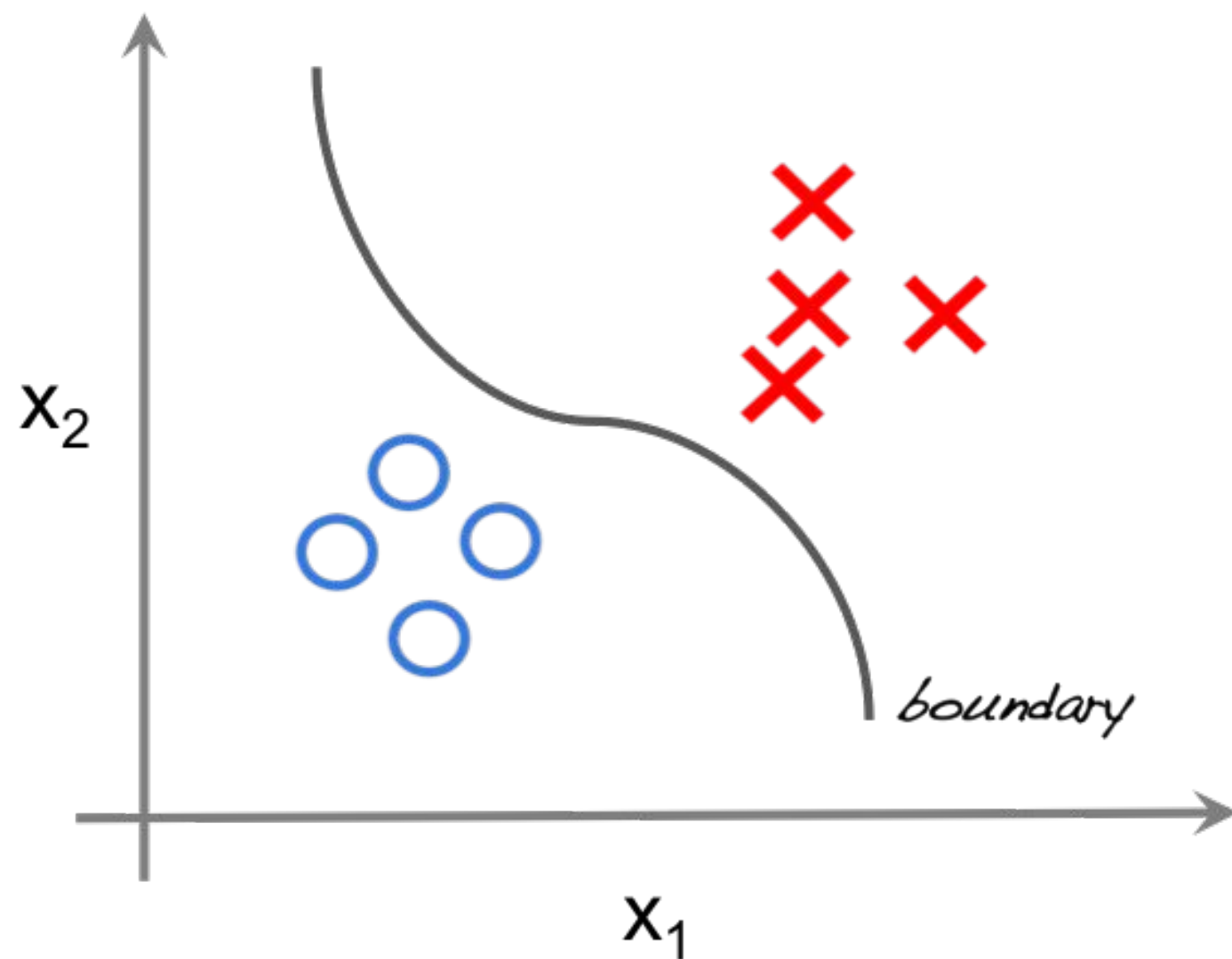


# Основная идея задач кластеризации

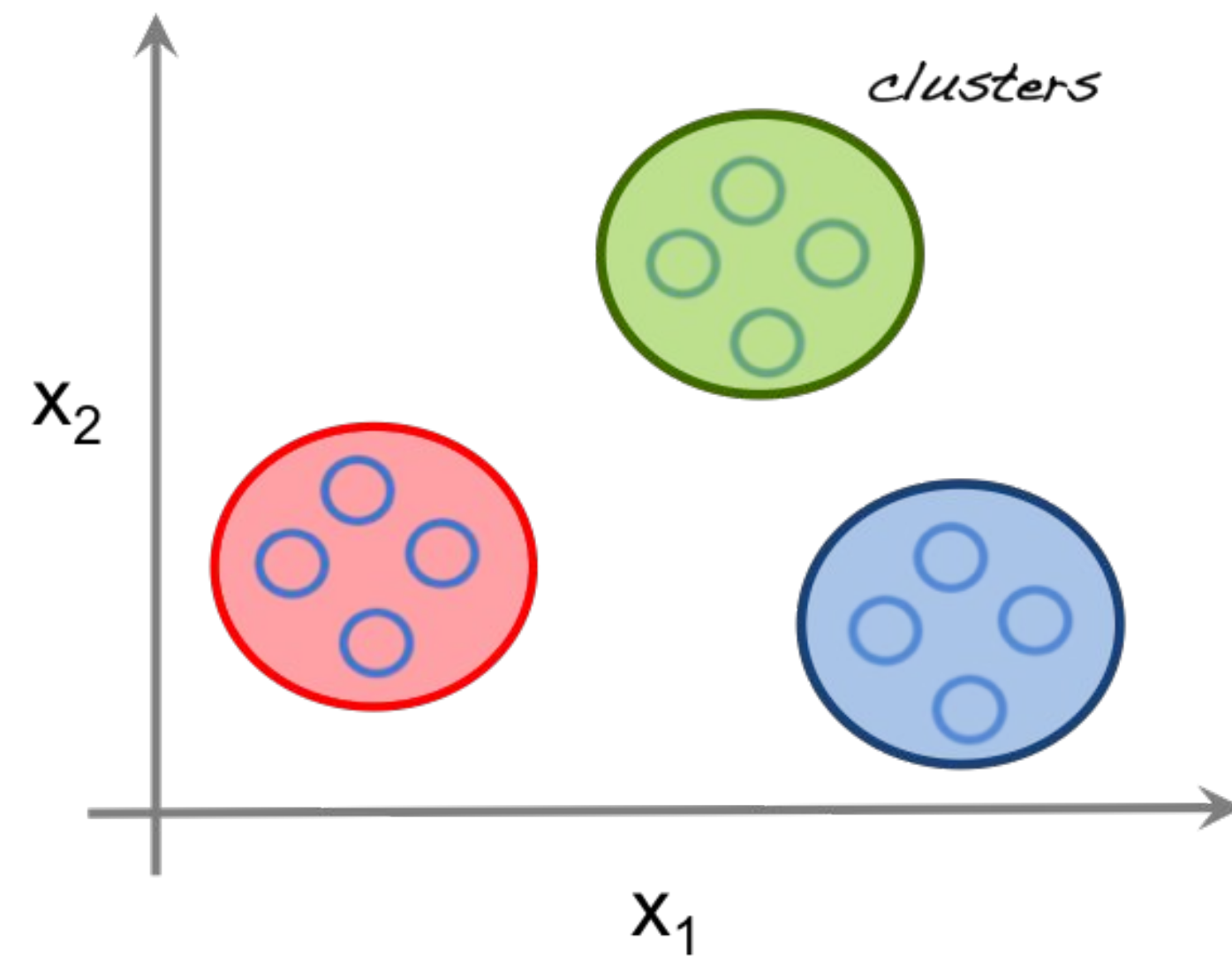


# Основная идея задач кластеризации

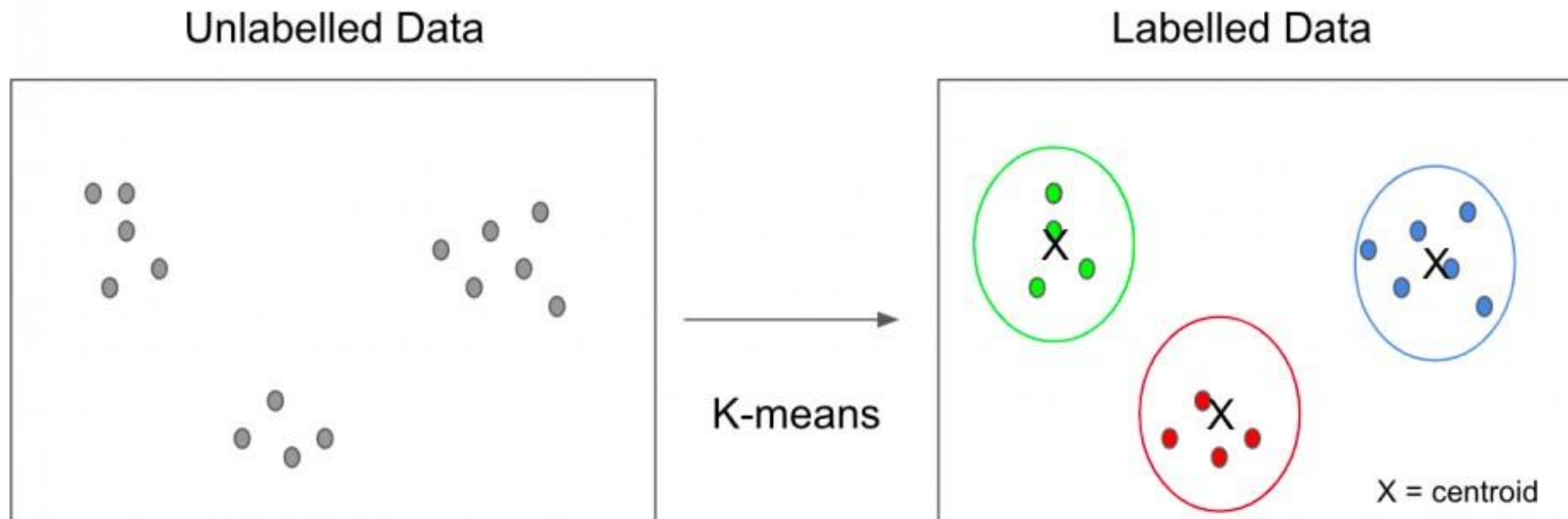
Supervised learning



Unsupervised learning

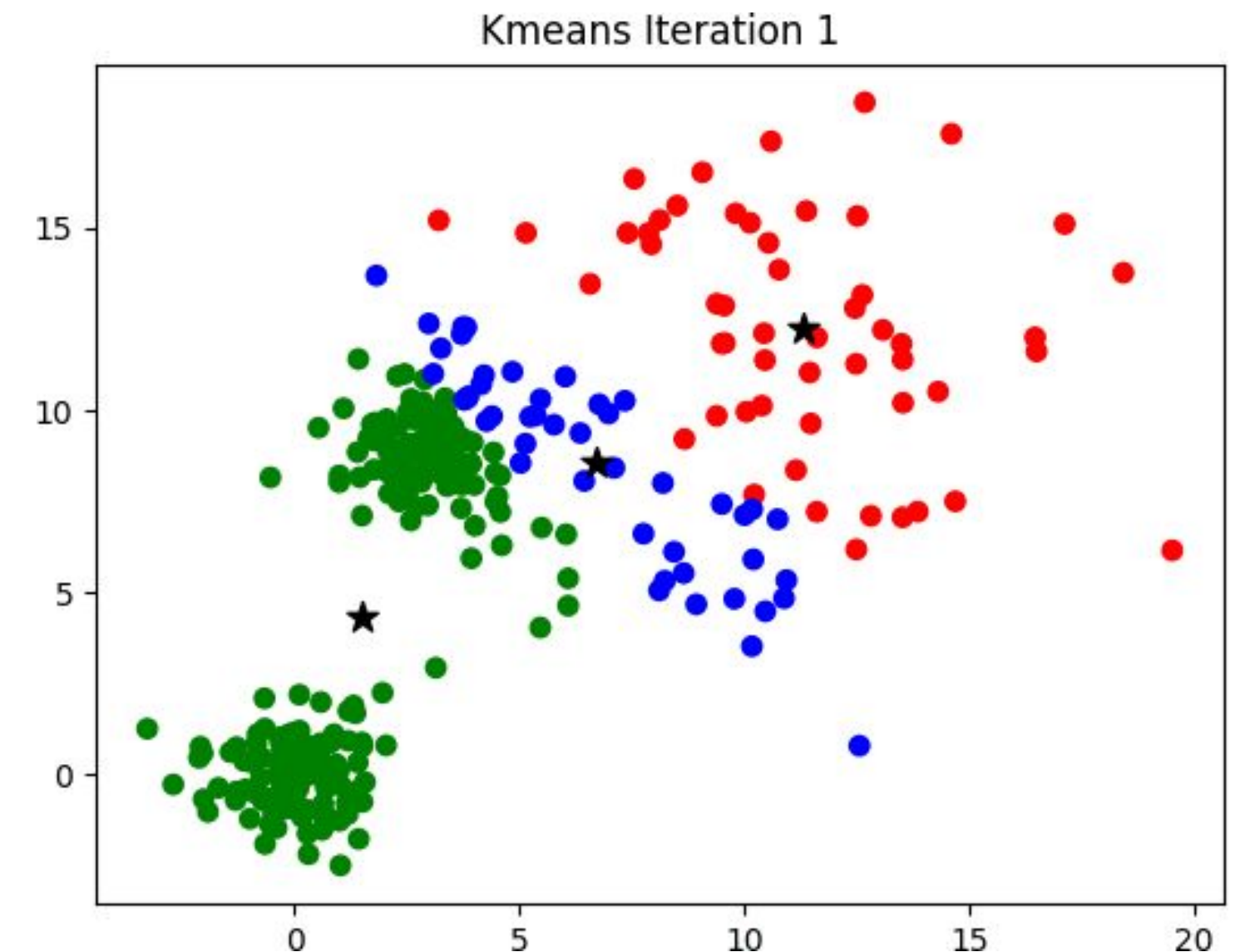


# Алгоритм K-Means



# sklearn.neighbors.KMeans

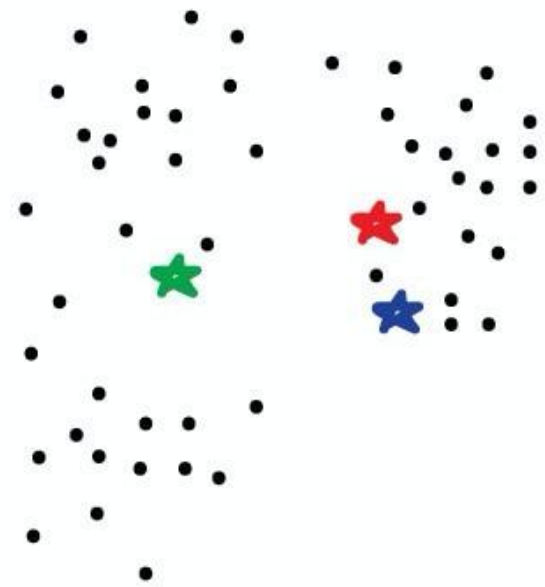
- **Выбор гиперпараметров:** Выбираем  $k$  и **подходящую метрику** (в смысле расстояния между объектами).
- **Обучение:**
  - Случайно иницилируем  $k$  центроидов.
  - Для каждой точки находим ближайший центроид, назначаем соотв. метку.
