

# Классификация

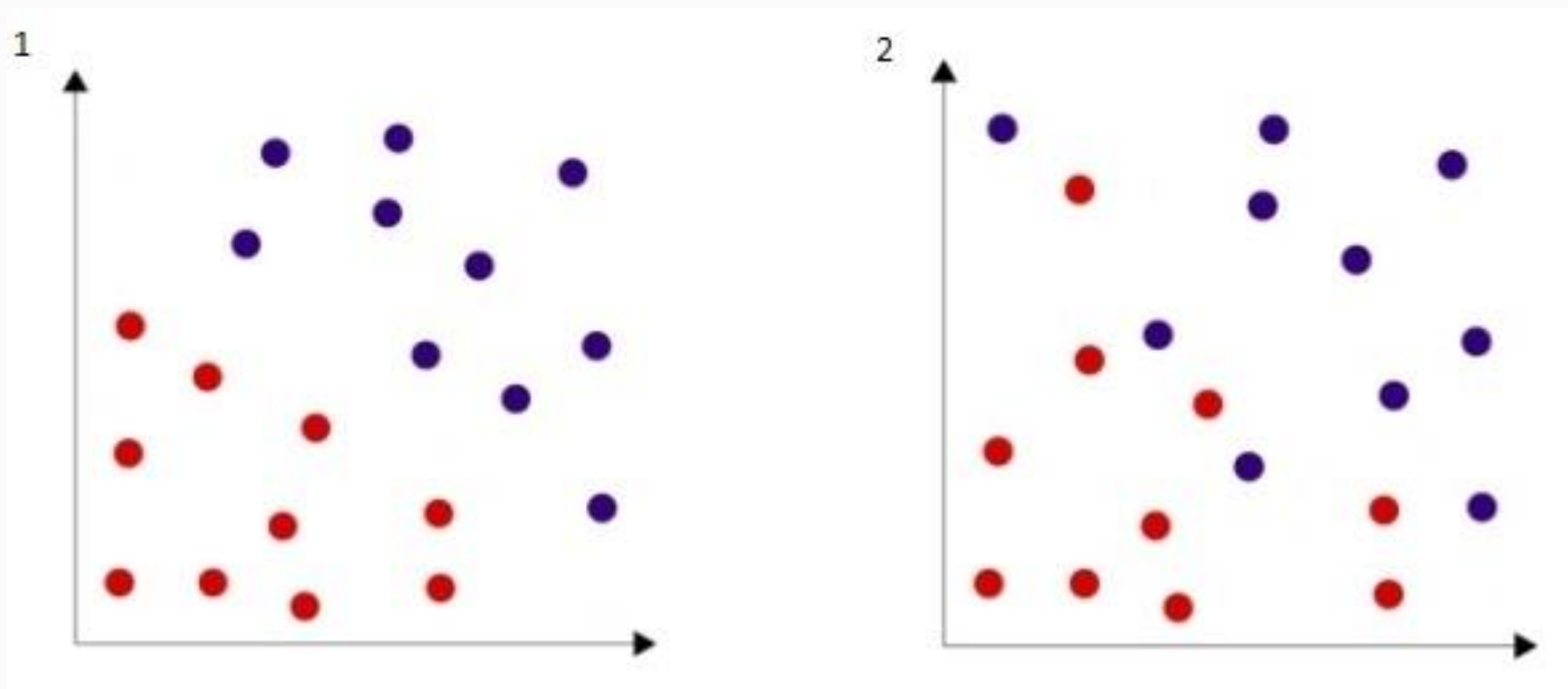
Будем решать задачу бинарной классификации, то есть классификации на два класса.

# Пример: классификация больных

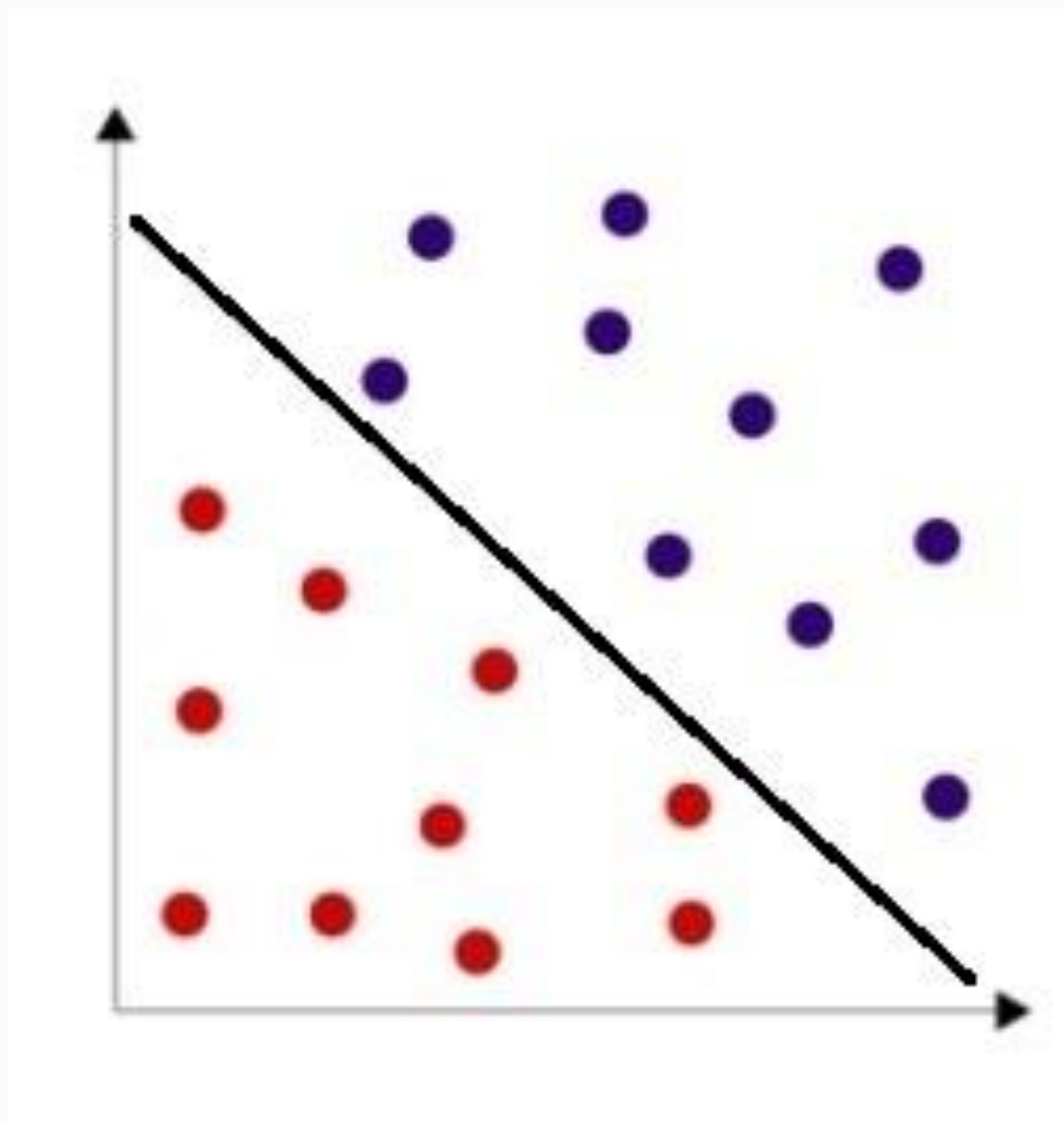
- есть данные о пациентах
- хочется построить модель предсказания болен человек или здоров



