



# **Деревья решений и ансамбли моделей**

Занятие №7

**Журавлёв Вадим**

Показывать

Ближайшие две недели Весь семестр

Дисциплина

Основы машинного обучения

Тип события

Все типы

Группа

Все группы

30 сентября 18:00 — 21:00 среда Основы машинного обучения Смешанное занятие 1 Уточняется ML-11

7 октября 18:00 — 21:00 среда Основы машинного обучения Смешанное занятие 2 Уточняется ML-11

14 октября 18:00 — 21:00 среда Основы машинного обучения Смешанное занятие 3 Уточняется ML-11

21 октября 18:00 — 21:00 среда Основы машинного обучения Смешанное занятие 4 Уточняется ML-11

28 октября 18:00 — 21:00 среда Основы машинного обучения Смешанное занятие 5 Уточняется ML-11

3 ноября вторник 18:00 — 21:00 Основы машинного обучения Смешанное занятие 6 Уточняется ML-11

11 ноября среда 18:00 — 21:00 Основы машинного обучения Смешанное занятие 7 Уточняется ML-11

сентябрь

Пн	Вт	Ср	Чт	Пт	Сб	Вс
1	2	3	4	5	6	
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

октябрь

Пн	Вт	Ср	Чт	Пт	Сб	Вс
		1	2	3	4	
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

ноябрь

Пн	Вт	Ср	Чт	Пт	Сб	Вс

## Приходя на лекцию



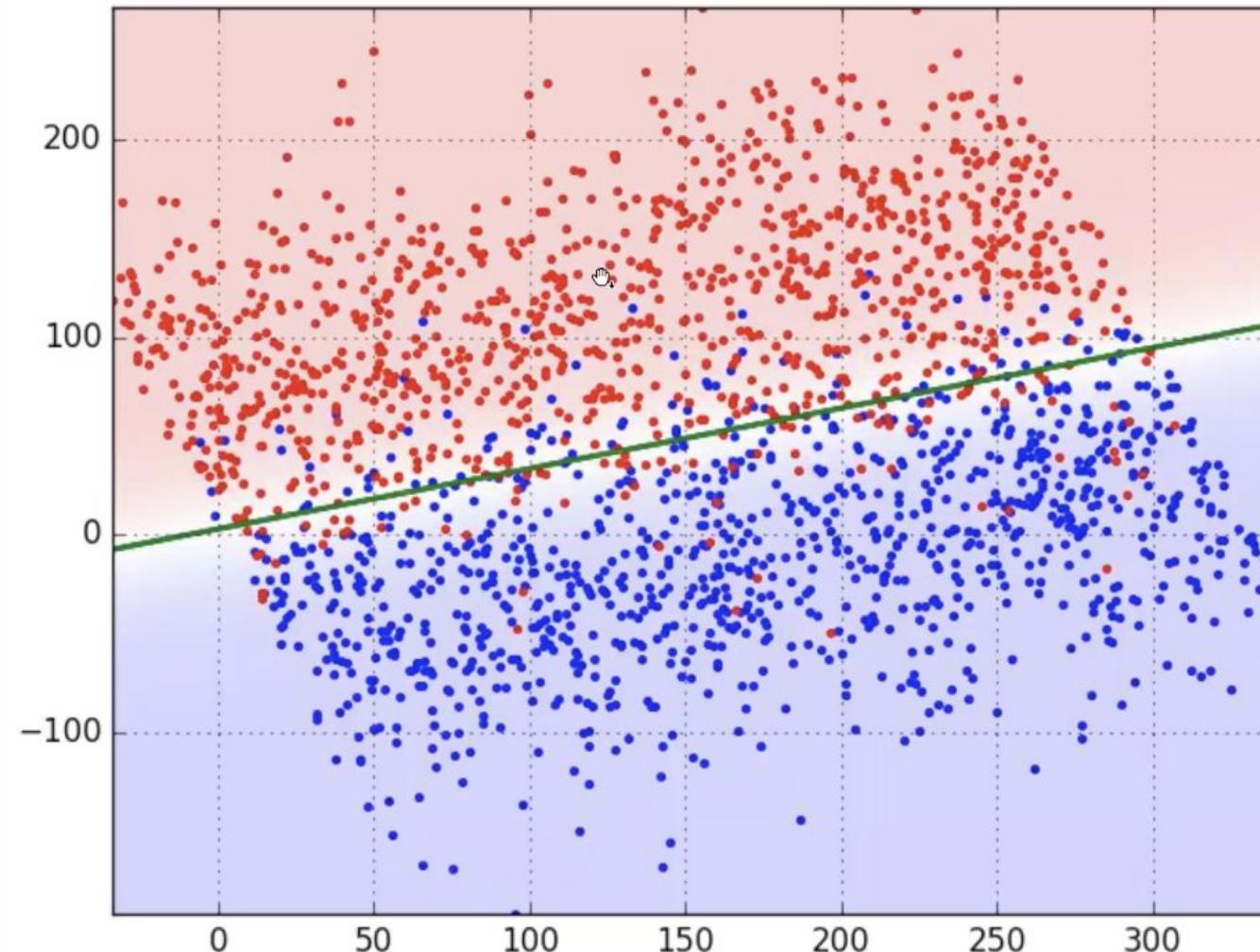
отметиться не  
забудь ты

---

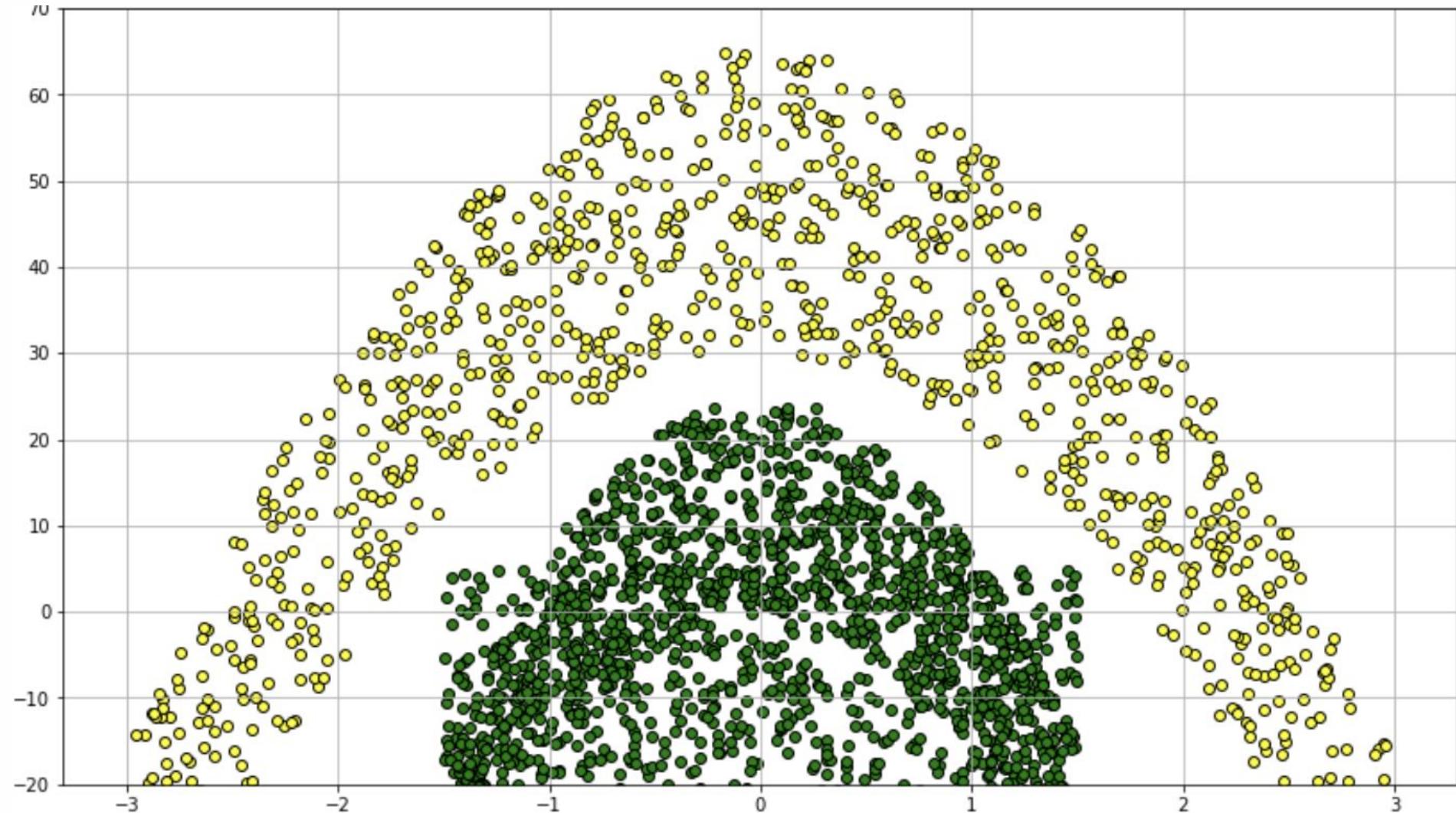
## **Содержание занятия**

- 1. Деревья**
- 2. Ансамбли моделей**
- 3. Стэкинг**
- 4. Бэггинг и бустинг**
- 5. Random Forest**
- 6. Gradient Boosting**

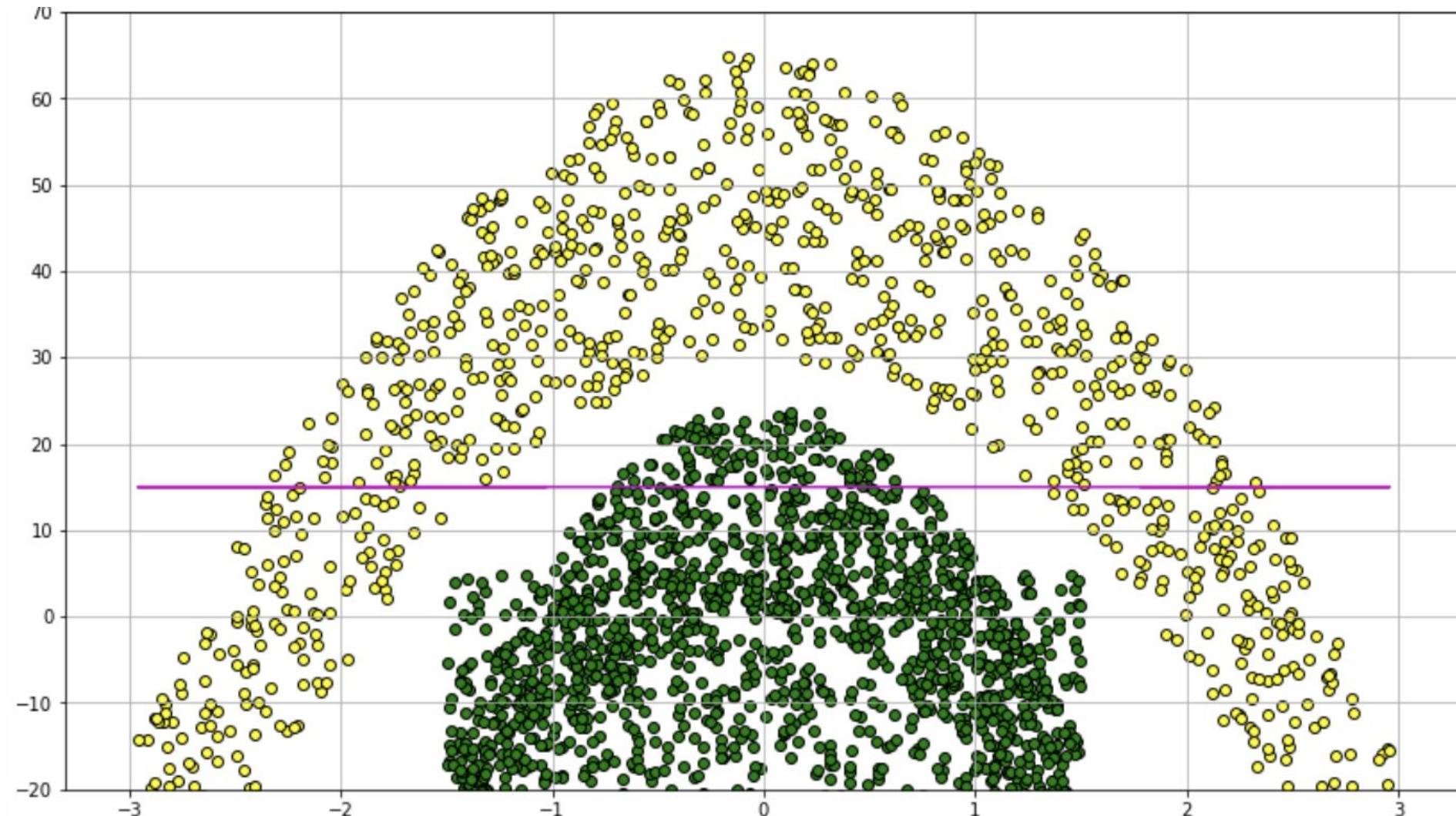
# Всегда ли регрессия нас спасает?



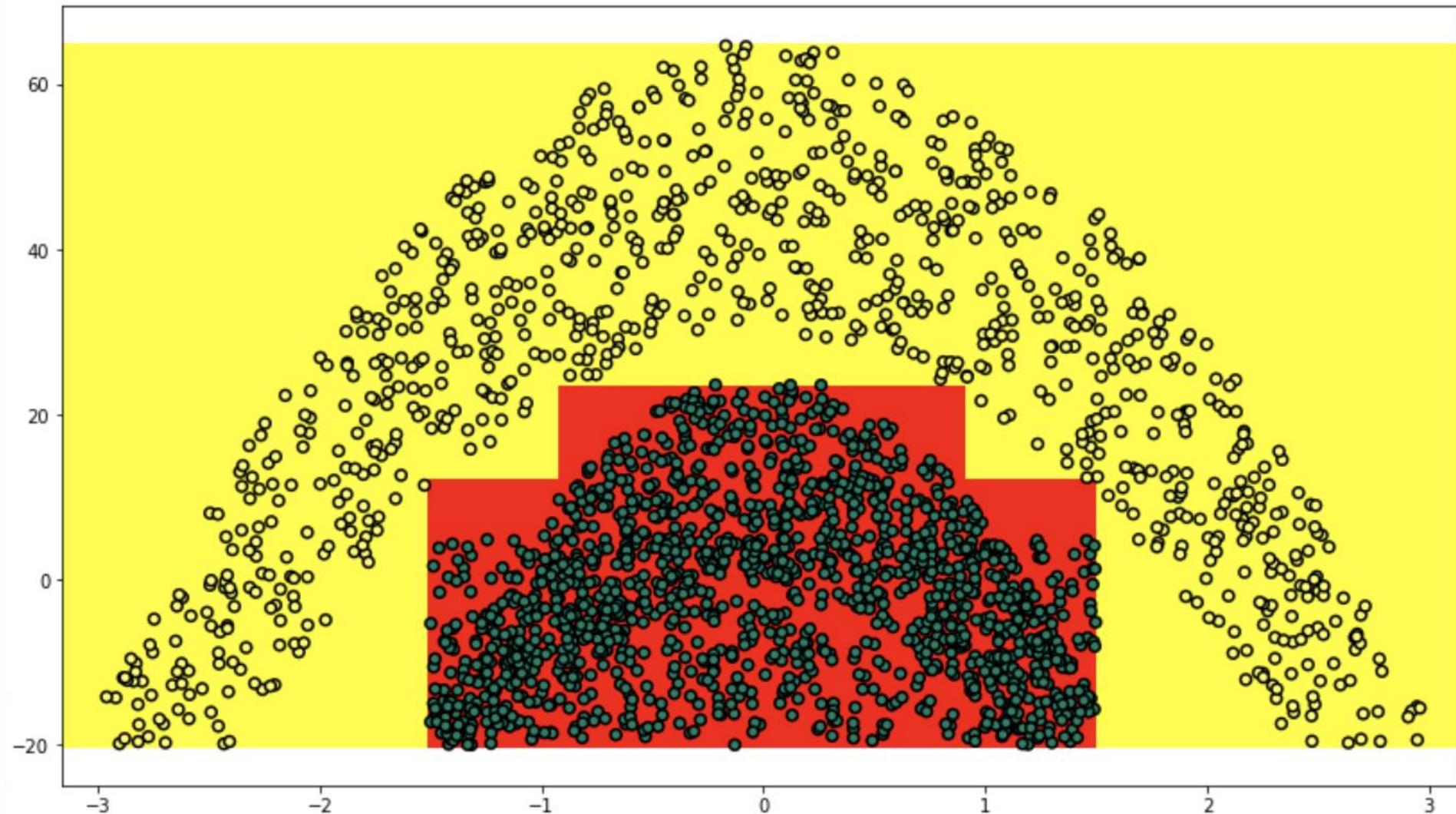
# А в таком случае?



# А в таком случае?

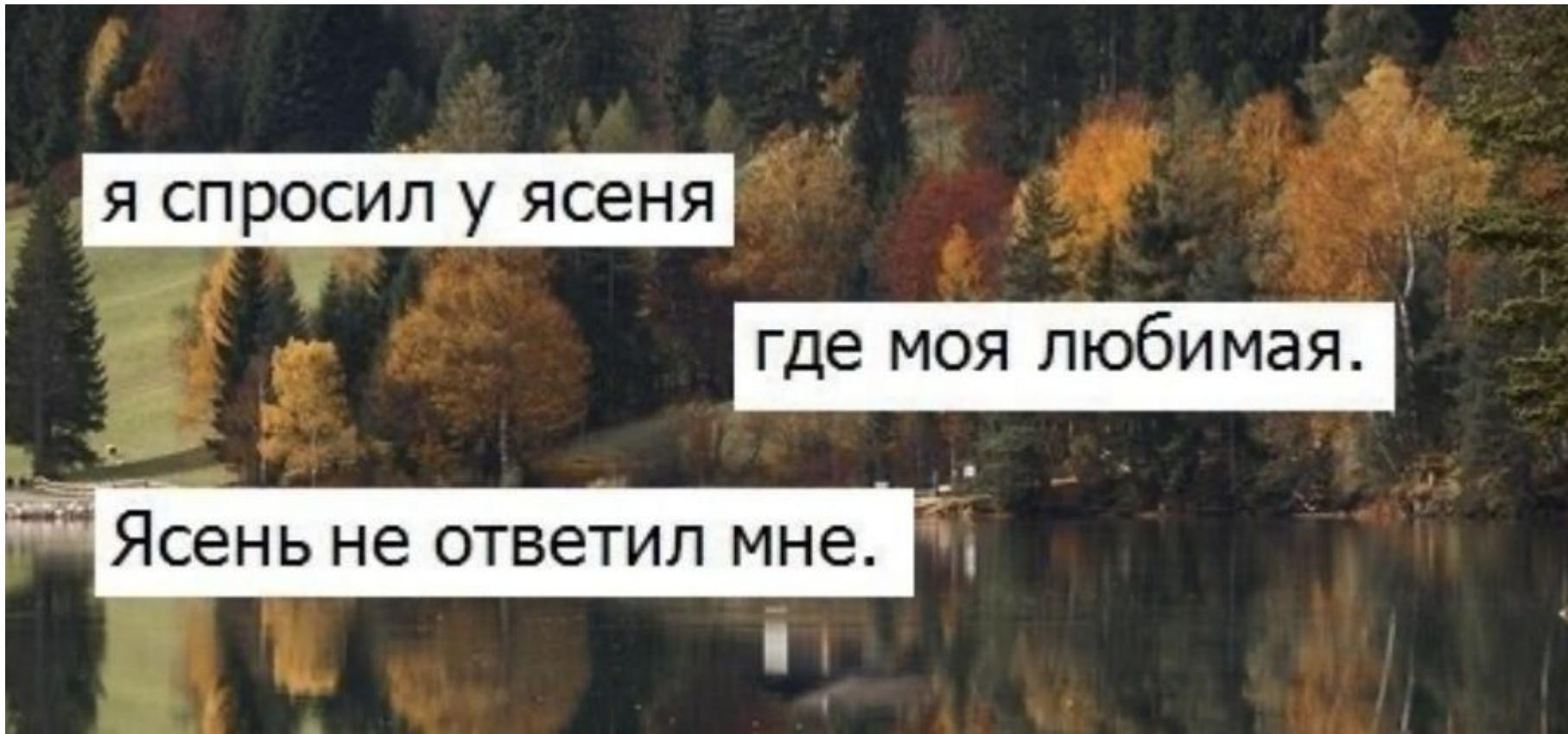


**А в таком случае?**



---

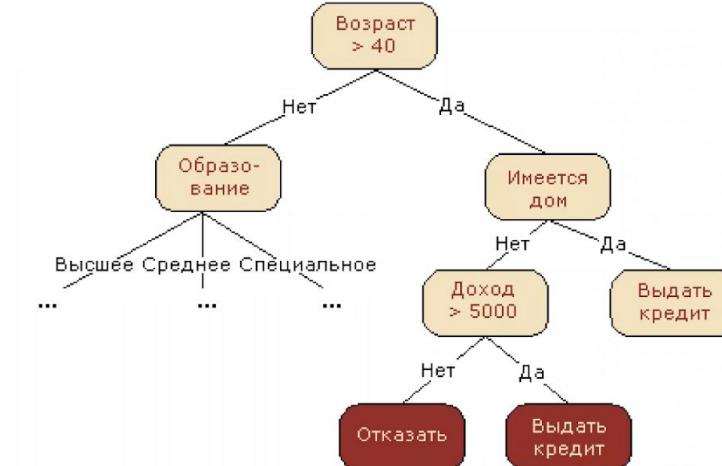
## Деревья решений



# Деревья решений

Деревья решений относятся к логическим методам классификации, так как ищут в данных логические закономерности.