  - Пересчитываем позиции центроидов как центры масс соотв. кластеров
- **Предсказание:** Находим ближайший центроид, возвращаем его метку.



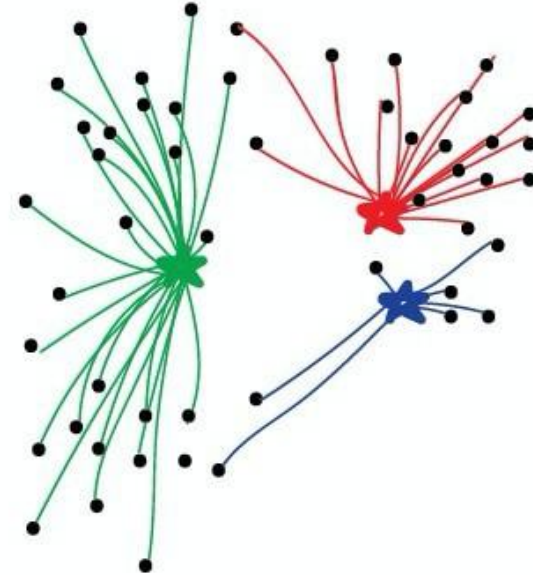


# Ставим три ларька с шаурмой оптимальным образом

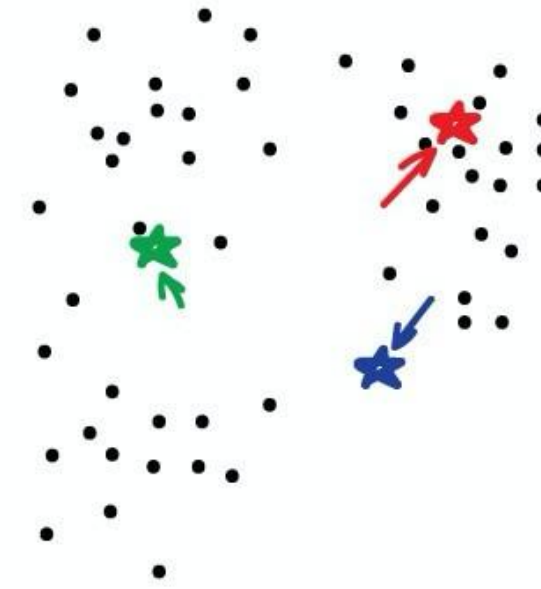
(иллюстрируя метод К-средних)



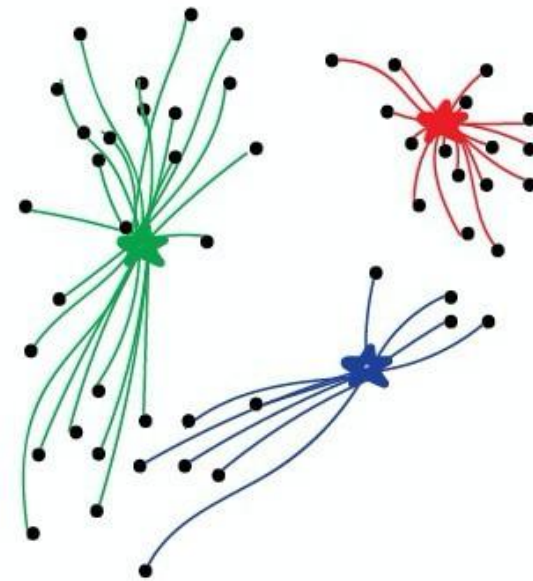
1. Ставим ларьки с шаурмой в случайных местах



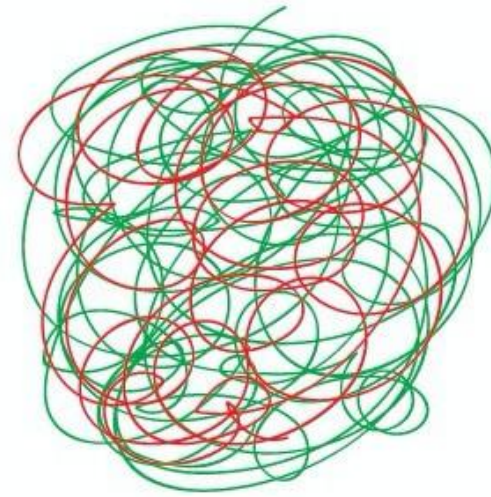
2. Смотрим в какой кому ближе идти



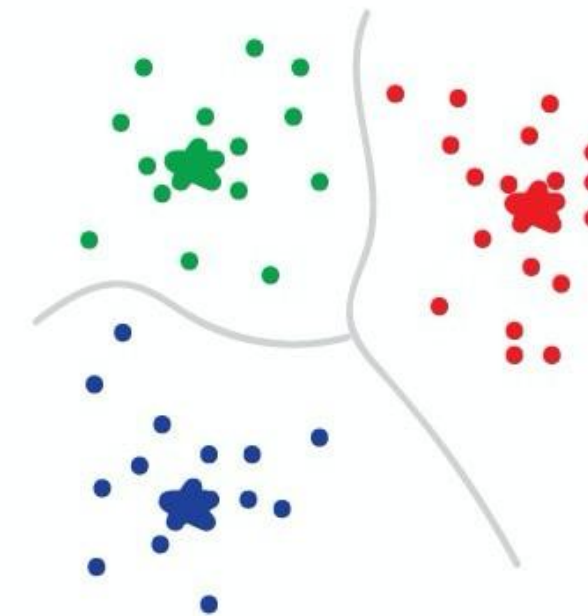
3. Двигаем ларьки ближе к центрам их популярности