# Два типа данных

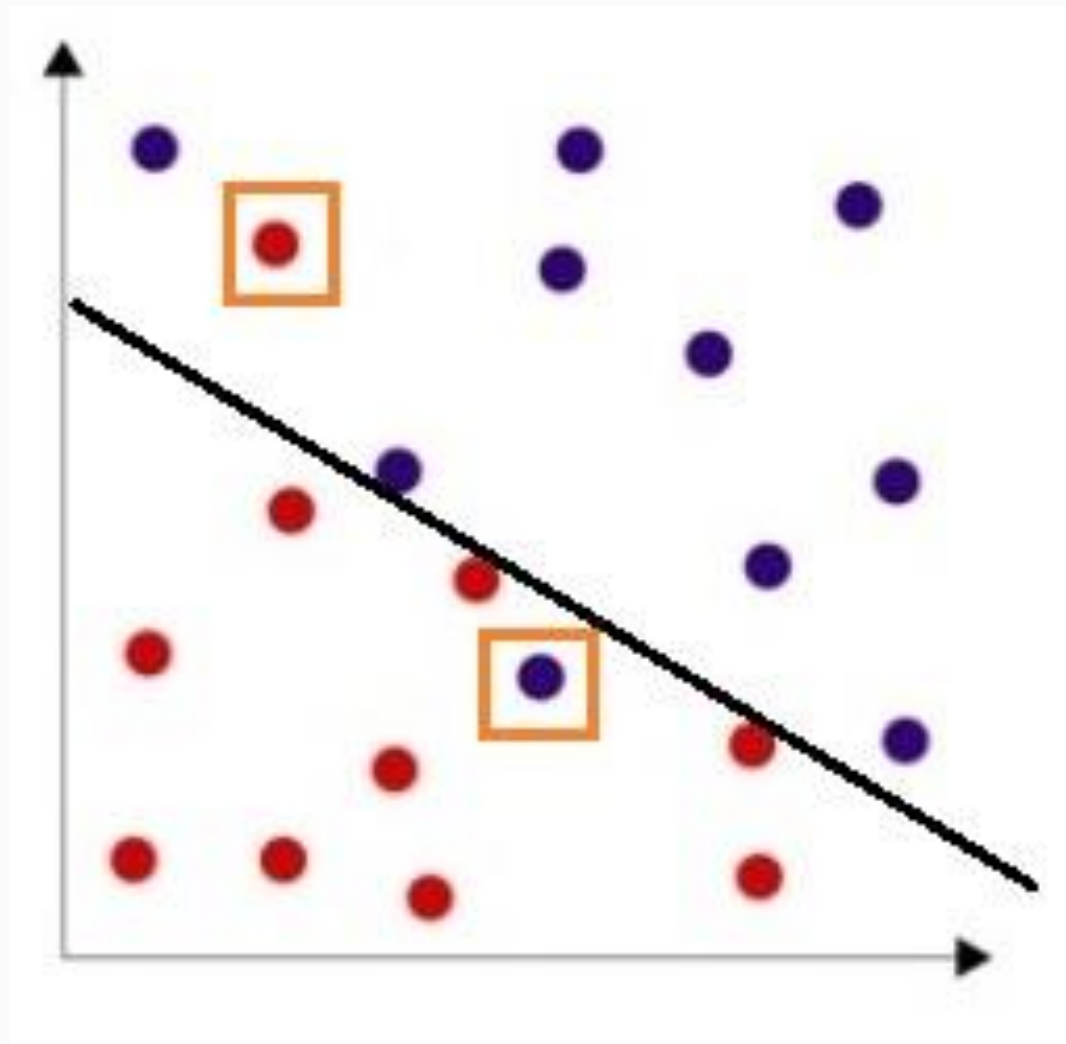


# Первая выборка



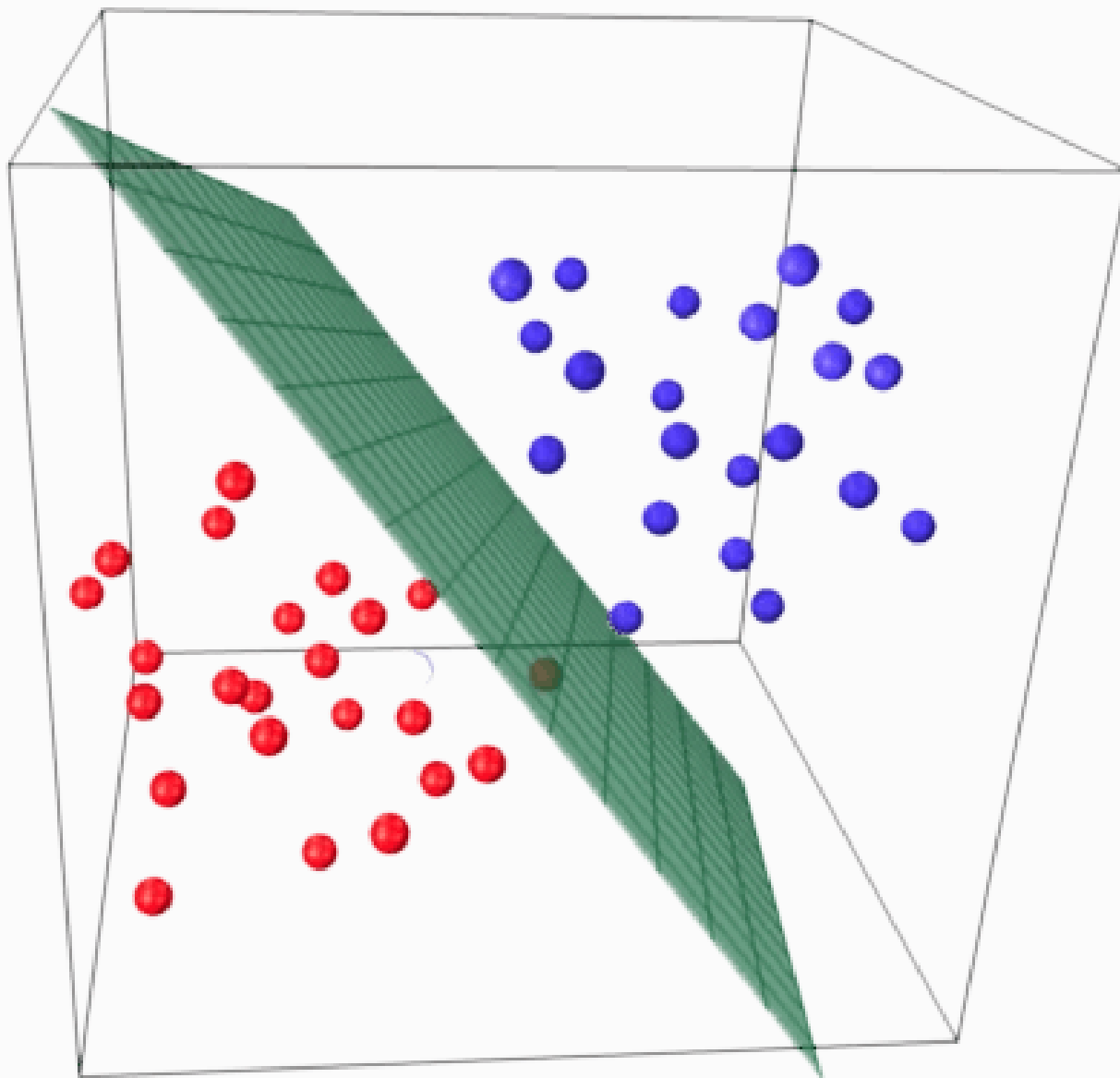
Линейно разделимая выборка

# Вторая выборка



Не является линейно разделимой

# Многомерный случай



Линейно разделимая выборка

# Формула линейной регрессии

$$a(x) = w_0 + w_1x_1 + w_2x_2 + \dots + w_dx_d,$$

- $a(x)$  - предсказание модели
- $x_1, x_2, \dots$  - признаки объекта
- $w_0, w_1, w_2, \dots$  - веса модели

# Формула линейной регрессии

$$a(x) = (w, x)$$



# Бинарный классификатор

$$a(x) = \textit{sign}(w, x)$$