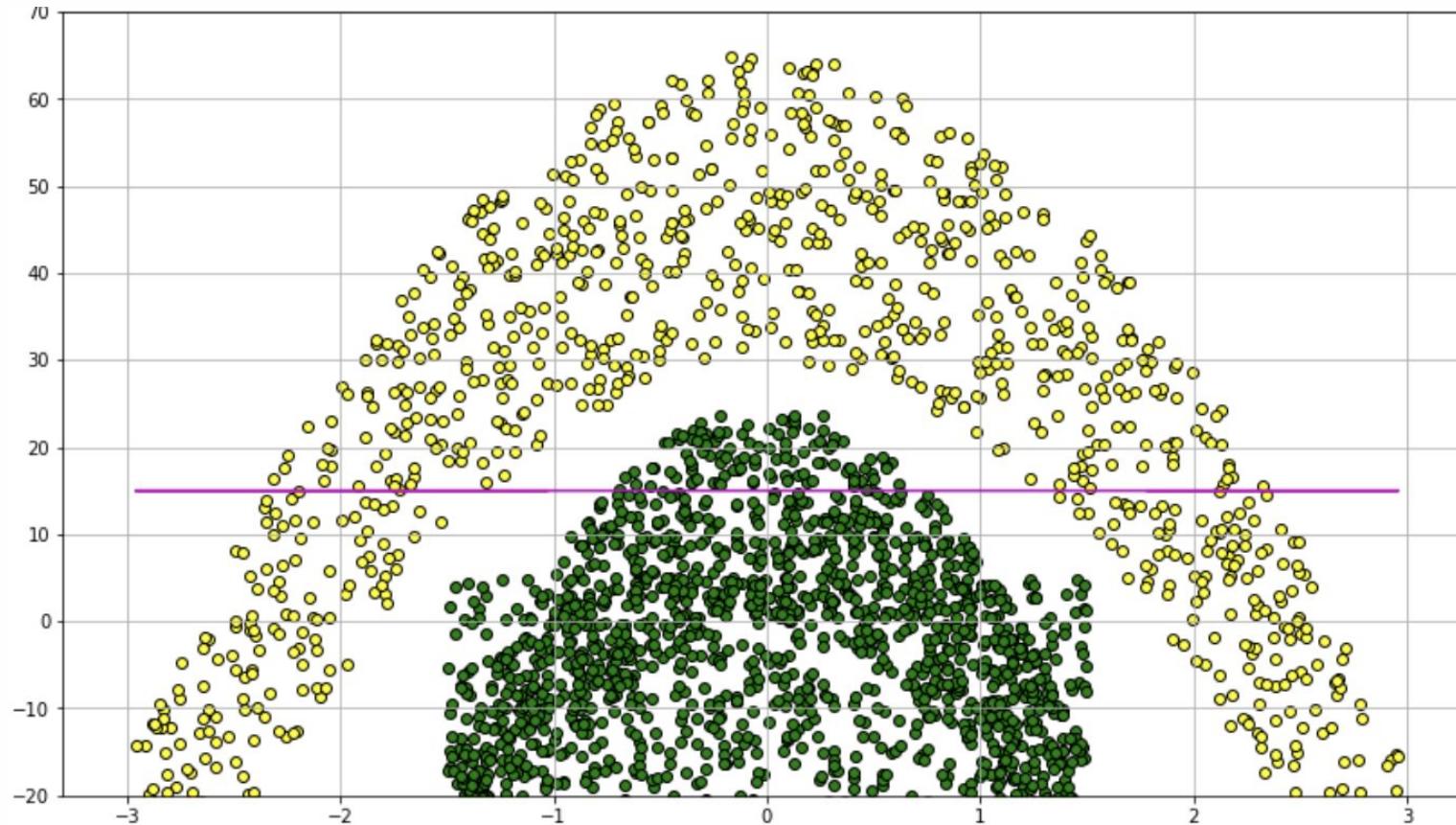
- Температура  $> 38$ ? Да -> Есть кашель? Да -> Кашель влажный? Да -> **Назначать антибиотики**
- Возраст  $> 40$ ? Да -> Имеется дом? Нет -> Доход  $> 5000$ ? Да -> **Выдать кредит**



В ходе лекции будем рассматривать задачу  
бинарной классификации, потом обсудим как  
можно масштабировать

# Мотивация 1. Пример нелинейного датасета

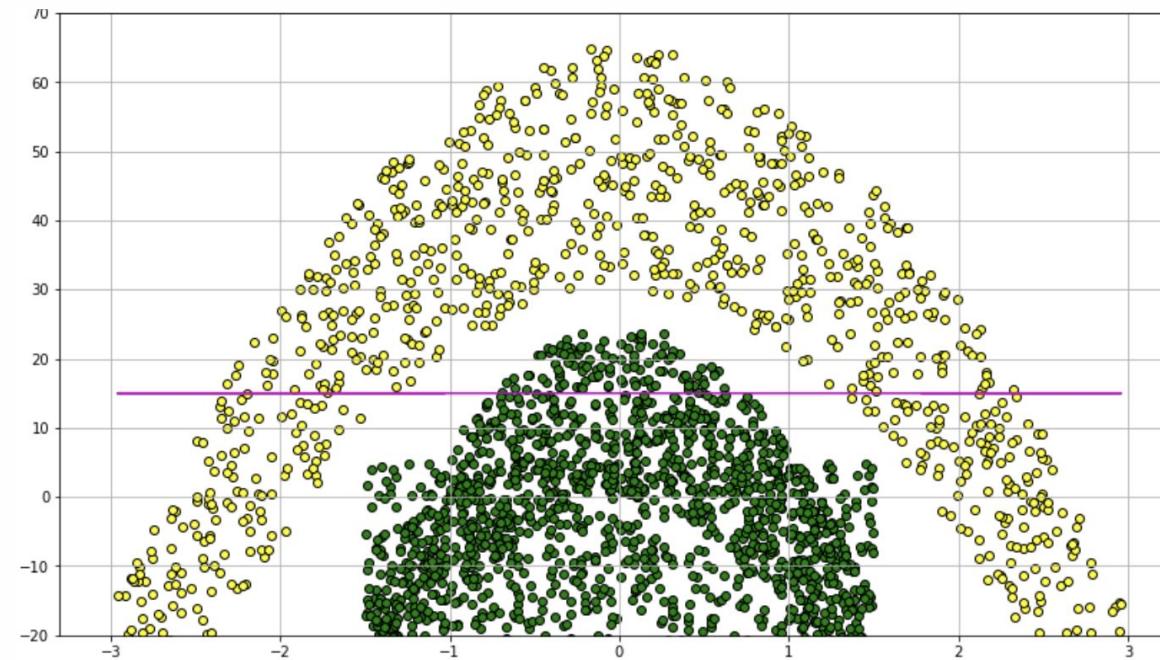
Logistic Regression, f-measure ~0.755



#010 Целевая переменная **нелинейно** зависит от признаков

## Мотивация 2. Лог. рег. и наш пример

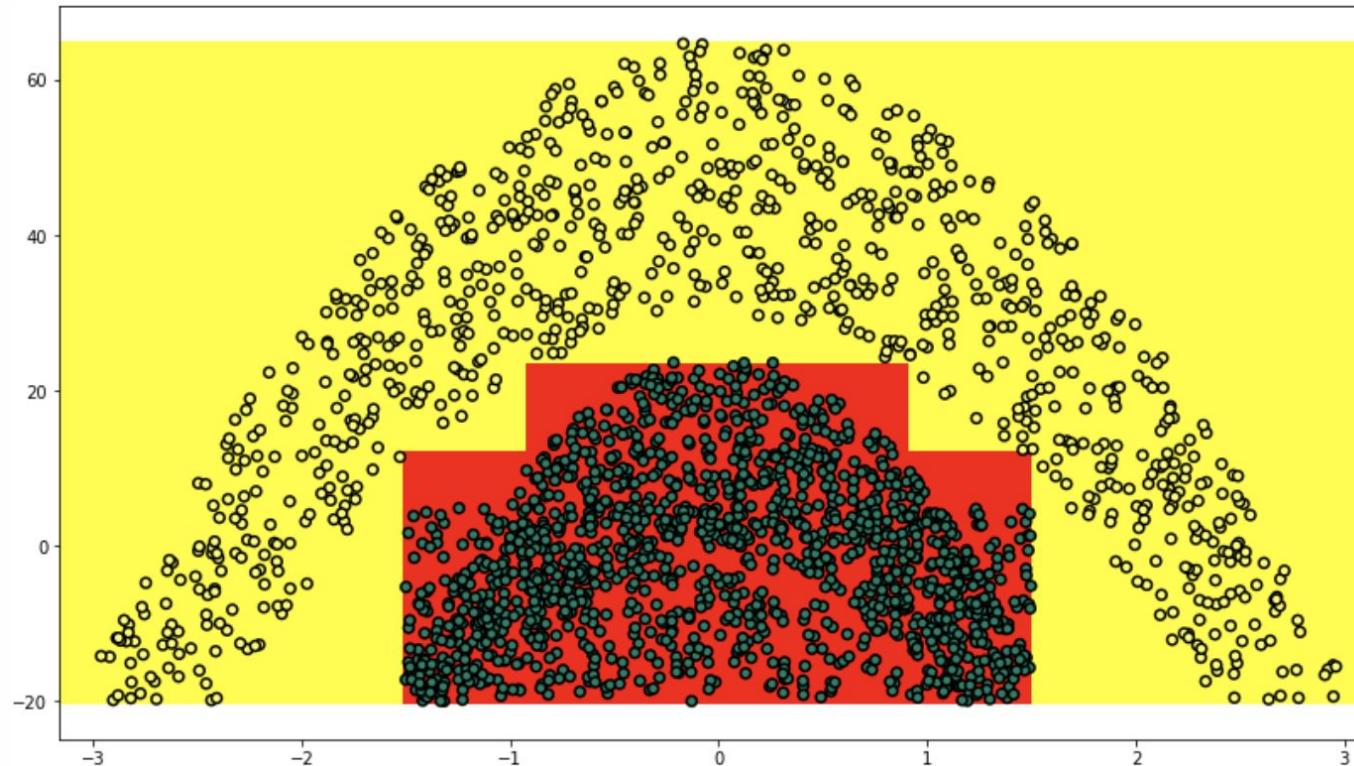
Logistic Regression, f-measure ~0.755



- Лог. рег. хорошо работает при линейной зависимости признаков и целевой переменной
- Экспериментировать с преобразованием признаков и добиться более хорошего качества, но такой подход является эвристиком и вы можете потратить много времени и при этом не получить желаемый результат

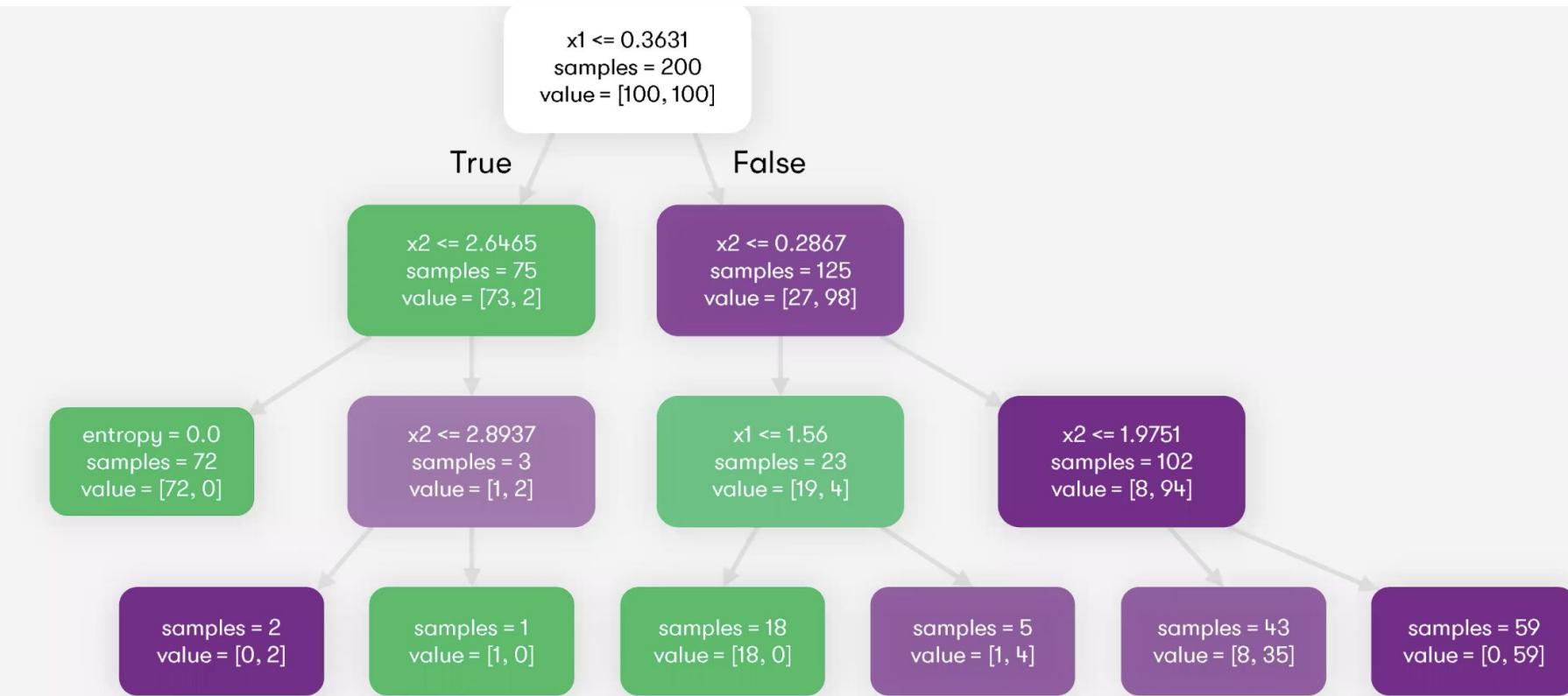
# Мотивация 3. Дерево решений и наш пример

Decision Tree, f-measure = 1



- разделяет пространство на многомерные прямоугольники (подпространства)
- в подпространстве формируется ответ на основе обучающей выборки

# Мотивация 4. Представление дерева



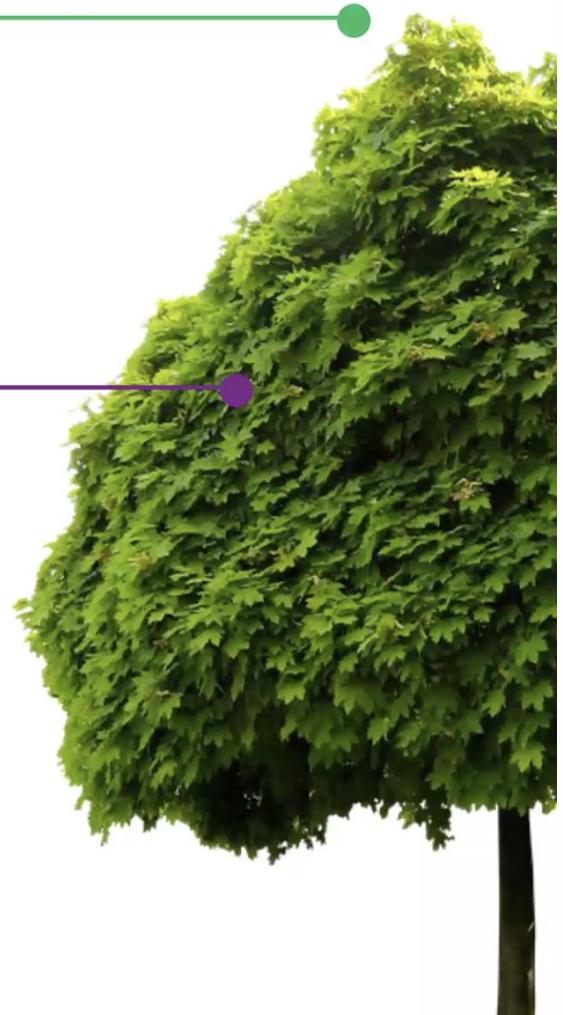
- Последовательность логических правил
- Константа в листьях

# Бинарное дерево решений

**Вершины** - логические правила

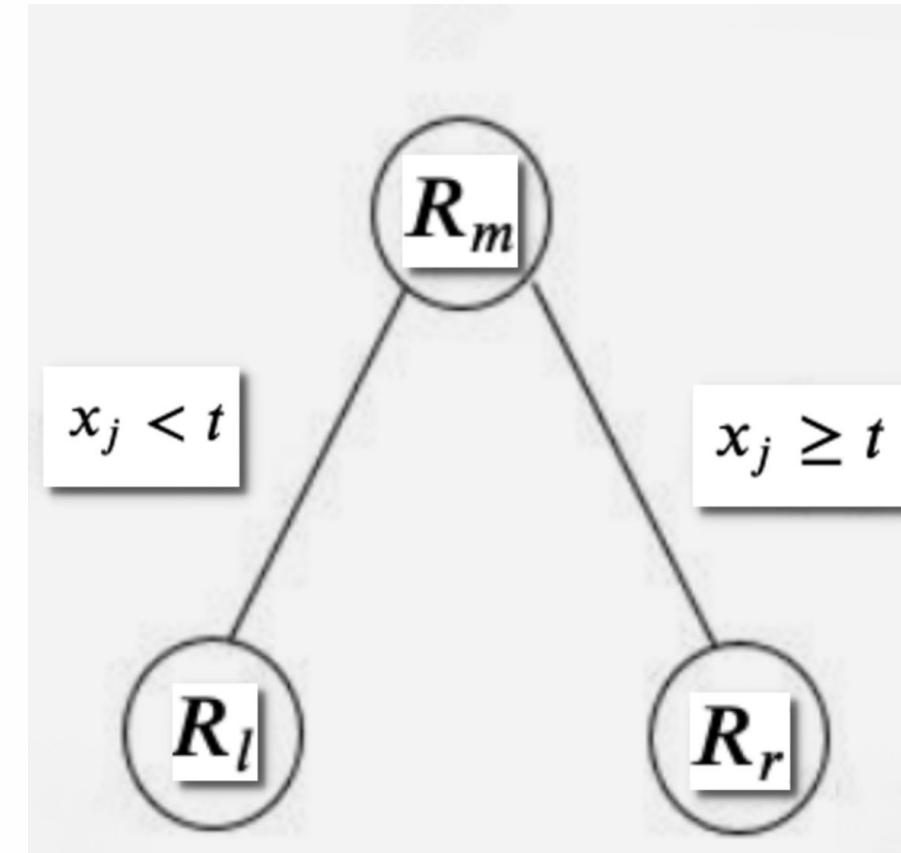
1. Кол-во этажей в доме  $\geq 5$ ?
2. Квартира студия?