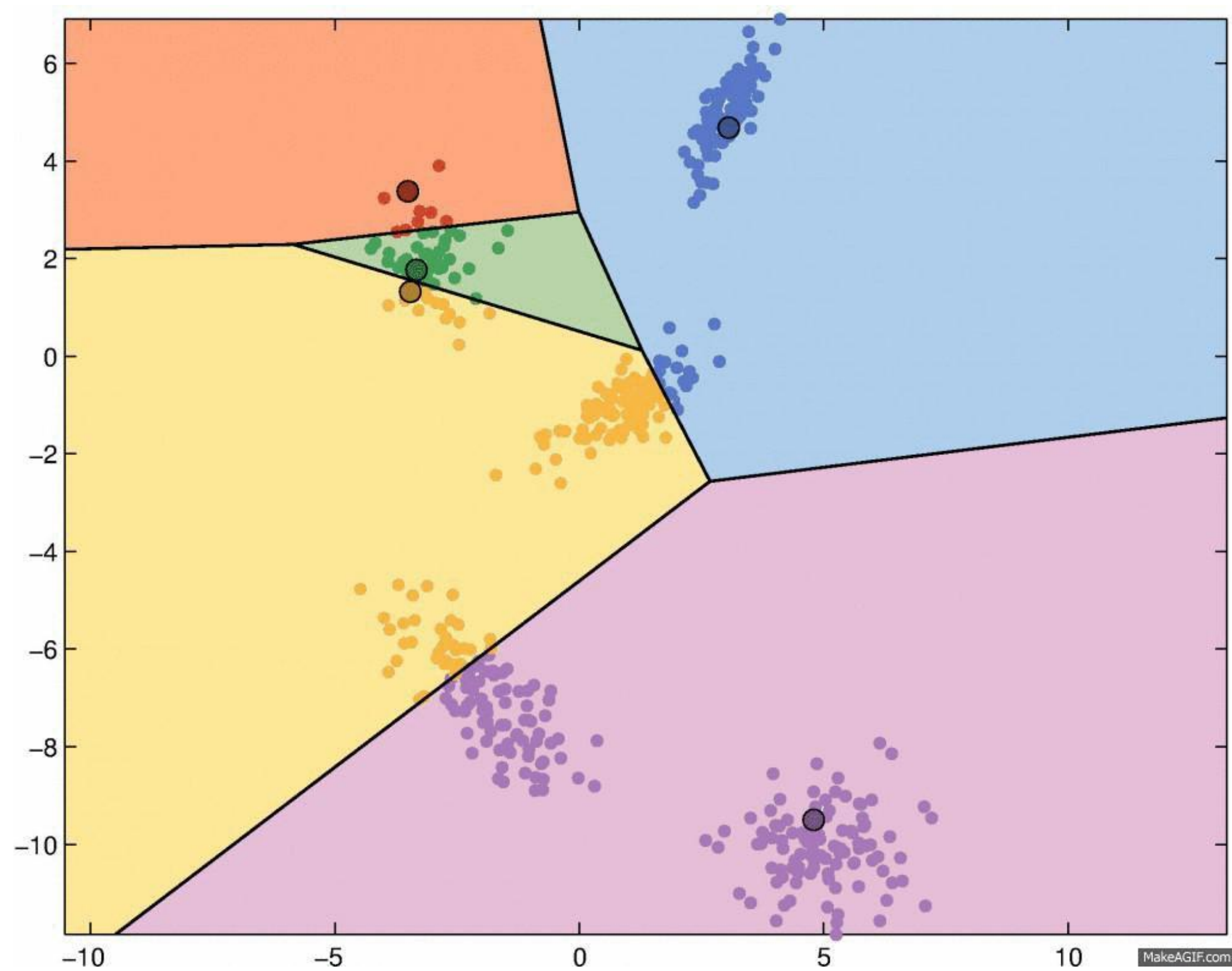
4. Снова смотрим и двигаем



5. Повторяем много раз



6. Готово, вы великолепны!



к-Means выстраивает приближение т.н.  
диаграммы Вороного по данным

# k-Means. Применения

## Понижение разреженности данных.

Представьте себе матрицу “пользователи x контент”. Она очень разреженная: **каждый пользователь взаимодействует с малой долей контента**. Обучать алгоритмы ML на таких данных очень трудно.

**Решение:** Сгруппируем пользователей при помощи k-means, агрегируем предпочтения в пределах каждой группы, будем работать с матрицей “группа x контент”, а при необходимости сделать предсказание для нового пользователя — находить наиболее похожую на него группу и брать предсказание для неё.

# k-Means. Плюсы

- **Простой интерпретируемый алгоритм.**

Это хороший baseline, с него стоит начать.

- **Даёт качественную кластеризацию при грамотном подборе метрики.**
- **Сложность предсказания —  $O(k \log k)$  в среднем и  $O(k^2)$  в худшем случае. Нужно  $O(k)$  дополнительной памяти**

Это практически мгновенно и не зависит от размера входных данных.

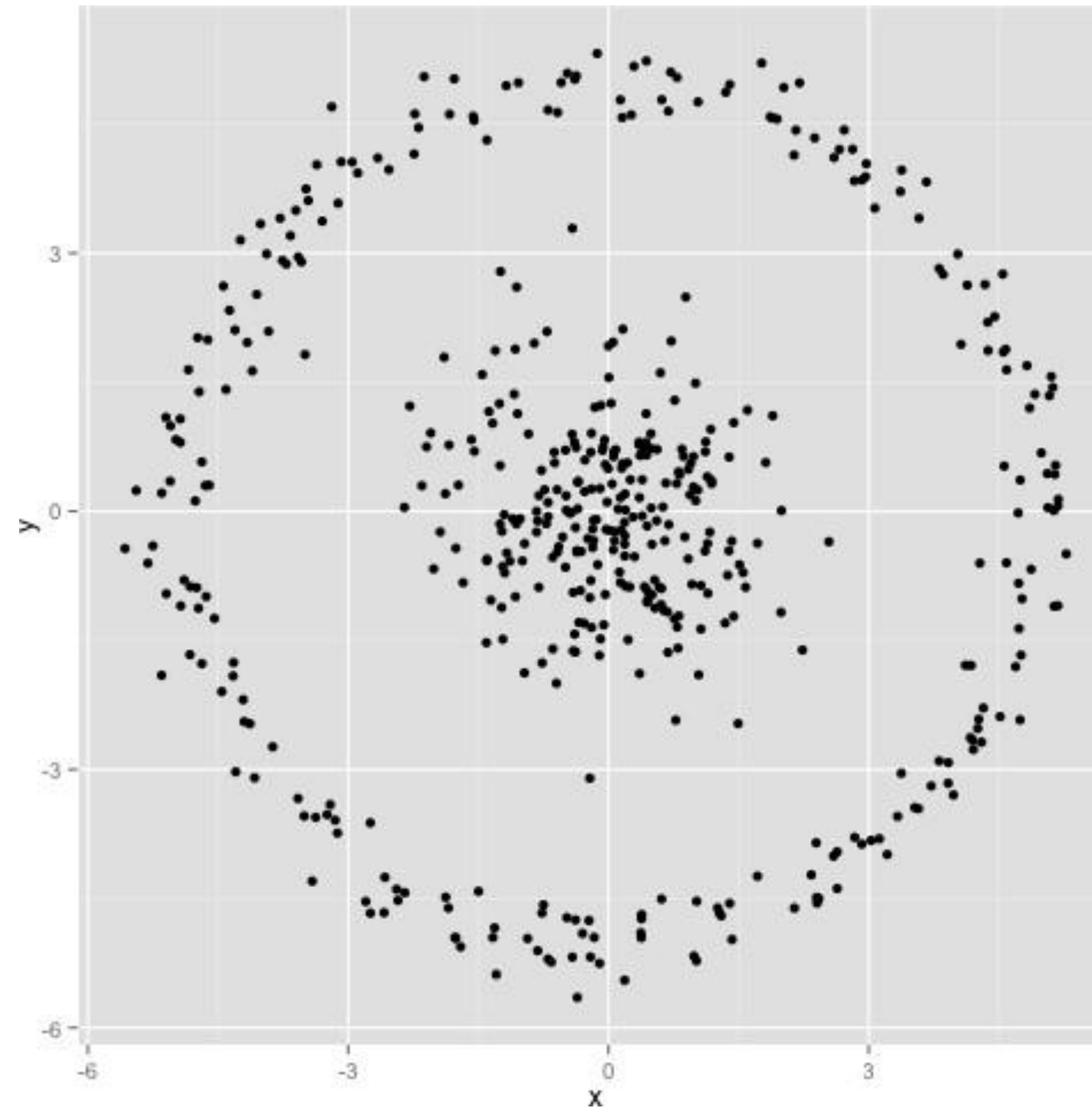
- **Понятно, как устроено оптимальное решение**  
(диаграмма Вороного с  $k$  ячейками)
- **Понятно, как пересчитать центроиды при поступлении новой точки.**

