

# Метрические методы

k-Nearest Neighbors (kNN)

# Метрические методы

## Идея

**Гипотеза непрерывности** (для регрессии):  
близким объектам соответствуют близкие ответы.

**Гипотеза компактности** (для классификации):  
близкие объекты, как правило, лежат в одном классе.

**Формализация понятия «близости»:**  
задана функция расстояния  $\rho: X \times X \rightarrow [0, \infty)$ .

# План

- Метод k ближайших соседей
- Настройка параметров в kNN
- Метрики в kNN
- Проклятие размерности
- kNN в scikit-learn

# План

- Метод k ближайших соседей
- Настройка параметров в kNN
- Метрики в kNN
- Проклятие размерности
- kNN в scikit-learn

# Метод k ближайших соседей (kNN)

## Метрические алгоритмы

- › Метрические алгоритмы: в пространстве признаков введено понятие метрики
- › Начнём с метода ближайшего соседа в задаче классификации

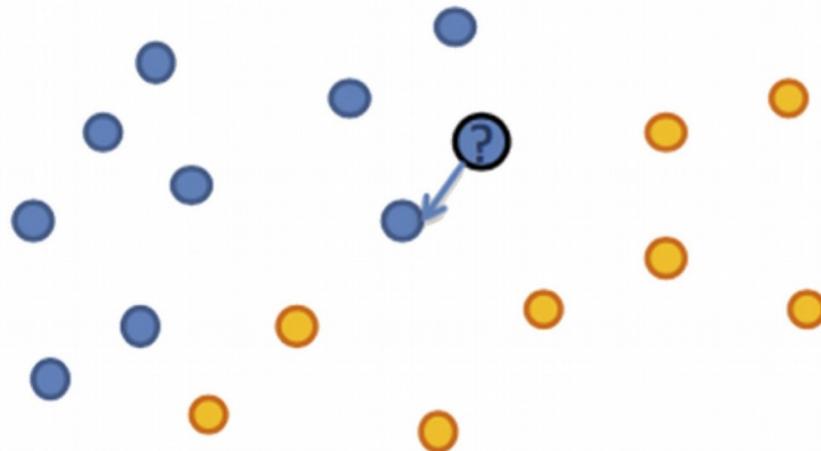
# Метод k ближайших соседей (kNN)

Метод ближайшего соседа



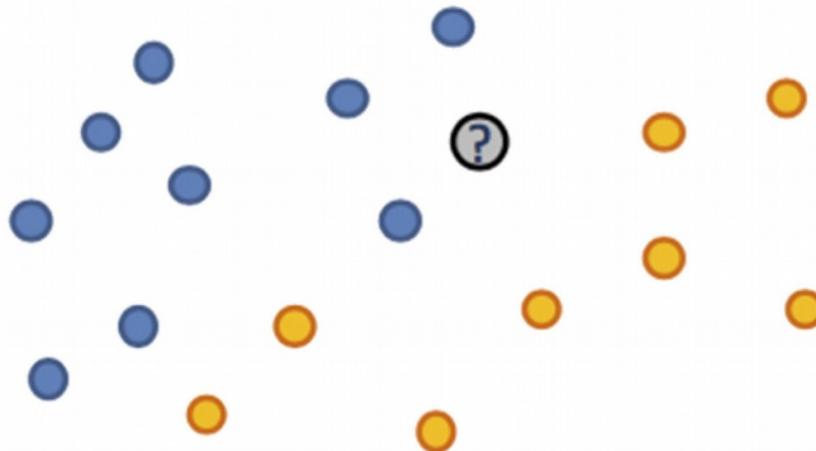
# Метод k ближайших соседей (kNN)

Метод ближайшего соседа



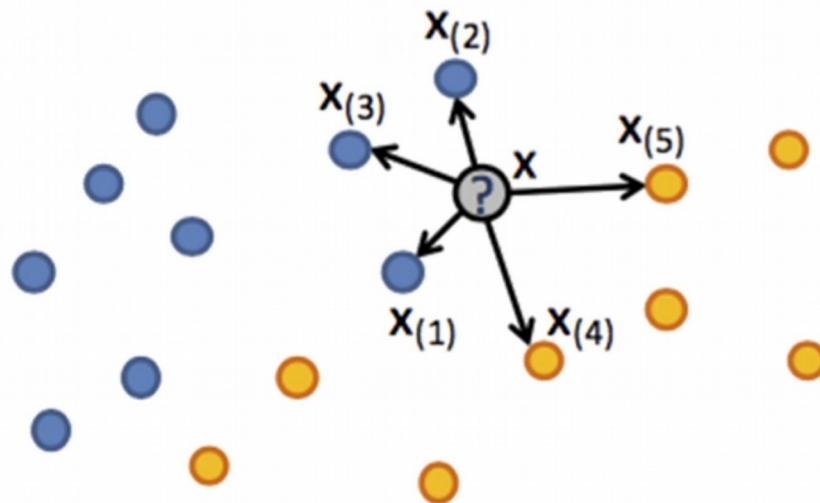
# Метод k ближайших соседей (kNN)

Пример классификации ( $k = 5$ )



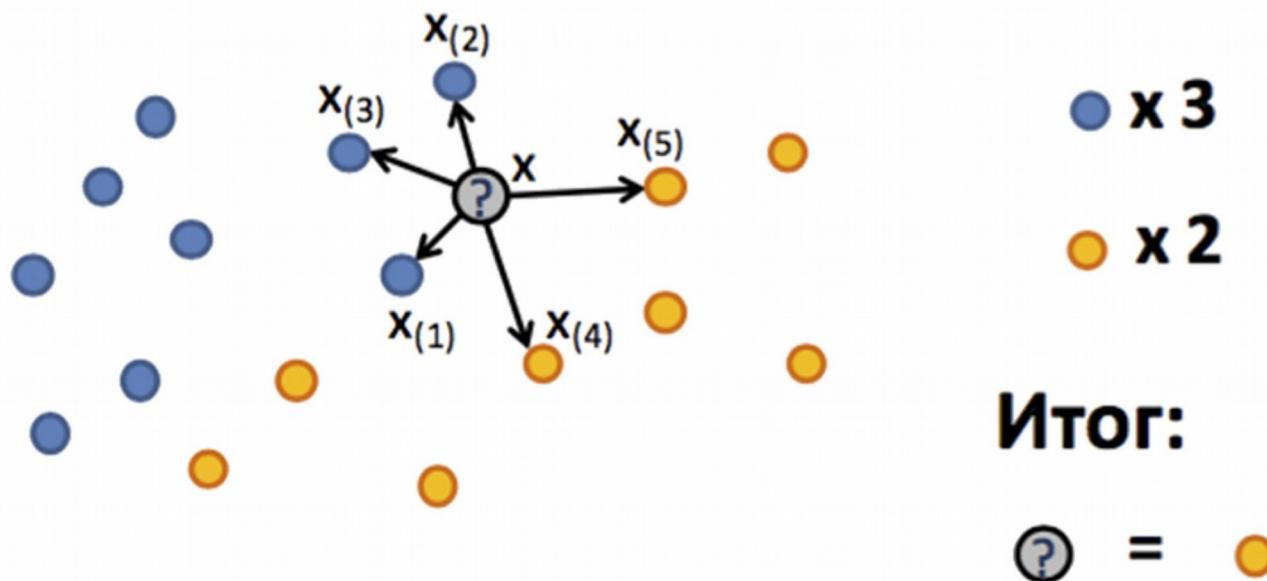
# Метод k ближайших соседей (kNN)

Пример классификации ( $k = 5$ )



# Метод k ближайших соседей (kNN)

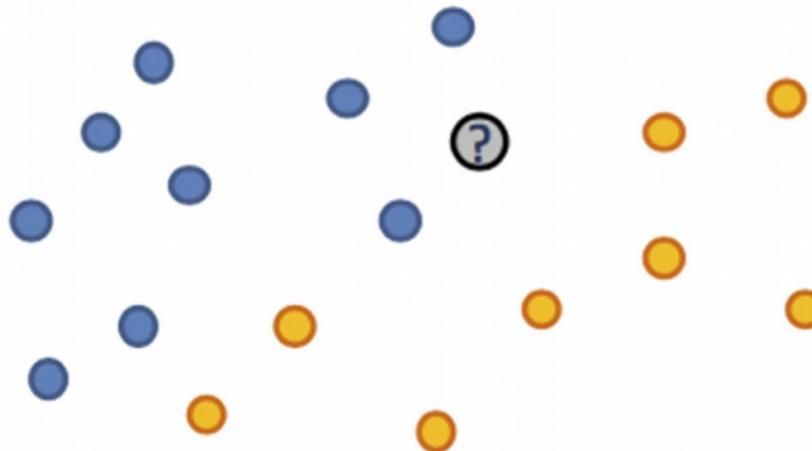
Пример классификации ( $k = 5$ )



# Метод k ближайших соседей (kNN)

## Взвешенный kNN

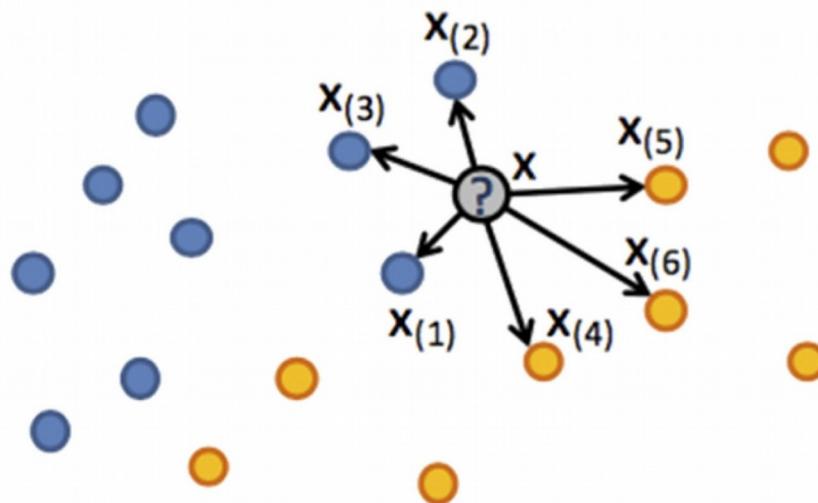
Пример классификации ( $k = 6$ )



# Метод k ближайших соседей (kNN)

## Взвешенный kNN

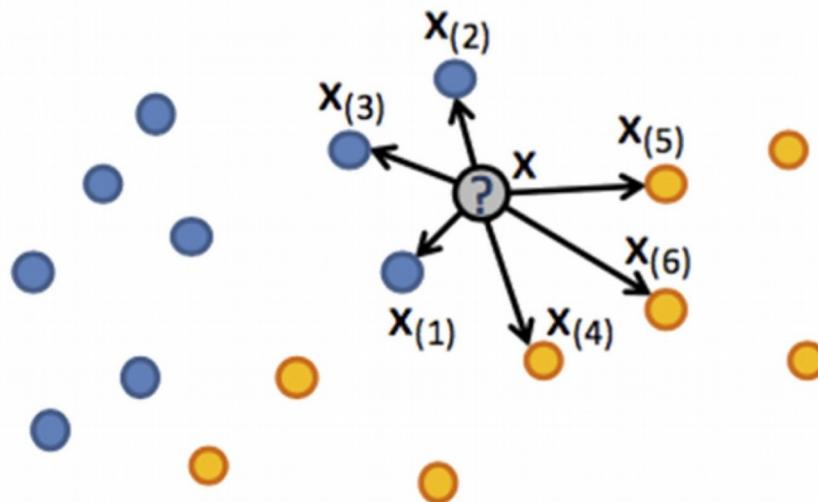
Пример классификации ( $k = 6$ )



# Метод k ближайших соседей (kNN)

## Взвешенный kNN

Пример классификации ( $k = 6$ )

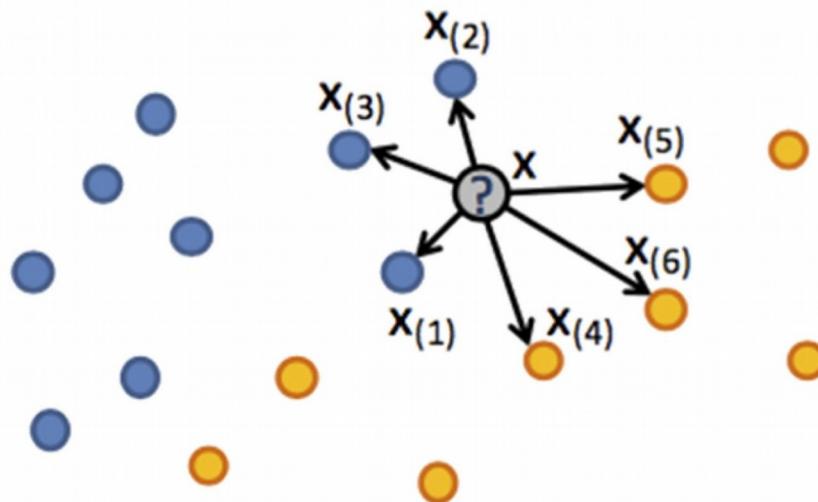


Веса можно определять как функцию от соседа или его номера:  $w(x_{(i)}) = w(i)$

# Метод k ближайших соседей (kNN)

## Взвешенный kNN

Пример классификации ( $k = 6$ )