**Листья** - предсказания в виде константы



# Метод построения решающего дерева определяется:

- Предикаты
- Критерий информативности
- Критерий останова
- Обработка пропущенных данных
- Стрижка



# Критерии информативности

$$Q(R_m, j, t) = H(R_m) - \frac{|R_l|}{|R_m|} H(R_l) - \frac{|R_r|}{|R_m|} H(R_r)$$

H - критерий информативности (impurity)

# Критерии информативности

$$Q(R_m, j, t) = H(R_m) - \frac{|R_l|}{|R_m|} H(R_l) - \frac{|R_r|}{|R_m|} H(R_r)$$

$H$  - критерий информативности (impurity)

$$H(R) = \min_{c \in Y} \frac{1}{|R|} \sum_{(x,y) \in R} L(y, c)$$

$R_m$  - выборка в текущей вершине

$j$  - индекс признака

$t$  - порог для признака

$L(y, c)$  - некоторая функция потерь

# Критерии информативности

$$Q(R_m, j, t) = H(R_m) - \frac{|R_l|}{|R_m|} H(R_l) - \frac{|R_r|}{|R_m|} H(R_r)$$

$H$  - критерий информативности (impurity)

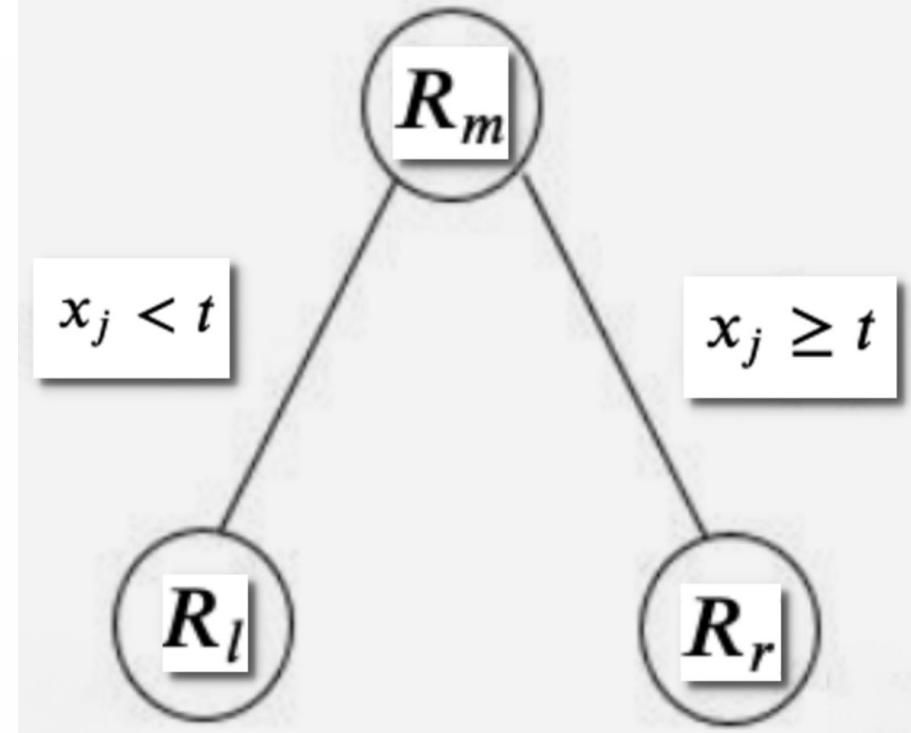
$$H(R) = \min_{c \in Y} \frac{1}{|R|} \sum_{(x,y) \in R} L(y, c)$$

$R_m$  - выборка в текущей вершине

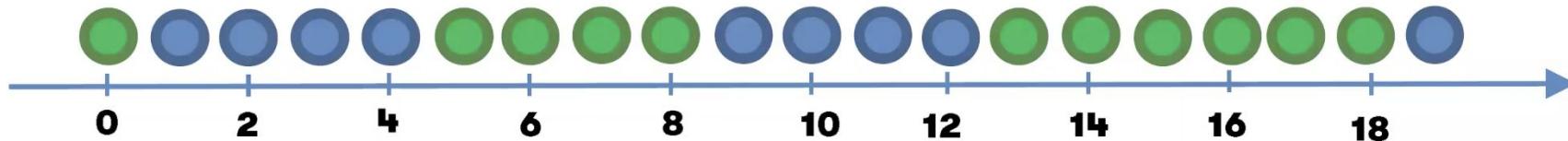
$j$  - индекс признака

$t$  - порог для признака

$L(y, c)$  - некоторая функция потерь



# Как выгоднее всего строить дерево?



# Как же определить меру беспорядка?

Энтропия Шеннона – мера беспорядка системы:

$$S = - \sum_{i=1}^N p_i \log_2 p_i,$$

В случае бинарной классификации:

$$S = -p_+ \log_2 p_+ - p_- \log_2 p_- = -p_+ \log_2 p_+ - (1 - p_+) \log_2 (1 - p_+);$$

Прирост информации:

$$IG(Q) = S_O - \sum_{i=1}^q \frac{N_i}{N} S_i,$$

# Энтропия в наших терминах

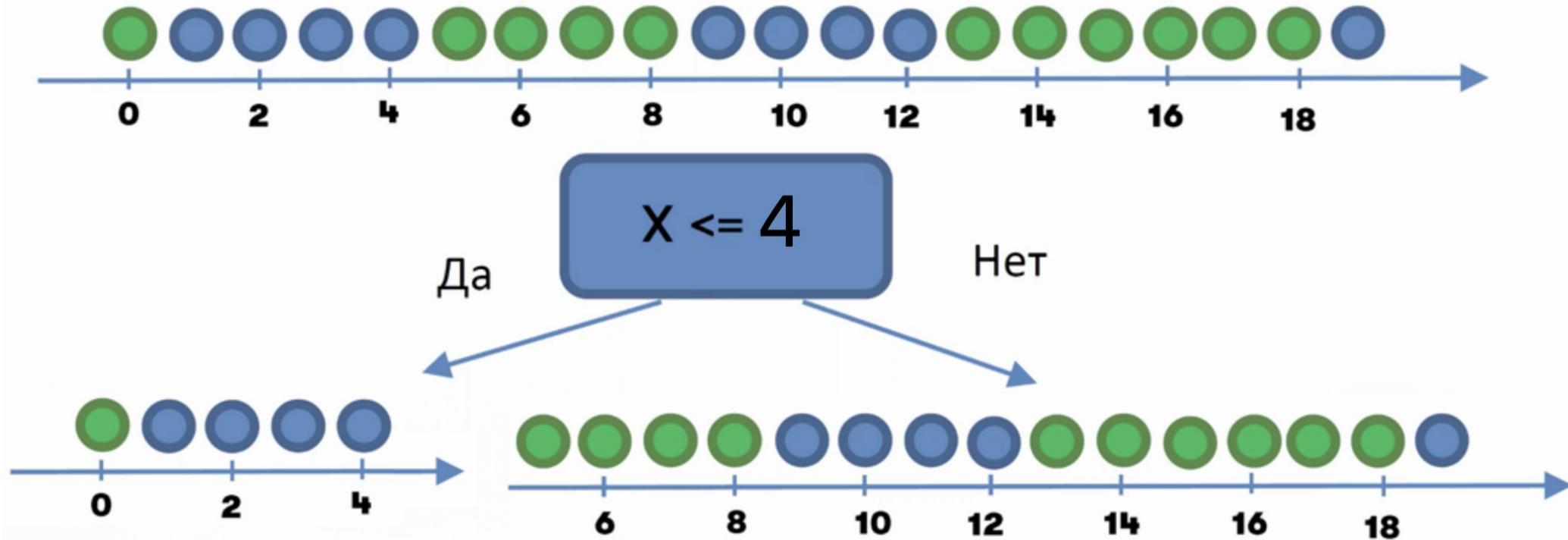
$$S_0 = H(R_m) \quad N = |R_m|$$

$$S_1 = H(R_l) \quad N_1 = |R_l|$$

$$S_2 = H(R_r) \quad N_2 = |R_r|$$

$$S = H(R) = - \sum_{i=1}^K p_i \log_2 p_i$$

# Как выгоднее всего строить дерево?



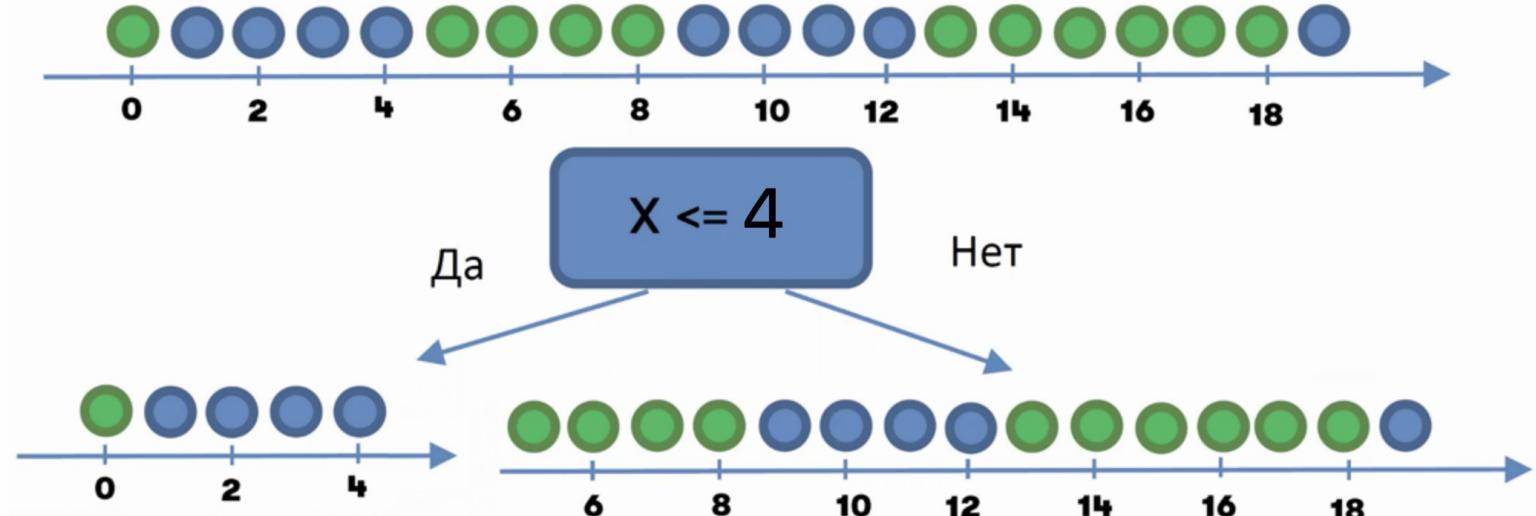
#022

# Как выгоднее всего строить дерево?

$$S = - \sum_{i=1}^N p_i \log_2 p_i$$

$$IG(Q) = S_0 - \sum_{i=1}^q \frac{N_i}{N} S_i$$

$$p_3 = ?, p_c = ?$$



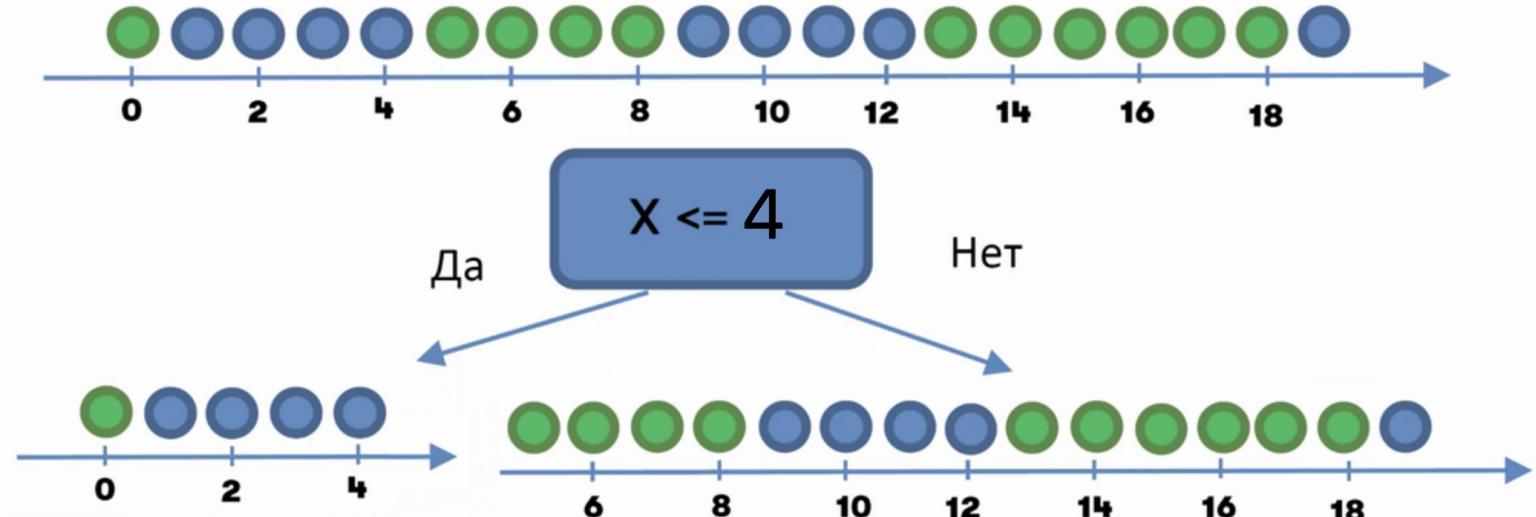
# Как выгоднее всего строить дерево?

$$S = - \sum_{i=1}^N p_i \log_2 p_i$$

$$IG(Q) = S_0 - \sum_{i=1}^q \frac{N_i}{N} S_i$$

$$p_z = \frac{11}{20}, p_c = \frac{9}{20}$$

$$S_0 = - \sum_{i=1}^N p_i \log_2 p_i = - \sum_{i=1}^2 p_i \log_2 p_i$$



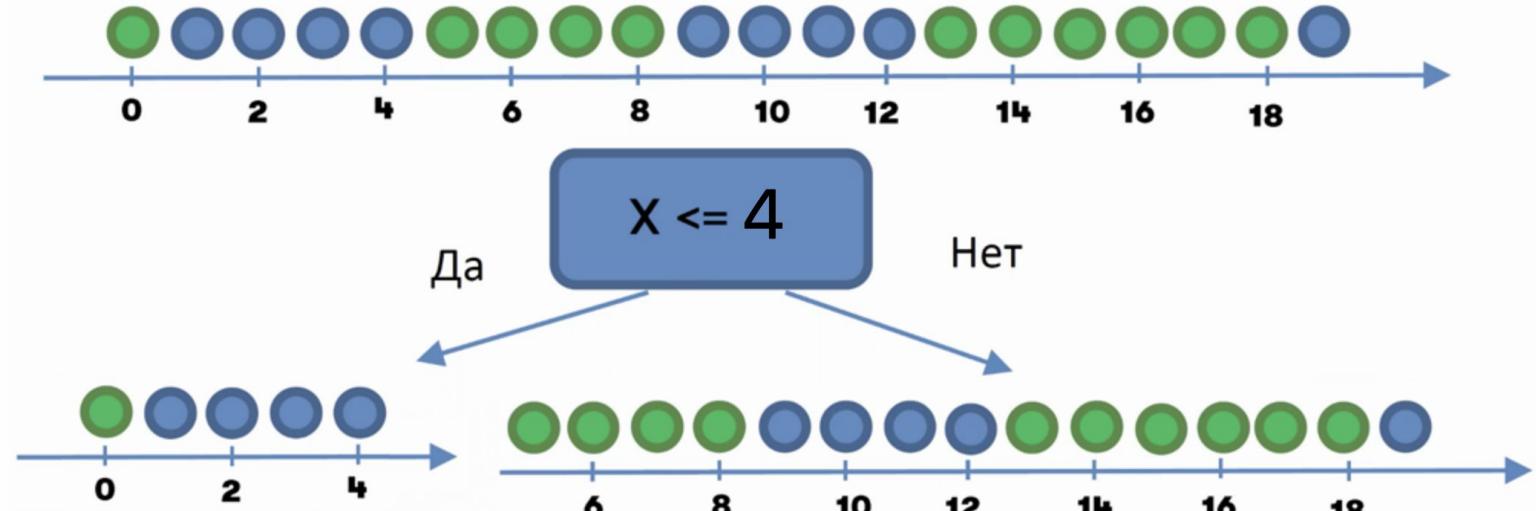
# Как выгоднее всего строить дерево?

$$S = - \sum_{i=1}^N p_i \log_2 p_i$$

$$IG(Q) = S_0 - \sum_{i=1}^q \frac{N_i}{N} S_i$$

$$p_z = \frac{11}{20}, p_c = \frac{9}{20}$$

$$S_0 = - \sum_{i=1}^N p_i \log_2 p_i = - \sum_{i=1}^2 p_i \log_2 p_i = -\frac{11}{20} \log_2 \frac{11}{20} - \frac{9}{20} \log_2 \frac{9}{20} \approx 0.99277$$



# Как выгоднее всего строить дерево?

$$S = - \sum_{i=1}^N p_i \log_2 p_i$$

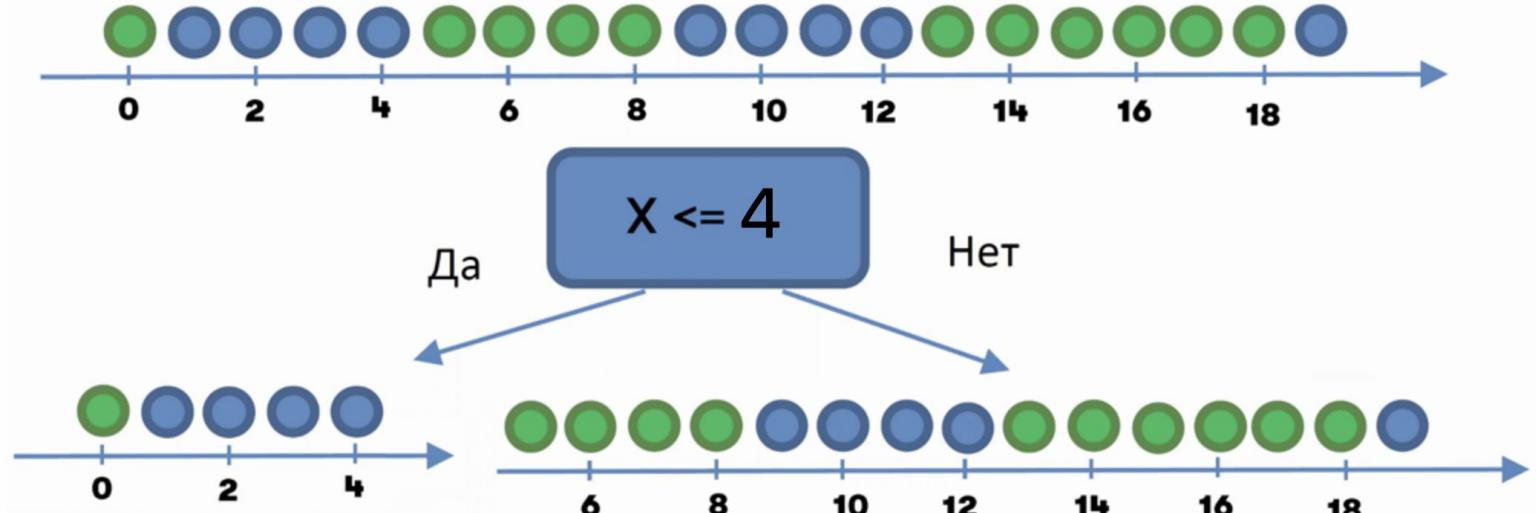
$$IG(Q) = S_0 - \sum_{i=1}^q \frac{N_i}{N} S_i$$

$$p_z = \frac{11}{20}, p_c = \frac{9}{20}$$

$$S_0 = - \sum_{i=1}^N p_i \log_2 p_i = - \sum_{i=1}^2 p_i \log_2 p_i = -\frac{11}{20} \log_2 \frac{11}{20} - \frac{9}{20} \log_2 \frac{9}{20} \approx 0.99277$$

$$S_1 = ?$$

$$S_2 = ?$$



#026

# Как выгоднее всего строить дерево?

$$S = - \sum_{i=1}^N p_i \log_2 p_i$$

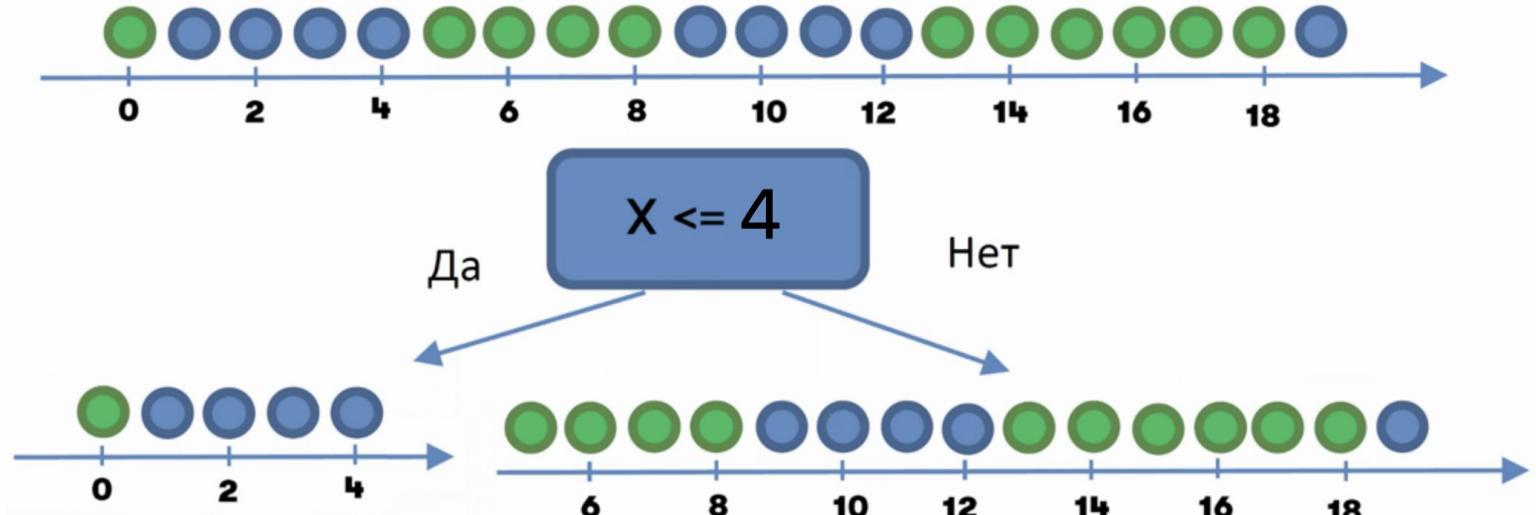
$$IG(Q) = S_0 - \sum_{i=1}^q \frac{N_i}{N} S_i$$

$$p_z = \frac{11}{20}, p_c = \frac{9}{20}$$

$$S_0 = - \sum_{i=1}^N p_i \log_2 p_i = - \sum_{i=1}^2 p_i \log_2 p_i = -\frac{11}{20} \log_2 \frac{11}{20} - \frac{9}{20} \log_2 \frac{9}{20} \approx 0.99277$$

$$S_1 = -\frac{1}{5} \log_2 \frac{1}{5} - \frac{4}{5} \log_2 \frac{4}{5} \approx 0.72193$$

$$S_2 = -\frac{10}{15} \log_2 \frac{10}{15} - \frac{5}{15} \log_2 \frac{5}{15} \approx 0.9183$$