# k-Means. Минусы

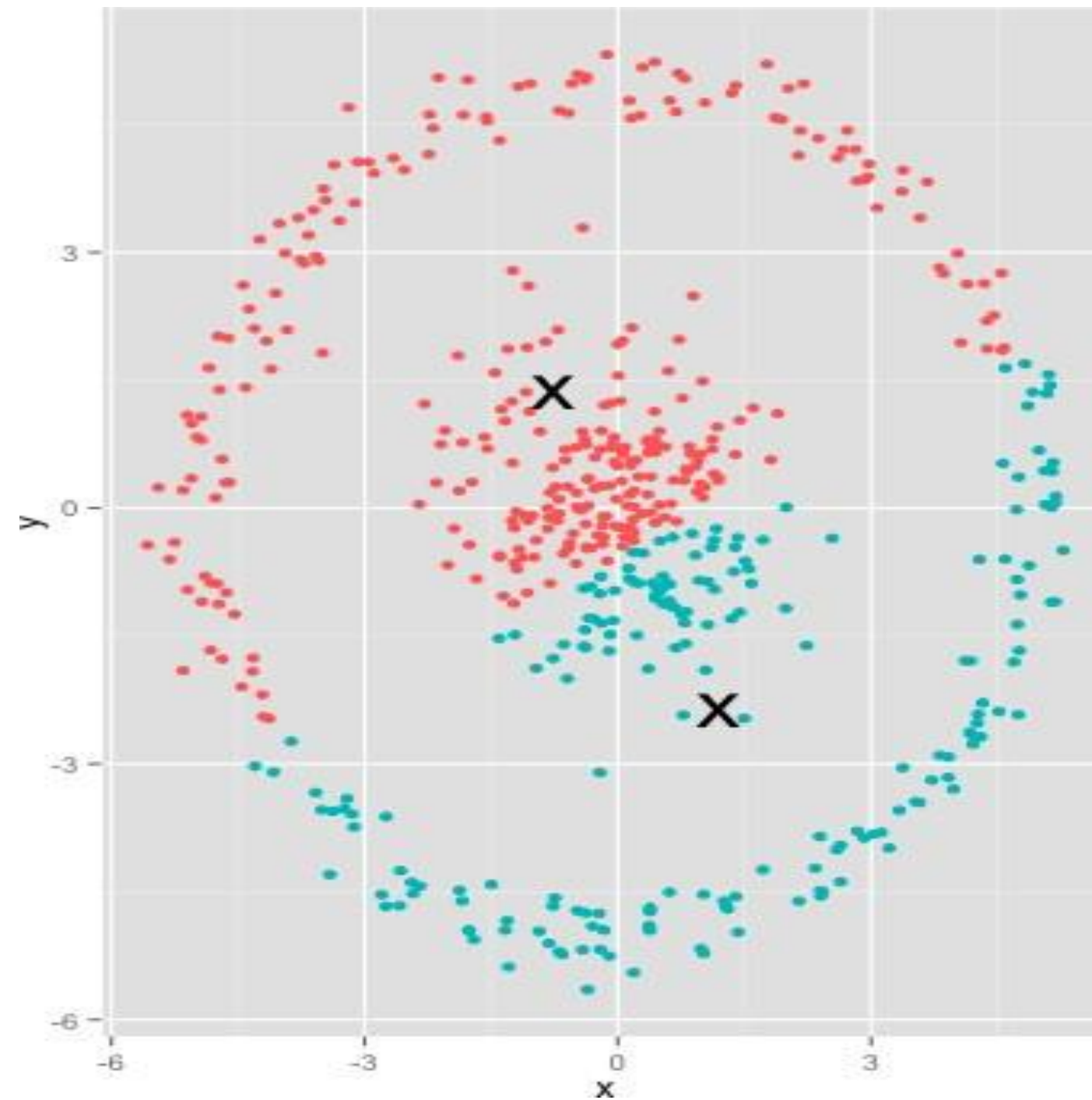
- **Непонятно, как подобрать  $k$  и правильную метрику** (здесь это важно, т.к. евклидова метрика может быть адекватна локальной геометрии данных, но глобальную структуру она чаще всего описывает неправильно).
- **Ответ сильно зависит от начального приближения.** С ним может и не повезти.
- **Итеративный процесс обучения.** Непонятно, сколько итераций потребуется до сходимости.



**Как k-Means разделит такие точки на кластеры при  $k=2$ ?**

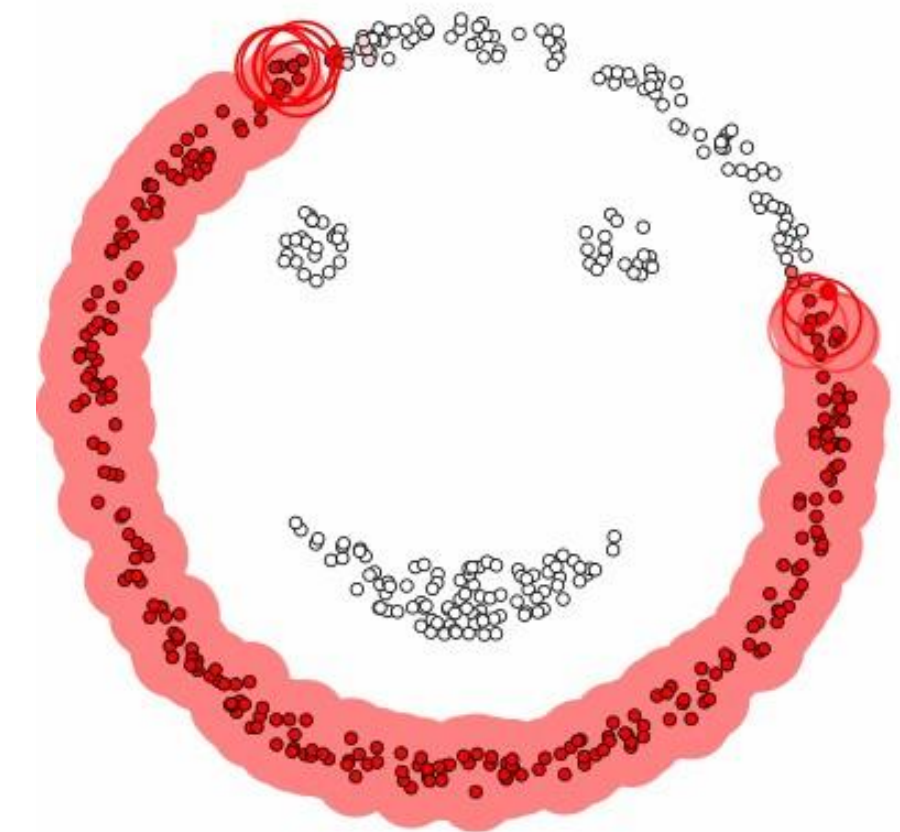
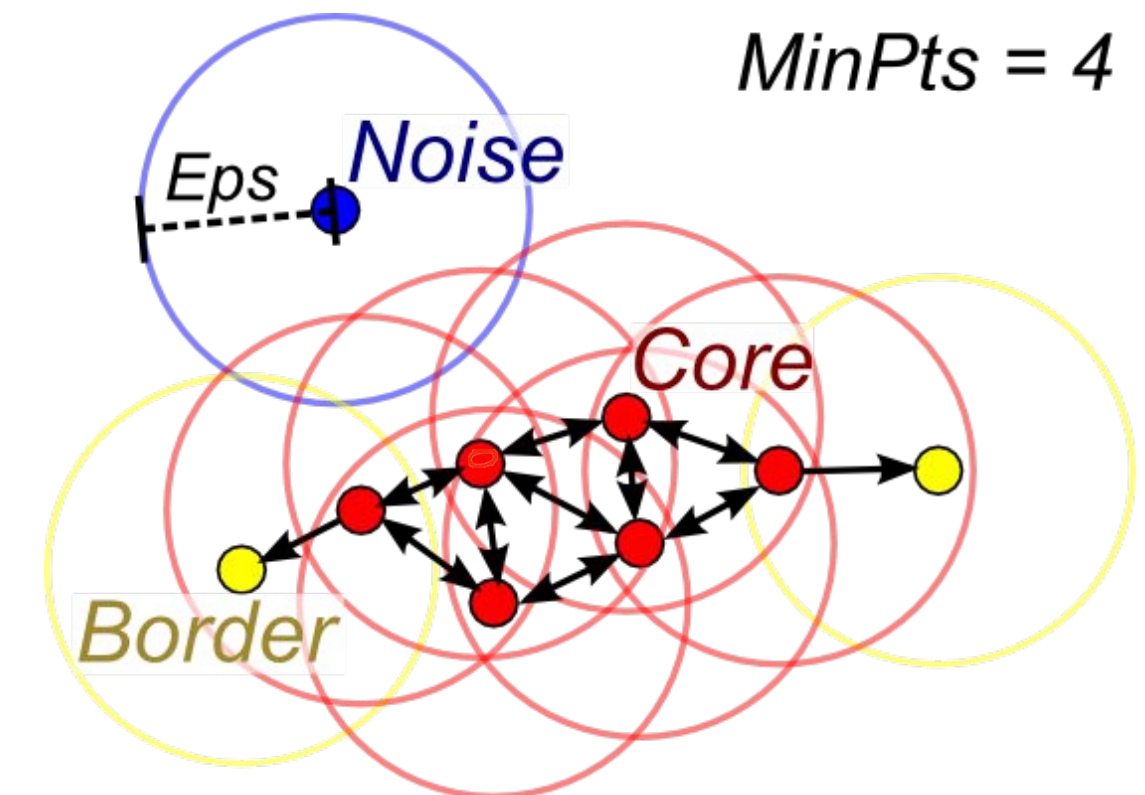