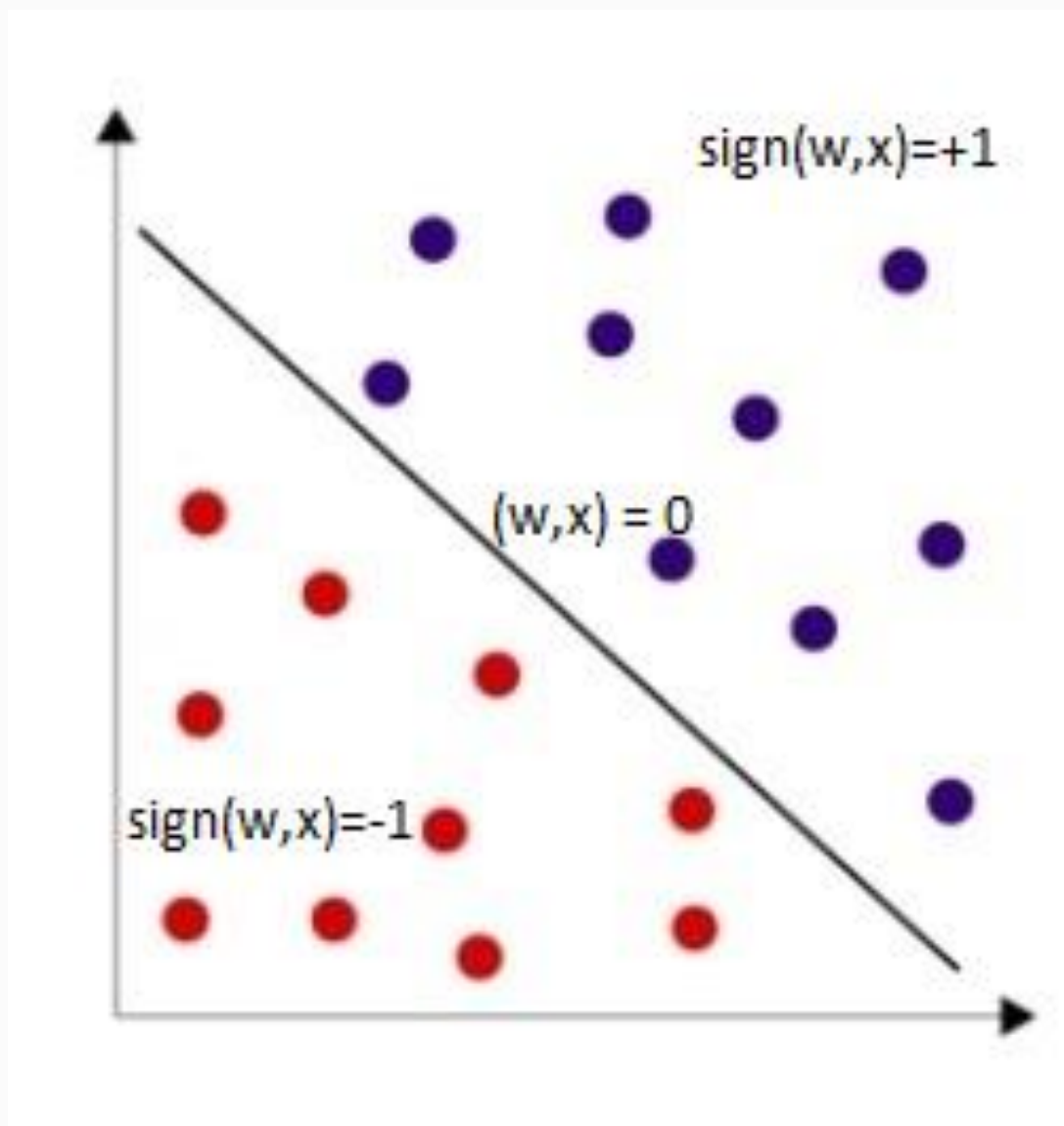
$$\textit{sign}(w, x) = +1, \text{ если } (w, x) > 0$$

и

$$\textit{sign}(w, x) = -1, \text{ если } (w, x) < 0$$

# Линейный классификатор

$$a(x) = \text{sign}(w, x)$$



# Пример

Признаки пациента:

- $x_1$  - возраст
- $x_2$  - температура
- $x_3$  - показатель давления



Классификатор:

$$a(x) = \text{sign}(w_0 + w_1x_1 + w_2x_2 + w_3x_3)$$

**4 параметра** -  $w_0, w_1, w_2, w_3$

# Логистическая регрессия

Логистическая регрессия – это линейный классификатор! Попробуем применить его для решения задачи.