Веса можно определять как функцию от соседа или его номера:  $w(x_{(i)}) = w(i)$

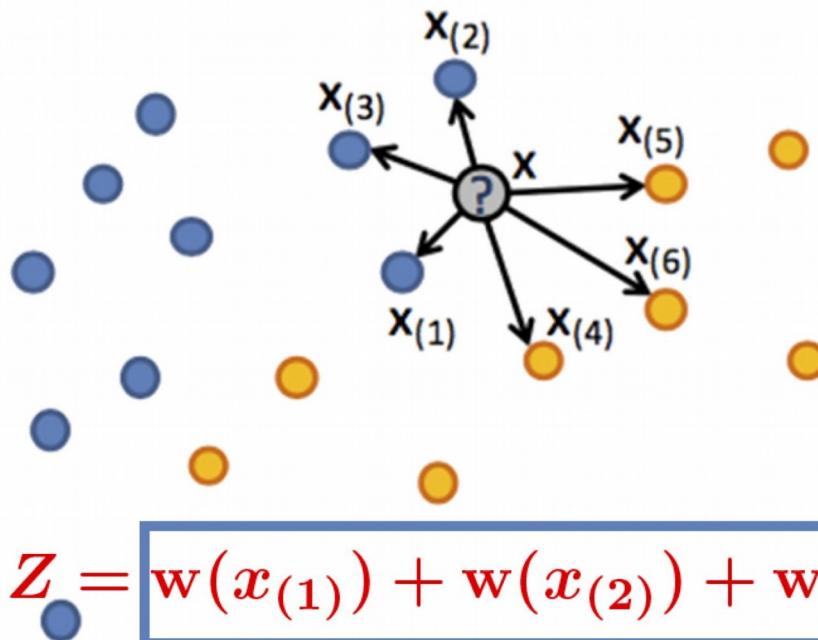
Или как функцию расстояния:

$$w(x(i)) = w(d(x, x_{(i)}))$$

# Метод k ближайших соседей (kNN)

## Взвешенный kNN

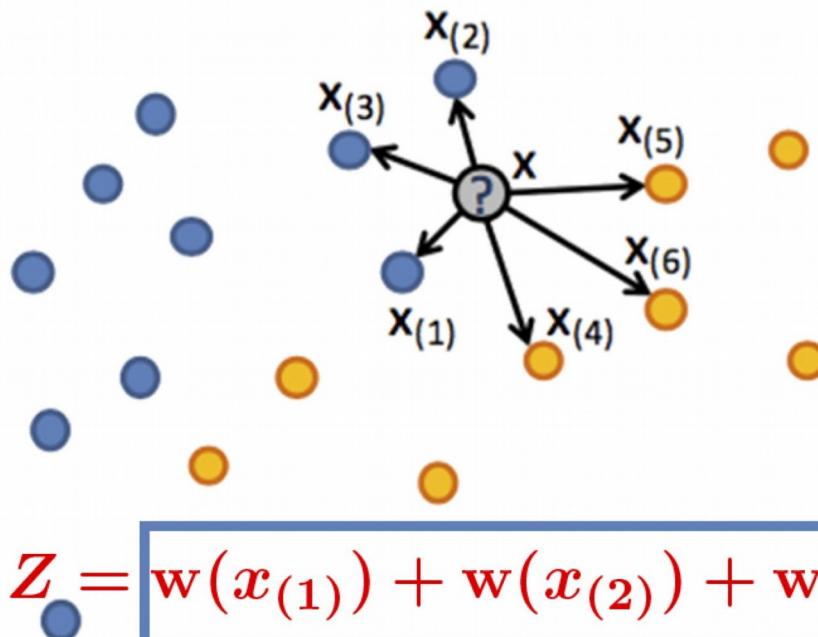
Пример классификации ( $k = 6$ )



# Метод k ближайших соседей (kNN)

## Взвешенный kNN

Пример классификации ( $k = 6$ )

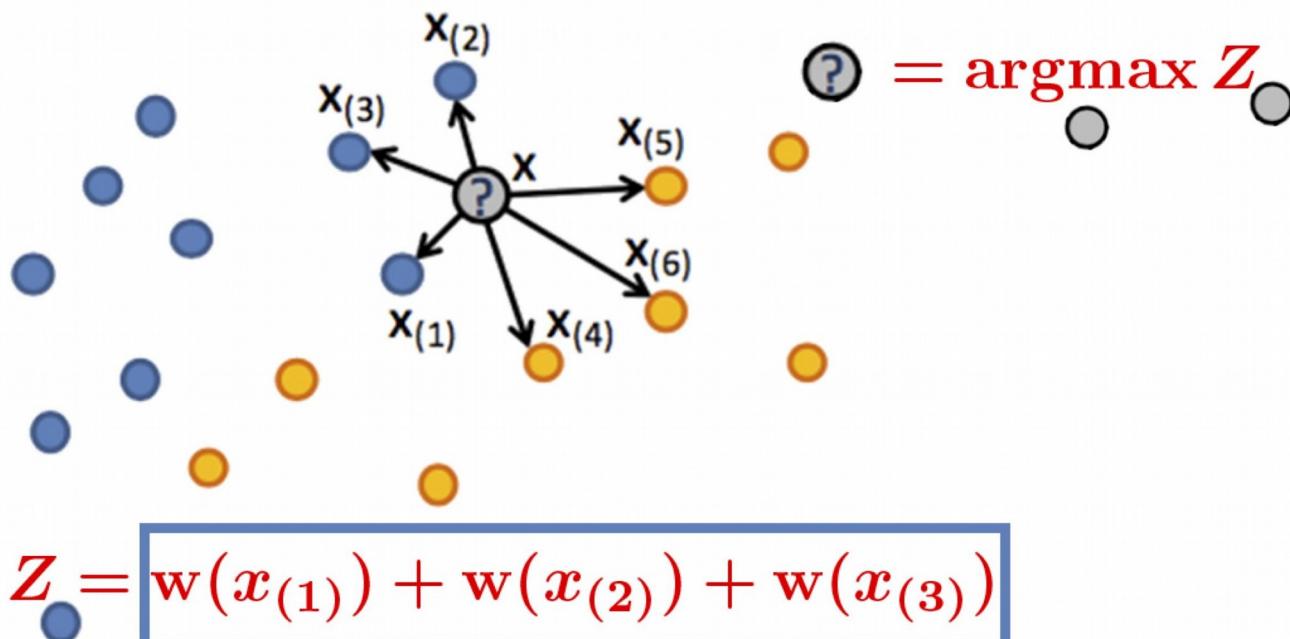


$$Z = w(x_{(4)}) + w(x_{(5)}) + w(x_{(6)})$$

# Метод k ближайших соседей (kNN)

## Взвешенный kNN

Пример классификации ( $k = 6$ )

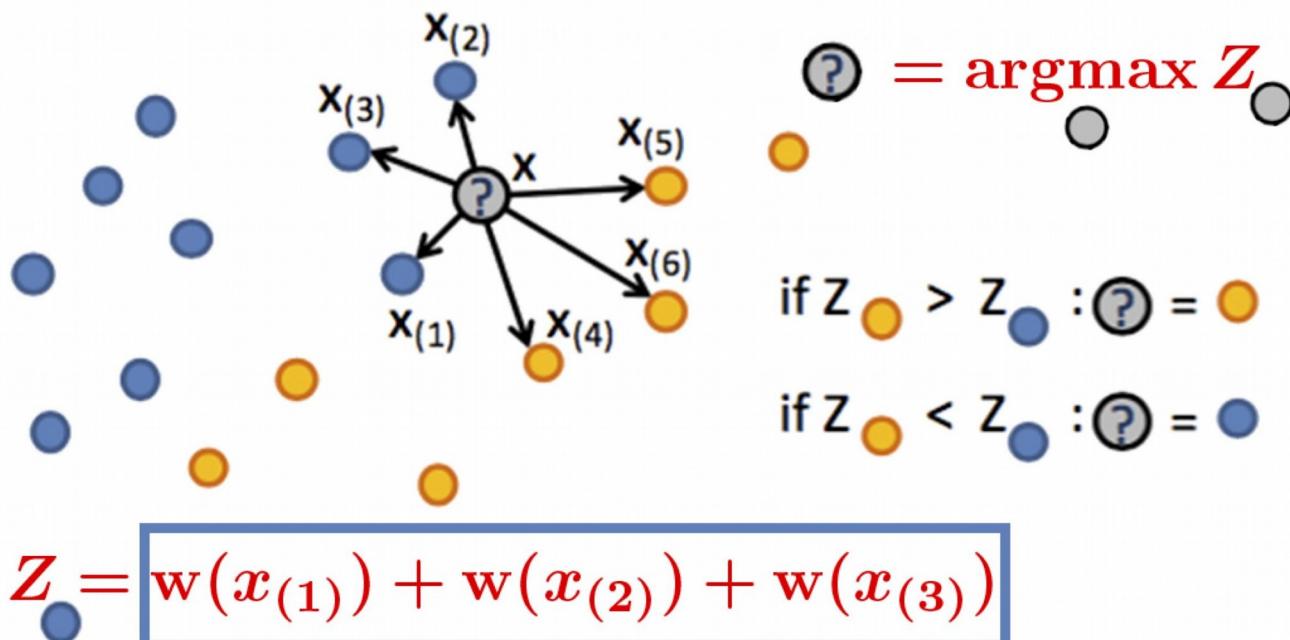


$$Z_1 = w(x_{(4)}) + w(x_{(5)}) + w(x_{(6)})$$

# Метод k ближайших соседей (kNN)

## Взвешенный kNN

Пример классификации ( $k = 6$ )



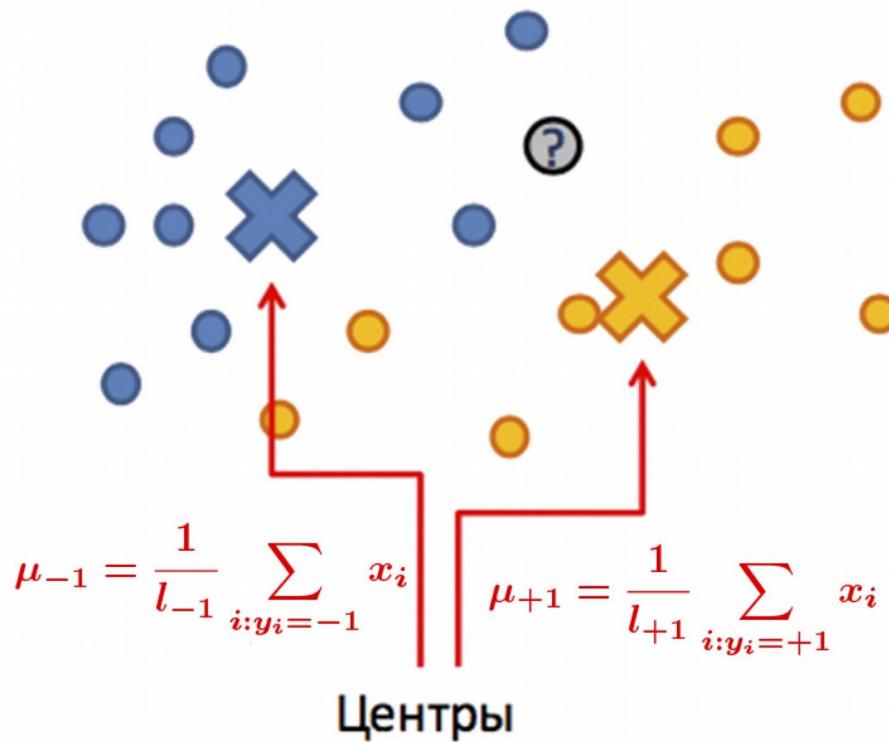
# Метод k ближайших соседей (kNN)