**Как k-Means разделит такие точки на кластеры при  $k=2$ ?**



# sklearn.cluster.DBSCAN

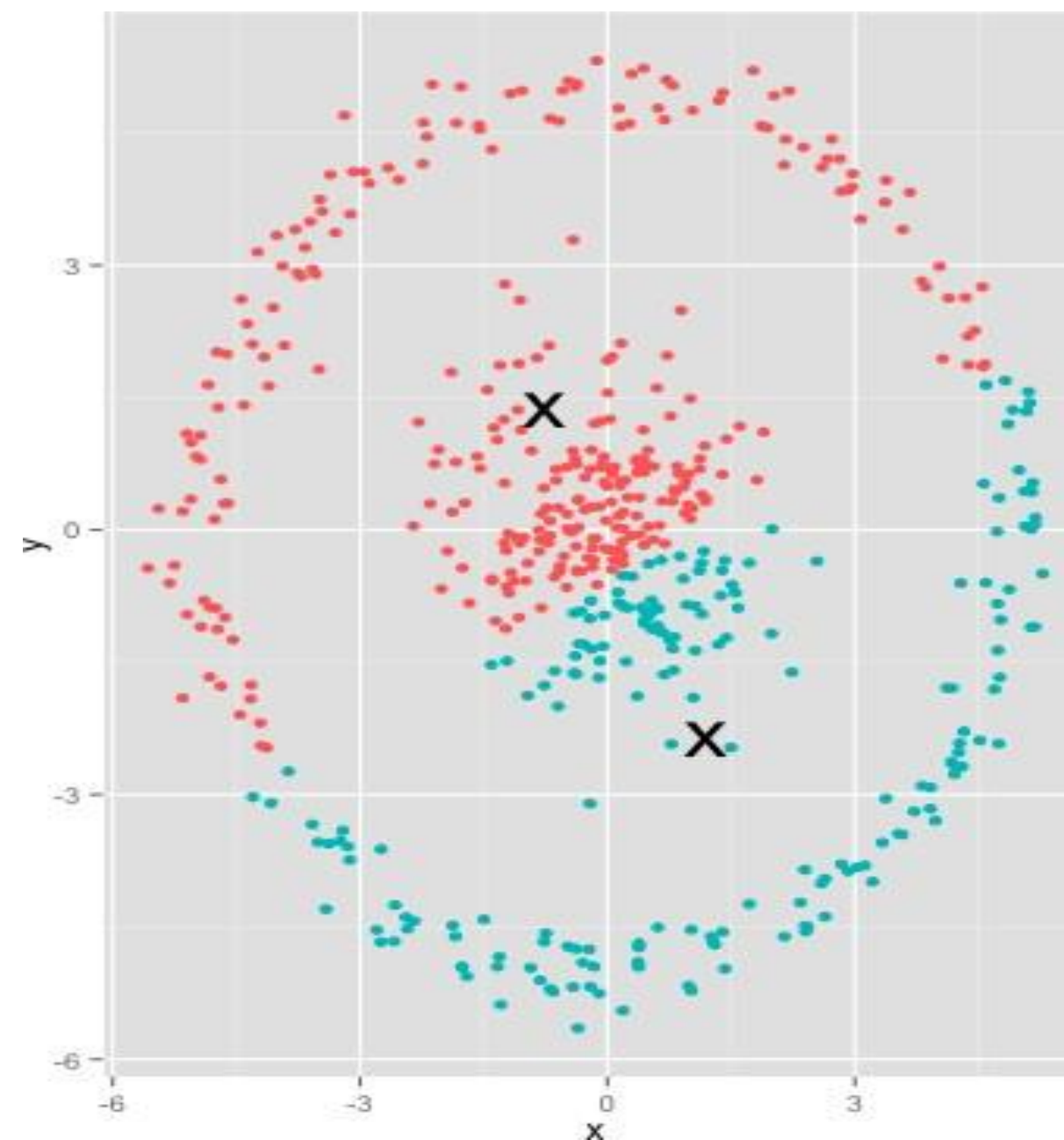
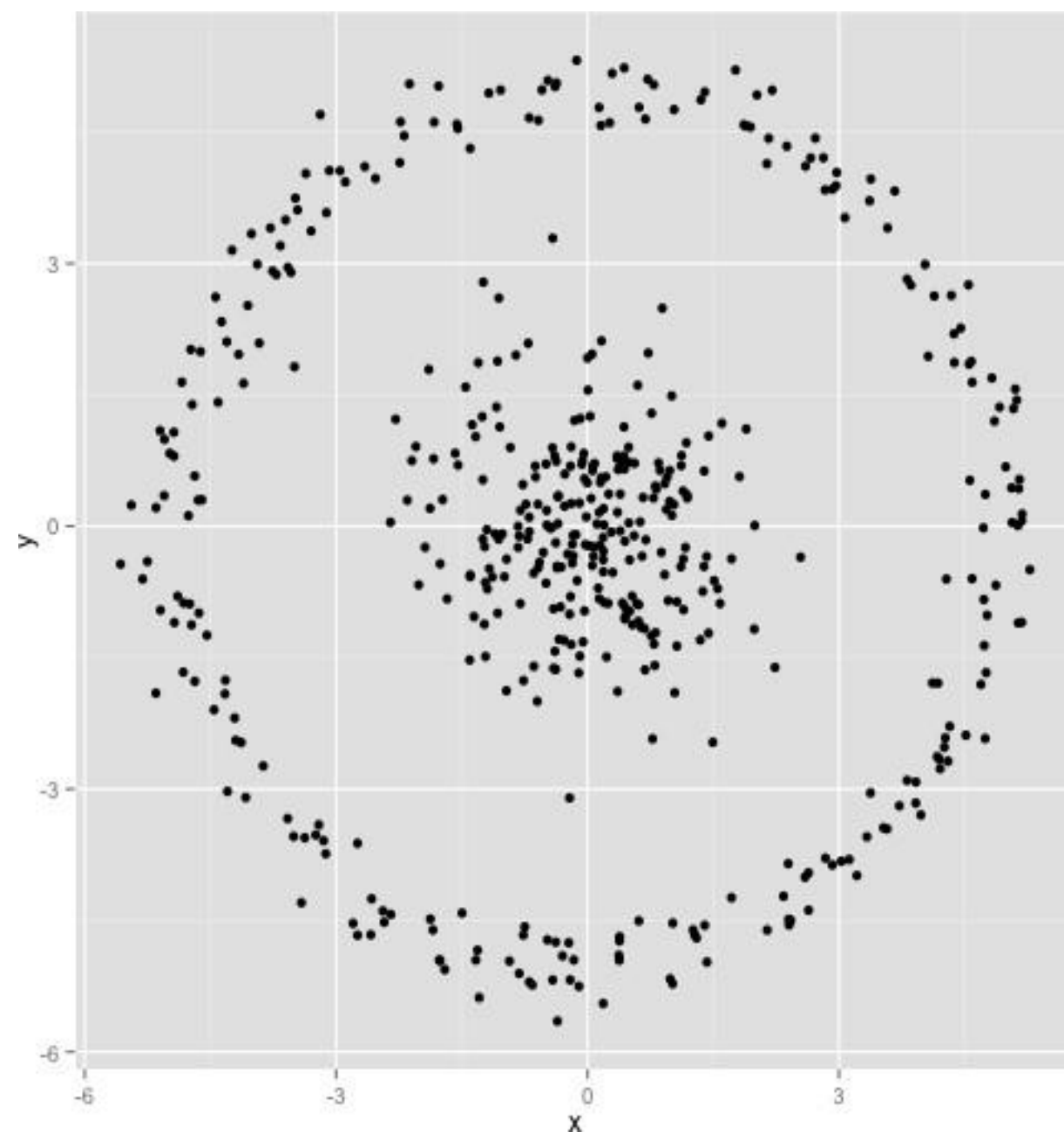
- **Выбор гиперпараметров:** Выбираем метрику, радиус окрестности и минимальное количество точек в пределах радиуса.
- **Обучение:**
  - Выстраиваем окрестность вокруг каждой точки данных. Перебираем окрестности в порядке убывания плотности
  - Если в пределах окрестности содержится хотя бы  $MinPts$  точек, то классифицируем соотв. точку как core
  - В противном случае классифицируем точку либо как border, если в её окрестности есть хотя бы одна core-точка, и как noise иначе.
- **Предсказание:** Core- и border-точки в пределах одной окрестности соединяются рёбрами. Кластерами будут — компоненты связности полученного графа. Noise-точки рапортуются отдельно.



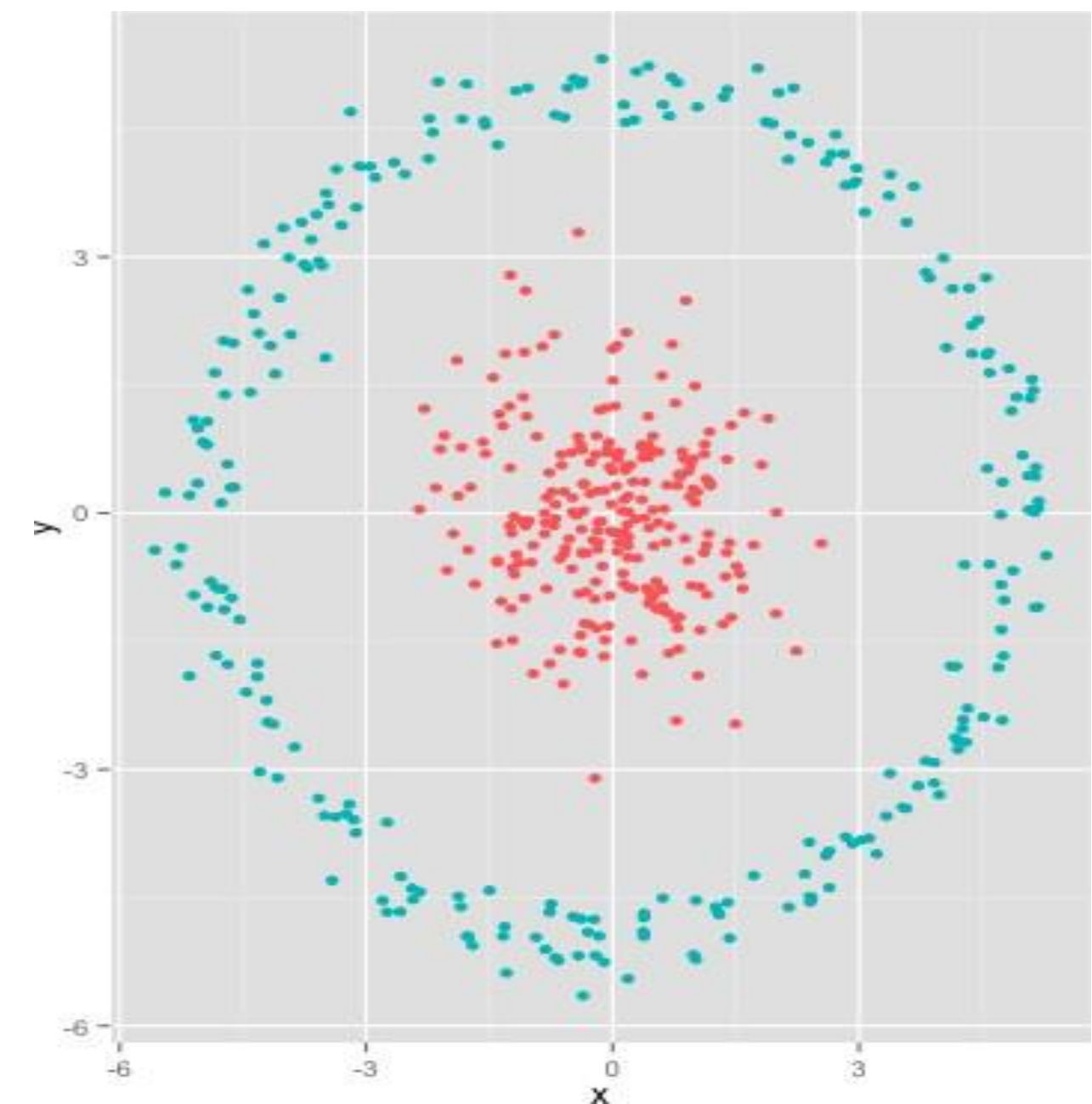


# Сравнение K-Means и DBSCAN

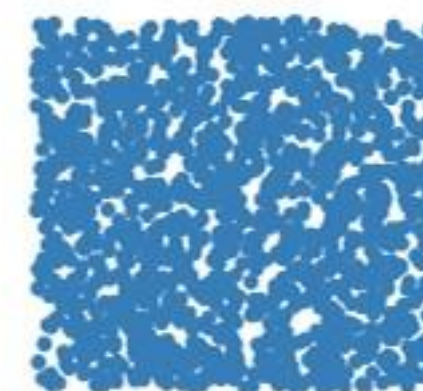
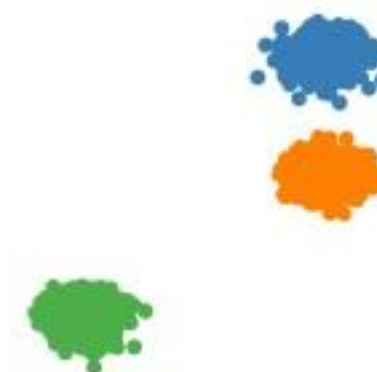
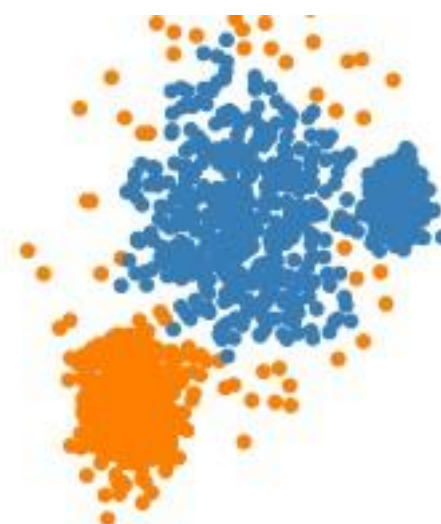
Оригинальный K-Means