# Практика!

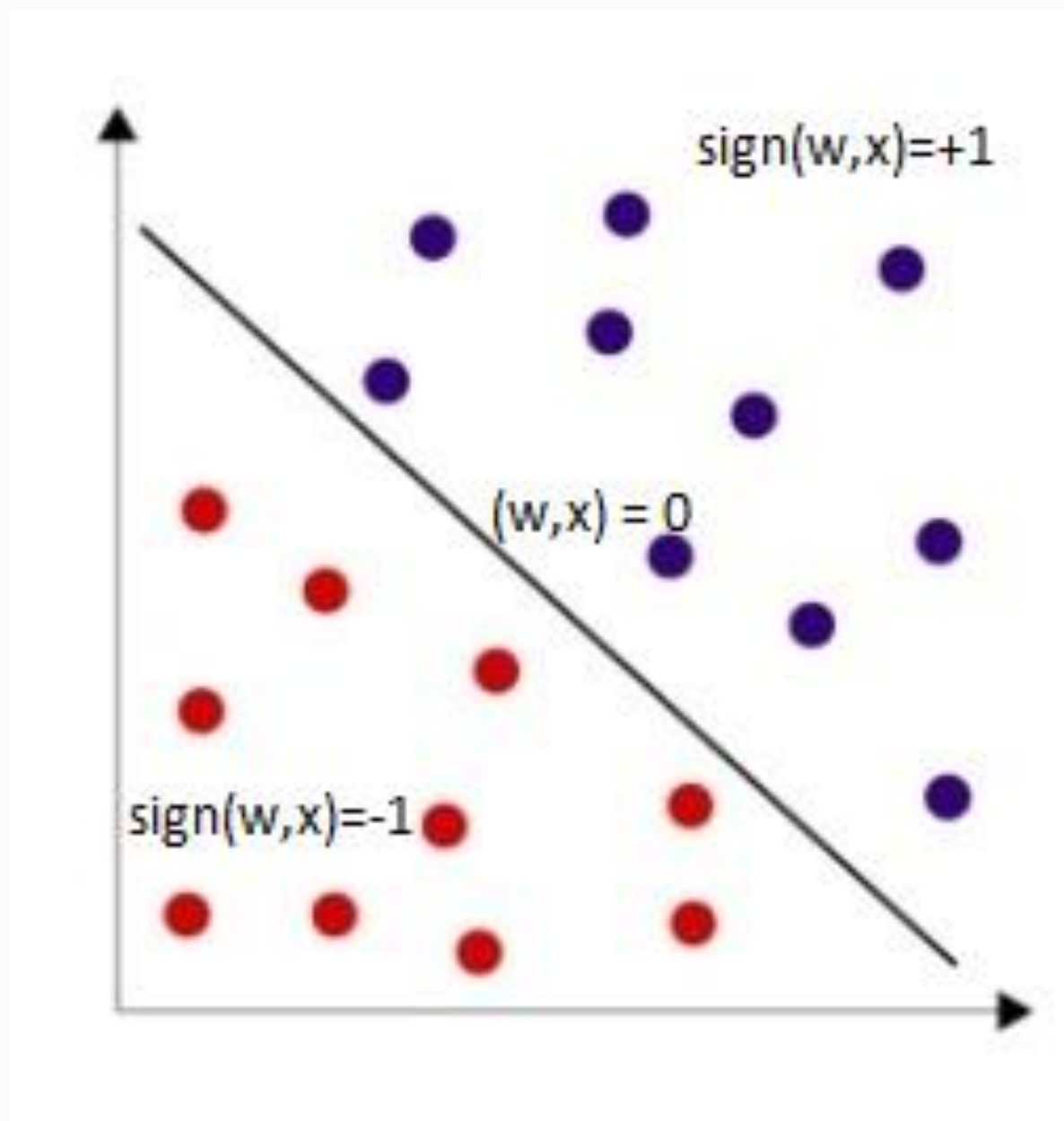
[https://colab.research.google.com/drive/1sas\\_HMWThGSiEZvN4hiq-dxU6xLHKks7#scrollTo=z79gbIw0bWJZ](https://colab.research.google.com/drive/1sas_HMWThGSiEZvN4hiq-dxU6xLHKks7#scrollTo=z79gbIw0bWJZ)

# Мягкая классификация

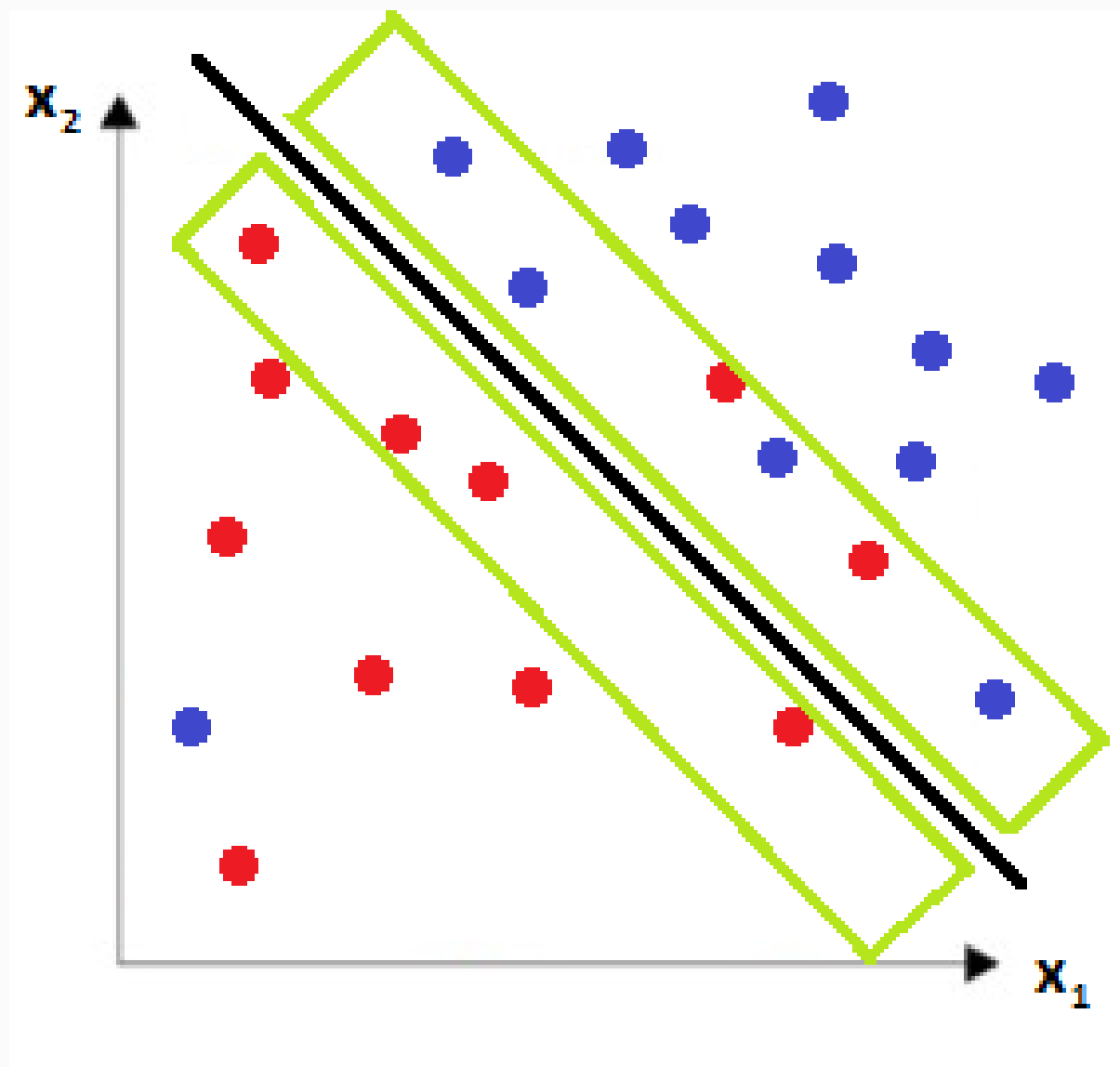
- нам хочется знать не только ответ, но и вероятность ответа