## Центроидный классификатор



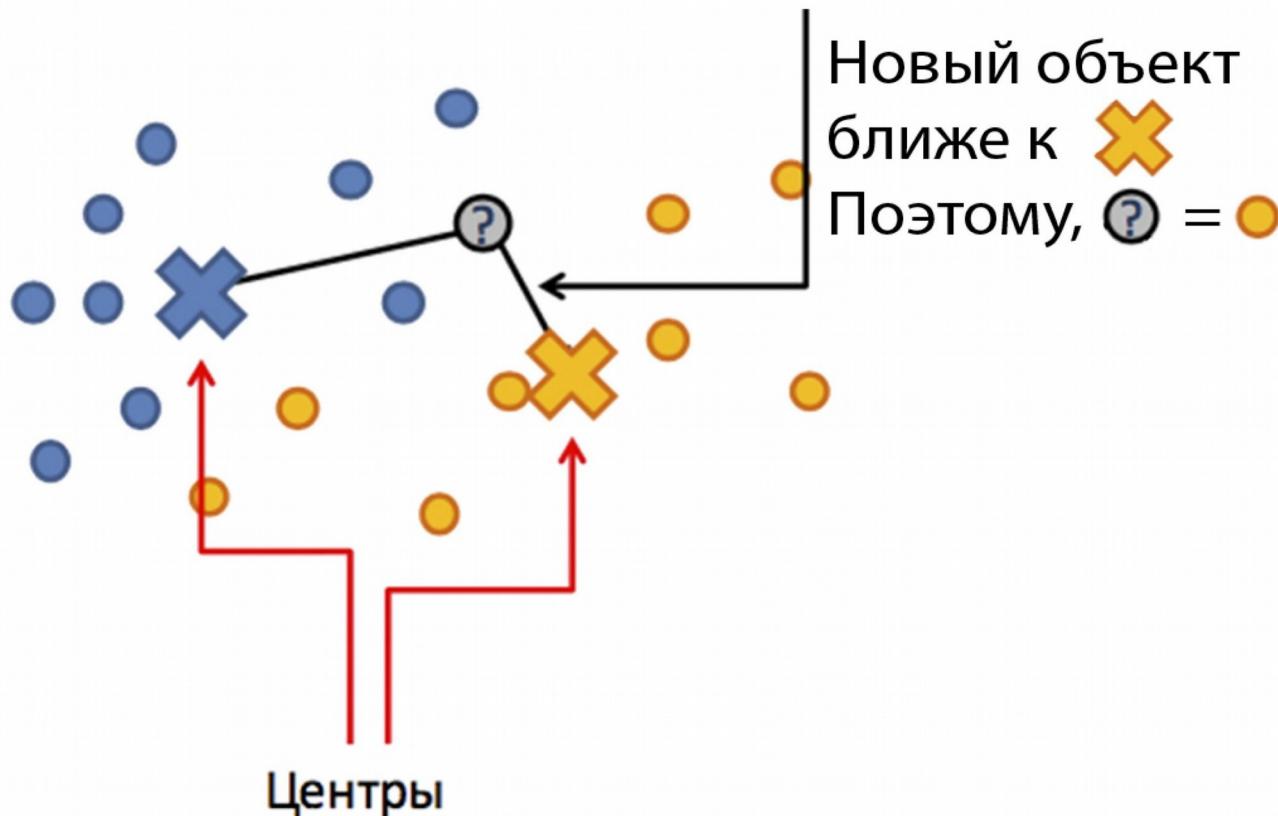
# Метод k ближайших соседей (kNN)

## Центроидный классификатор



# Метод k ближайших соседей (kNN)

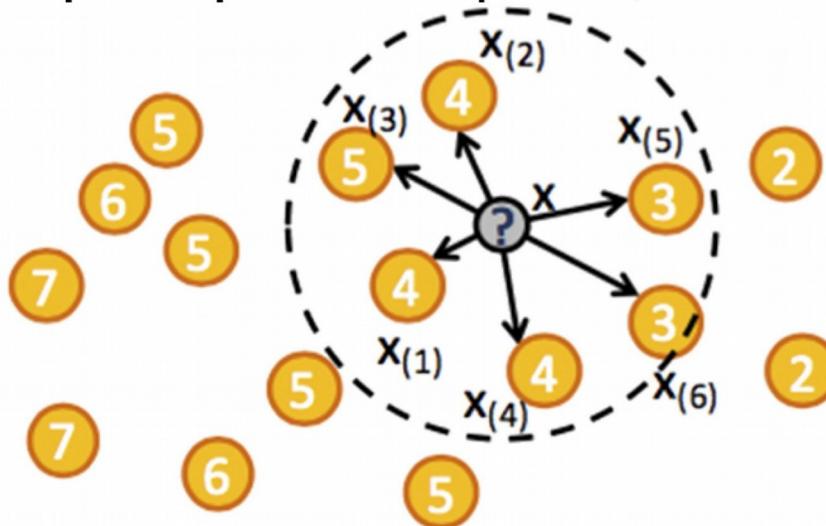
## Центроидный классификатор



# Метод k ближайших соседей (kNN)

## Взвешенный kNN для регрессии

Пример классификации ( $k = 6$ )



Веса можно определять как функцию от соседа или его номера:  $w(x_{(i)}) = w(i)$

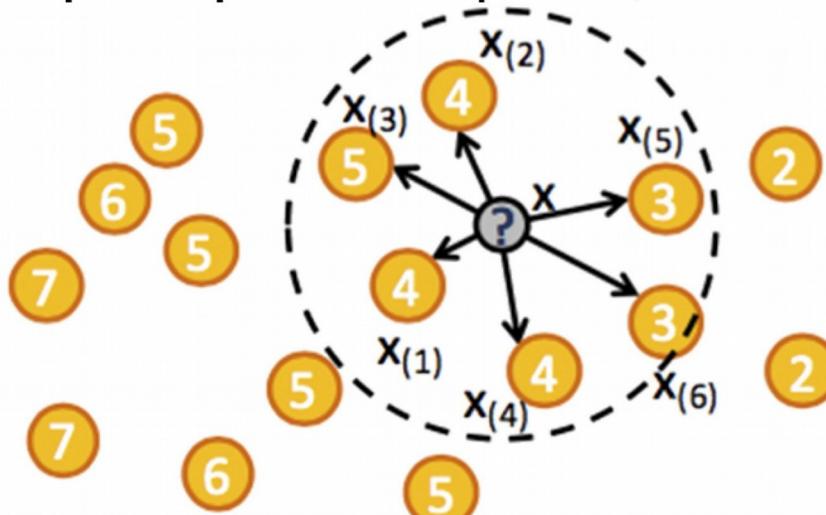
Или как функцию расстояния:

$$w(x(i)) = w(d(x, x_{(i)}))$$

# Метод k ближайших соседей (kNN)

## Взвешенный kNN для регрессии

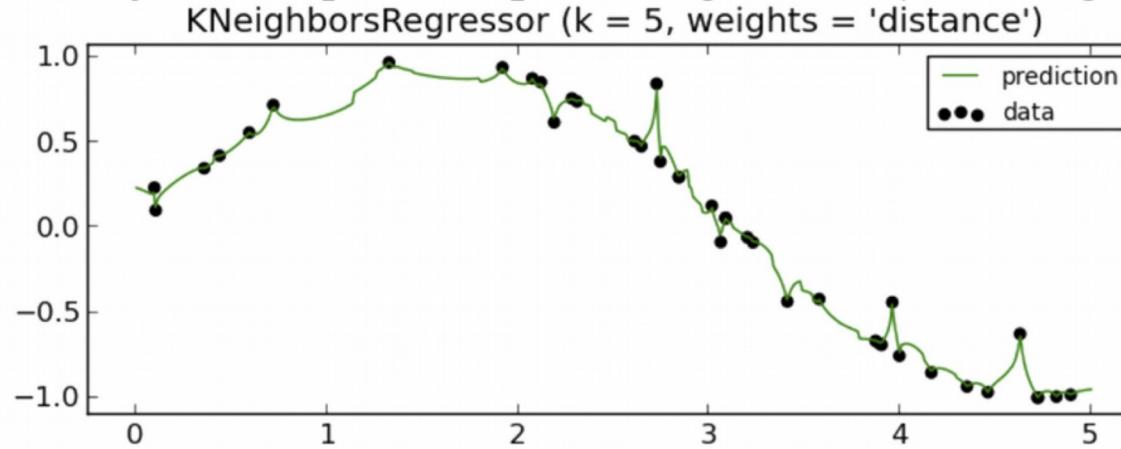
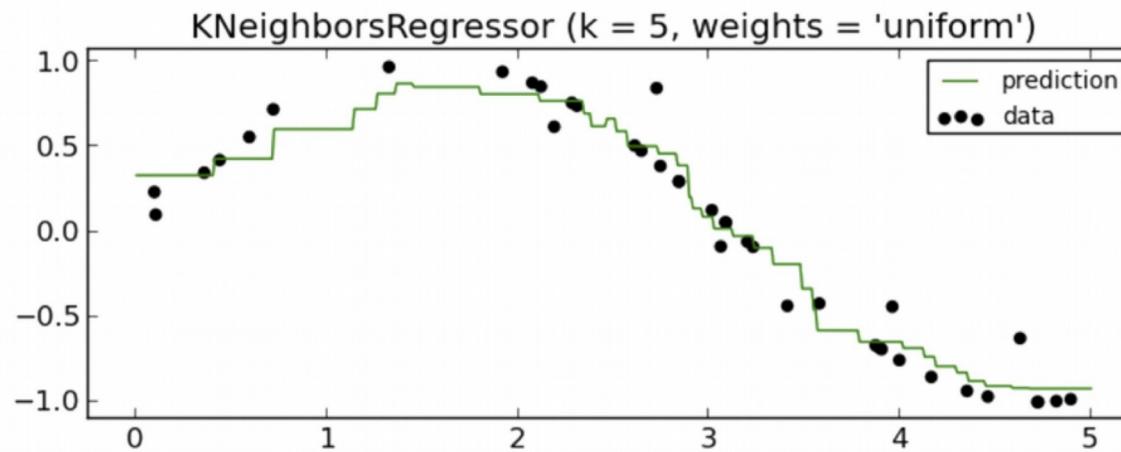
Пример классификации ( $k = 6$ )



$$\begin{aligned} \textcircled{?} = & \frac{4 w(x_{(1)}) + 4 w(x_{(2)}) + 5 w(x_{(3)}) +}{w(x_{(1)}) + w(x_{(2)}) + w(x_{(3)}) +} \\ & + \frac{4 w(x_{(4)}) + 3 w(x_{(5)}) + 3 w(x_{(6)})}{w(x_{(4)}) + w(x_{(5)}) + w(x_{(6)})} \end{aligned}$$

# Метод k ближайших соседей (kNN)

## Выбор весов в kNN



# Метод $k$ ближайших соседей (kNN)

## Выбор весов в kNN

- › Метод  $k$  ближайших соседей
- › Веса соседей
- › kNN в задаче классификации и в задаче регрессии

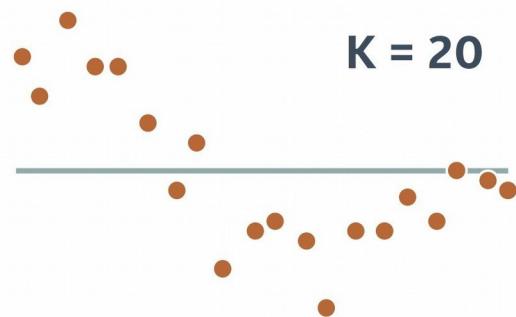
# Настройка параметров в kNN

- › У kNN нет параметров, настраиваемых на обучающей выборке
- › Но есть параметры, выбираемые исследователем
- › Количество соседей, веса объектов и расстояние
- › Эти параметры можно подобрать