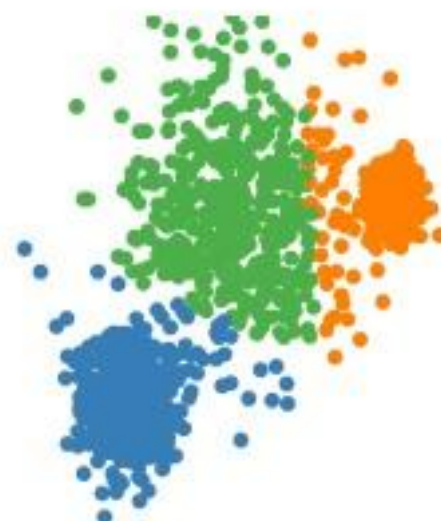
DBSCAN



DBSCAN



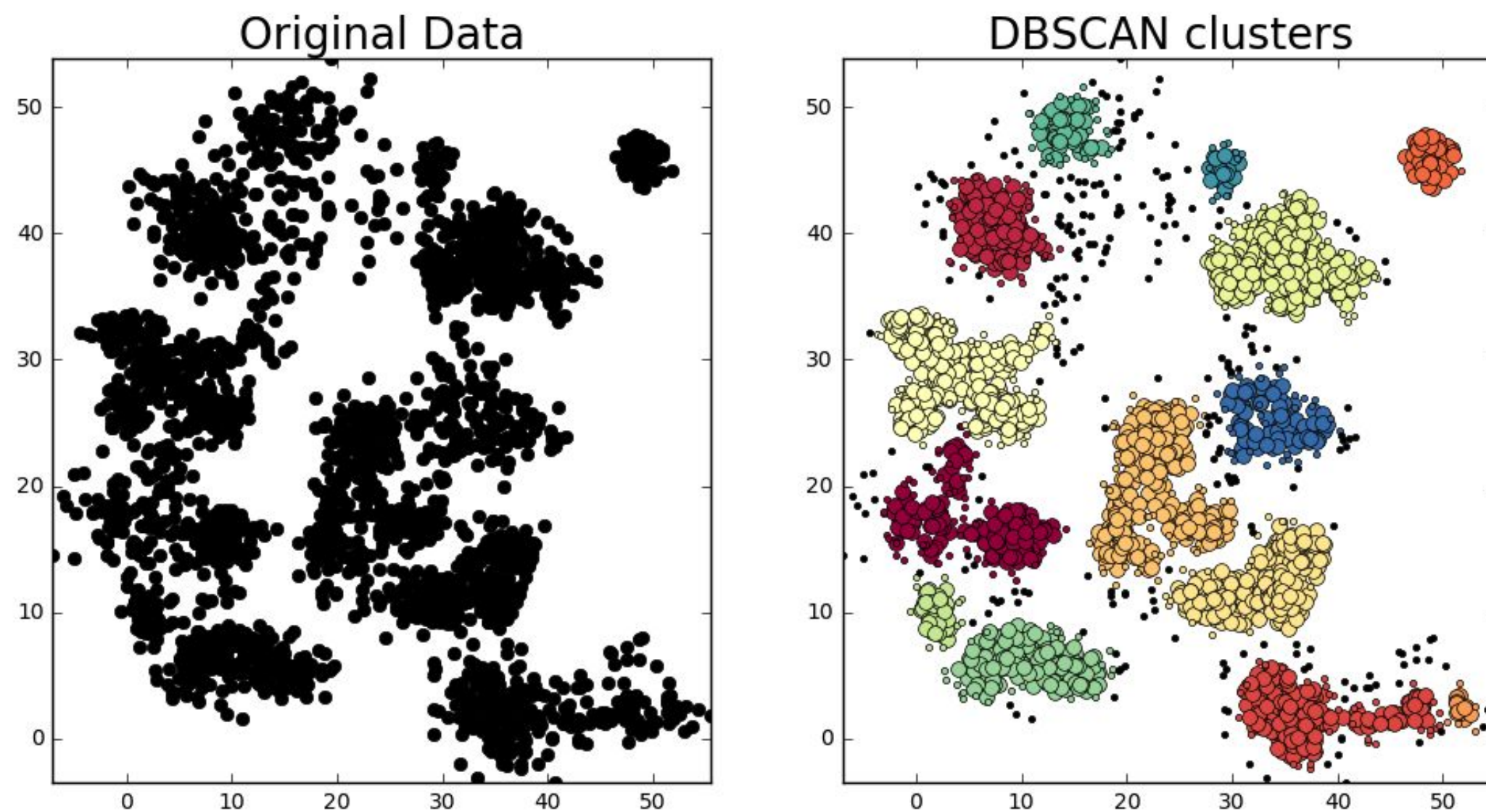
k-means





# DBSCAN. Применения

Всё те же, что и у k-Means, особенно когда эксперт по предметной области не может оценить  $k$  заранее.



# DBSCAN. Плюсы

- **Простой интерпретируемый алгоритм.**  
Результаты зачастую лучше, чем у k-Means.
- **Улавливает более тонкие локальные особенности** в данных. Многообразия локально неотличимы от  $\mathbb{R}^n$ , потому можно использовать обычную Евклидову метрику.
- **Не требует заранее указывать количество кластеров.**
- **Находит заодно и выбросы.**
- **Быстро обучается**, не требует итеративного уточнения.

# DBSCAN. Минусы

- **Нужно подбирать радиус, MinPts.**

Непонятно, как это сделать из интуитивных соображений.

- **Трудно делать предсказания для новых точек,** т.к. каждая новая точка изменяет плотность в окрестностях имеющихся точек.  
Нужны более сложные структуры данных, чем в k-Means.

- **Не учитывается структура построенного графа,** хотя в ней содержится много полезной информации.

# Оценка качества предсказания

И особенности процесса обучения ML-алгоритмов



# train-val-test split

В задачах supervised learning принято делить выборку на три непересекающиеся части:

- **Обучающая** (training sample). На ней происходит обучение модели.
- **Валидационная** (validation sample). На ней считают метрики качества, а по ним уже подбирают гиперпараметры.
- **Тестовая** (test sample). По ней оценивают качество обученной модели.



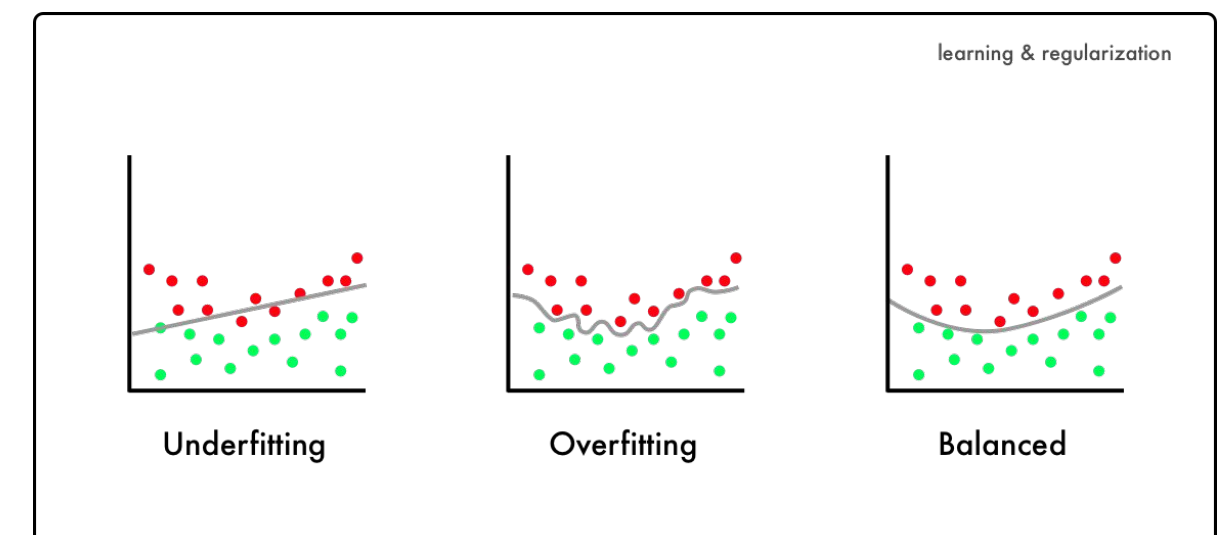
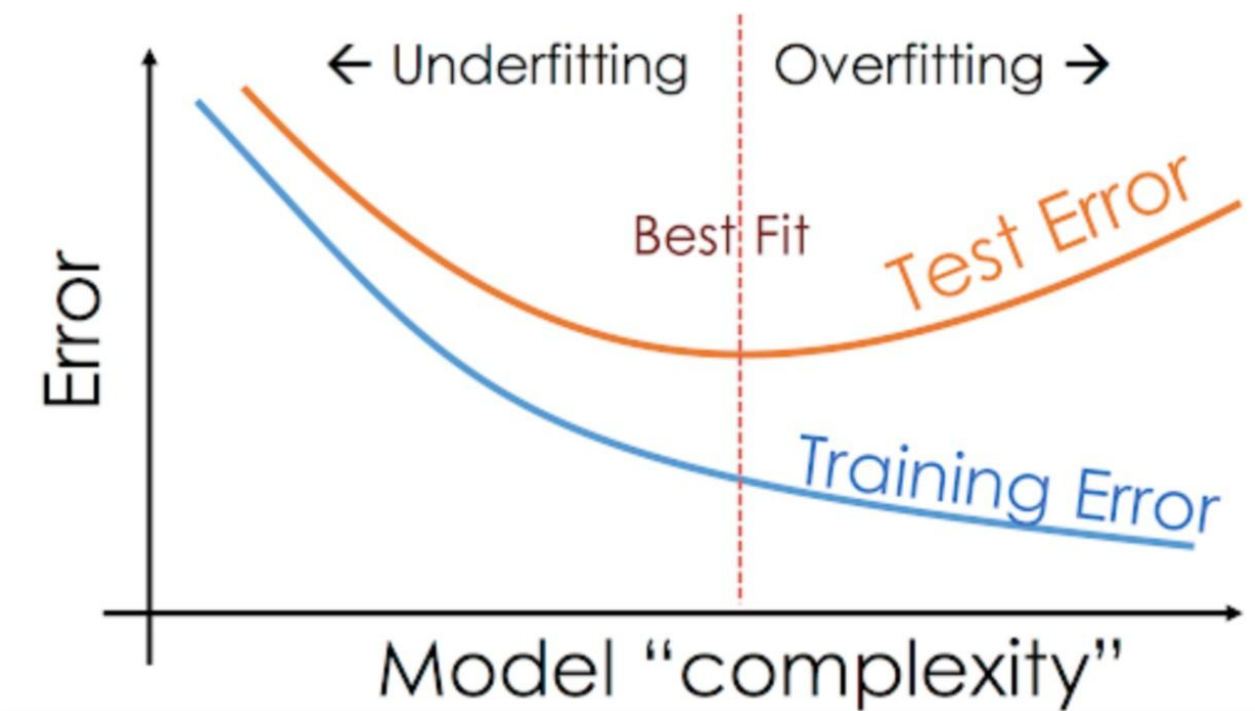
Валидационную выборку используют не всегда.  
Когда используют, то стараются брать её того же  
размера, что и тестовую.