# Бинарный линейный классификатор

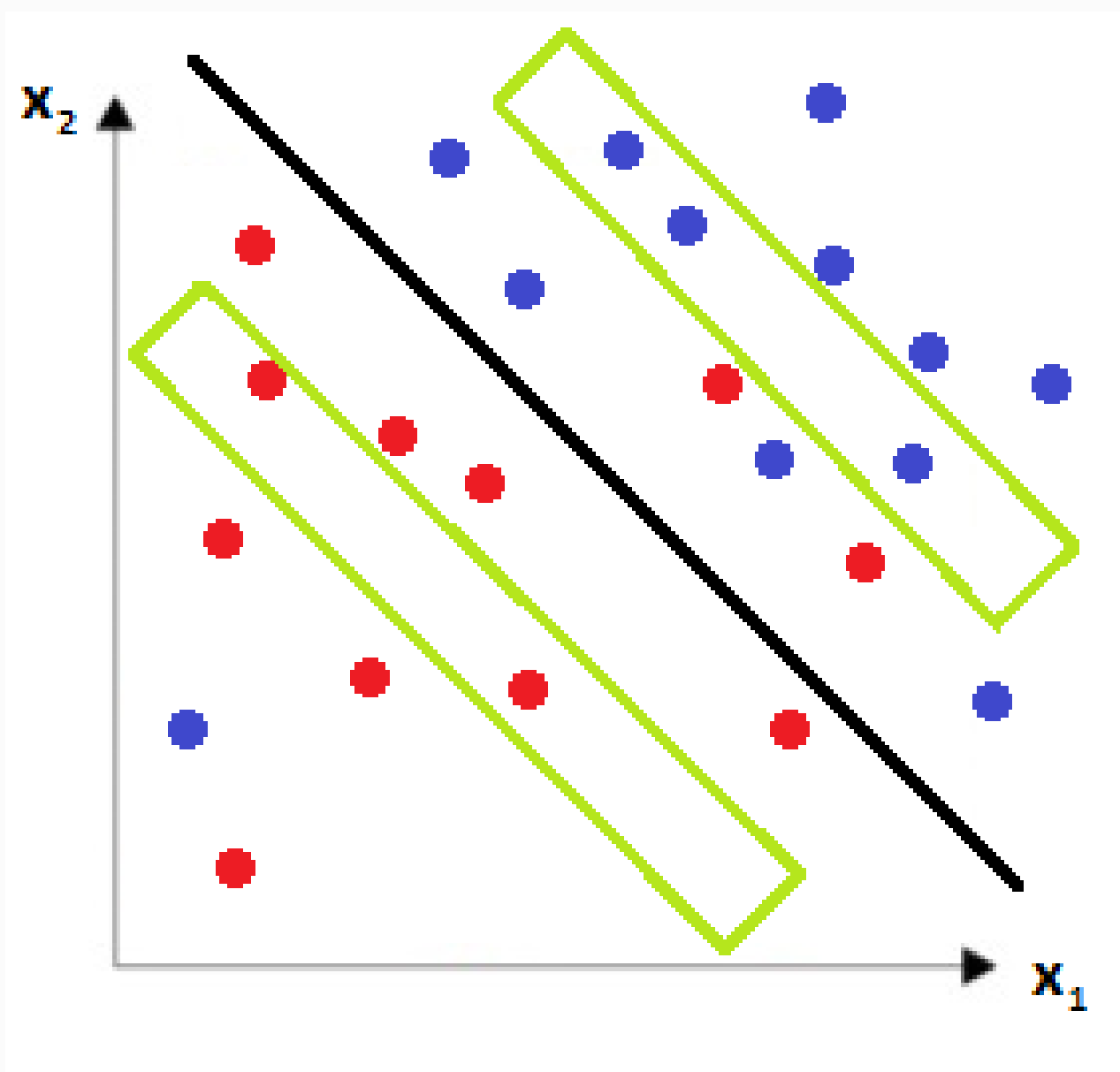


# Уверенность классификатора





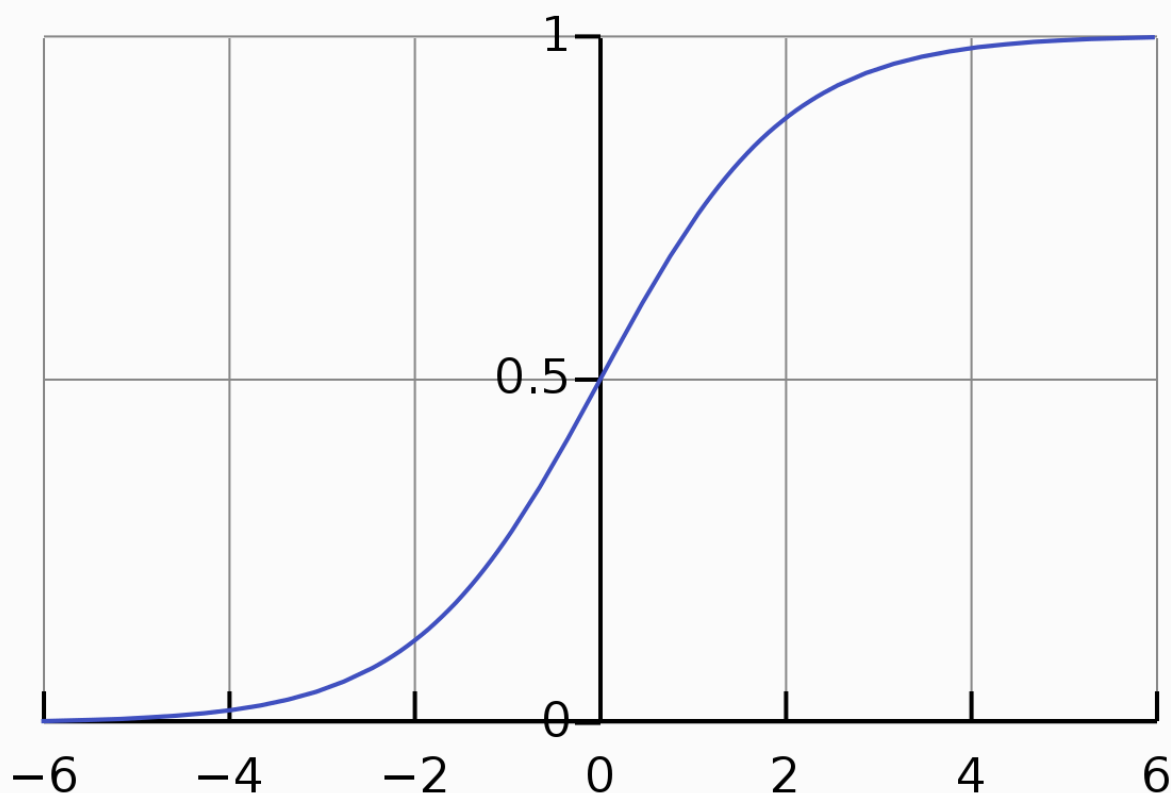
# Уверенность классификатора



# Сигмоида

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

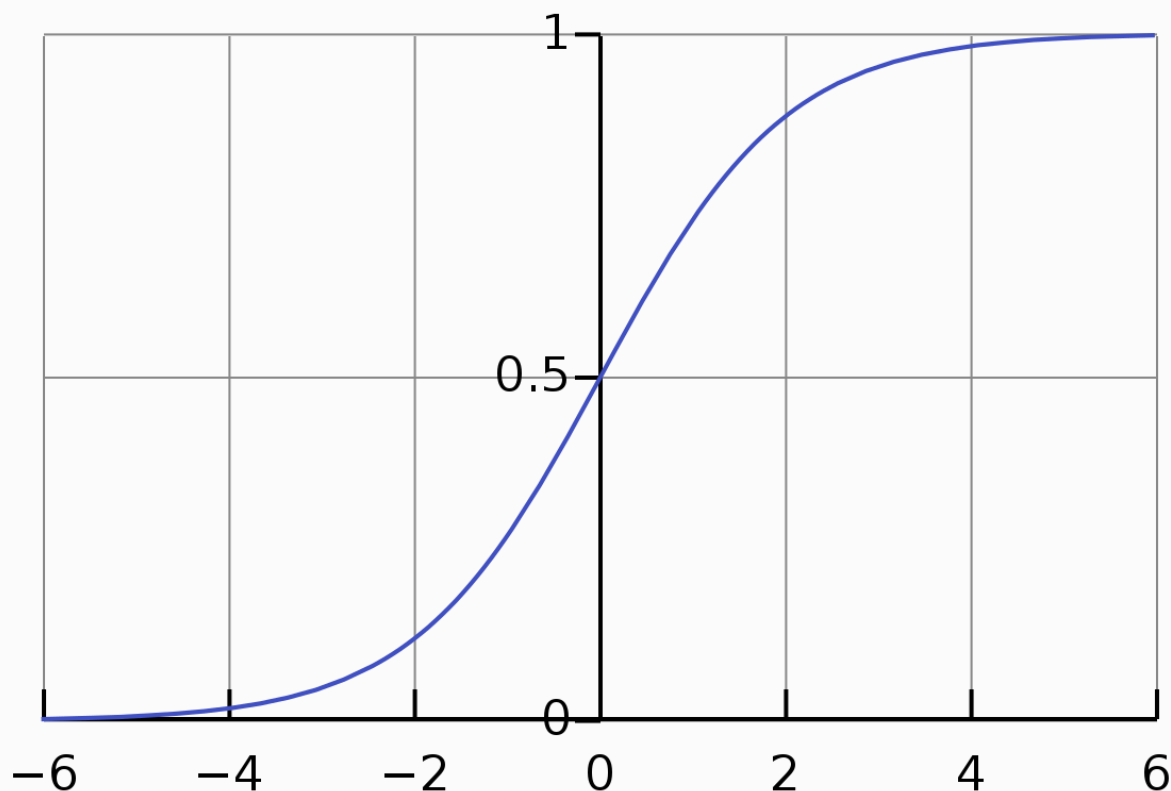
Сигмоида переводит произвольное действительное число (расстояние от объекта до разделяющей границы) в отрезок  $[0; 1]$ .



# Предсказываем вероятность

$$a(x) = \sigma(w, x) = \frac{1}{1 + e^{-(w, x)}}$$

Использование сигмоиды позволяет получить число из отрезка  $[0; 1]$ .



# Логистическая регрессия

$$a(x) = \frac{1}{1 + e^{-(w,x)}}$$

Логистическая регрессия – это линейный классификатор, который умеет предсказывать вероятности классов!

# Логистическая регрессия для выявления больных

- получаем не только класс (болен пациент или нет), но и вероятность того, что пациент болен
- получаем хорошо интерпретируемую модель, например,

$$a(x) = \sigma(1 + 10 \cdot \text{давление} - 5 \cdot \text{температура})$$



# Пример: предсказание модели

id	Предсказанная вероятность	Правильный ответ	Предсказанный класс
1	0.6	-1	1