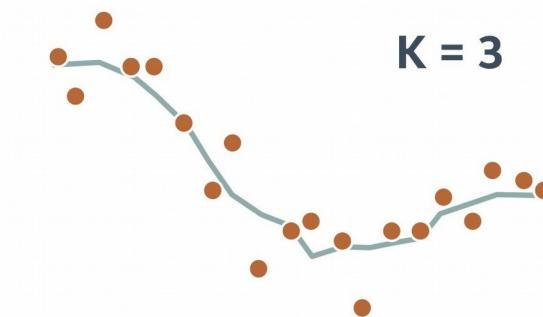
# Метрики в kNN

## Количество соседей

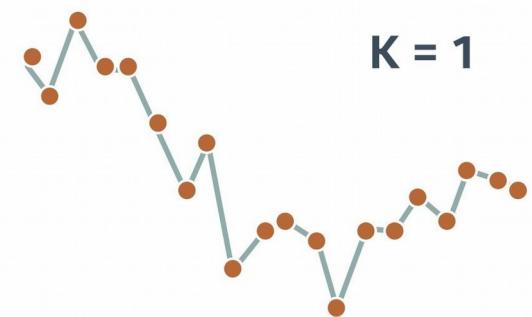
### REGRESSION WITH KNN



$K = 20$



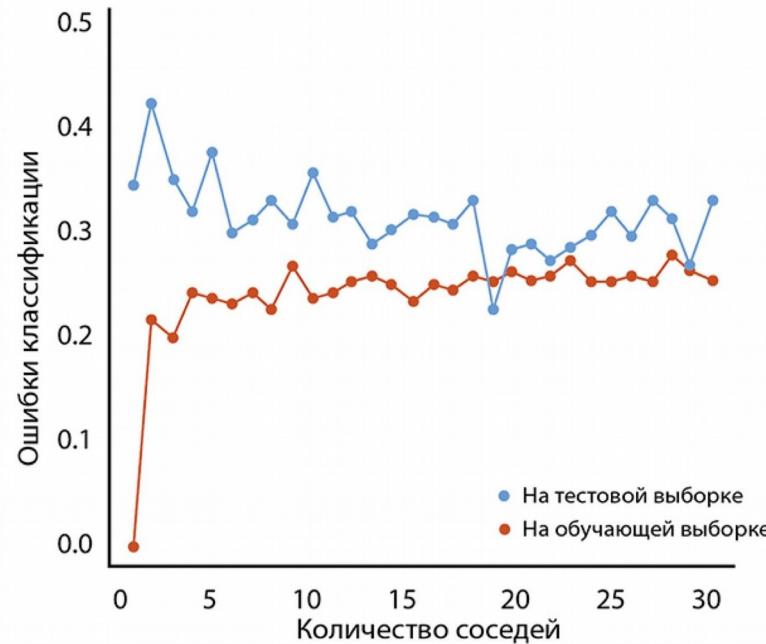
$K = 3$



$K = 1$

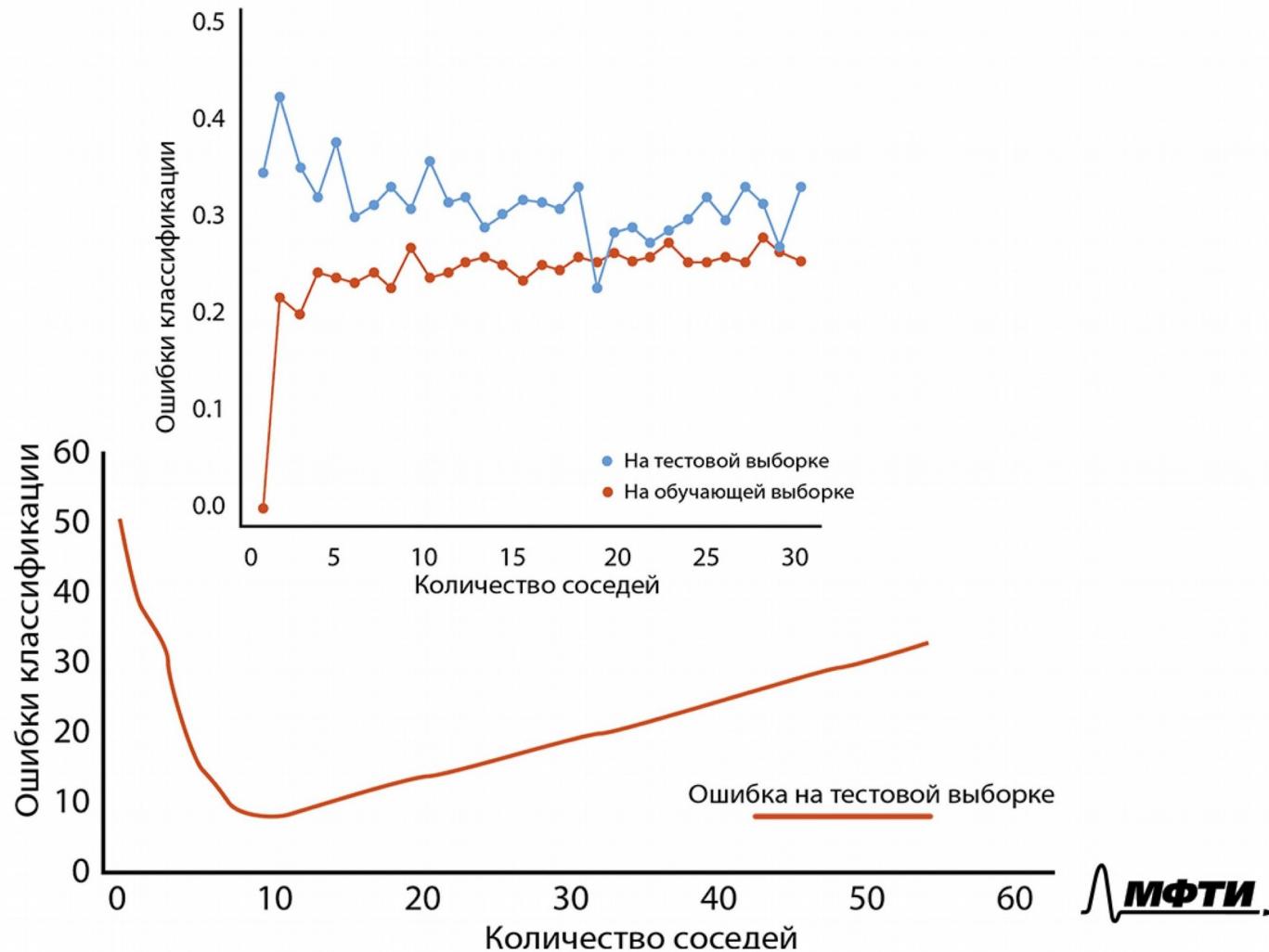
# Настройка параметров в kNN

## Количество соседей



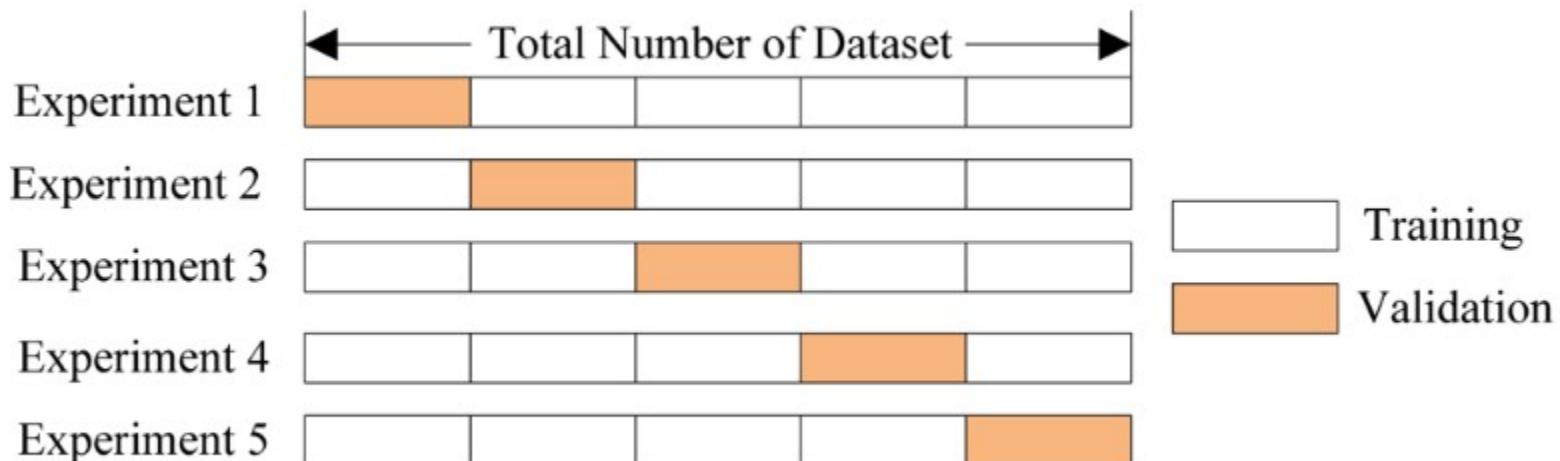
# Настройка параметров в kNN

## Количество соседей



# Настройка параметров в kNN

1. отложенная выборка (held-out/hold-out set)
2. кросс-валидация (cross-validation)



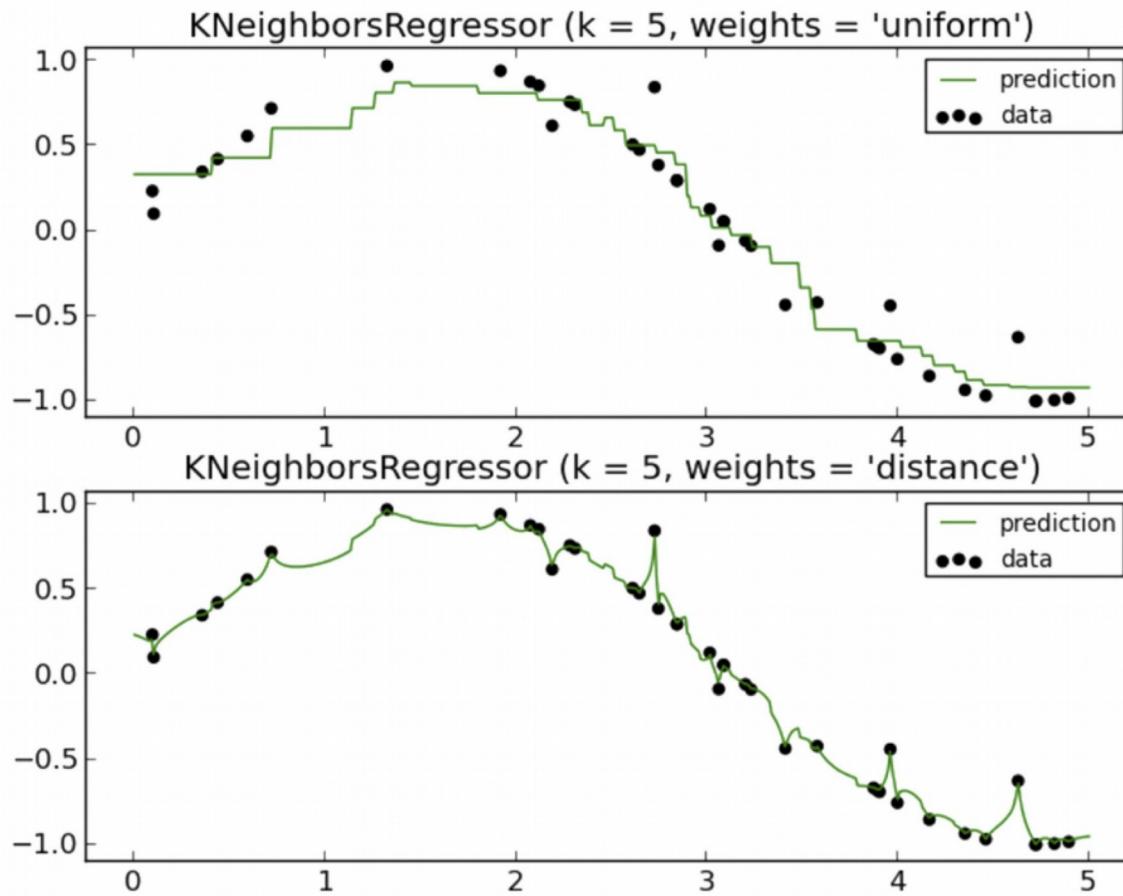
# Настройка параметров в kNN

## Веса соседей как функция от номера

- »  $w(i)$  должна не возрастать
- » Простейший вариант:  $w(i) = 1$
- »  $w(i) = q^i, \ 0 < q < 1$
- »  $w(i) = \frac{1}{i}, \ w(i) = \frac{1}{i+a}, \ w(i) = \frac{1}{(i+a)^b}$
- » Не очень удачный вариант:  $w(i) = 1 - \frac{i-1}{k}$

# Метод k ближайших соседей (kNN)

## Выбор весов в kNN



# Настройка параметров в kNN

## Веса соседей как функция от расстояния

- »  $w(d) = 1$
- »  $w(d) = \frac{1}{d}$
- »  $w(d) = \frac{1}{(q+a)^b}$
- »  $w(d) = q^d, 0 < q < 1$
- » Ядра

# Настройка параметров в kNN

## Веса соседей как функция от расстояния

- › В kNN можно настраивать количество соседей, функции весов объектов и метрику
- › Делать это нужно не на обучающей выборке, а по качеству в кросс-валидации
- › Уже обсудили: выбор количества соседей и весов объектов
- › Далее: метрики

# Метрики в kNN

Что такое метрика?