**Важно!** Каждая выборка  
должна быть репрезентативна!

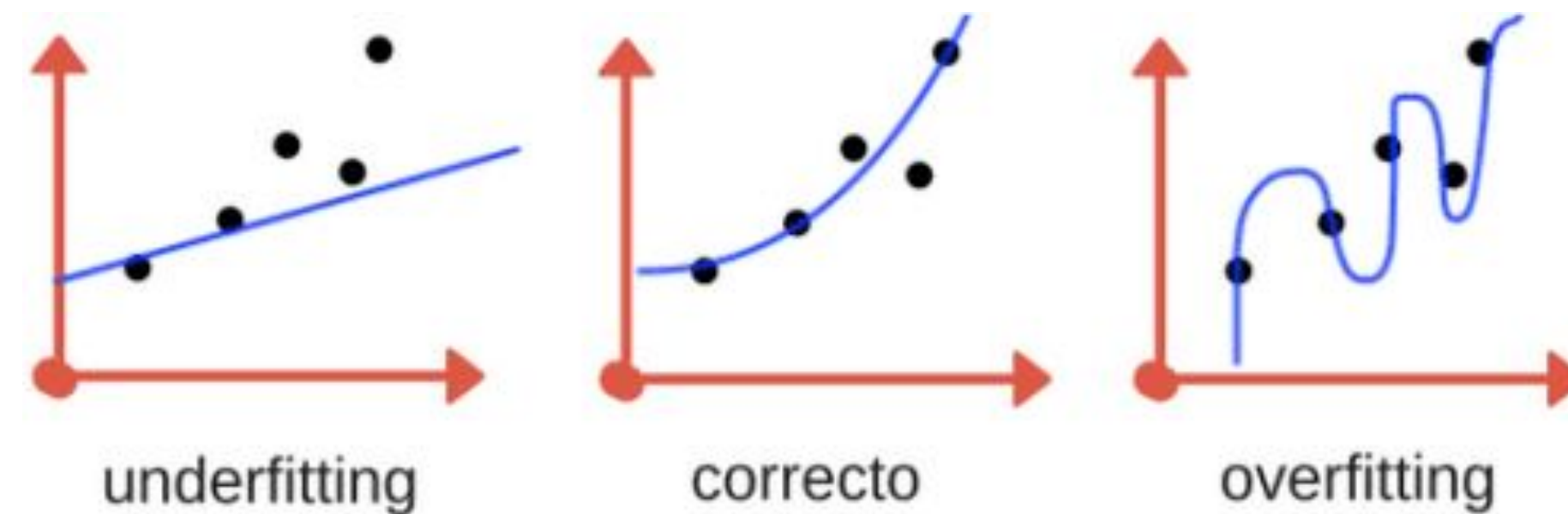
# Under- and over- fitting

По функции ошибок и метрикам качества легко понять, что происходит с алгоритмом.

- До тех пор, пока **качество и на обучающей, и на тестовой выборках растёт**, модель **обучается**.
- Как только **качество на тесте начинает стабильно падать**, это **переобучение**, нужно прекращать.



# Несмещенная оценка



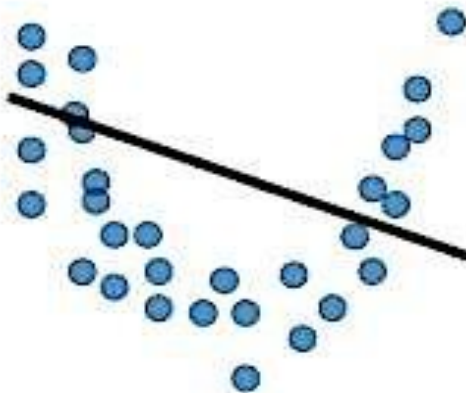
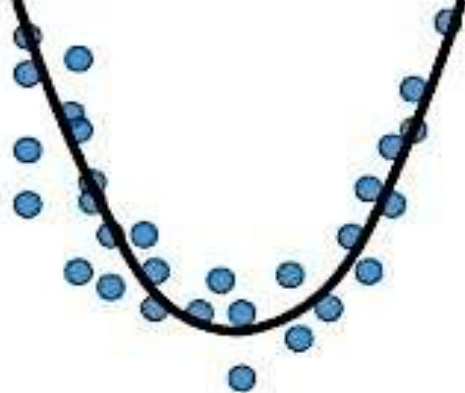
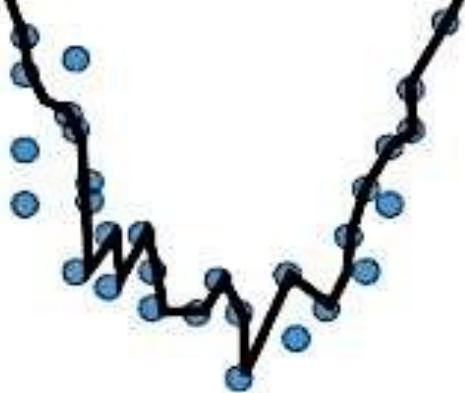
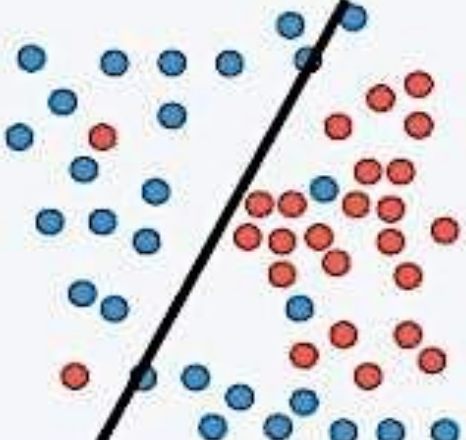
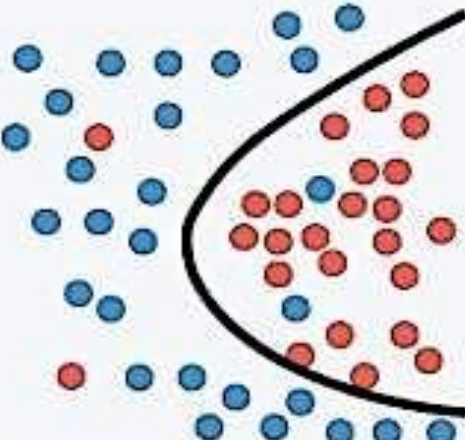
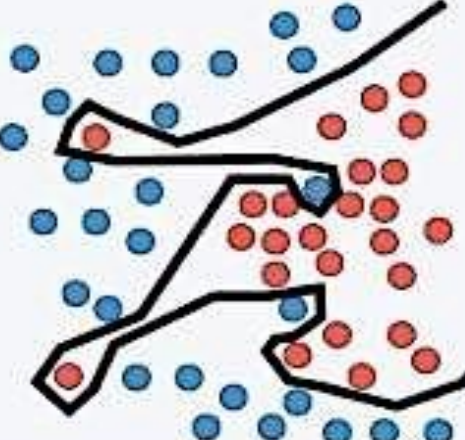

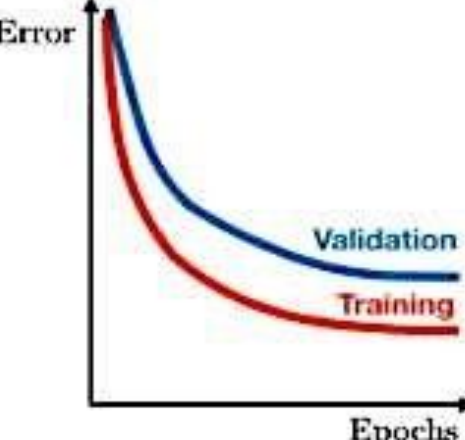

**Вопрос:** какое предсказание лучше по метрикам, а какое на самом деле?

Если тестировать модель на той же выборке, на которой она обучалась, то оценка получится смещенной. В таком случае "самая лучшая" модель - это та, которая просто запомнила все данные.