2	0.8	1	1
3	0.3	-1	-1
4	0.55	-1	1
5	0.1	-1	-1
6	0.96	1	1
7	0.33	1	-1
8	0.2	-1	-1
9	0.14	-1	-1
10	0.88	1	1

# Практика!

[https://colab.research.google.com/drive/1sas\\_HMWThGSiEZvN4hiq-dxU6xLHKks7#scrollTo=z79gbIw0bWJZ](https://colab.research.google.com/drive/1sas_HMWThGSiEZvN4hiq-dxU6xLHKks7#scrollTo=z79gbIw0bWJZ)

# Accuracy

- **Accuracy** – это доля правильных ответов алгоритма



Accuracy = 0.7

id	Предсказанная вероятность	Правильный ответ	Предсказанный класс
1	0.6	-1	1
2	0.8	1	1
3	0.3	-1	-1
4	0.55	-1	1
5	0.1	-1	-1
6	0.96	1	1
7	0.33	1	-1
8	0.2	-1	-1
9	0.14	-1	-1
10	0.88	1	1

# Accuracy

- 1000 объектов:

950 – не мошенники (класс 0)

50 – мошенники (класс +1)

- Модель:  $a(x) = 0$

**Accuracy?**

# Accuracy

- 1000 объектов:

950 – не мошенники (класс 0)

50 – мошенники (класс +1)

- Модель:  $a(x) = 0$

**Accuracy = 0.95**

*Если классы несбалансированы, то accuracy не надо использовать!*

# Матрица ошибок

# Confusion Matrix

<i><u>Actual \ Predict</u></i>	0	1
0	TN	FN
1	FP	TP

# ROC-AUC: интуиция

- Пример:

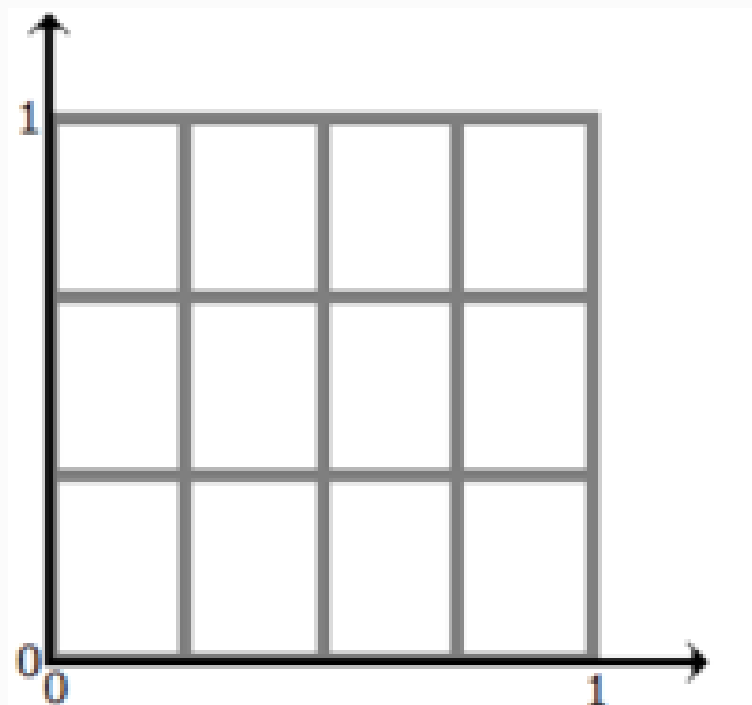
р	класс
0.5	0
0.1	0
0.25	0
0.6	1
0.2	1
0.3	1
0.0	0