# Как выгоднее всего строить дерево?

$$S = - \sum_{i=1}^N p_i \log_2 p_i$$

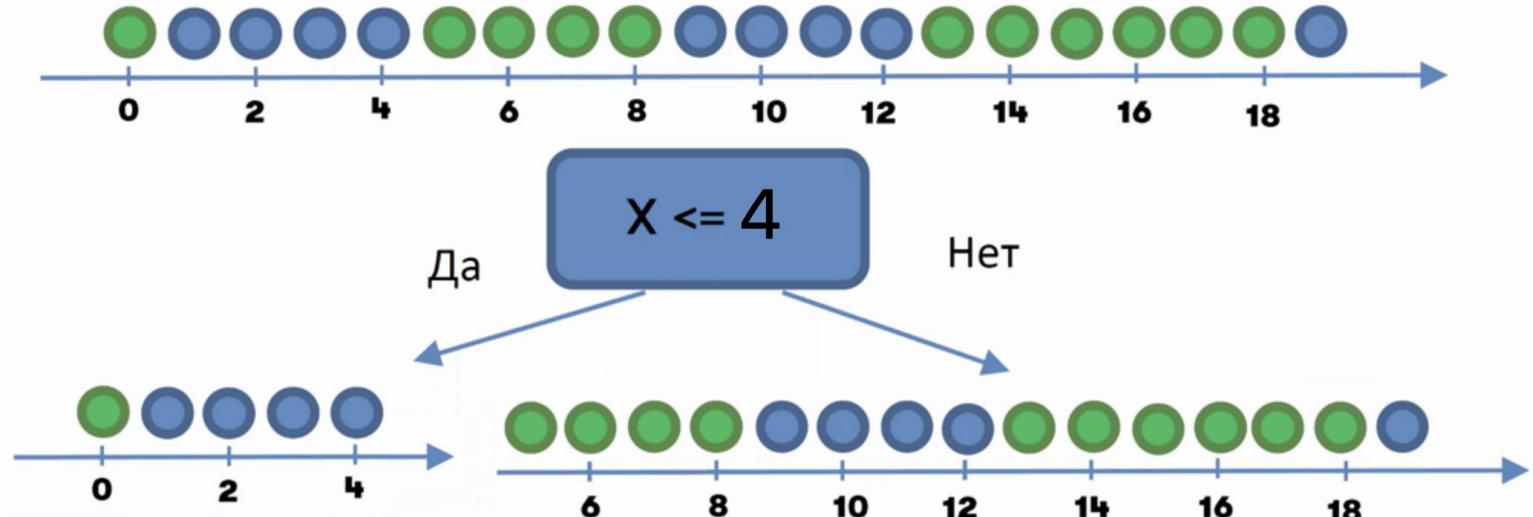
$$IG(Q) = S_0 - \sum_{i=1}^q \frac{N_i}{N} S_i$$

$$p_z = \frac{11}{20}, p_c = \frac{9}{20}$$

$$S_0 = - \sum_{i=1}^N p_i \log_2 p_i = - \sum_{i=1}^2 p_i \log_2 p_i = -\frac{11}{20} \log_2 \frac{11}{20} - \frac{9}{20} \log_2 \frac{9}{20} \approx 0.99277$$

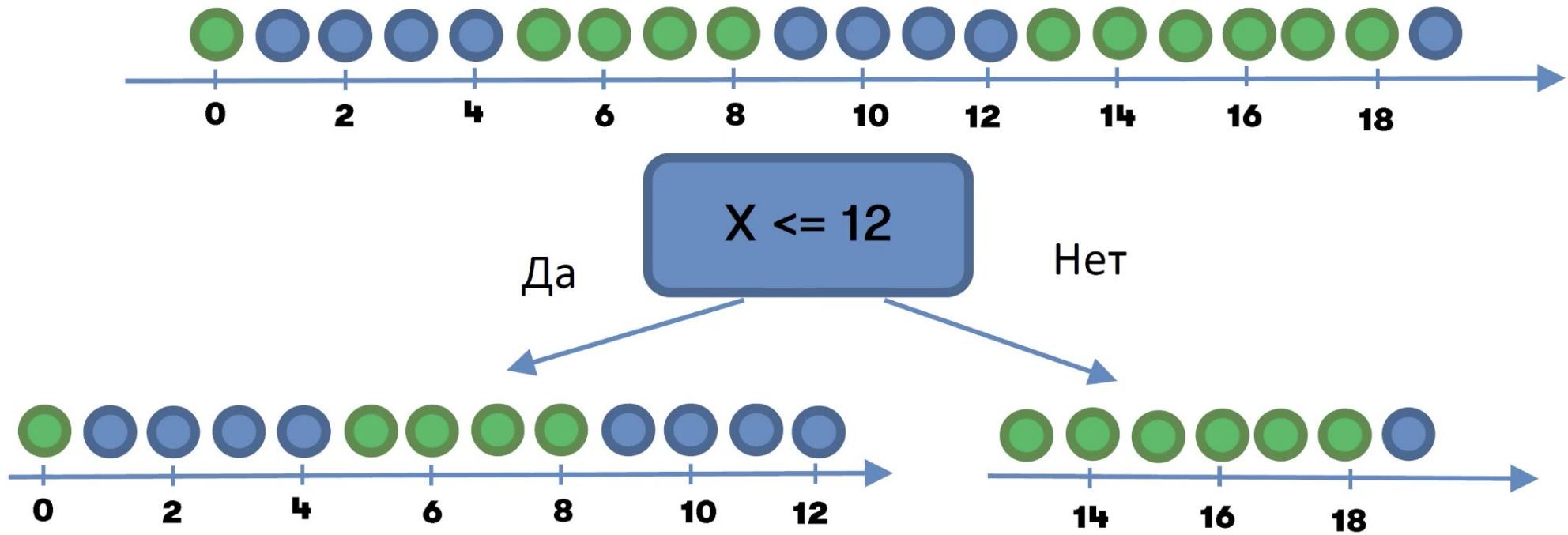
$$S_1 = -\frac{1}{5} \log_2 \frac{1}{5} - \frac{4}{5} \log_2 \frac{4}{5} \approx 0.72193$$

$$S_2 = -\frac{10}{15} \log_2 \frac{10}{15} - \frac{5}{15} \log_2 \frac{5}{15} \approx 0.9183$$



$$IG(Q) = S_0 - \sum_{i=1}^q \frac{N_i}{N} S_i = S_0 - \frac{5}{20} S_1 - \frac{15}{20} S_2 \approx 0.1236$$

# Как выгоднее всего строить дерево?



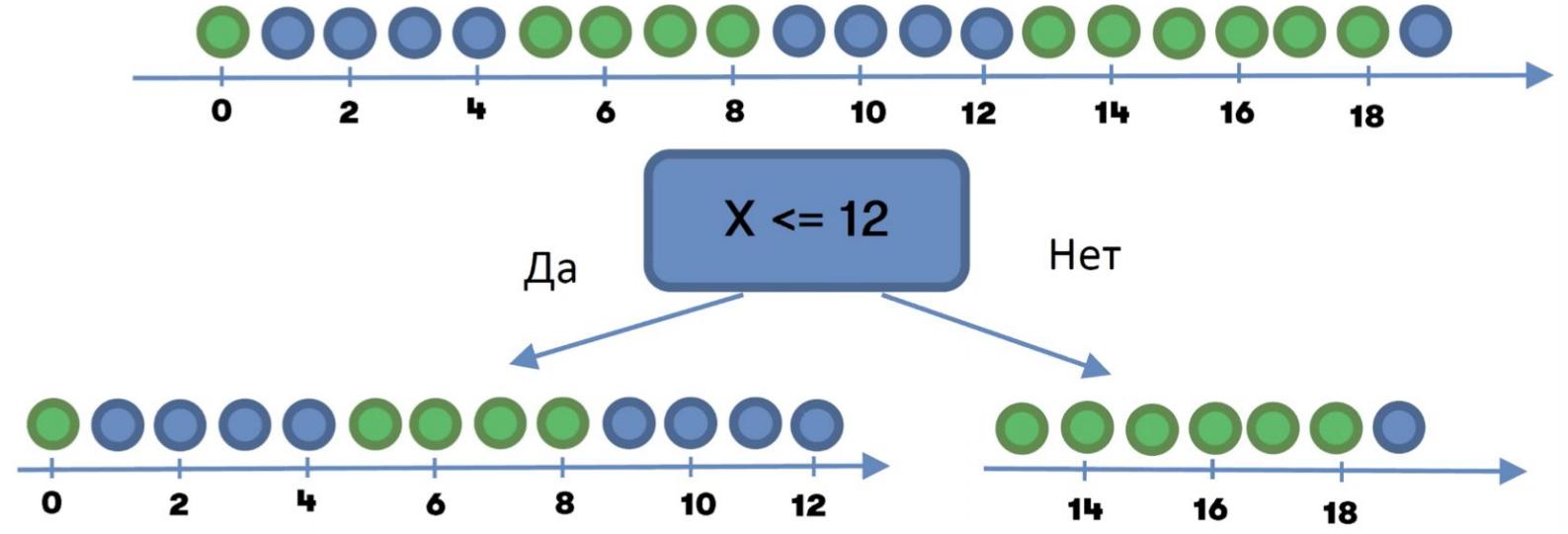
# Как выгоднее всего строить дерево?

$$S = - \sum_{i=1}^N p_i \log_2 p_i$$

$$IG(Q) = S_0 - \sum_{i=1}^q \frac{N_i}{N} S_i$$

$$p_3 = \frac{11}{20}, p_c = \frac{9}{20}$$

$$S_0 = - \sum_{i=1}^N p_i \log_2 p_i = - \sum_{i=1}^2 p_i \log_2 p_i = -\frac{11}{20} \log_2 \frac{11}{20} - \frac{9}{20} \log_2 \frac{9}{20} \approx 0.99277$$



# Как выгоднее всего строить дерево?

$$S = - \sum_{i=1}^N p_i \log_2 p_i$$

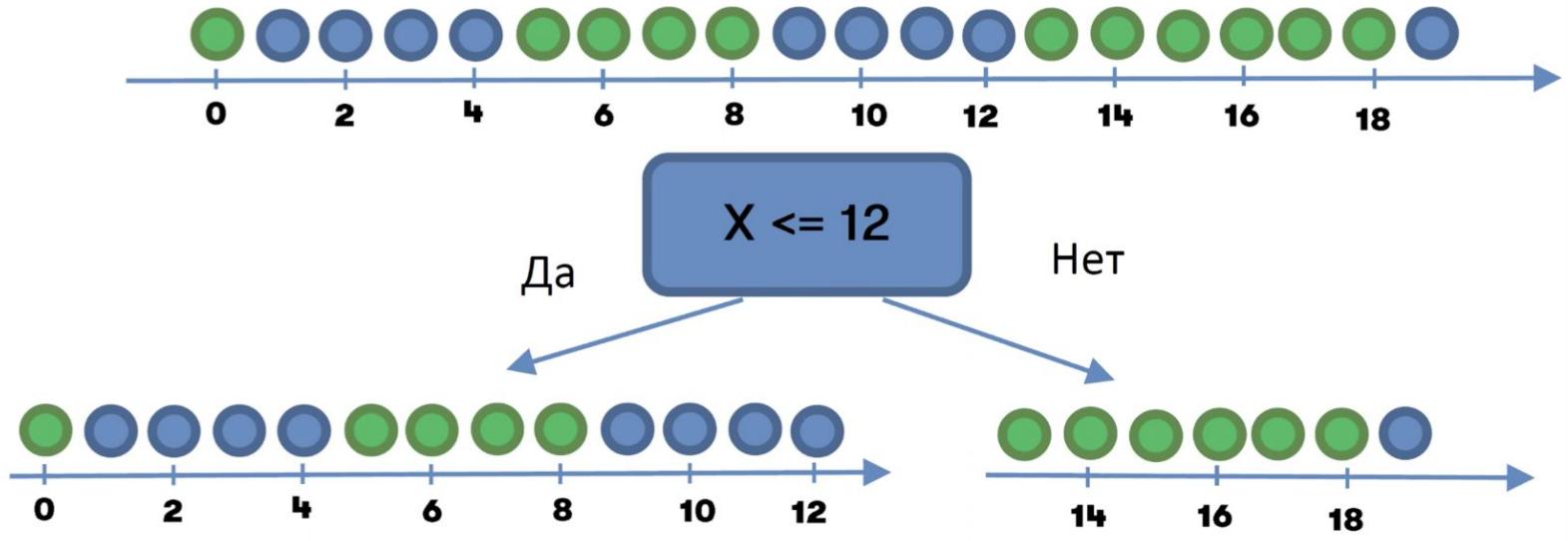
$$IG(Q) = S_0 - \sum_{i=1}^q \frac{N_i}{N} S_i$$

$$p_z = \frac{11}{20}, p_c = \frac{9}{20}$$

$$S_0 = - \sum_{i=1}^N p_i \log_2 p_i = - \sum_{i=1}^2 p_i \log_2 p_i = -\frac{11}{20} \log_2 \frac{11}{20} - \frac{9}{20} \log_2 \frac{9}{20} \approx 0.99277$$

$$S_1 = -\frac{5}{13} \log_2 \frac{5}{13} - \frac{8}{13} \log_2 \frac{8}{13} \approx 0.9612$$

$$S_2 = -\frac{6}{7} \log_2 \frac{6}{7} - \frac{1}{7} \log_2 \frac{1}{7} \approx 0.5917$$

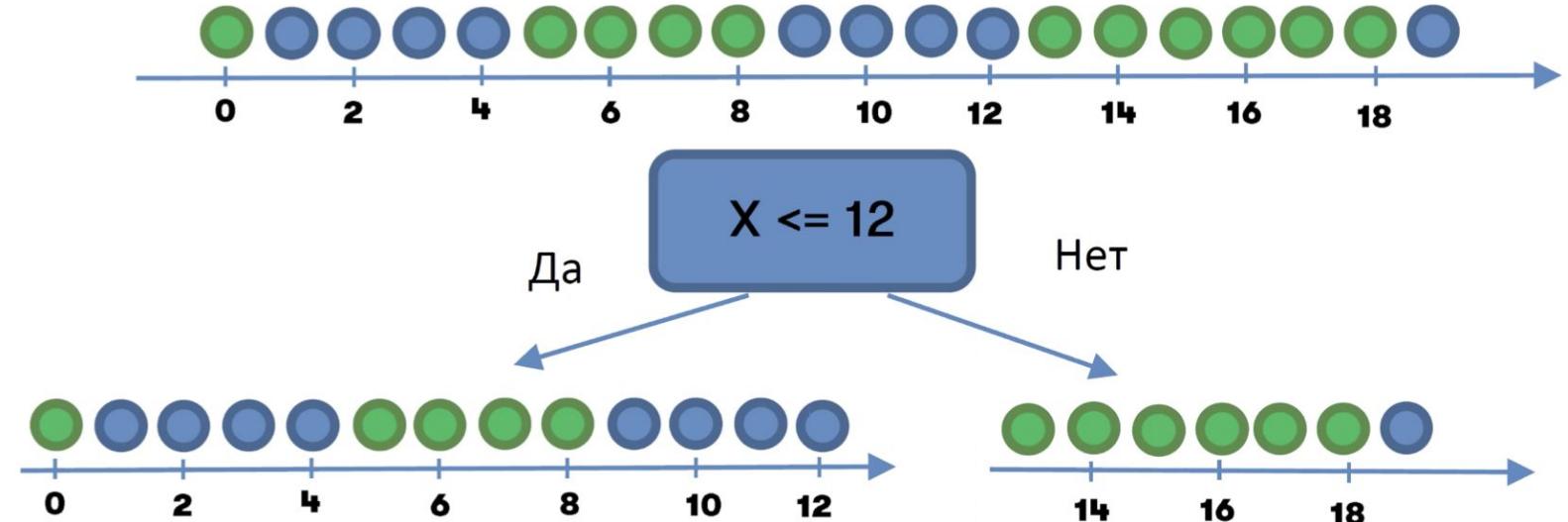


$$IG(Q) = S_0 - \sum_{i=1}^q \frac{N_i}{N} S_i = S_0 - \frac{13}{20} S_1 - \frac{7}{20} S_2 \approx 0.1609$$

# Как выгоднее всего строить дерево?

$$S = - \sum_{i=1}^N p_i \log_2 p_i$$

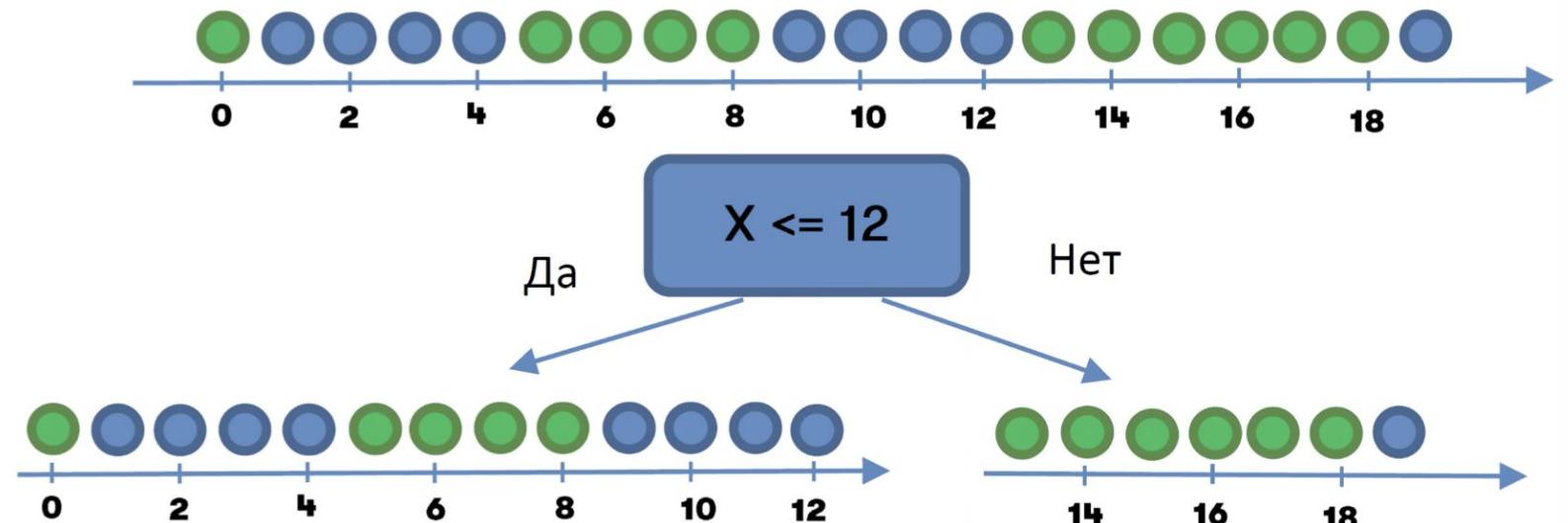
$$IG(Q) = S_0 - \sum_{i=1}^q \frac{N_i}{N} S_i$$