Хорошая модель должна делать хорошие предсказания на **НОВЫХ** для себя данных



# Under- and over-fitting

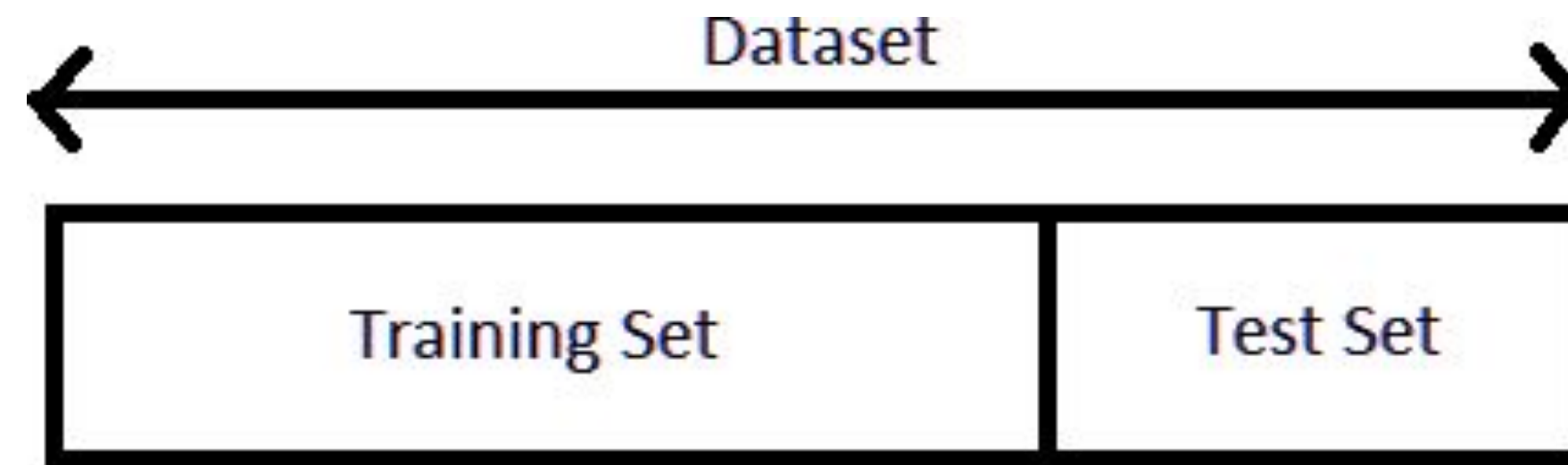
	Underfitting	Just right	Overfitting
Symptoms	<ul style="list-style-type: none"><li>• High training error</li><li>• Training error close to test error</li><li>• High bias</li></ul>	<ul style="list-style-type: none"><li>• Training error slightly lower than test error</li></ul>	<ul style="list-style-type: none"><li>• Very low training error</li><li>• Training error much lower than test error</li><li>• High variance</li></ul>
Regression illustration			
Classification illustration			
Deep learning illustration			
Possible remedies	<ul style="list-style-type: none"><li>• Complexify model</li><li>• Add more features</li><li>• Train longer</li></ul>		<ul style="list-style-type: none"><li>• Perform regularization</li><li>• Get more data</li></ul>



# Отложенная выборка

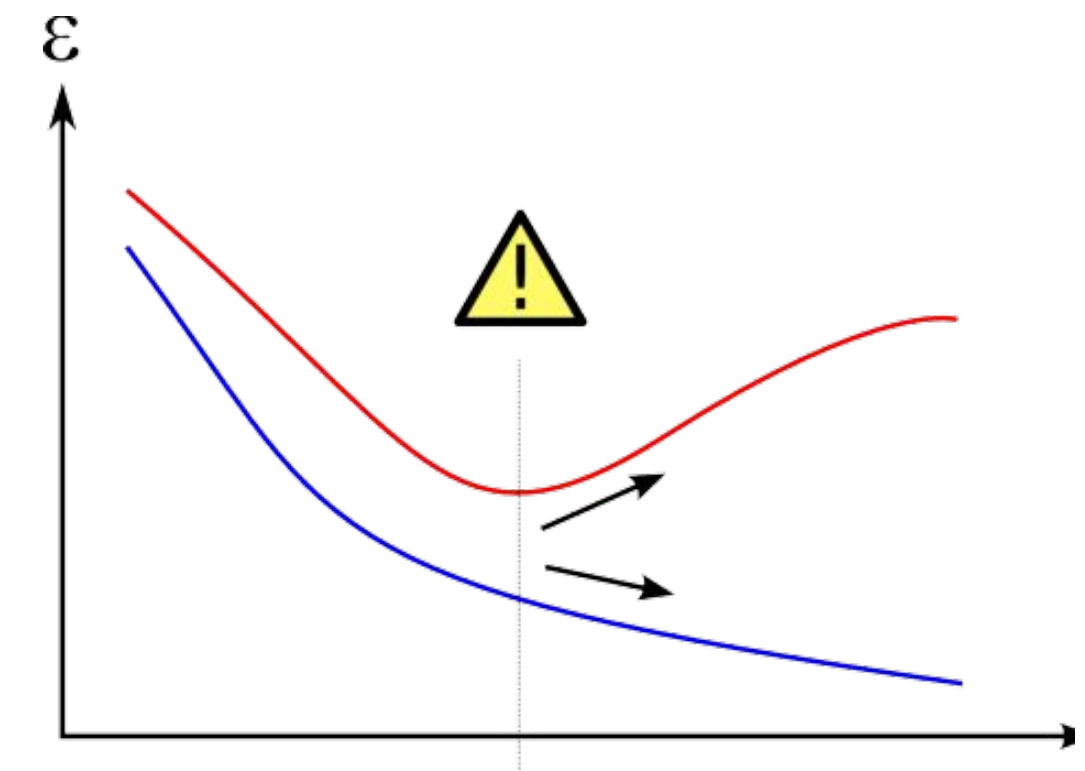
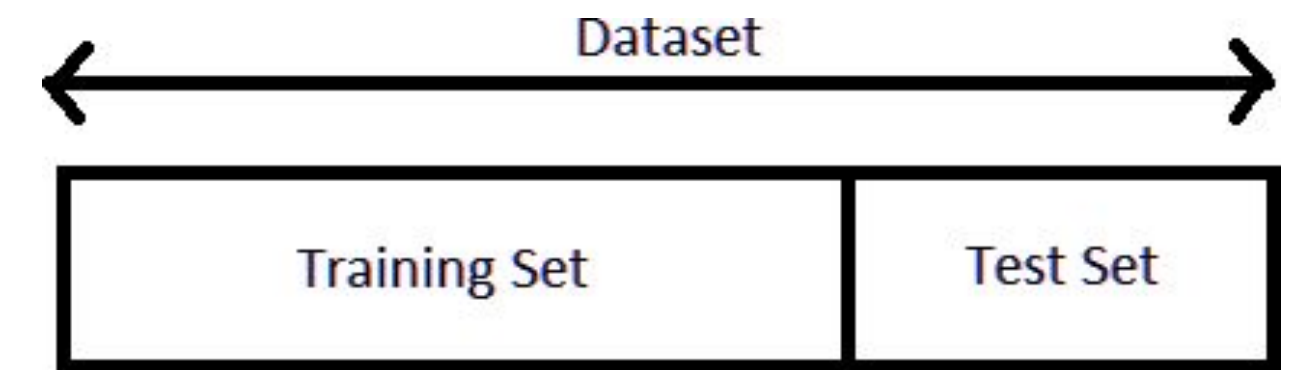
Можно "отложить", скажем, 20% обучающей выборки для валидации модели. Использовать 80% выборки для обучения и 20% для тестирования.

- Оценка на тестовой выборке будет несмещенной
- Тестовая выборка маленькая - оценка будет иметь погрешность



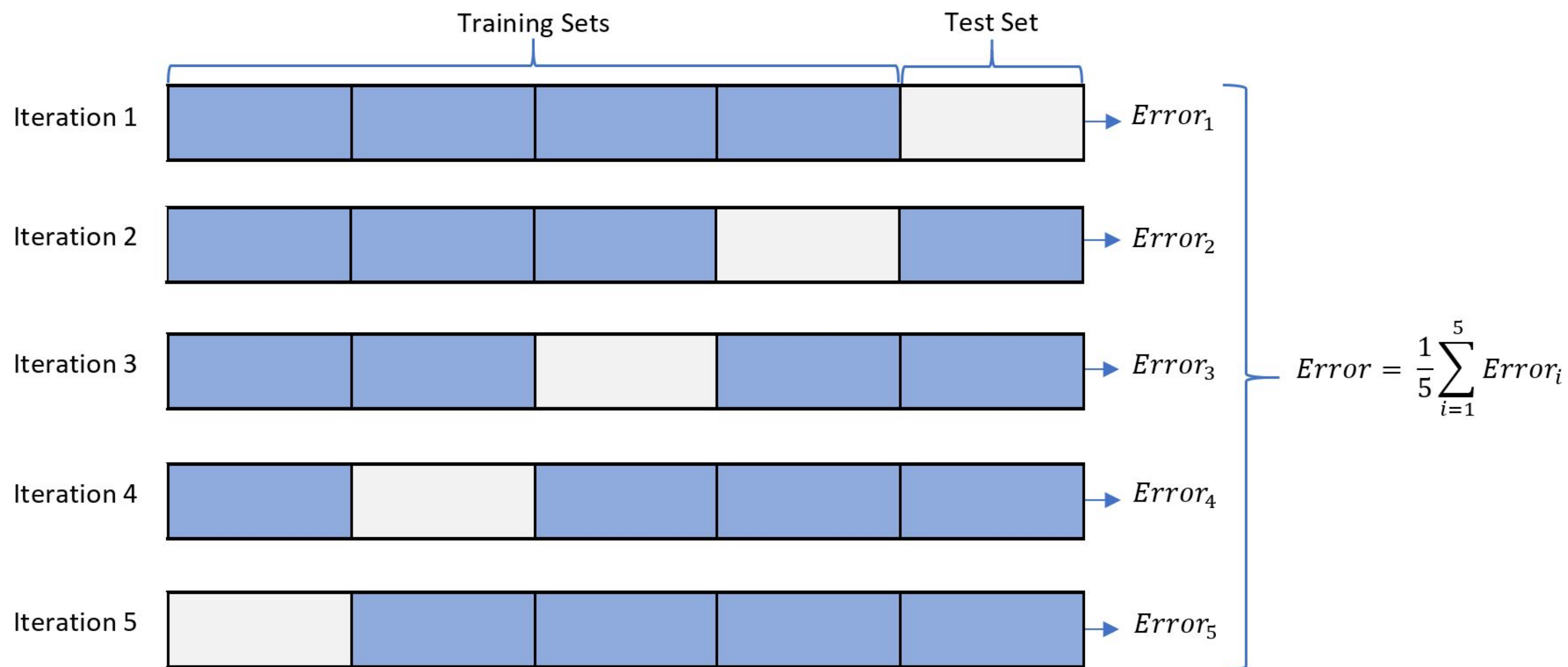
# Переобучение

- Как обнаружить? - Train/Test split
  - Разделить выборку на обучающую и контрольную
  - Следить за качеством на контрольной выборке
- Минусы?
  - Уменьшение размера обучающей выборки может негативно сказаться на качестве
  - Малый размер тестовой выборки может давать сильное смещение оценки.
  - Можно переобучиться под **тестовую выборку**



# Cross-validation

Если хочется оценить качество алгоритма совсем честно, можно посчитать метрики по **кросс-валидации** (её тоже можно делать со стратификацией).



# Cross-validation

- Разбиваем выборку на  $k$  частей
- $k-1$  частей используются для обучения и одна - для тестирования
- Процесс повторяется  $k$  раз. Каждый раз для тестирования выбирается разная часть
- Результаты тестирования усредняются

## Плюсы:

- Погрешность оценки уменьшается, т.к. используется весь набор

## Минусы:

- Обучение производится  $k$  раз. Для некоторых моделей это может быть очень долго