- »  $\rho(x, y) \geq 0$
- »  $\rho(x, y) = \rho(y, x)$
- »  $\rho(x, y) \leq \rho(x, z) + \rho(z, y)$

# Метрики в kNN

## Самое обычное расстояние

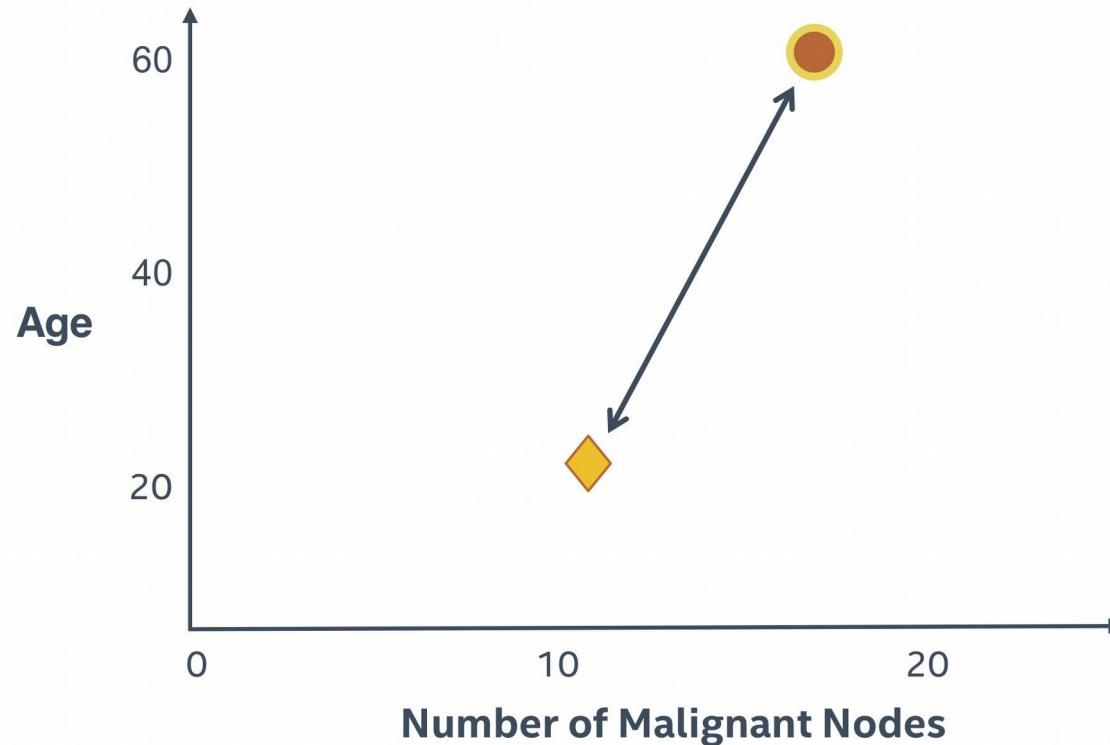
› Евклидова метрика:

$$\rho(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

# Метрики в kNN

## Эвклидово расстояние

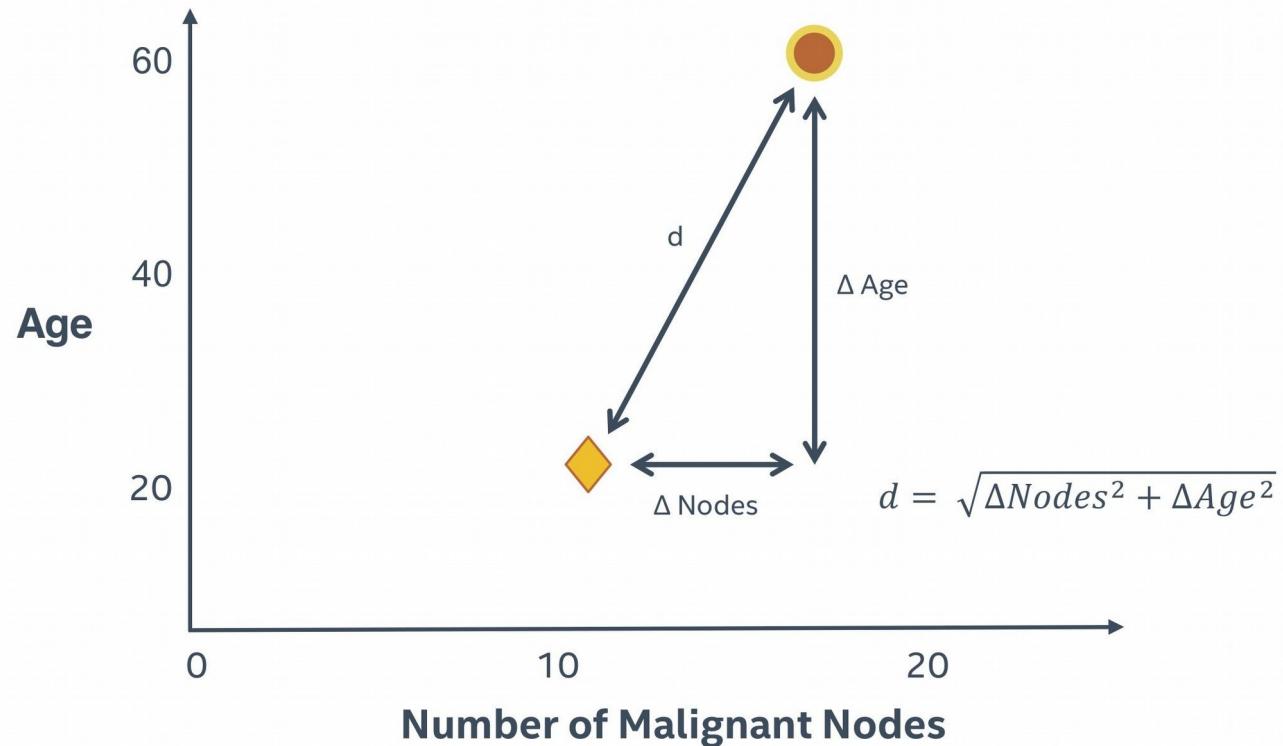
### EUCLIDEAN DISTANCE



# Метрики в kNN

## Эвклидово расстояние

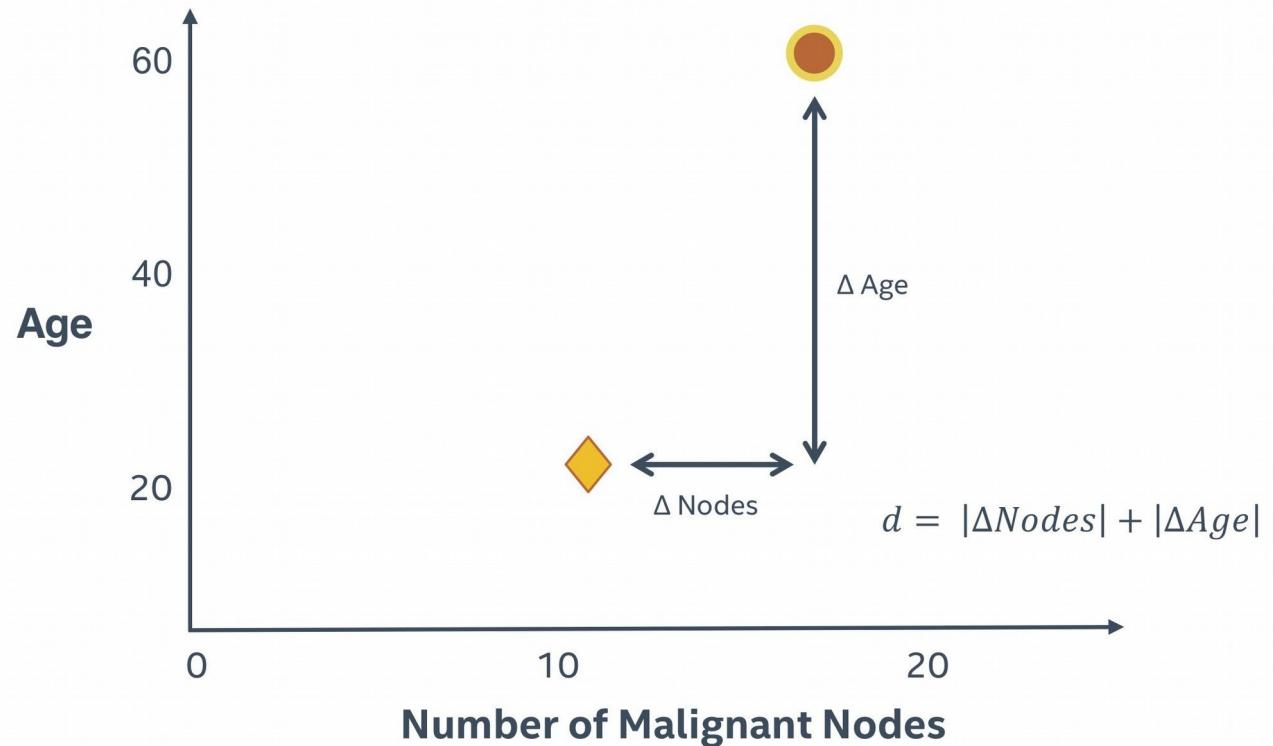
### EUCLIDEAN DISTANCE (L2 DISTANCE)



# Метрики в kNN

## Манхэттенское расстояние

MANHATTAN DISTANCE (L1 OR CITY BLOCK DISTANCE)



# Метрики в kNN

## Метрика Минковского

» Евклидова метрика:

$$\sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

» Манхэттенская метрика:

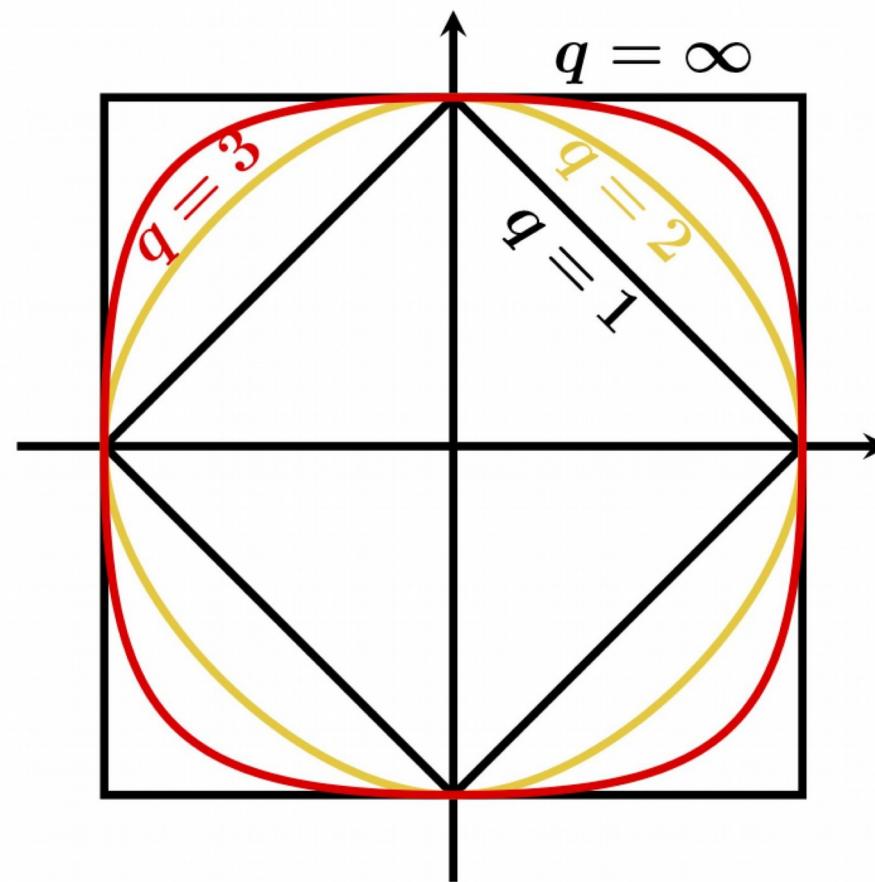
$$\sum_{i=1}^n |x_i - y_i|$$

» Метрика Минковского:

$$\left( \sum_{i=1}^n (|x_i - y_i|)^q \right)^{\frac{1}{q}}$$

# Метрики в kNN

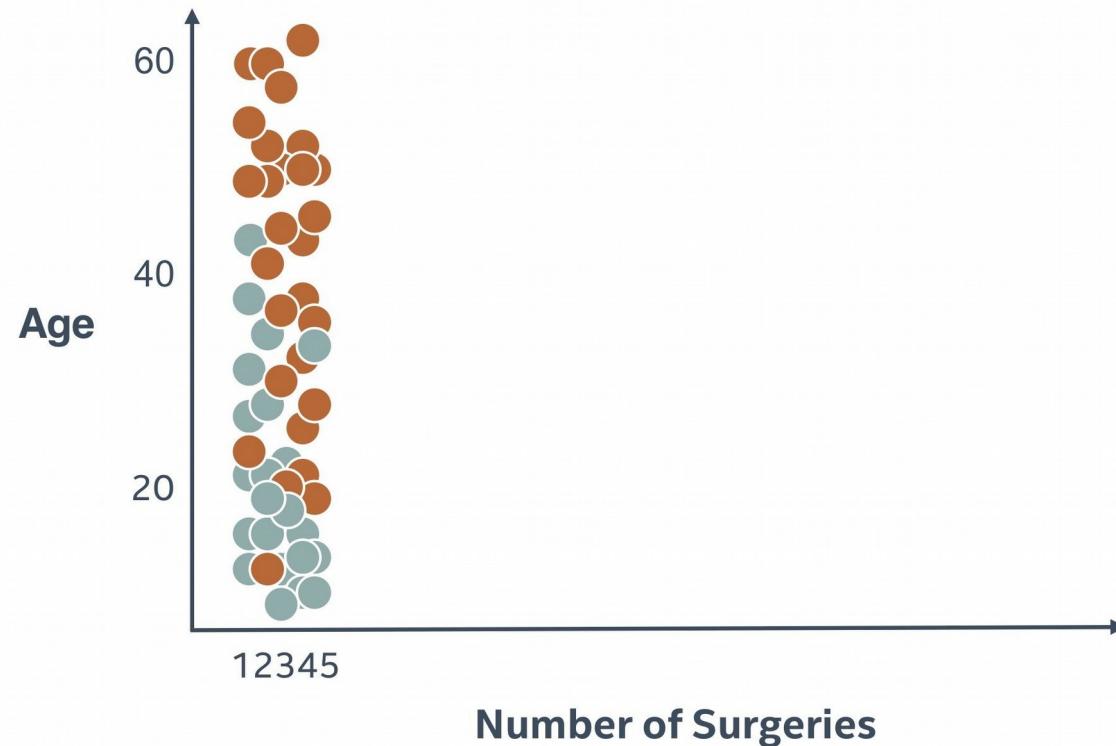
## Как “посмотреть” метрику



# Метрики в kNN

## Важность масштабирования

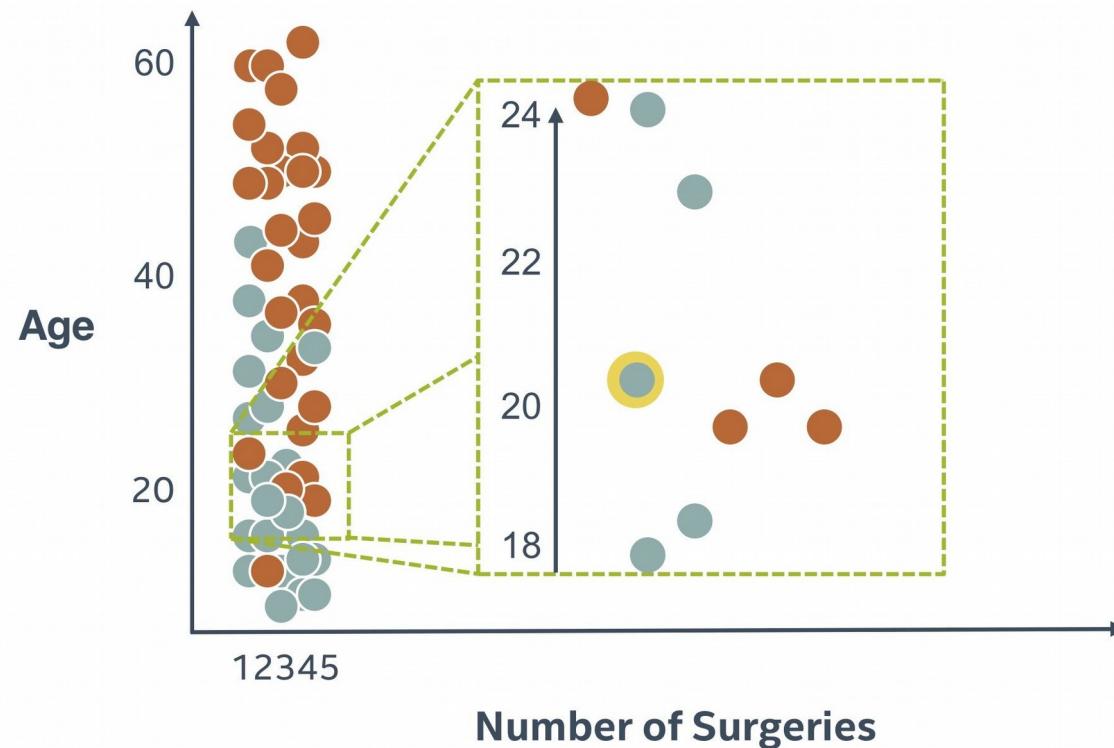
**SCALE IS IMPORTANT FOR DISTANCE MEASUREMENT**



# Метрики в kNN

## Важность масштабирования

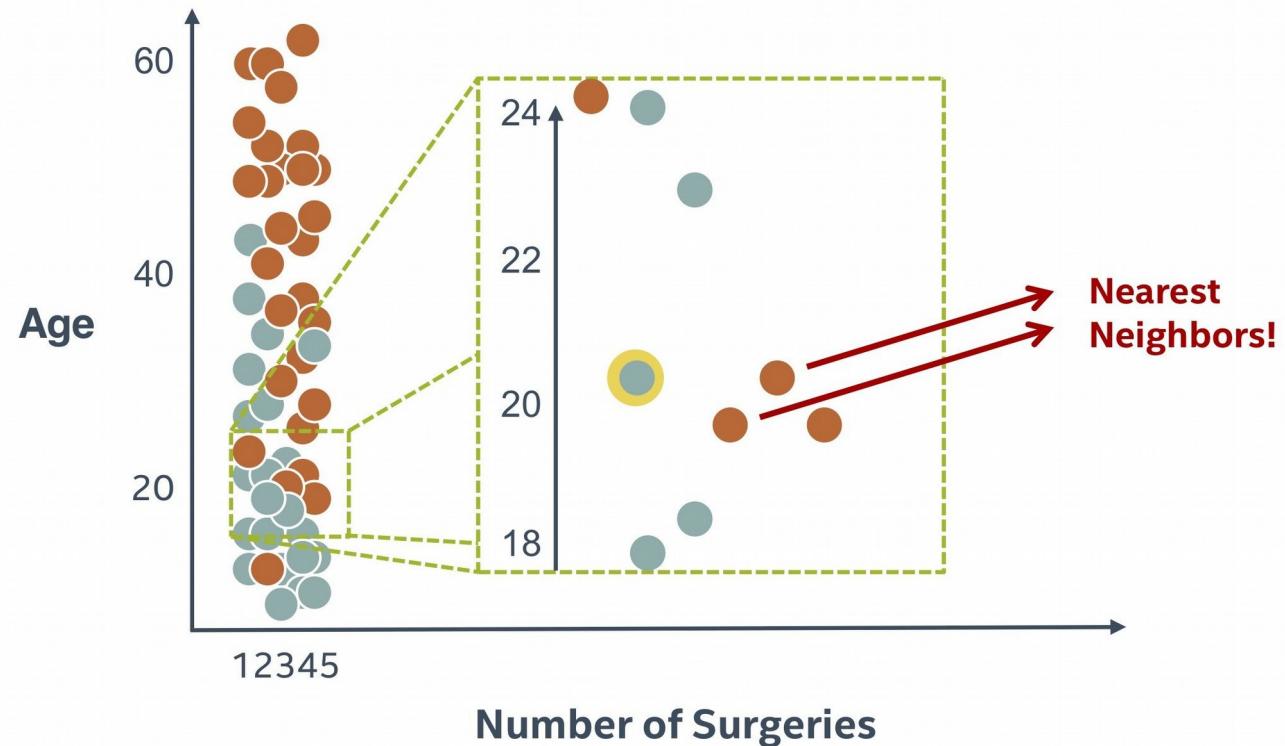
**SCALE IS IMPORTANT FOR DISTANCE MEASUREMENT**



# Метрики в kNN

## Важность масштабирования

SCALE IS IMPORTANT FOR DISTANCE MEASUREMENT

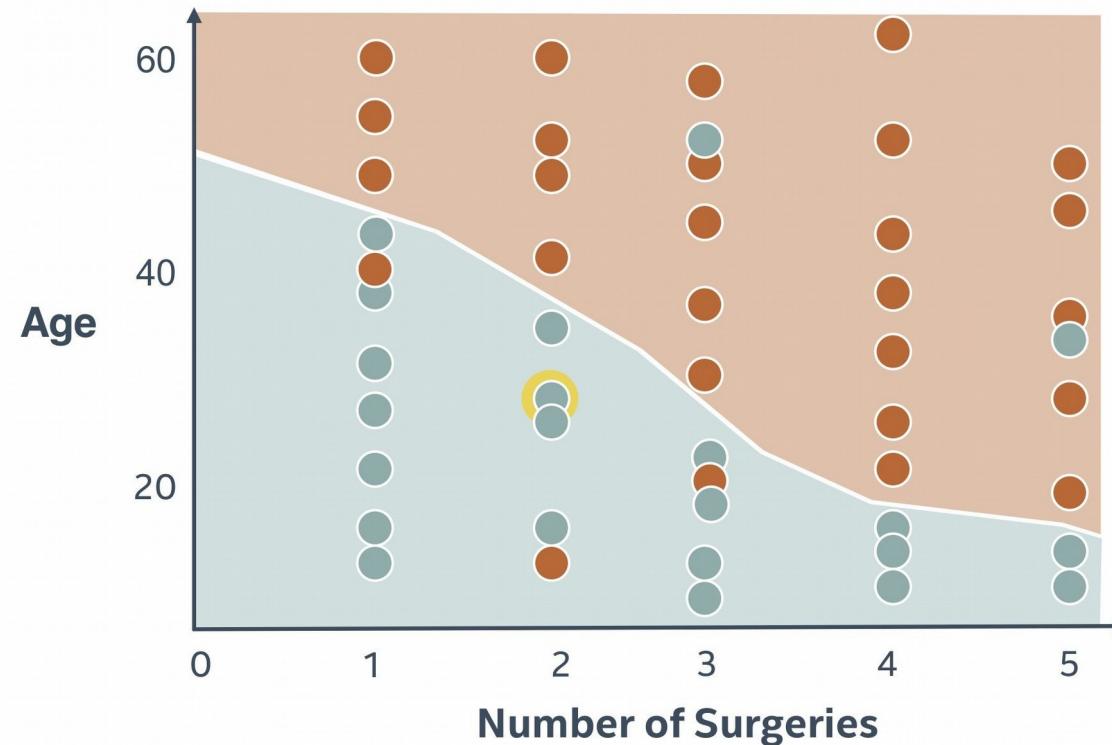


# Метрики в kNN

## Важность масштабирования

**SCALE IS IMPORTANT FOR DISTANCE MEASUREMENT**

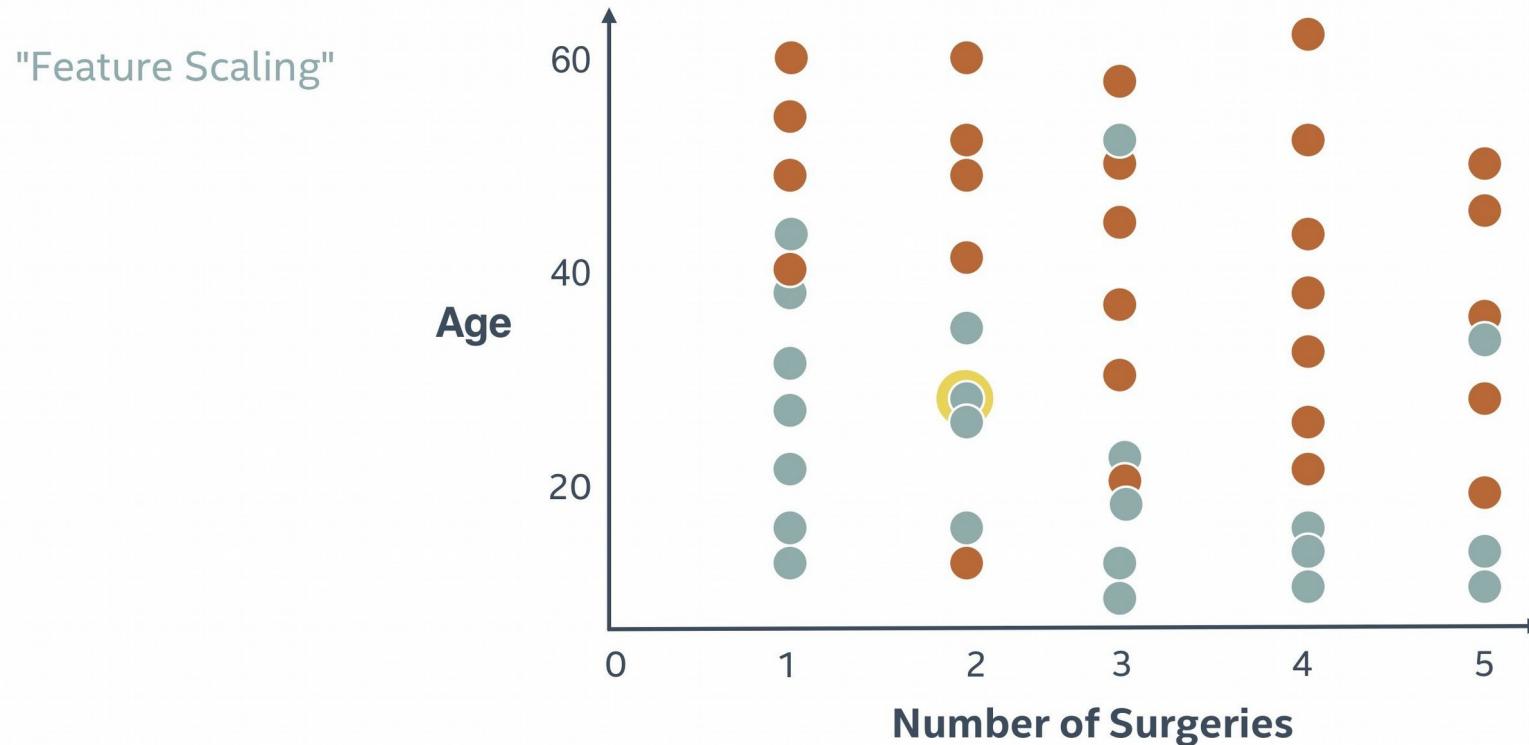
"Feature Scaling"