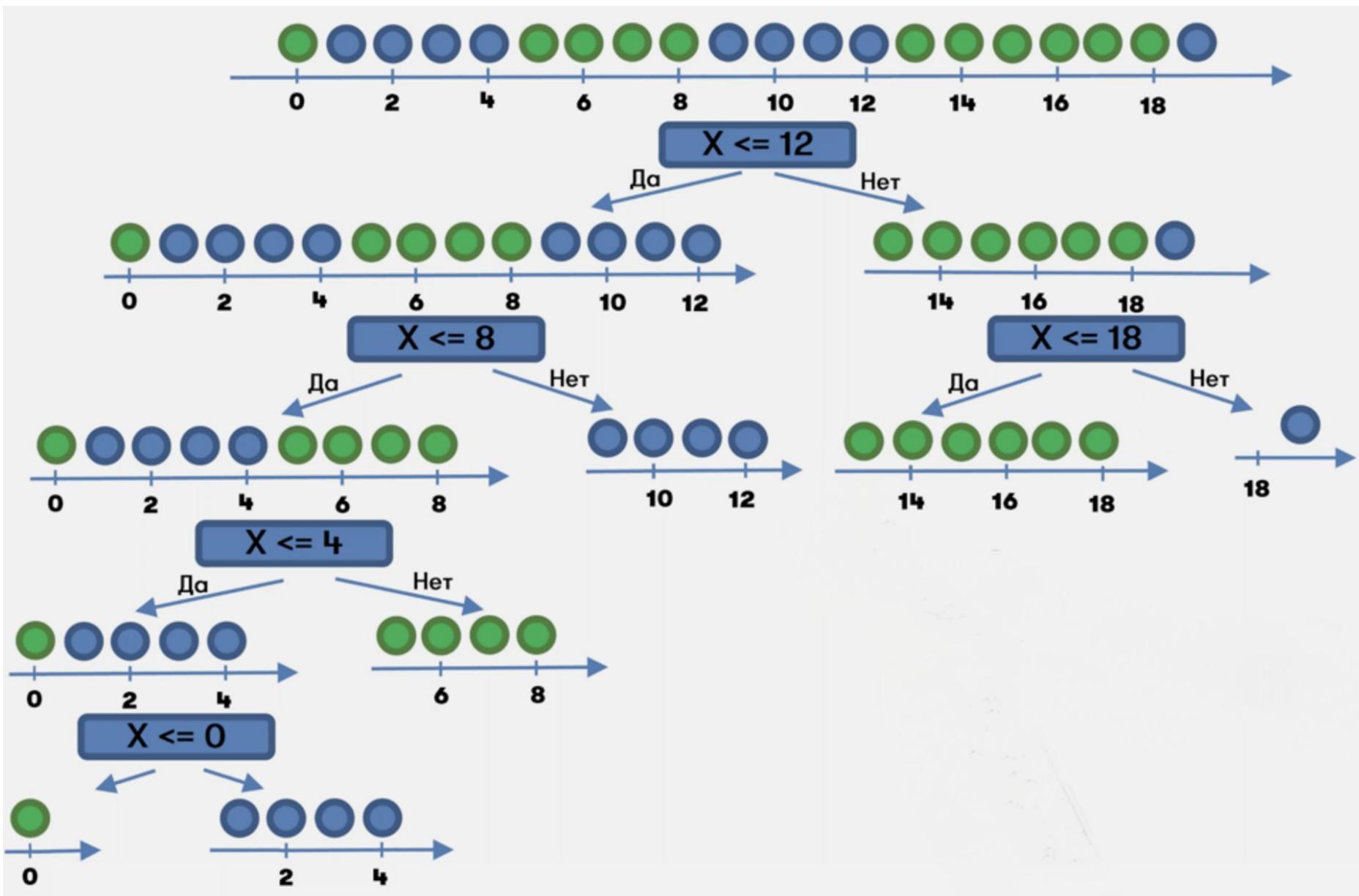
Выбираем разбиение с максимальным  $IG(Q)$

# Как выгоднее всего строить дерево?

Признак	IG(Q)
0	0.044674
1	0.000806
2	0.024225
3	0.066972
4	0.123571
5	0.059086
6	0.023141
7	0.004853
8	0.000074
9	0.007299
10	0.032825
11	0.080342
12	0.160885
13	0.108108
14	0.064699
15	0.030519
16	0.007153
17	0.000806
18	0.059931



#033



#034

# Алгоритм построения дерева

$$S = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\} \quad \mathbf{x} = (x_1, \dots, x_d)$$

$$x_j, y \in \{0, 1\}$$

GROWTREE( $S$ )

**if** ( $y = 0$  for all  $\langle \mathbf{x}, y \rangle \in S$ ) **return** new leaf(0)

**else if** ( $y = 1$  for all  $\langle \mathbf{x}, y \rangle \in S$ ) **return** new leaf(1)

**else**

choose best attribute  $x_j$

$S_0 = \text{all } \langle \mathbf{x}, y \rangle \in S \text{ with } x_j = 0;$

$S_1 = \text{all } \langle \mathbf{x}, y \rangle \in S \text{ with } x_j = 1;$

**return** new node( $x_j$ , GROWTREE( $S_0$ ), GROWTREE( $S_1$ ))



# Кроме энтропии:

Джини:

$$G = 1 - \sum_k (p_k)^2$$

$$G = 1 - p_+^2 - p_-^2 = 1 - p_+^2 - (1 - p_+)^2 = 2p_+(1 - p_+).$$

Misclassification error:

$$E = 1 - \max_k p_k$$

## Рассмотрим индикатор ошибки как функцию потерь

$$Q(R_m, j, t) = H(R_m) - \frac{|R_l|}{|R_m|} H(R_l) - \frac{|R_r|}{|R_m|} H(R_r)$$

$H$  - критерий информативности (impurity)

$$H(R) = \min_{c \in Y} \frac{1}{|R|} \sum_{(x,y) \in R} L(y, c)$$

$$L(y, c) = [y \neq c]$$

# Рассмотрим индикатор ошибки как функцию потерь

$$Q(R_m, j, t) = H(R_m) - \frac{|R_l|}{|R_m|} H(R_l) - \frac{|R_r|}{|R_m|} H(R_r)$$

$H$  - критерий информативности (impurity)

$$H(R) = \min_{c \in Y} \frac{1}{|R|} \sum_{(x,y) \in R} [y \neq c]$$

Как выбрать  $c$ ?

## Рассмотрим индикатор ошибки как функцию потерь

$$Q(R_m, j, t) = H(R_m) - \frac{|R_l|}{|R_m|} H(R_l) - \frac{|R_r|}{|R_m|} H(R_r)$$

$H$  - критерий информативности (impurity)

$$H(R) = \min_{c \in Y} \frac{1}{|R|} \sum_{(x,y) \in R} [y \neq c]$$

$c^*$  - самый популярный класс

# Рассмотрим индикатор ошибки как функцию потерь

$$Q(R_m, j, t) = H(R_m) - \frac{|R_l|}{|R_m|} H(R_l) - \frac{|R_r|}{|R_m|} H(R_r)$$

$H$  - критерий информативности (impurity)

$$H(R) = \min_{c \in Y} \frac{1}{|R|} \sum_{(x,y) \in R} [y \neq c]$$

$c = k_*$  - самый популярный класс

$$H(R) = \min_{c \in Y} \frac{1}{|R|} \sum_{(x,y) \in R} [y \neq k_*] = 1 - \max_{c \in Y} \frac{1}{|R|} \sum_{(x,y) \in R} [y = k_*] = 1 - p_{k_*} = 1 - \max_k p_k$$

# Рассмотрим индикатор ошибки как функцию потерь

$$Q(R_m, j, t) = H(R_m) - \frac{|R_l|}{|R_m|} H(R_l) - \frac{|R_r|}{|R_m|} H(R_r)$$

$H$  - критерий информативности (impurity)

$$H(R) = \min_{c \in Y} \frac{1}{|R|} \sum_{(x,y) \in R} [y \neq c]$$

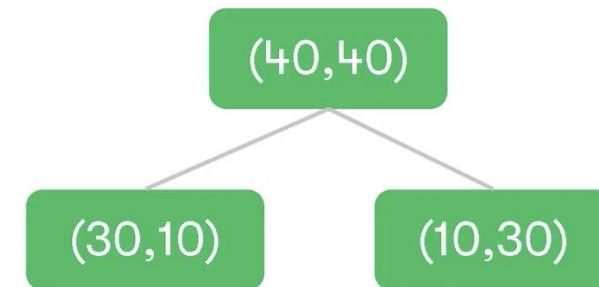
$c = k_*$  - самый популярный класс

$$H(R) = \min_{c \in Y} \frac{1}{|R|} \sum_{(x,y) \in R} [y \neq k_*] = 1 - \max_{c \in Y} \frac{1}{|R|} \sum_{(x,y) \in R} [y = k_*] = 1 - p_{k_*} = 1 - \max_k p_k$$

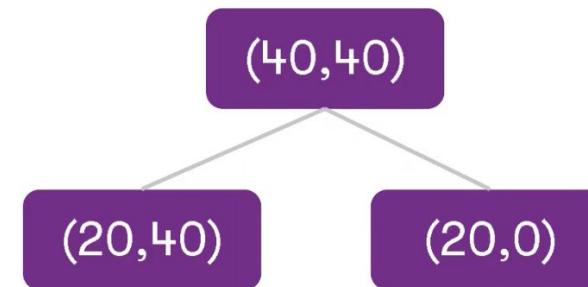
#041      Данный критерий является достаточно грубым, поскольку учитывает частоту лишь одного класса.

# Пример. Чувствительность КИ для классификации

**A**

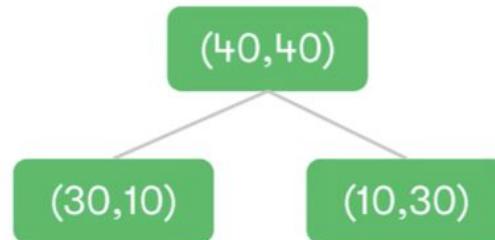


**B**

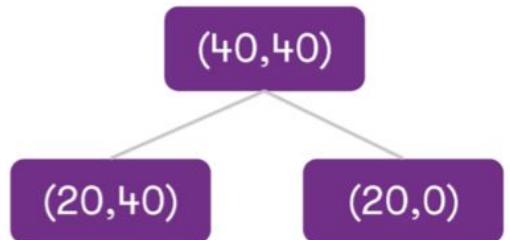


# Энтропия:

**A**



**B**



A:

$$S_0 = - \sum_{i=1}^N p_i \log_2 p_i = - \sum_{i=1}^2 p_i \log_2 p_i = -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = 1$$

$$S_1 =$$

$$S_2 =$$

$$IG(Q) = S_0 - \sum_{i=1}^q \frac{N_i}{N} S_i =$$

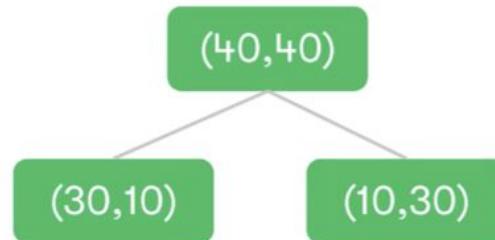
B:  $S_1 \approx 0.91$

$$S_2 \approx 0$$

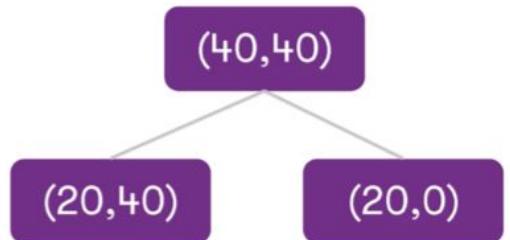
$$IG(Q) \approx 0.31$$

# Энтропия:

**A**



**B**



A:

$$S_0 = - \sum_{i=1}^N p_i \log_2 p_i = - \sum_{i=1}^2 p_i \log_2 p_i = -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = 1$$

$$S_1 = -\frac{3}{4} \log_2 \frac{3}{4} - \frac{1}{4} \log_2 \frac{1}{4} \approx 0.8113$$

$$S_2 = -\frac{1}{4} \log_2 \frac{1}{4} - \frac{3}{4} \log_2 \frac{3}{4} \approx 0.8113$$

$$IG(Q) = S_0 - \sum_{i=1}^q \frac{N_i}{N} S_i = 1 - \frac{1}{2} S_1 - \frac{1}{2} S_2 \approx 0.1887$$

B:  $S_1 \approx 0.91$

$$S_2 \approx 0$$

$$IG(Q) \approx 0.31$$

## Джини:

$$G = 1 - \sum_k (p_k)^2$$

A:  $G = 1 - ((0, 5)^2 + (0.5)^2) = 0.5$

$$G_1 =$$

$$G_2 =$$

$$IG =$$

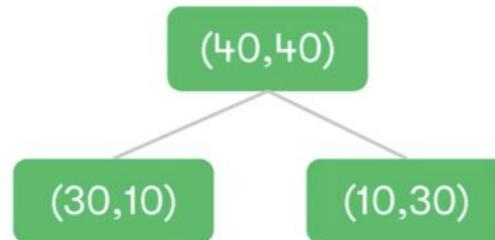
$$G = 1 - ((0, 5)^2 + (0.5)^2) = 0.5$$

B:  $G_1 =$

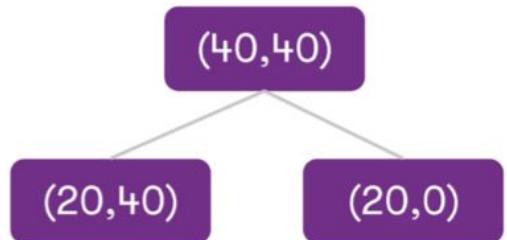
$$G_2 =$$

$$IG =$$

**A**



**B**



## Джини:

$$G = 1 - \sum_k (p_k)^2$$

A:  $G = 1 - ((0, 5)^2 + (0.5)^2) = 0.5$

$$G_1 = 1 - ((0, 25)^2 + (0.75)^2) = 0.375$$

$$G_2 = 1 - ((0, 75)^2 + (0.25)^2) = 0.375$$

$$IG = 0.5 - \frac{1}{2}0.375 - \frac{1}{2}0.375 \approx 0.125$$

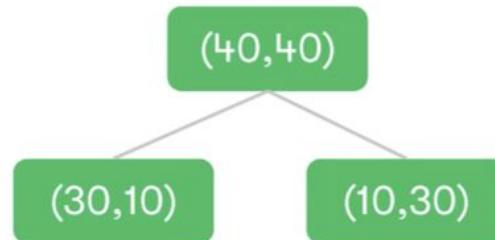
$$G = 1 - ((0, 5)^2 + (0.5)^2) = 0.5$$

B:  $G_1 =$

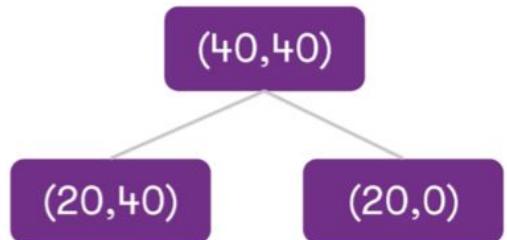
$$G_2 =$$

$$IG =$$

**A**



**B**



## Джини:

$$G = 1 - \sum_k (p_k)^2$$

A:  $G = 1 - ((0, 5)^2 + (0.5)^2) = 0.5$

$$G_1 = 1 - ((0, 25)^2 + (0.75)^2) = 0.375$$

$$G_2 = 1 - ((0, 75)^2 + (0.25)^2) = 0.375$$

$$IG = 0.5 - \frac{1}{2}0.375 - \frac{1}{2}0.375 \approx 0.125$$

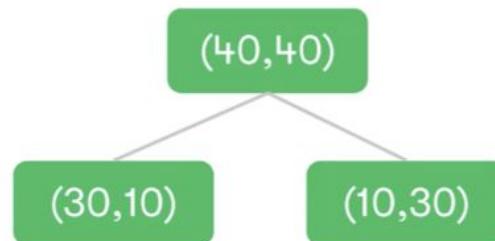
$$G = 1 - ((0, 5)^2 + (0.5)^2) = 0.5$$

B:  $G_1 = 1 - ((\frac{1}{3})^2 + (\frac{2}{3})^2) \approx 0.44$

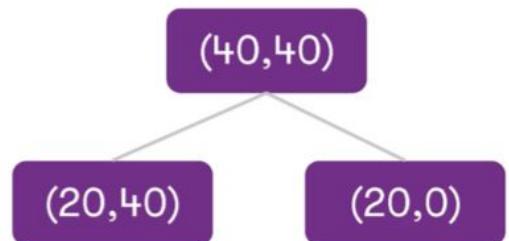
$$G_2 = 1 - ((1)^2 + (0)^2) = 0$$

$$IG = 0.5 - \frac{3}{4}0.44 \approx 0.17$$

**A**



**B**



# Missclassification error

$$E = 1 - \max_k p_k$$

A:  $E = 1 - 0.5 = 0.5$

$$E_1 =$$

$$E_2 =$$

$$IG =$$

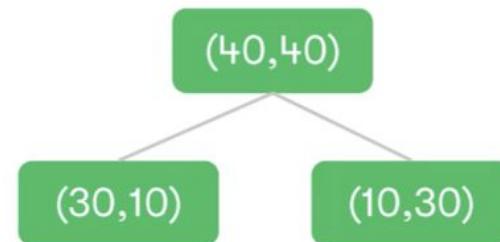
$$E = 1 - 0.5 = 0.5$$

B:  $E_1 =$

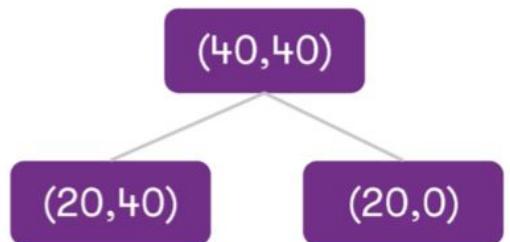
$$E_2 =$$

$$IG =$$

**A**



**B**



# Missclassification error

$$E = 1 - \max_k p_k$$

A:  $E = 1 - 0.5 = 0.5$

$$E_1 = 1 - \frac{3}{4} = 0.25$$

$$E_2 = 1 - \frac{3}{4} = 0.25$$

$$IG = 0.5 - \frac{1}{2}0.25 - \frac{1}{2}0.25 = 0.25$$

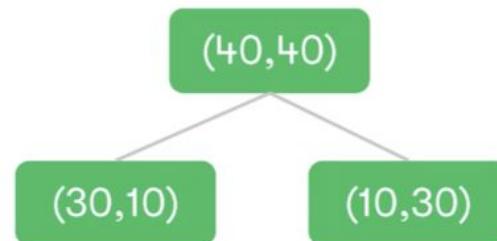
$$E = 1 - 0.5 = 0.5$$

B:  $E_1 = 1 - \frac{4}{6} \approx \frac{1}{3}$

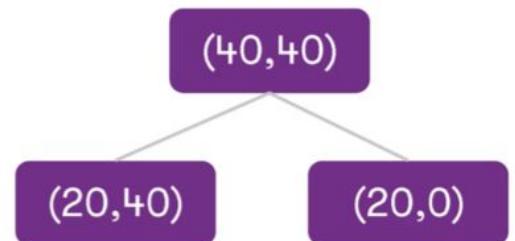
$$E_2 = 1 - 1 = 0$$

$$IG = 0.5 - \frac{3}{4} \frac{1}{3} = 0.25$$

**A**



**B**



# Регрессия

Как обычно, в регрессии выберем квадрат отклонения в качестве функции потерь. В этом случае критерий информативности будет выглядеть как

$$H(R) = \min_{c \in \mathbb{Y}} \frac{1}{|R|} \sum_{(x_i, y_i) \in R} (y_i - c)^2.$$

Как известно, минимум в этом выражении будет достигаться на среднем значении целевой переменной. Значит, критерий можно переписать в следующем виде:

$$H(R) = \frac{1}{|R|} \sum_{(x_i, y_i) \in R} \left( y_i - \frac{1}{|R|} \sum_{(x_j, y_j) \in R} y_j \right)^2.$$

# Регрессия

Как обычно, в регрессии выберем квадрат отклонения в качестве функции потерь. В этом случае критерий информативности будет выглядеть как

$$H(R) = \min_{c \in \mathbb{Y}} \frac{1}{|R|} \sum_{(x_i, y_i) \in R} (y_i - c)^2.$$

Как известно, минимум в этом выражении будет достигаться на среднем значении целевой переменной. Значит, критерий можно переписать в следующем виде:

$$H(R) = \frac{1}{|R|} \sum_{(x_i, y_i) \in R} \left( y_i - \frac{1}{|R|} \sum_{(x_j, y_j) \in R} y_j \right)^2.$$

Мы получили, что информативность вершины измеряется её дисперсией — чем ниже разброс целевой переменной, тем лучше вершина. Разумеется, можно использовать и другие функции ошибки  $L$  — например, при выборе абсолютного отклонения мы получим в качестве критерия среднее абсолютное отклонение от медианы.

# Критерии останова

Можно придумать большое количество критериев останова. Перечислим некоторые ограничения и критерии:

- Ограничение максимальной глубины дерева.
- Ограничение минимального числа объектов в листе.
- Ограничение максимального количества листьев в дереве.
- Останов в случае, если все объекты в листе относятся к одному классу.
- Требование, что функционал качества при дроблении улучшался как минимум на  $s$  процентов.

С помощью грамотного выбора подобных критериев и их параметров можно существенно повлиять на качество дерева. Тем не менее, такой подбор является трудозатратным и требует проведения кросс-валидации.