р	класс
0.6	1
0.5	0
0.3	1
0.25	0
0.2	1
0.1	0
0.0	0

# ROC-AUC: алгоритм

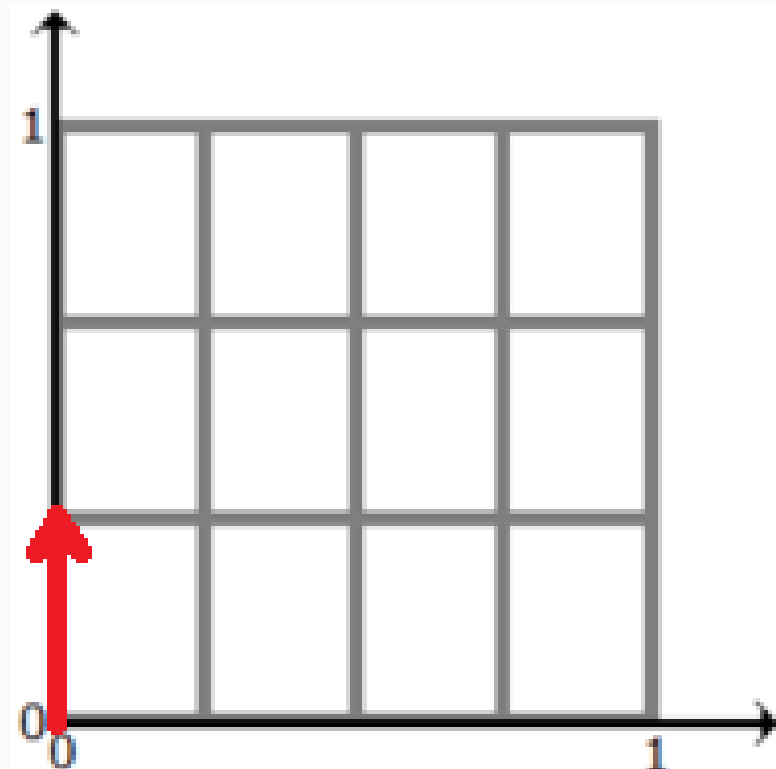
- Нарисуем квадрат 1 на 1.
- Горизонтальную сторону квадрата разобьем на равные отрезки, число которых равно числу 0 в данных
- Вертикальную сторону разобьем на равные отрезки, число которых равно числу 1



# ROC-AUC: алгоритм

- Нарисуем квадрат 1 на 1.
- Горизонтальную сторону квадрата разобьем на равные отрезки, число которых равно числу 0 в данных
- Вертикальную сторону разобьем на равные отрезки, число которых равно числу 1

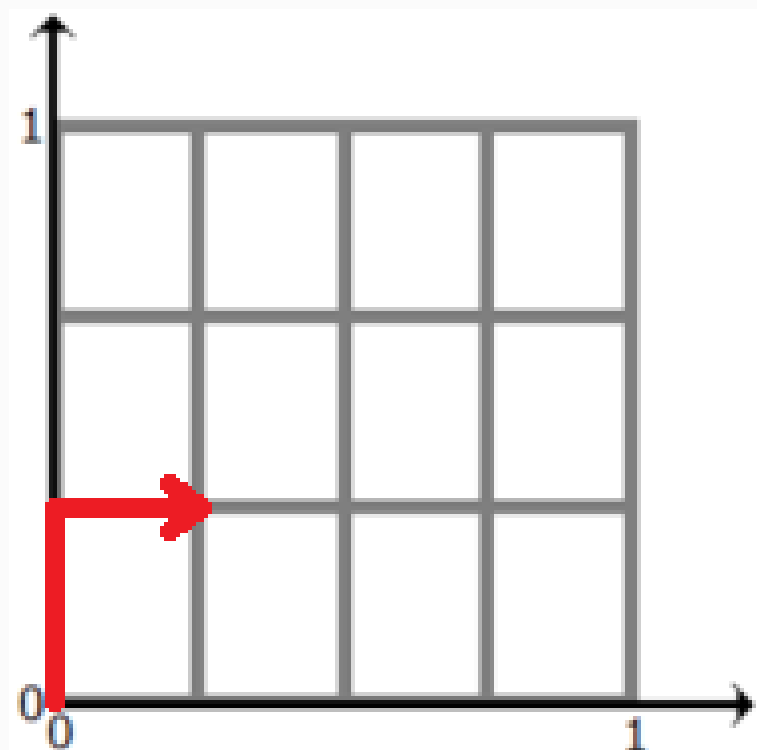
<b>p</b>	<b>класс</b>
<b>0.6</b>	<b>1</b>
<b>0.5</b>	<b>0</b>
<b>0.3</b>	<b>1</b>
<b>0.25</b>	<b>0</b>
<b>0.2</b>	<b>1</b>
<b>0.1</b>	<b>0</b>
<b>0.0</b>	<b>0</b>



# ROC-AUC: алгоритм

- Нарисуем квадрат 1 на 1.
- Горизонтальную сторону квадрата разобьем на равные отрезки, число которых равно числу 0 в данных
- Вертикальную сторону разобьем на равные отрезки, число которых равно числу 1