# Метрики в kNN

## Важность масштабирования

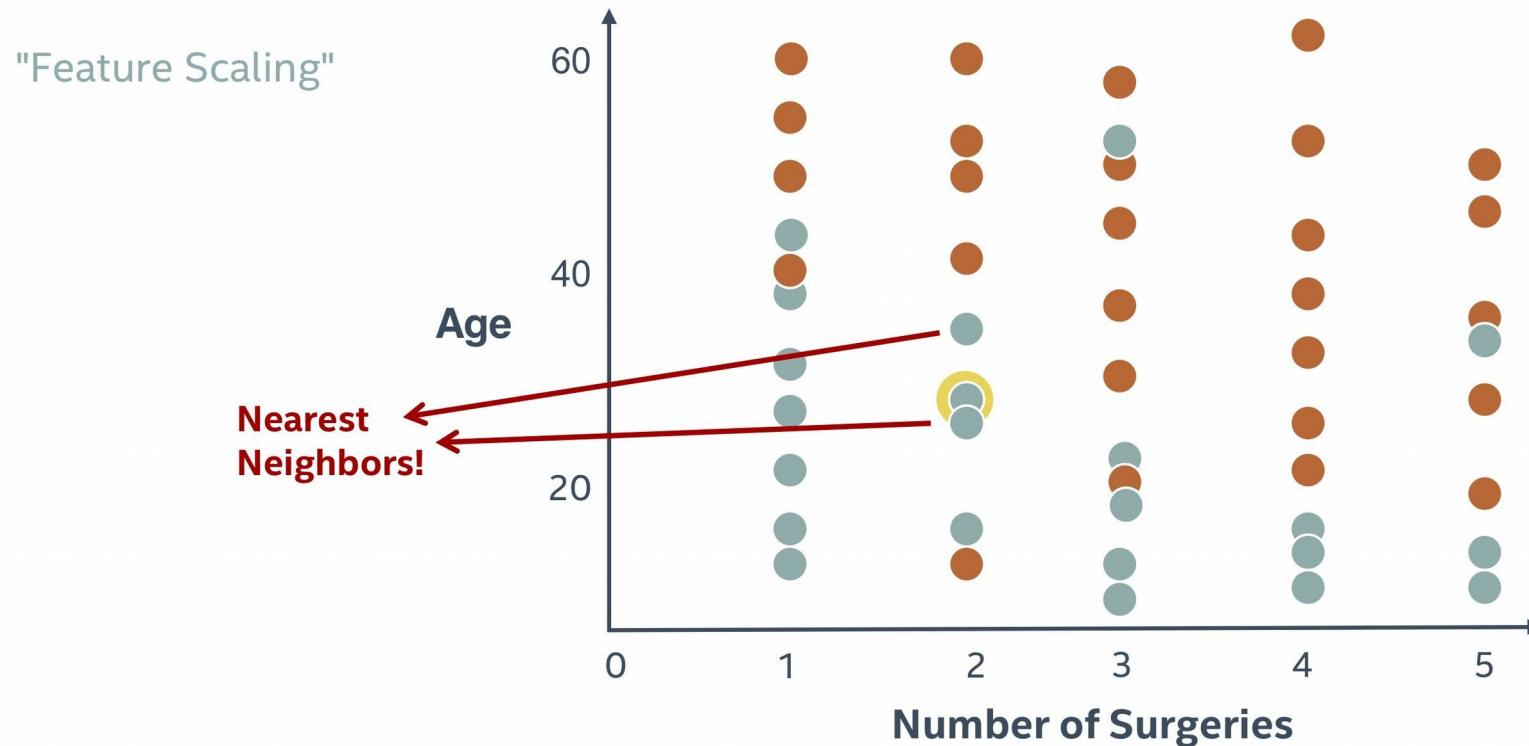
**SCALE IS IMPORTANT FOR DISTANCE MEASUREMENT**



# Метрики в kNN

## Важность масштабирования

SCALE IS IMPORTANT FOR DISTANCE MEASUREMENT



# Метрики в kNN

## Важность масштабирования

Задача определения пола человека по двум признакам:

- росту (в сантиметрах, принимает значения примерно от 150 до 200)
- уровню экспрессии гена SRY (безразмерная величина от нуля до единицы; у мужчин ближе к единице, у женщин ближе к нулю).

Обучающая выборка состоит из двух объектов:

- $x_1 = (180, 0.2)$ , девочка
- $x_2 = (173, 0.9)$ , мальчик.

Требуется классифицировать новый объект  $u = (178, 0.85)$ .

Евклидовы расстояния равны  $\rho(u, x_1) \approx 2.1$  и  $\rho(u, x_2) \approx 5$ .

Мы признаем новый объект девочкой, хотя это не так — высокий уровень экспрессии гена SRY позволяет с уверенностью сказать, что это мальчик.

Из-за сильных различий в масштабе признаков уровень экспрессии практически не учитывается при классификации, что совершенно неправильно.

# Метрики в kNN

## Важность масштабирования

### 1. Standardisation:

Standardisation replaces the values by their Z scores.

$$x' = \frac{x - \bar{x}}{\sigma}$$

### 2. Mean Normalisation:

$$x' = \frac{x - \text{mean}(x)}{\max(x) - \min(x)}$$

This distribution will have values between **-1 and 1** with  $\mu=0$ .

# Метрики в kNN

## Важность масштабирования

### 3. Min-Max Scaling:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

This scaling brings the value between 0 and 1.

### 4. Unit Vector:

$$x' = \frac{x}{\|x\|}$$

Scaling is done considering the whole feature vector to be of unit length.

# Метрики в kNN

## Методы масштабирования

### FEATURE SCALING: THE SYNTAX

Import the class containing the scaling method

```
from sklearn.preprocessing import StandardScaler
```

# Метрики в kNN

## Методы масштабирования

### FEATURE SCALING: THE SYNTAX

Import the class containing the scaling method

```
from sklearn.preprocessing import StandardScaler
```

Create an instance of the class

```
StdSc = StandardScaler()
```

# Метрики в kNN

## Методы масштабирования

### FEATURE SCALING: THE SYNTAX

Import the class containing the scaling method

```
from sklearn.preprocessing import StandardScaler
```

Create an instance of the class

```
StdSc = StandardScaler()
```

Fit the scaling parameters and then transform the data

```
StdSc = StdSc.fit(X_data)  
X_scaled = KNN.transform(X_data)
```

# Метрики в kNN

## Методы масштабирования

### FEATURE SCALING: THE SYNTAX

Import the class containing the scaling method

```
from sklearn.preprocessing import StandardScaler
```

Create an instance of the class

```
StdSc = StandardScaler()
```

Fit the scaling parameters and then transform the data

```
StdSc = StdSc.fit(X_data)  
X_scaled = KNN.transform(X_data)
```

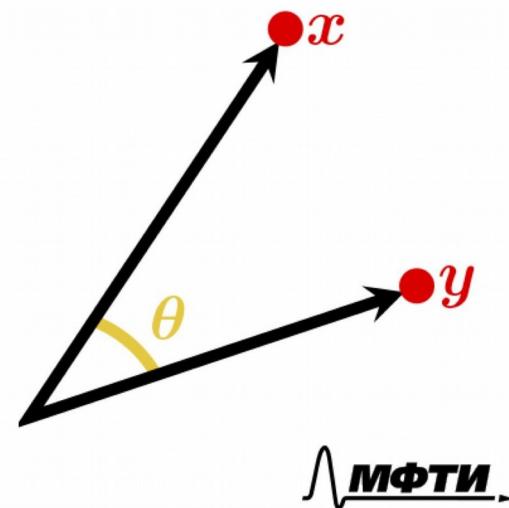
Other scaling methods exist: [MinMaxScaler](#), [MaxAbsScaler](#).

# Метрики в kNN

## Косинусная мера

$$\text{similarity} = \cos(\theta) = \frac{x \cdot y}{\|x\| \cdot \|y\|} = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}}$$

» Функция близости, а не расстояние



# Метрики в kNN

## Коэффициент корреляции

$$r = \frac{\sum_{i=1}^n ((x_i - \bar{x})(y_i - \bar{y}))}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}$$

# Метрики в kNN

## Другие функции близости

» Скалярное произведение:  $\sum x_i \cdot y_i$

» Коэффициент Дайса:  $\frac{2 \sum x_i \cdot y_i}{\sum x_i^2 + \sum y_i^2}$

» Косинусная мера:  $\frac{\sum x_i \cdot y_i}{\sqrt{\sum x_i^2} \cdot \sqrt{\sum y_i^2}}$

» Коэффициент Жаккара:  $\frac{\sum x_i \cdot y_i}{\sum x_i^2 + \sum y_i^2 - \sum x_i \cdot y_i}$

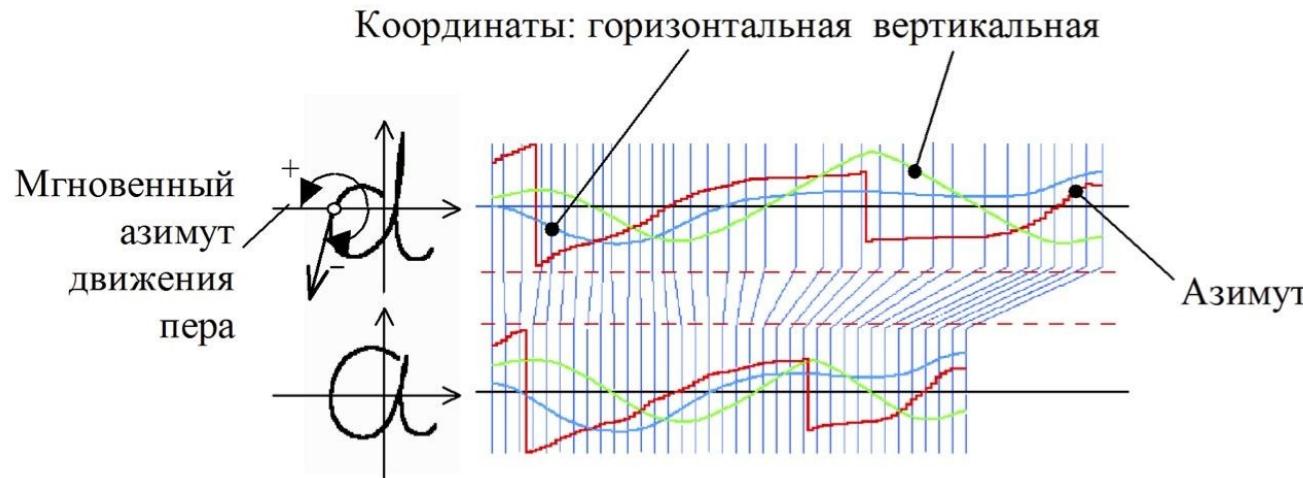
# Метрики в kNN

## Другие функции близости

- между текстами (редакторское расстояние Левенштейна):

CTGGG**CTA**AAA**GGTCC**CTTAGCC..TTTAG**AAAAAA**.GGGCCATTAGG**AAA**TTGC  
CTGGG**ACT**AAA....CCTTAGC**C**TTTA**C**AAAAA**T**GGGCCATTAGG...TTGC

- между сигналами (энергия сжатий и растяжений):



# Метрики в kNN

## Резюме

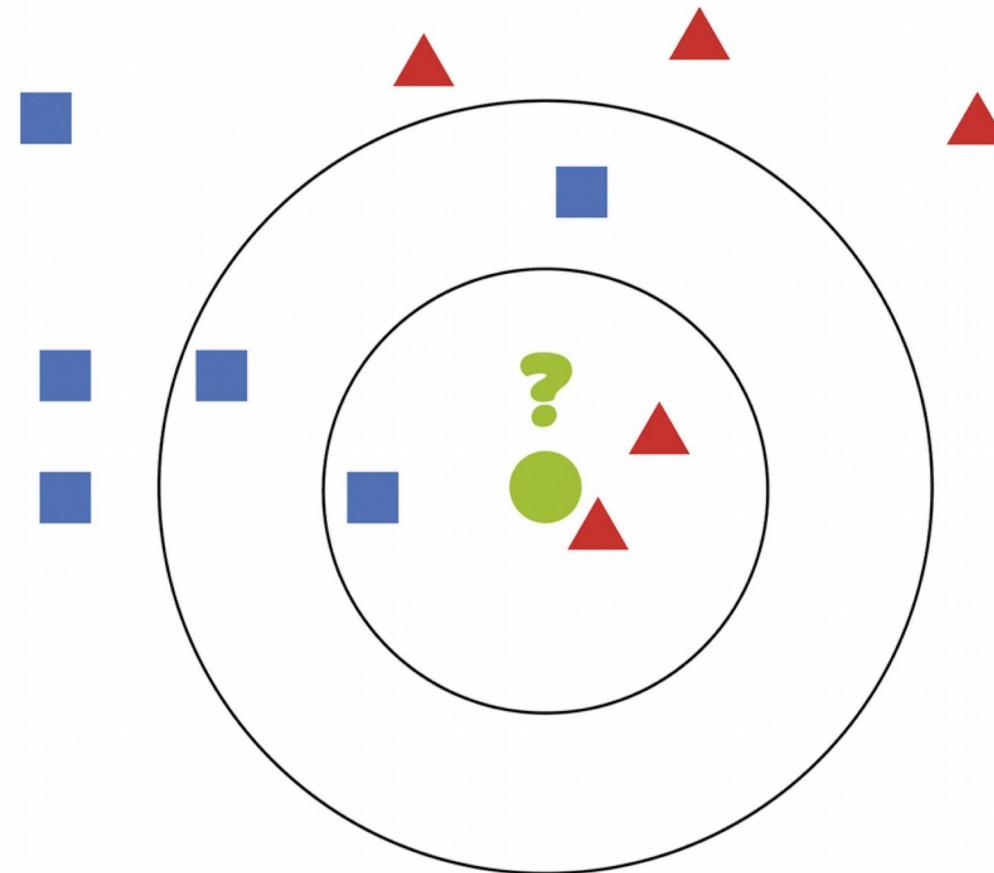
- › Можно придумать множество разных метрик
- › Чтобы понять, какая метрика лучше подходит в задаче, нужно просто попробовать разные
- › Некоторые метрики и функции близости часто используются в задачах из какой-то конкретной области, но все равно полезно экспериментировать