# Популярные методы построения деревьев

- ID3: использует энтропийный критерий. Строит дерево до тех пор, пока в каждом листе не окажутся объекты одного класса, либо пока разбиение вершины дает уменьшение энтропийного критерия.
- C4.5: использует критерий Gain Ratio (нормированный энтропийный критерий). Критерий останова— ограничение на число объектов в листе. Стрижка производится с помощью метода Error-Based Pruning, который использует оценки обобщающей способности для принятия решения об удалении вершины. Обработка пропущенных значений осуществляется с помощью метода, который игнорирует объекты с пропущенными значениями при вычислении критерия ветвления, а затем переносит такие объекты в оба поддерева с определенными весами.
- CART: использует критерий Джини. Стрижка осуществляется с помощью Cost-Complexity Pruning. Для обработки пропусков используется метод суррогатных предикатов.

# Класс DecisionTreeClassifier в Scikit-learn

Основные параметры класса `sklearn.tree.DecisionTreeClassifier`:

- `max_depth` — максимальная глубина дерева
- `max_features` — максимальное число признаков, по которым ищется лучшее разбиение в дереве (это нужно потому, что при большом количестве признаков будет "дорого" искать лучшее (по критерию типа прироста информации) разбиение среди всех признаков)
- `min_samples_leaf` — минимальное число объектов в листе. У этого параметра есть понятная интерпретация: скажем, если он равен 5, то дерево будет порождать только те классифицирующие правила, которые верны как минимум для 5 объектов

Параметры дерева надо настраивать в зависимости от входных данных, и делается это обычно с помощью *кросс-валидации*

# Плюсы и минусы деревьев решений

## Плюсы и минусы деревьев решений

### Плюсы:

- Порождение четких правил классификации, понятных человеку, например, "если возраст < 25 и интерес к мотоциклам, то отказать в кредите". Это свойство называют интерпретируемостью модели;
- Деревья решений могут легко визуализироваться, то есть может "интерпретироваться" (строгого определения я не видел) как сама модель (дерево), так и прогноз для отдельного взятого тестового объекта (путь в дереве);
- Быстрые процессы обучения и прогнозирования;
- Малое число параметров модели;
- Поддержка числовых, и категориальных признаков.

# Минусы

- У порождения четких правил классификации есть и другая сторона: деревья очень чувствительны к шумам во входных данных, вся модель может кардинально измениться, если немного изменится обучающая выборка (например, если убрать один из признаков или добавить несколько объектов), поэтому и правила классификации могут сильно изменяться, что ухудшает интерпретируемость модели;
- Разделяющая граница, построенная деревом решений, имеет свои ограничения (состоит из гиперплоскостей, перпендикулярных какой-то из координатной оси), и на практике дерево решений по качеству классификации уступает некоторым другим методам;
- Необходимость отсекать ветви дерева (pruning) или устанавливать минимальное число элементов в листьях дерева или максимальную глубину дерева для борьбы с переобучением. Впрочем, переобучение — проблема всех методов машинного обучения;
- Нестабильность. Небольшие изменения в данных могут существенно изменять построенное дерево решений. С этой проблемой борются с помощью ансамблей деревьев решений (рассмотрим далее);
- Сложно поддерживаются пропуски в данных. Friedman оценил, что на поддержку пропусков в данных ушло около 50% кода CART
- Модель умеет только интерполировать, но не экстраполировать (это же верно и для леса и бустинга на деревьях). То есть дерево решений делает константный прогноз для объектов, находящихся в признаковом пространстве вне параллелепипеда, охватывающего все объекты обучающей выборки. В нашем примере с зелёными и синими шариками это значит, что модель дает одинаковый прогноз для всех шариков с координатой  $> 19$  или  $< 0$ .

---

# Ансамбли



#057

# О значимости ансамблей

Лучшие алгоритмы машинного обучения по точности:

- Градиентный бустинг для классических задач
- Искусственные нейронные сети для изображений, видео, звука

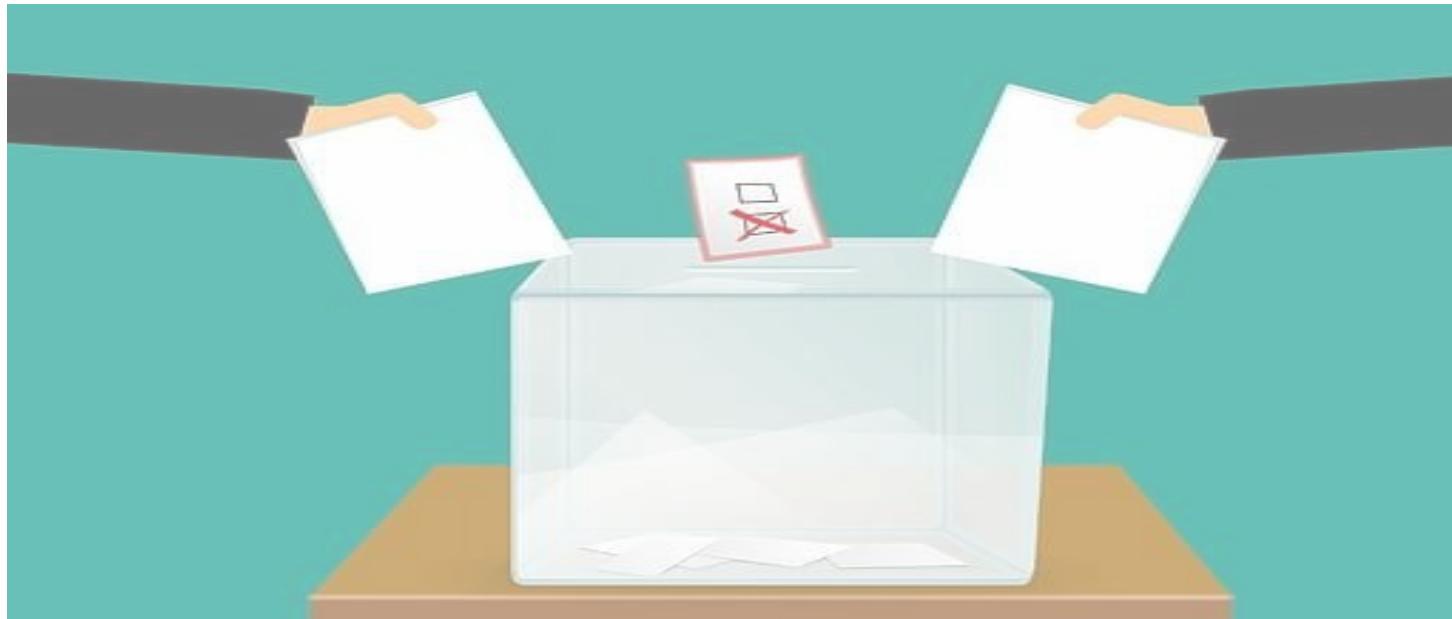
В соревнованиях kaggle всегда\* побеждают ансамбли

## Ансамбли моделей

Коллективное принятие решений как правило превосходит по качеству индивидуальное принятие решений



# Простое голосование



**Классификация:** класс определяется большинством голосов или усреднением скоров

**Регрессия:** среднее значение

---

## Взвешенное голосование



**Классификация:** класс определяется большинством голосов с учетом веса, или усреднением скоров с учетом веса

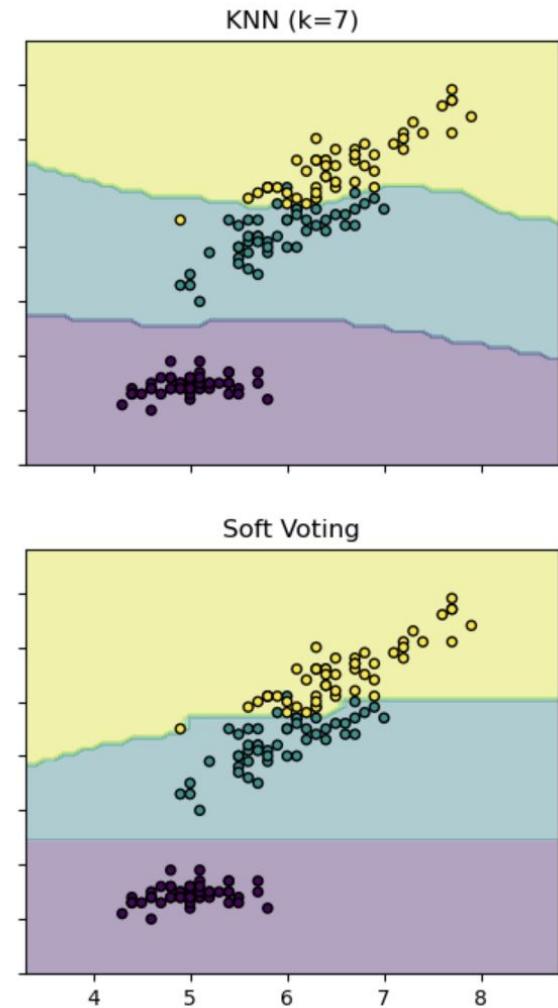
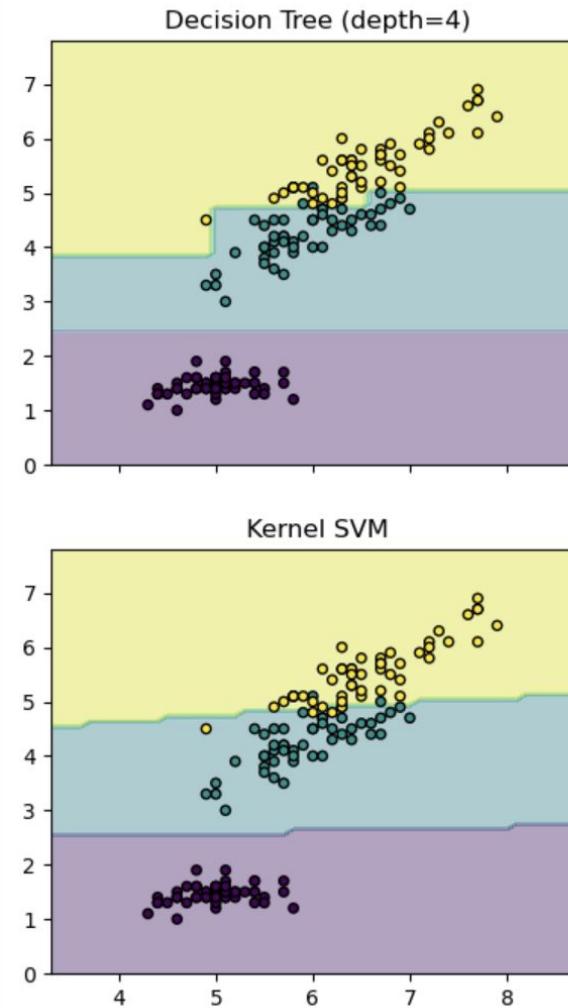
**Регрессия:** среднее взвешенное значение

# Пример голосования

Датасет - [Iris Dataset](#)

Soft Voting - голосование трёх моделей с разным весом (в данном случае веса - [2, 1, 2])

```
from sklearn.ensemble import  
VotingClassifier  
  
clf1 = SVC()  
clf2 = KNeighborsClassifier(n_neighbors=7)  
clf3 = DecisionTreeClassifier(max_depth=4)  
  
eclf = VotingClassifier(estimators=[('dt', clf1), ('knn',  
clf2), ('svc', clf3)], voting='soft', weights=[2, 1, 2])
```



#062

<https://scikit-learn.org/stable/modules/ensemble.html>

---

## **Недостатки голосования**

1. Голосование не учитывает особенностей поведения отдельных моделей
2. Голосование по сути является простой моделью

---

## Вопрос

Можно ли использовать ансамбли для линейных моделей?



---

# Стэкинг

## Идея:

Построим модель, которая будет предсказывать правильный ответ на основе предсказаний других моделей

# Блендинг

Простейшая схема стекинга – блендинг

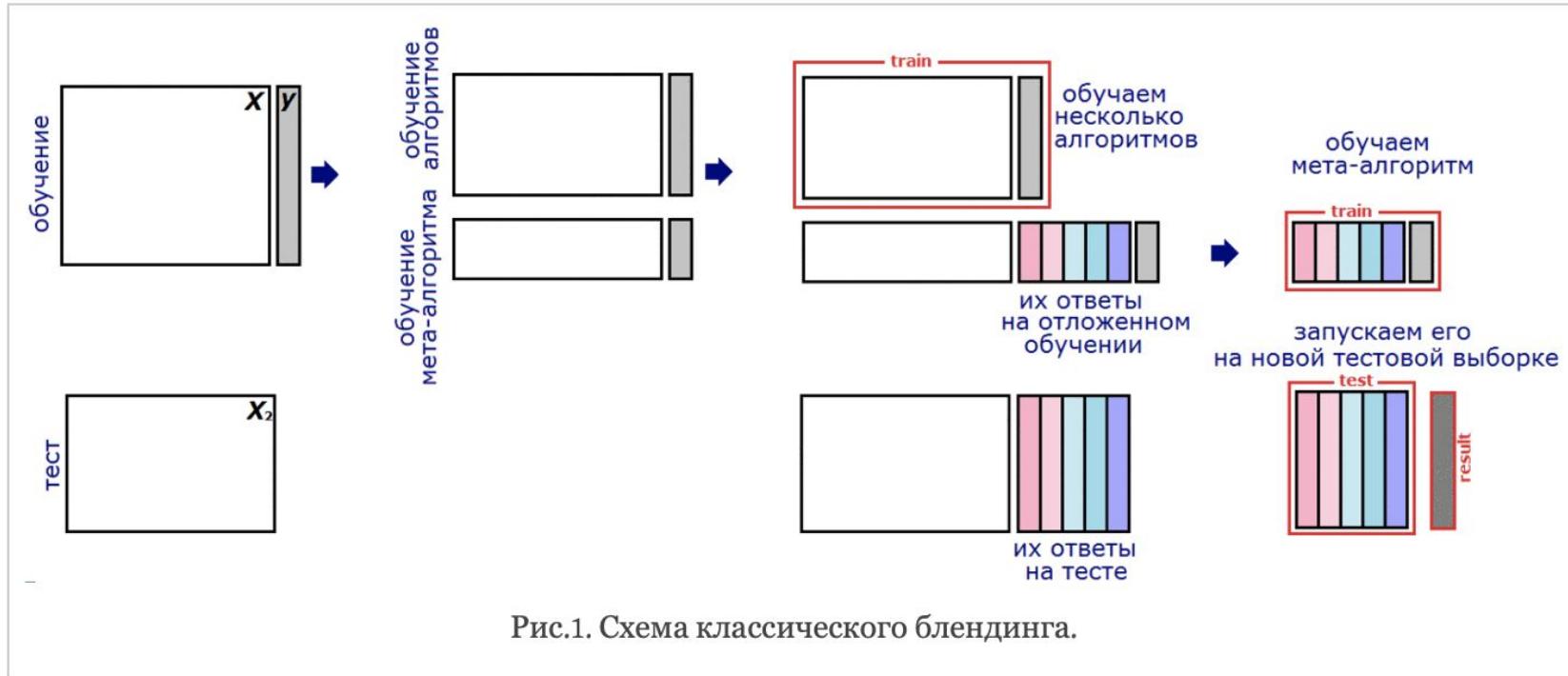


Рис.1. Схема классического блендинга.

---

## **Недостаток блендинга**

Самый большой недостаток классического блендинга — деление обучающей выборки.

## **Как бороться?**

---

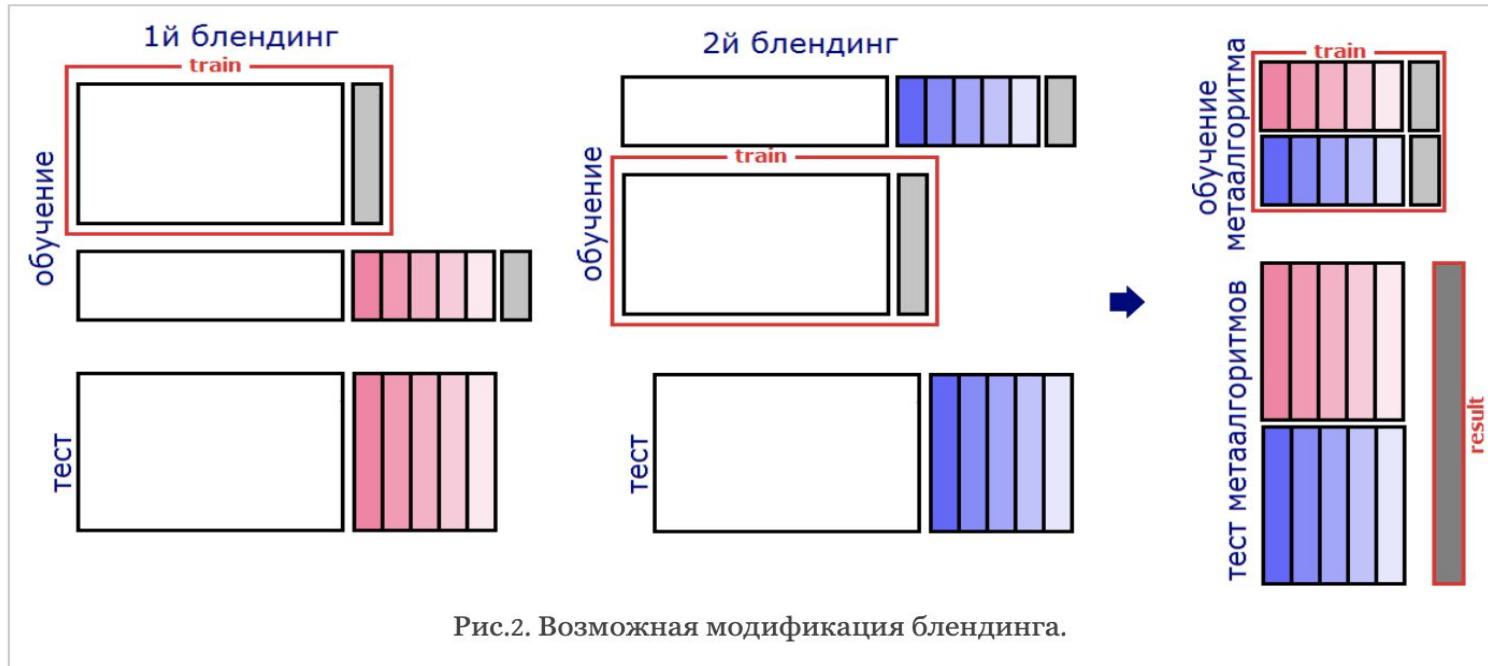
## **Недостаток блендинга**

Самый большой недостаток классического блендинга — деление обучающей выборки.

### **Как бороться?**

Для повышения качества надо усреднить несколько блендингов с разными разбиениями обучения

# Блендинг



На практике такая схема блендинга сложнее в реализации и более медленная, а по качеству может не превосходить обычного усреднения.