р	класс
0.6	1
0.5	0
0.3	1
0.25	0
0.2	1
0.1	0
0.0	0

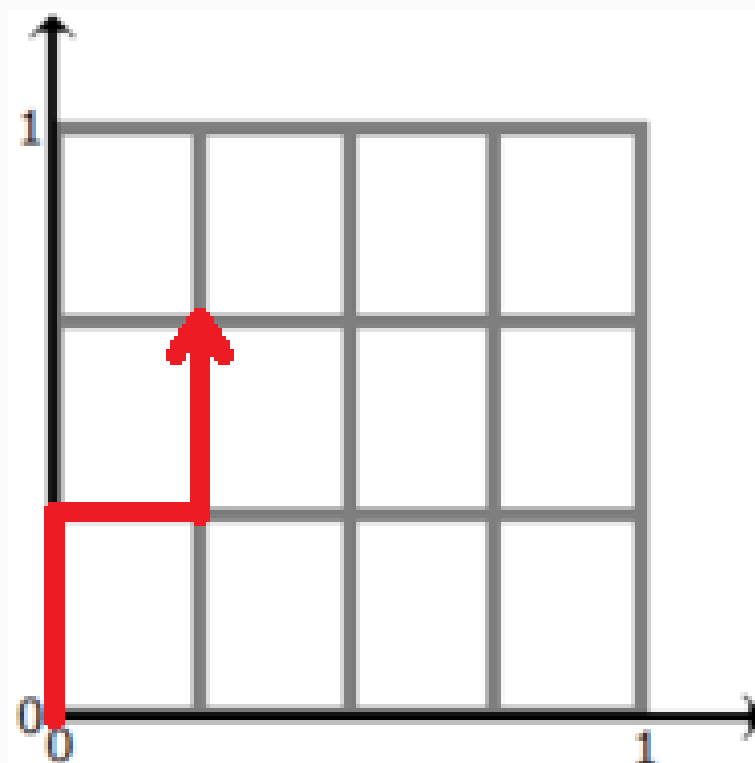




# ROC-AUC: алгоритм

- Нарисуем квадрат 1 на 1.
- Горизонтальную сторону квадрата разобьем на равные отрезки, число которых равно числу 0 в данных
- Вертикальную сторону разобьем на равные отрезки, число которых равно числу 1

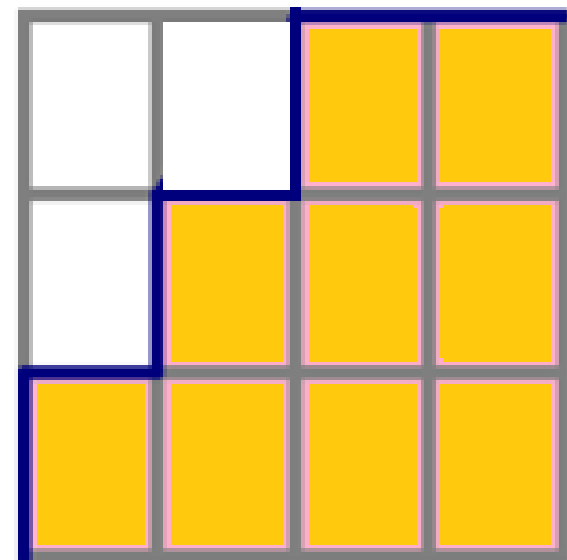
р	класс
0.6	1
0.5	0
0.3	1
0.25	0
0.2	1
0.1	0
0.0	0



# ROC-AUC: алгоритм

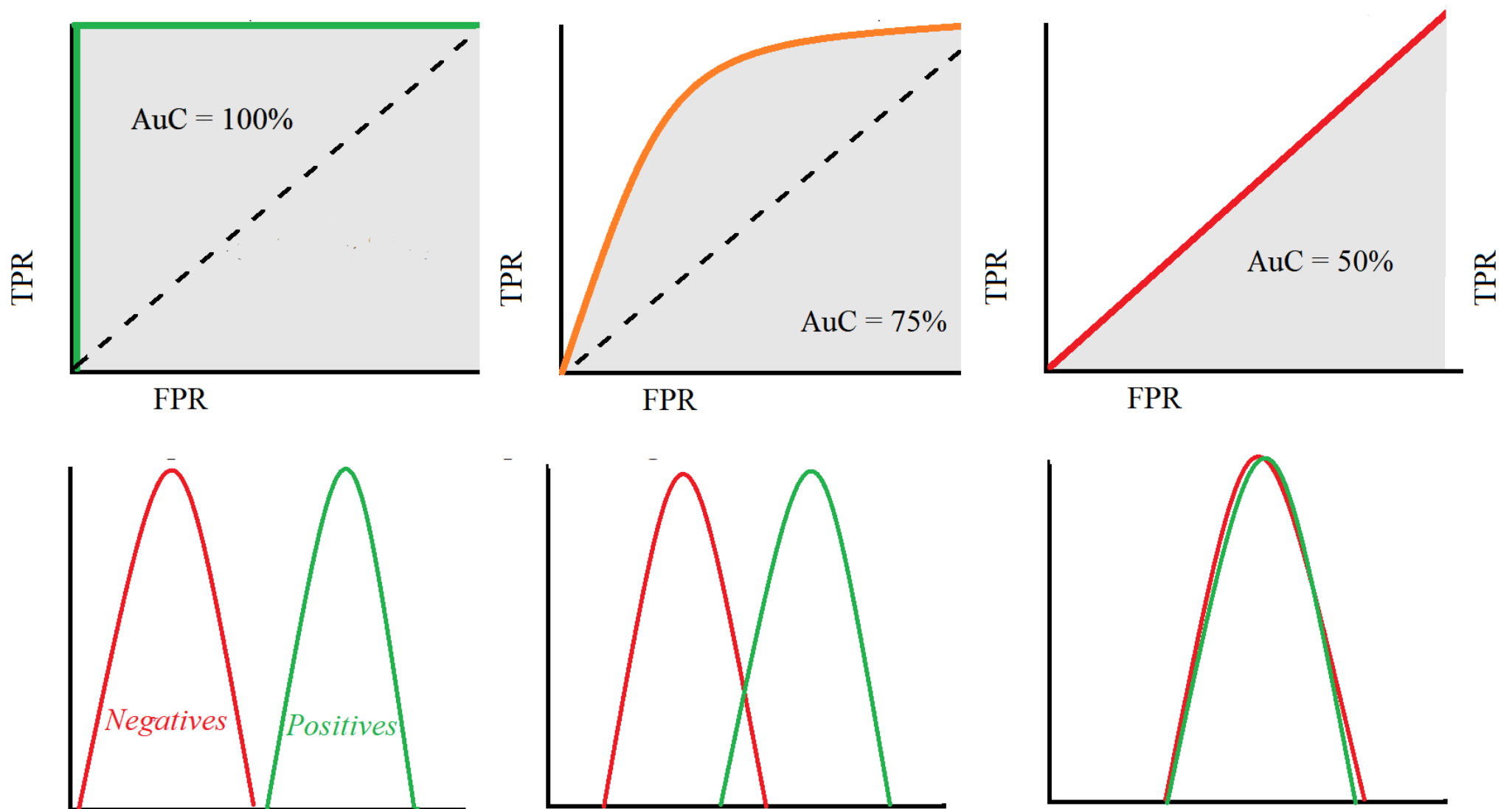
- Пойдем по отсортированной таблице по столбцу класс сверху вниз
- Будем стартовать из точки (0,0) на квадрате. И если мы встречаем 1, сдвигаемся на одну клеточку вверх, а если 0 - то вправо
- В итоге мы придём в точку (1,1).

р	класс
0.6	1
0.5	0
0.3	1
0.25	0
0.2	1
0.1	0
0.0	0



Полученная кривая называется ROC-кривой, а метрика, равная площади под ней - AUC-ROC.

# ROC-AUC: примеры



# Практика!

[https://colab.research.google.com/drive/1sas\\_HMWThGSiEZvN4hiq-dxU6xLHKks7#scrollTo=z79gbIw0bWJZ](https://colab.research.google.com/drive/1sas_HMWThGSiEZvN4hiq-dxU6xLHKks7#scrollTo=z79gbIw0bWJZ)