# План

- Метод k ближайших соседей
- Настройка параметров в kNN
- Метрики в kNN
- Проклятие размерности
- Метод окна Парзена
- Рекомендации фильмов с помощью kNN
- kNN в scikit-learn

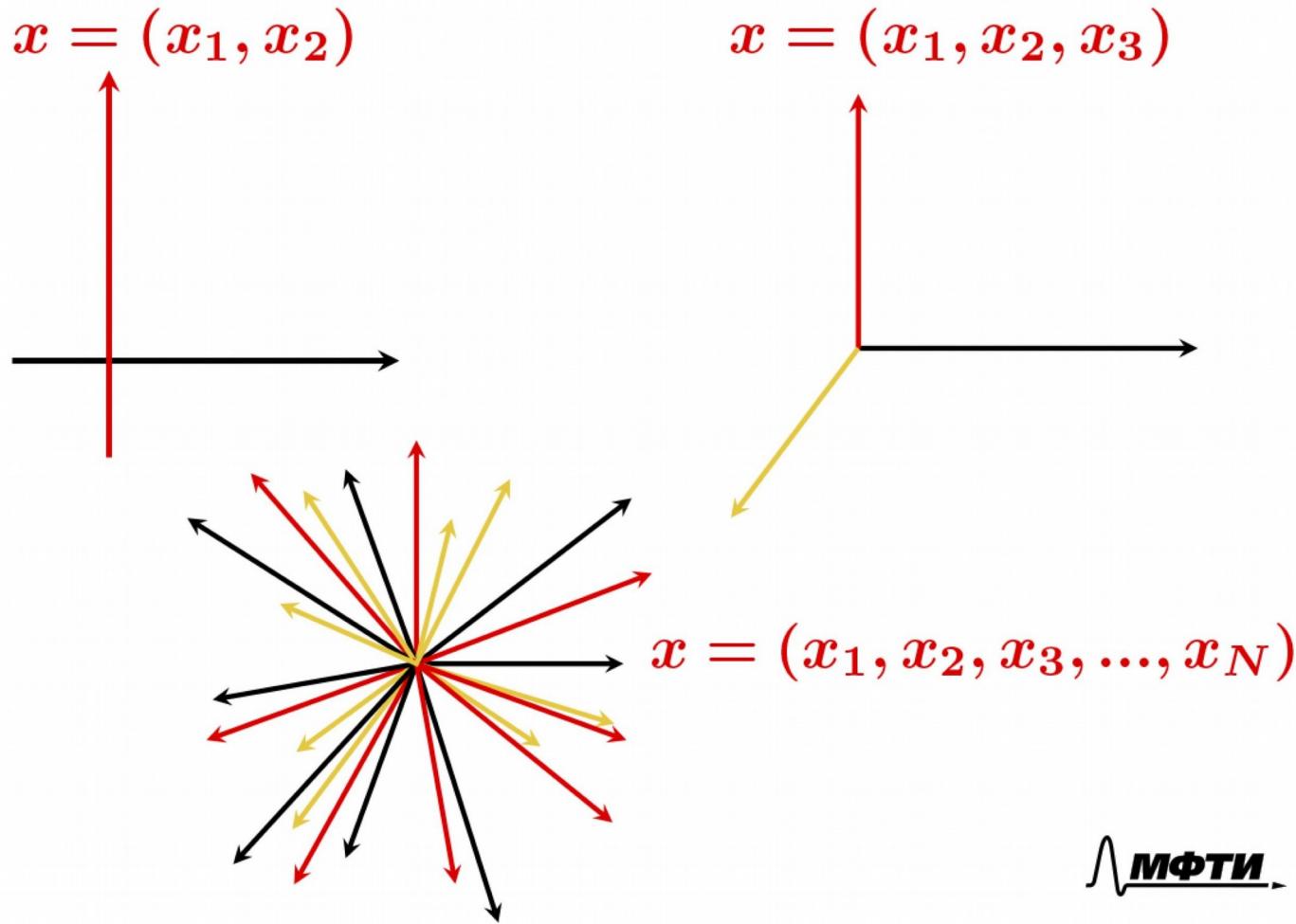
# Проклятие размерности

## Роль расстояния в метрических методах



# Проклятие размерности

## Роль расстояния в метрических методах



# Проклятие размерности

## Расстояние в пространстве высокой размерности

- › Нет ли чего-то необычного в свойствах расстояния в пространствах высокой размерности?

# Проклятие размерности

Идея 1: небольшие отличия в большом числе координат

$$x_1 = (a_1, a_2, \dots, a_N)$$

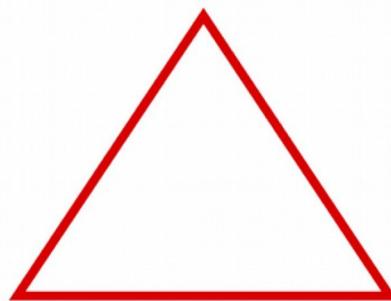
$$x_2 = (a_1 + \varepsilon, a_2 + \varepsilon, \dots, a_N + \varepsilon)$$

$$x_3 = (a_1, a_2 + \Delta, a_3, \dots, a_N)$$

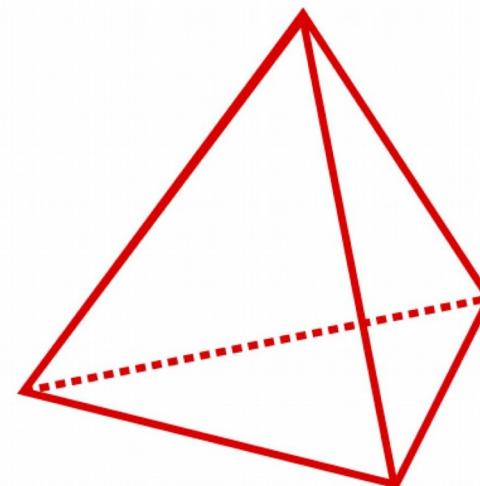
# Проклятие размерности

Идея 2: почти одинаковые расстояния

Треугольник



Тетраэдр



В  $N$ -мерном пространстве — до  $N + 1$   
равноудалённой точки

# Проклятие размерности

Идея 3: экспоненциальный рост необходимых данных

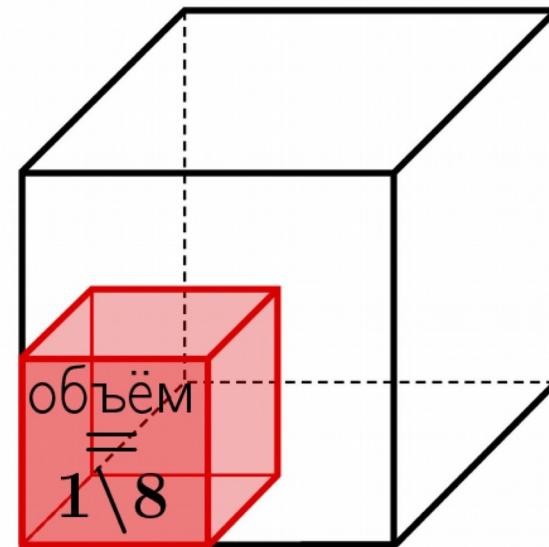
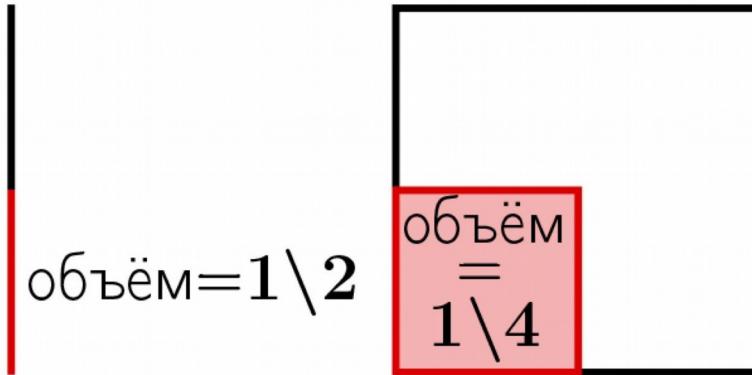
- › Рассмотрим векторы размерности  $N$  из бинарных признаков:

$$X = (0, 0, 1, 0, 1, 1, \dots, 1)$$

- › Всевозможных комбинаций значений признаков —  $2^N$
- › С ростом  $N$  экспоненциально увеличивается необходимое количество данных

# Проклятие размерности

Пример: вероятность попадания в куб



# Проклятие размерности

Пример: вероятность попадания в куб

- › Куб с длиной ребра **0.99** будет составлять  **$0.99^n$**  часть от объема единичного куба

$$\lim_{n \rightarrow \infty} 0.99^n = 0$$

# Проклятие размерности

## Резюме

- › Расстояние в пространствах высокой размерности
- › Рост необходимого объёма данных
- › Примерно одинаковое расстояние между большинством пар объектов
- › Проклятие размерности

# kNN B scikit-learn

## K NEAREST NEIGHBORS: THE SYNTAX

Import the class containing the classification method

```
from sklearn.neighbors import KNeighborsClassifier
```

# kNN B scikit-learn

## K NEAREST NEIGHBORS: THE SYNTAX

**Import the class containing the classification method**

```
from sklearn.neighbors import KNeighborsClassifier
```

**Create an instance of the class**

```
KNN = KNeighborsClassifier(n_neighbors=3)
```

# kNN B scikit-learn

## K NEAREST NEIGHBORS: THE SYNTAX

Import the class containing the classification method

```
from sklearn.neighbors import KNeighborsClassifier
```

Create an instance of the class

```
KNN = KNeighborsClassifier(n_neighbors=3)
```

Fit the instance on the data and then predict the expected value

```
KNN = KNN.fit(X_data, y_data)
```

```
y_predict = KNN.predict(X_data)
```

# kNN B scikit-learn

## K NEAREST NEIGHBORS: THE SYNTAX

Import the class containing the classification method

```
from sklearn.neighbors import KNeighborsClassifier
```

Create an instance of the class

```
KNN = KNeighborsClassifier(n_neighbors=3)
```

Fit the instance on the data and then predict the expected value

```
KNN = KNN.fit(X_data, y_data)  
y_predict = KNN.predict(X_data)
```

The **fit** and **predict/transform** syntax will show up throughout the course.

# kNN B scikit-learn

## K NEAREST NEIGHBORS: THE SYNTAX

Import the class containing the classification method

```
from sklearn.neighbors import KNeighborsClassifier
```

Create an instance of the class

```
KNN = KNeighborsClassifier(n_neighbors=3)
```

Fit the instance on the data and then predict the expected value

```
KNN = KNN.fit(X_data, y_data)  
y_predict = KNN.predict(X_data)
```

Regression can be done with [KNeighborsRegressor](#).

Thank You

"

,

# References:

- Силен, Мейсман, Али. Основы Data Science и Big Data. Python и наука о данных (2017)
- Sarkar, Bali, Sharma. Practical Machine Learning with Python (2017)
- Рашка. Python и машинное обучение (2017)
- Николенко, Кадурин, Архангельская. Глубокое обучение. Погружение в мир нейронных сетей (2018)

# References:

- Battle of the Data Science Venn Diagrams
  - <https://www.kdnuggets.com/2016/10/battle-data-science-venn-diagrams.html>
- Becoming a Data Scientist – Curriculum via Metromap
  - <http://nirvacana.com/thoughts/2013/07/08/becoming-a-data-scientist/>
- 8 Skills You Need to Be a Data Scientist
  - <https://blog.udacity.com/2014/11/data-science-job-skills.html>
- Machine Learning: What it is and Why it Matters
  - <https://www.simplilearn.com/what-is-machine-learning>