# Стэкинг

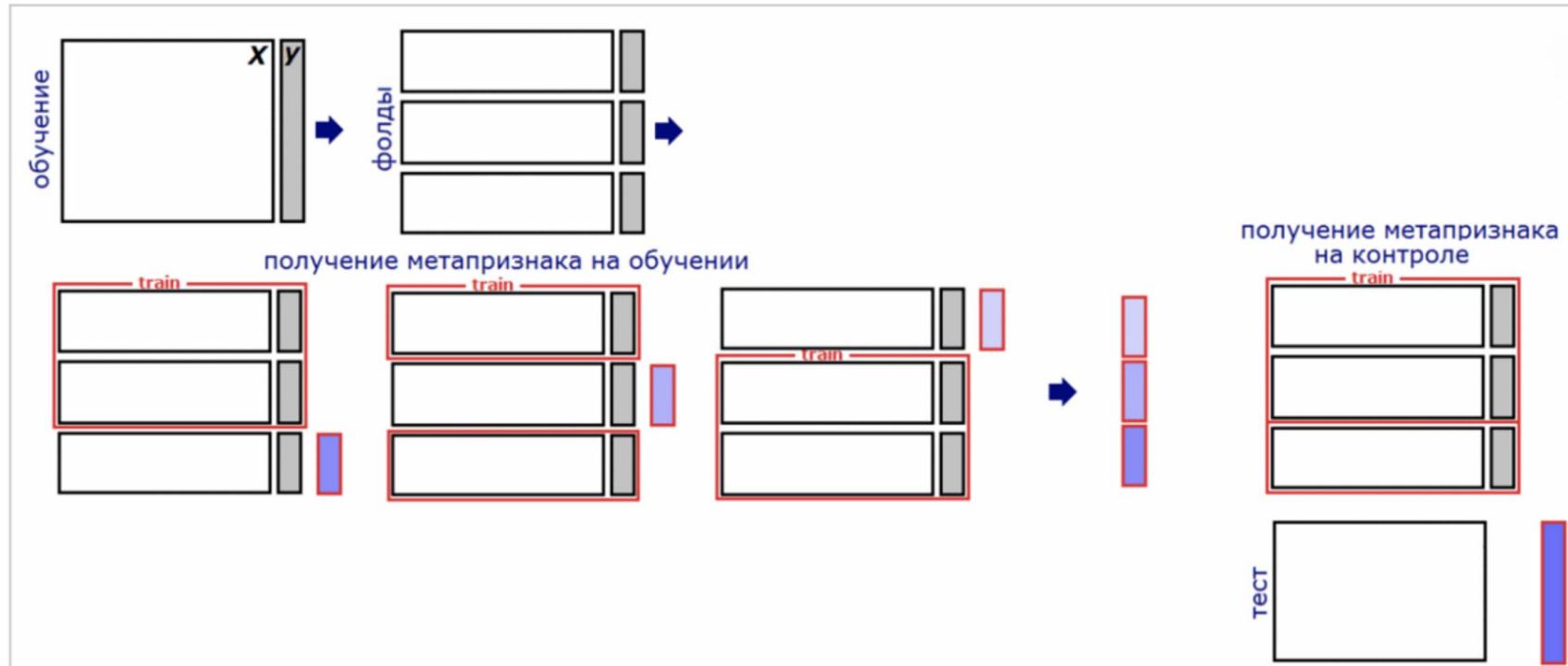


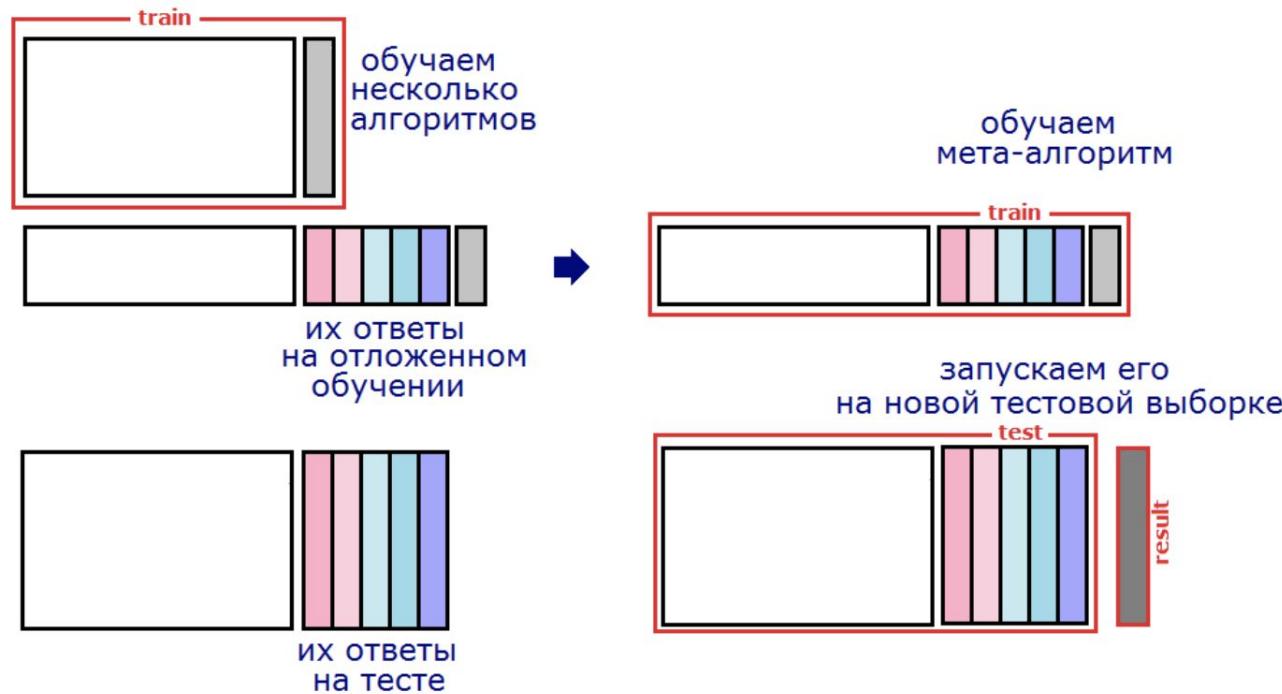
Рис.3. Получение метапризнака в классическом стекинге.

Кросс-валидационное предсказание будем называть **метапризнаком**.

# Стэкинг

Стэкинг можно делать как на наборе метапризнаков, так и на наборе метапризнаков + набор исходных признаков.

Стэкинг может быть многоуровневым.



---

# **Бэггинг и Бустинг**

## **Идея бэггинга:**

1. Построим много слегка различающихся моделей
2. Методом усреднения выберем итоговый ответ

## **Идея бустинга:**

Каждая следующая модель в ансамбле пытается предсказать ошибку всех предыдущих моделей ансамбля

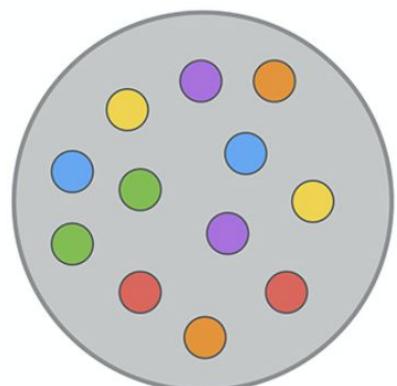
## Бутстрэп

**Бутстрэп** (bootstrap) – метод исследования распределения статистик вероятностных распределений, основанный на многократной генерации псевдовыборок на базе имеющейся выборки.

1. Из исходной выборки генерируем псевдовыборки методом случайного выбора с возвращением.
2. На псевдовыборках считаем целевую статистику.
3. Анализируем распределение целевой статистики на псевдовыборках.

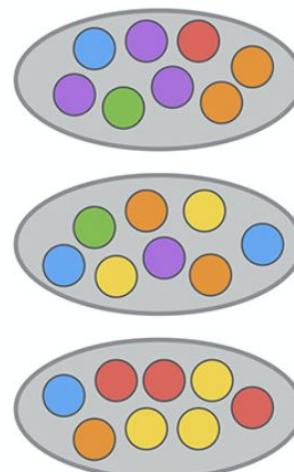
# Бутстрэп

Исходная выборка



Статистика по  
выборке

Бутстрэп выборки



Статистики по  
бутстрэп выборкам

Статистика 1

Статистика 2

Статистика 3

Бутстрэп  
распределение

#074

## Бэггинг

Bagging – Bootstrap aggregating (Leo Breiman, 1994)

- Из Train генерим методом случайного выбора сэмплов с возвращением  $\text{Train}'_1 \dots \text{Train}'_N$
- На каждом  $\text{Train}'$  строим модель
- Итоговое предсказание получаем усреднением предсказаний всех моделей или простым голосованием

## Бэггинг

### Задача:

В обучающей выборке  $n$  объектов. Из этих  $n$  объектов мы сэмплируем с возвращением  $n$  объектов. Какова доля уникальных объектов, которые попадают в выборку бустрэпа?

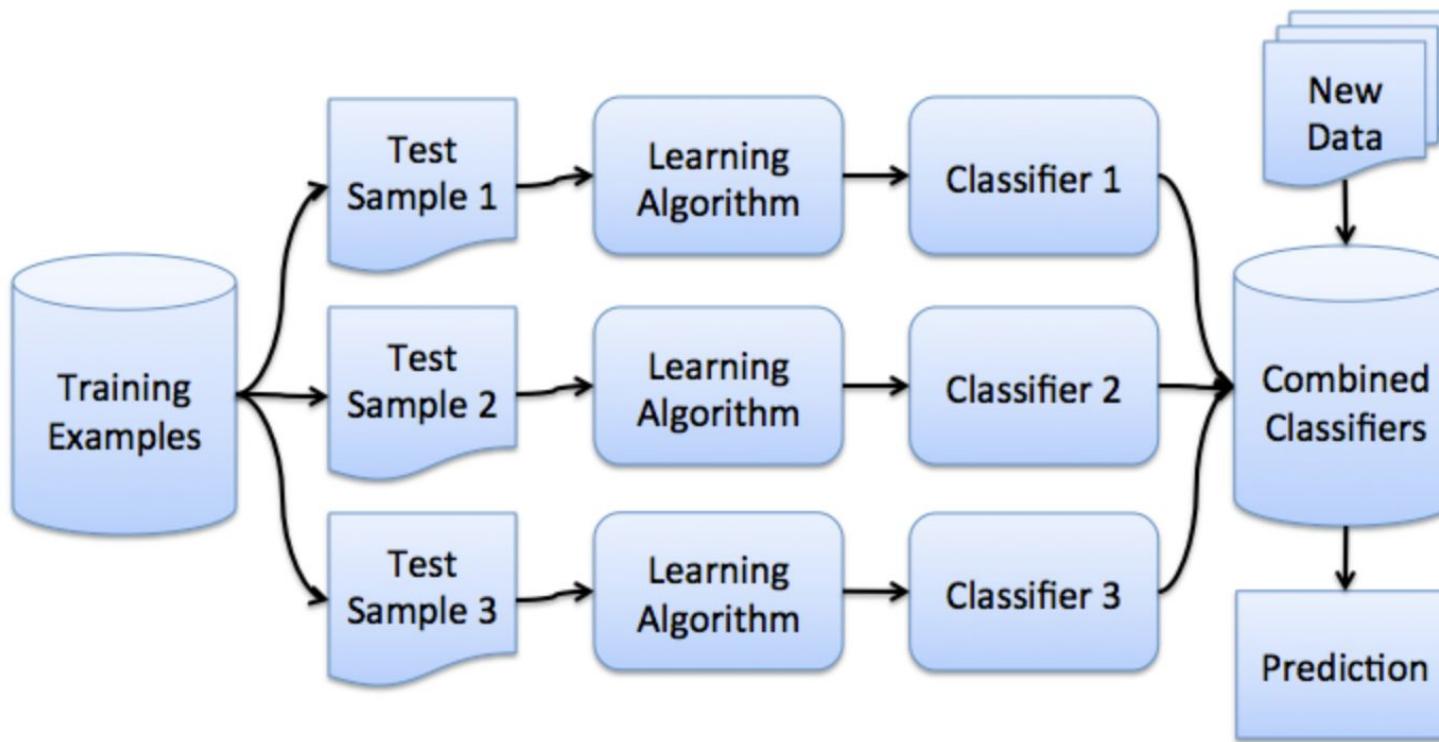
## БЭГГИНГ

Сколько объектов попадают в выборку бустрэпа? примерно 63%

Это можно легко доказать: пусть в выборке  $\ell$  объектов. На каждом шаге все объекты попадают в подвыборку с возвращением равновероятно, т.е отдельный объект — с вероятностью  $\frac{1}{\ell}$ .

Вероятность того, что объект НЕ попадет в подвыборку (т.е. его не взяли  $\ell$  раз):  $(1 - \frac{1}{\ell})^\ell$ . При  $\ell \rightarrow +\infty$  получаем один из "замечательных" пределов  $\frac{1}{e}$ . Тогда вероятность попадания конкретного объекта в подвыборку  $\approx 1 - \frac{1}{e} \approx 63\%$ .

# БЭГГИНГ



## Почему это работает?

Рассмотрим задачу регрессии с базовыми алгоритмами  $b_1(x), \dots, b_n(x)$

$$\varepsilon_i(x) = b_i(x) - y(x), i = 1, \dots, n$$

$$E_x(b_i(x) - y(x))^2 = E_x \varepsilon_i^2(x)$$

$$E_1 = \frac{1}{n} E_x \sum_{i=1}^n \varepsilon_i^2(x)$$

## БЭГГИНГ

Предположим, что ошибки несмещены и некоррелированы:

Построим новую функцию регрессии

Найдем её среднеквадратичную ошибку

$$\begin{aligned} E_x \varepsilon_i(x) &= 0, \\ E_x \varepsilon_i(x) \varepsilon_j(x) &= 0, i \neq j \end{aligned}$$

$$a(x) = \frac{1}{n} \sum_{i=1}^n b_i(x)$$

$$\begin{aligned} E_n &= E_x \left( \frac{1}{n} \sum_{i=1}^n b_i(x) - y(x) \right)^2 \\ &= E_x \left( \frac{1}{n} \sum_{i=1}^n \varepsilon_i \right)^2 \\ &= \frac{1}{n^2} E_x \left( \sum_{i=1}^n \varepsilon_i^2(x) + \sum_{i \neq j} \varepsilon_i(x) \varepsilon_j(x) \right) \\ &= \frac{1}{n} E_1 \end{aligned}$$

# Метод случайных подпространств

RSM – Random Subspace Method или feature bagging

- Из Train генерим методом случайного выбора признаков без возвращения  $\text{Train}'_1 \dots \text{Train}'_N$
- На каждом  $\text{Train}'$  строим модель
- Итоговое предсказание получаем усреднением предсказаний всех моделей

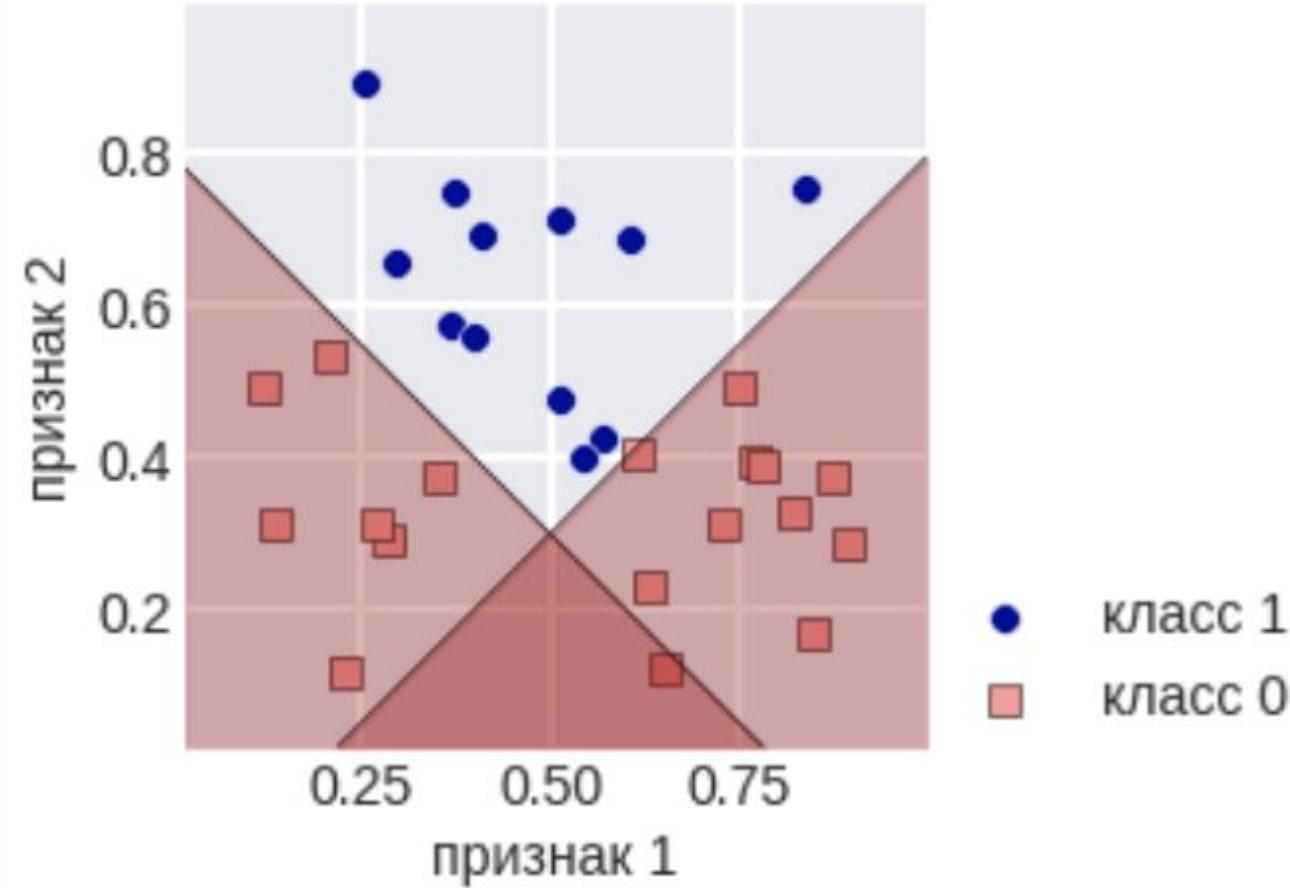
---

## Вернёмся к вопросу

Используются ли ансамбли для линейных моделей?



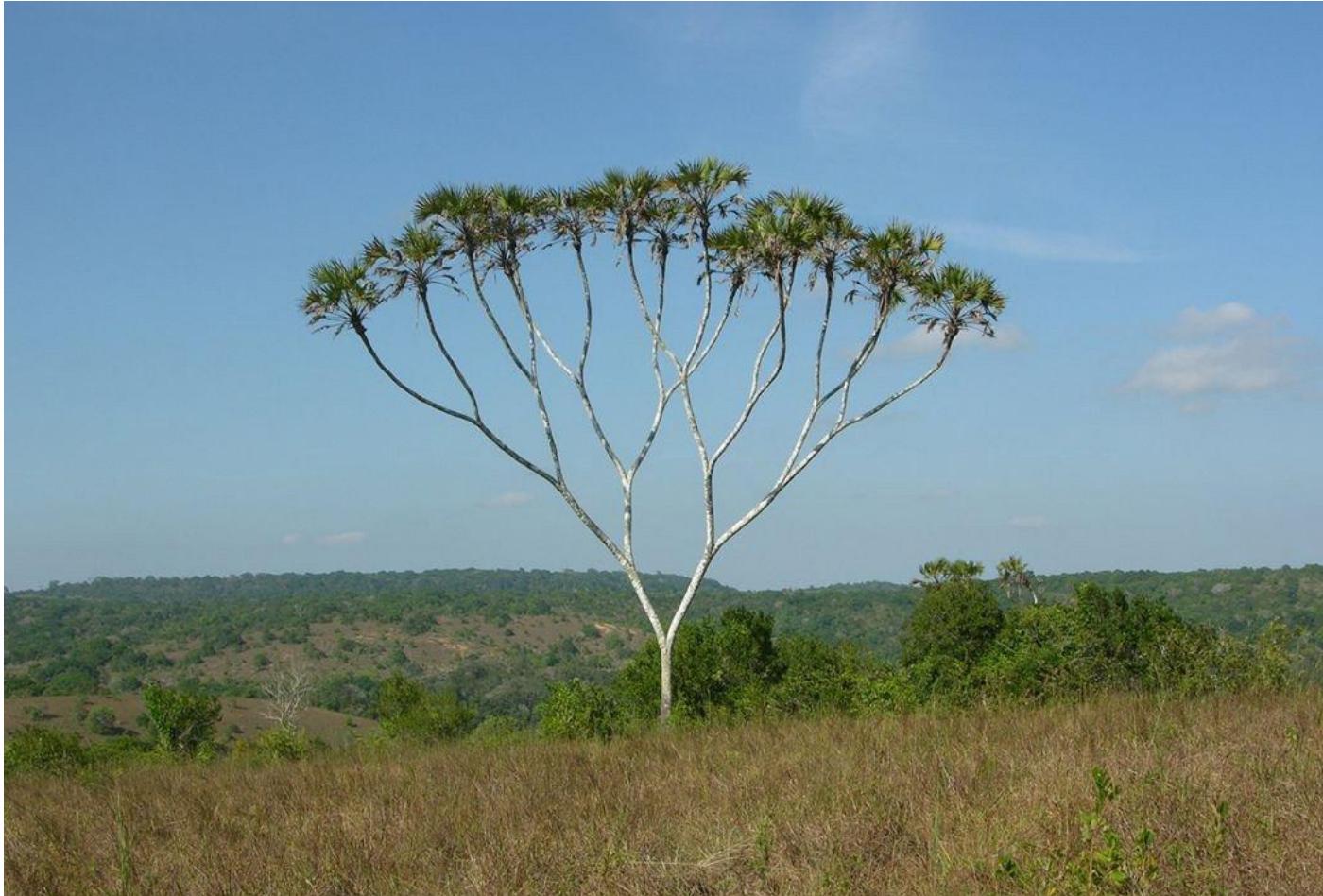
# Вопрос



#083

---

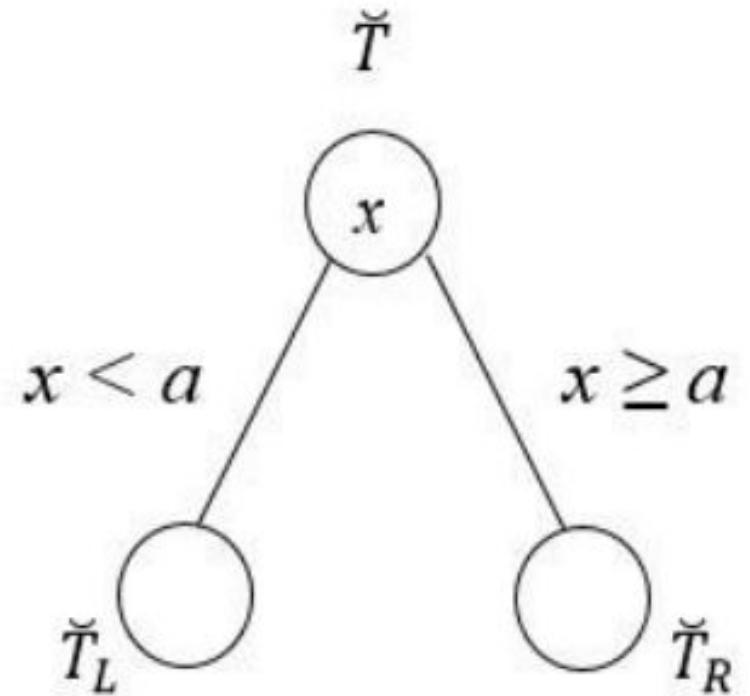
# Немного регрессионных деревьев



#084

# Немного о регрессионных деревьях

$$\check{T} = \{S_{i_1}, \dots, S_{i_u}\}$$
$$\check{T}_L(x, a) = \{S_1^L, \dots, S_p^L\} \quad \check{T}_R(x, a) = \{S_1^R, \dots, S_q^R\}$$

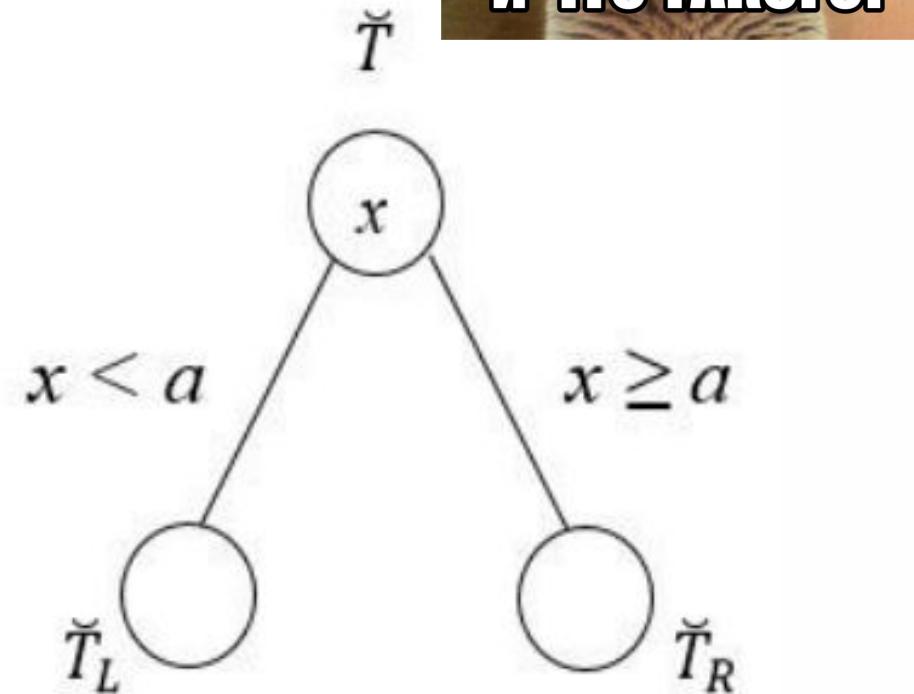


# Немного о регрессионных деревьях

$$\breve{T} = \{S_{i_1}, \dots, S_{i_u}\}$$

$$\breve{T}_L(x, a) = \{S_1^L, \dots, S_p^L\}$$

$$\breve{T}_R(x, a) = \{S_1^R, \dots, S_q^R\}$$



И ЧТО ТАКОГО?

# Регрессионные деревья

$$\bar{y}_{\check{T}} = \frac{1}{u} \sum_{t=1}^u y_{i_t};$$

$$\text{Variance} = \frac{1}{u} \sum_{t=1}^u (y_{i_t}^2) - \left[ \frac{1}{u} \sum_{t=1}^u (y_{i_t}) \right]^2;$$

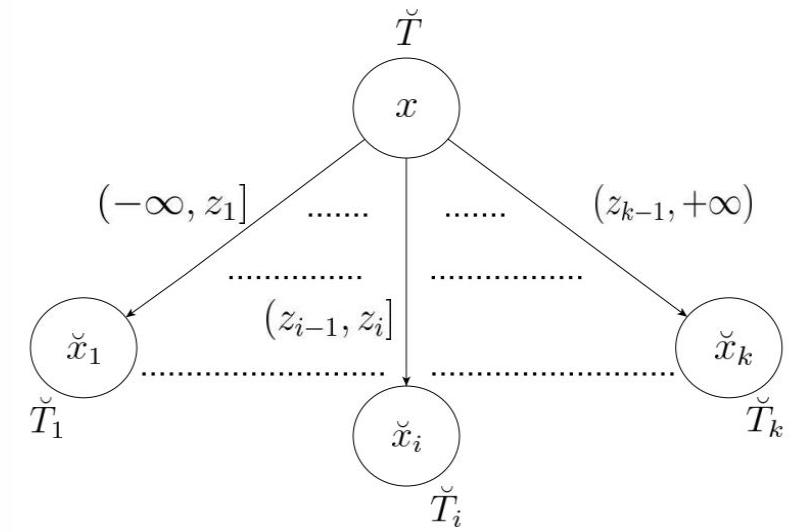
$$\text{SE}(x) = \frac{1}{u} \left\{ \sum_{i=1}^p (y_i^L - \bar{y}_{\check{T}_L})^2 + \sum_{j=1}^q (y_j^R - \bar{y}_{\check{T}_R})^2 \right\};$$

$$C(x) = \text{Variance} - \text{SE}(x).$$

Оптимальным считается разбиение с максимальным значением величины C(x).

# k-арные регрессионные деревья

Как будет выглядеть критерий ветвления  
для k-арного дерева?

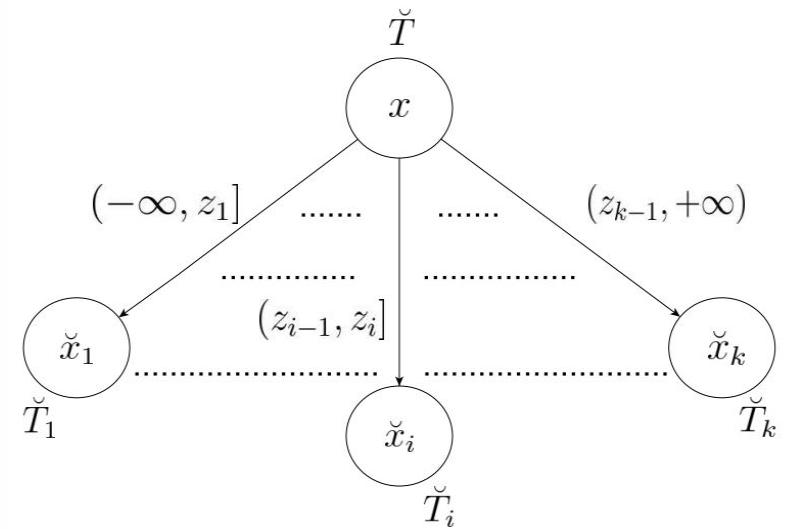


# k-арные регрессионные деревья

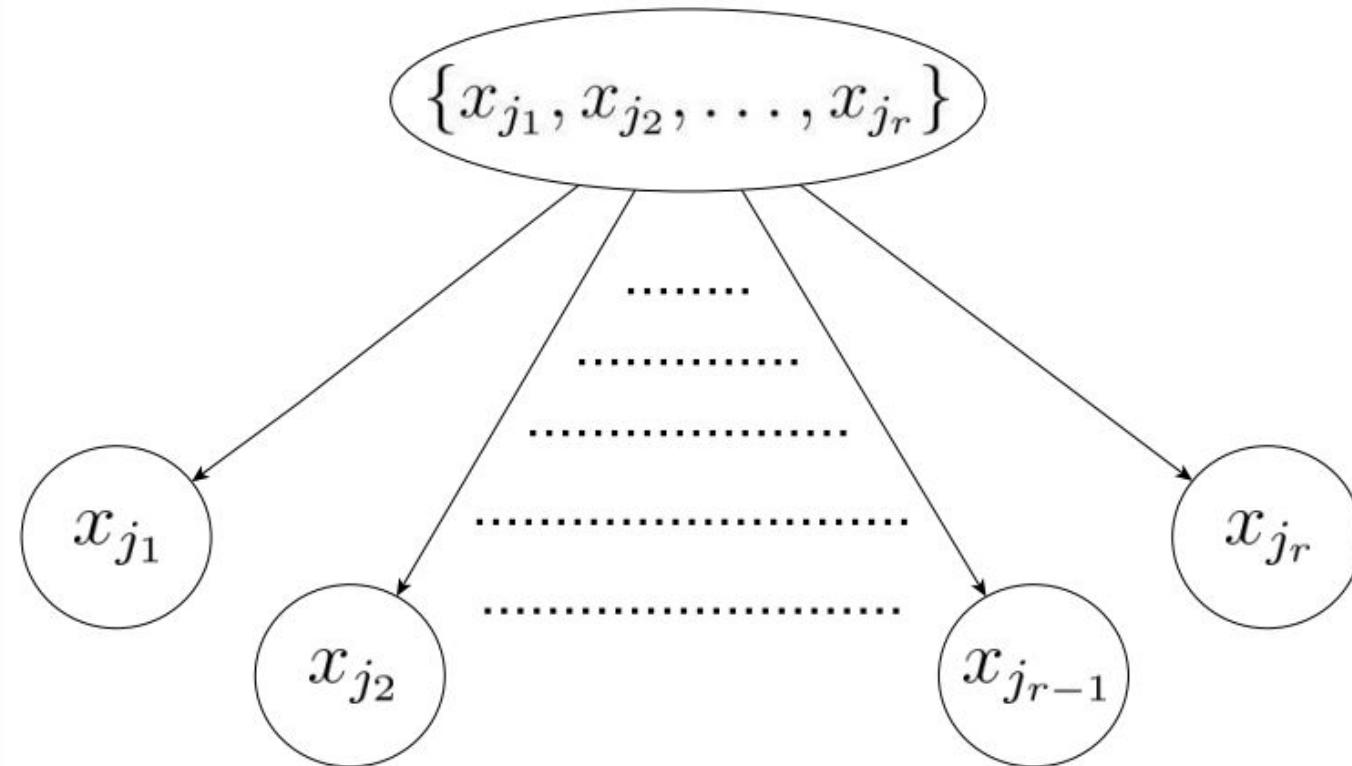
Как будет выглядеть критерий ветвления для k-арного дерева?

$$SE(x, k) = \frac{1}{u_i} \left\{ \sum_{S_t^i \in \check{T}_{i_1}} (y_t^i - \bar{y}_{\check{T}_{i_1}})^2 + \dots + \sum_{S_t^i \in \check{T}_{i_k}} (y_t^i - \bar{y}_{\check{T}_{i_k}})^2 \right\}$$

$$C(k, x) = Variance - SE(x, k).$$



# Полные k-арные регрессионные деревья



#090

# Полные k-арные регрессионные деревья



#091

Ссылка:

[Построение и исследование полных решающих деревьев ...jmlda.org › papers › doc](https://jmlda.org/papers/doc)

# Random Forest

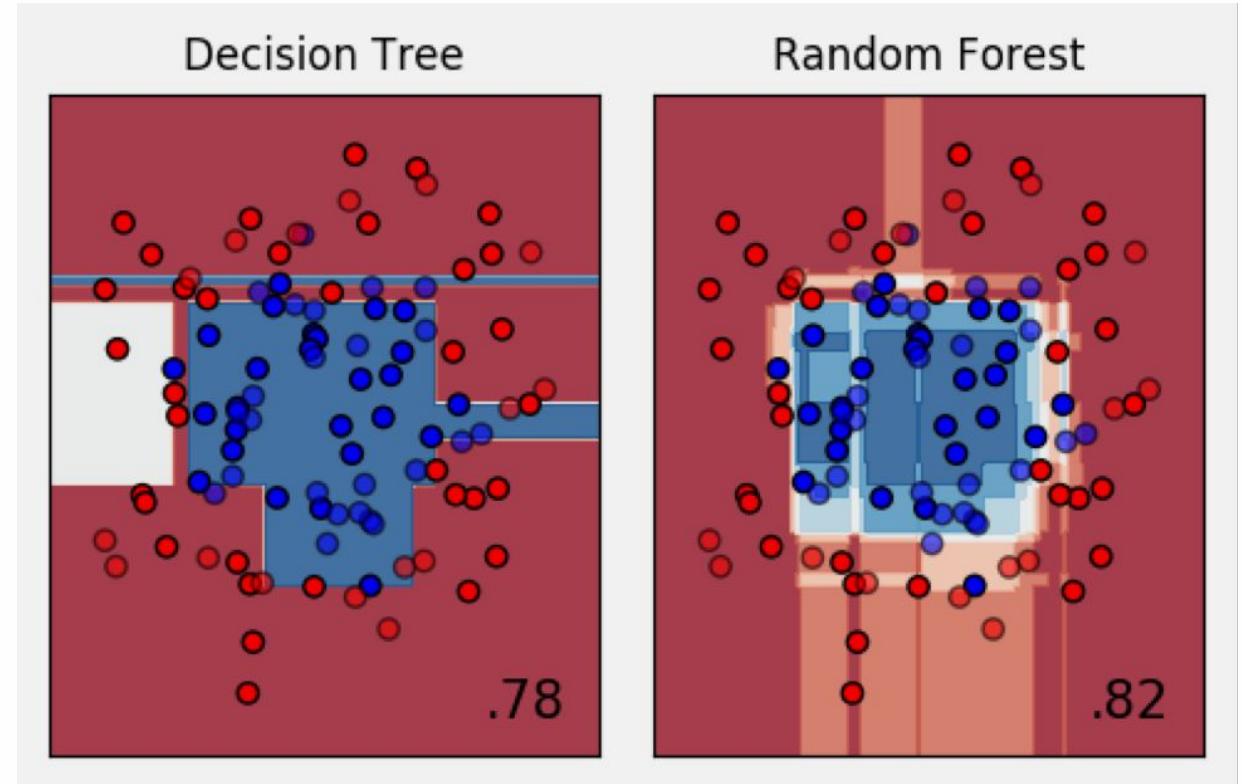
Алгоритм:

1. Выполняем N раз:
  - 1.1. Бутстрэп объектов
  - 1.2. Случайное подпространство признаков
  - 1.3. Построение дерева решений
2. Выбираем ответ модели методом усреднения предсказаний или простого голосования



# Random Forest

Random Forest выдает  
качество лучше, чем  
единичное решающее дерево



# Параметры Random Forest

```
RandomForestRegressor(n_estimators, criterion, max_depth,  
min_samples_split, min_samples_leaf, min_weight_fraction_leaf,  
max_features, max_leaf_nodes, min_impurity_decrease,  
min_impurity_split, bootstrap, oob_score, n_jobs, random_state,  
verbose, warm_start)
```

Параметры функции потерь

Параметры ансамбля

Параметры дерева

Параметры технические

# Random Forest

- `class sklearn.ensemble.RandomForestRegressor(`
- `n_estimators` – число деревьев в "лесу" (по дефолту – 10)
- `criterion` – функция, которая измеряет качество разбиения ветки дерева (по дефолту – "mse" , так же можно выбрать "mae")
- `max_features` – число признаков, по которым ищется разбиение. Вы можете указать конкретное число или процент признаков, либо выбрать из доступных значений: "auto" (все признаки), "sqrt", "log2". По дефолту стоит "auto".
- `max_depth` – максимальная глубина дерева (по дефолту глубина не ограничена)
- `min_samples_split` – минимальное количество объектов, необходимое для разделения внутреннего узла. Можно задать числом или процентом от общего числа объектов (по дефолту – 2)
- `min_samples_leaf` – минимальное число объектов в листе. Можно задать числом или процентом от общего числа объектов (по дефолту – 1)
- `min_weight_fraction_leaf` – минимальная взвешенная доля от общей суммы весов (всех входных объектов) должна быть в листе (по дефолту имеют одинаковый вес)
- `max_leaf_nodes` – максимальное количество листьев (по дефолту нет ограничения)
- `min_impurity_split` – порог для остановки наращивания дерева (по дефолту 1e-7)
- `bootstrap` – применять ли бустрэп для построения дерева (по дефолту True)
- `oob_score` – использовать ли out-of-bag объекты для оценки R^2 (по дефолту False)
- `n_jobs` – количество ядер для построения модели и предсказаний (по дефолту 1, если поставить -1, то будут использоваться все ядра)
- `random_state` – начальное значение для генерации случайных чисел (по дефолту его нет, если хотите воспроизводимые результаты, то нужно указать любое число типа int)
- `verbose` – вывод логов по построению деревьев (по дефолту 0)
- `warm_start` – использует уже натренированную модель и добавляет деревьев в ансамбль (по дефолту False)
- `)`

# Random Forest: Оценка важности признаков

Чем выше в среднем признак в дереве решений, тем он важнее в данной задаче



---

# Random Forest

Плюсы:

1. Алгоритм прост
2. Не чувствителен к выбросам в данных
3. Не переобучается
4. Хорошо параллелизируется
5. Не требует сложной настройки параметров
6. Не требует нормализации данных

# Random Forest

Минусы:

1. Модели не интерпретируемые
2. Работает хуже линейных моделей, когда есть разреженные признаки (например, тексты)
3. Большой размер получающихся моделей

# Градиентный бустинг

**Идея:**

1. Представляем итоговую модель  $f(x)$  как сумму слабых моделей  $h(x)$  (обычно решающие деревья малой глубины).
2. Пусть задана дифференцируемая функция потерь  $L(y, f(x))$
3. На каждом шаге мы ищем модель  $h(x)$ , которая бы аппроксимировала вектор антиградиента  $L$

# Градиентный бустинг

#0100

1. Инициализировать GBM константным значением  $\hat{f}(x) = \hat{f}_0, \hat{f}_0 = \gamma, \gamma \in \mathbb{R}$

$$\hat{f}_0 = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, \gamma)$$

2. Для каждой итерации  $t = 1, \dots, M$  повторять:

1. Посчитать псевдо-остатки  $r_t$

$$r_{it} = - \left[ \frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f(x)=\hat{f}(x)}, \quad \text{for } i = 1, \dots, n$$

2. Построить новый базовый алгоритм  $h_t(x)$  как регрессию на псевдо-остатках

$$\{(x_i, r_{it})\}_{i=1, \dots, n}$$

3. Найти оптимальный коэффициент  $\rho_t$  при  $h_t(x)$  относительно исходной функции потерь

$$\rho_t = \arg \min_{\rho} \sum_{i=1}^n L(y_i, \hat{f}(x_i) + \rho \cdot h(x_i, \theta))$$

4. Сохранить  $\hat{f}_t(x) = \rho_t \cdot h_t(x)$

5. Обновить текущее приближение  $\hat{f}(x)$

$$\hat{f}(x) \leftarrow \hat{f}(x) + \hat{f}_t(x) = \sum_{i=0}^t \hat{f}_i(x)$$

3. Скомпоновать итоговую GBM модель  $\hat{f}(x)$

$$\hat{f}(x) = \sum_{i=0}^M \hat{f}_i(x)$$

## Градиентный бустинг

Построили алгоритм  $a(x)$ , построим алгоритм  $b(x)$  такой, что

$$a(x_i) + b(x_i) = y_i, \quad i \in \{1, 2, \dots, m\},$$

Алгоритм  $b(x)$  поправляет ошибки алгоритма  $a(x)$  на невязку:  $\varepsilon_i = y_i - a(x_i)$

$$\sum_{i=1}^m L(y_i - a(x_i), b(x_i)) \rightarrow \min \neq \sum_{i=1}^m L(y_i, a(x_i) + b(x_i)) \rightarrow \min$$

## Градиентный бустинг

$$F = \sum_{i=1}^m L(y_i, a(x_i) + b_i) \rightarrow \min_{(b_1, \dots, b_m)}$$

Функция многих переменных максимально убывает

## Градиентный бустинг

$$F(b_1, \dots, b_m) = F = \sum_{i=1}^m L(y_i, a(x_i) + b_i) \rightarrow \min_{(b_1, \dots, b_m)}$$

Функция многих переменных максимально убывает в направлении своего антиградиента:

$$-(L'(y_1, a(x_1)), \dots, L'(y_m, a(x_m)))$$

# Градиентный бустинг

Выгодно считать

$$b_i = -L'(y_i, a(x_i)), \quad i \in \{1, 2, \dots, m\}$$

а это и есть наши ответы алгоритма b. Получается его следует настраивать на обучающей выборке:

$$(x_i, -L'(y_i, a(x_i)))_{i=1}^m$$

# Градиентный бустинг

```
GradientBoostingClassifier(loss, learning_rate, n_estimators,  
subsample, criterion, min_samples_split, min_samples_leaf,  
min_weight_fraction_leaf, max_depth, min_impurity_decrease,  
min_impurity_split, init, random_state, max_features, verbose,  
max_leaf_nodes, warm_start, presort, validation_fraction,  
n_iter_no_change, tol)
```

Параметры функции потерь

Параметры ансамбля

Параметры дерева

Параметры технические

# Недостатки градиентного бустинга

- **Проблема переобучения градиентного бустинга**

По мере увеличения числа деревьев ошибка на обучающей выборке постепенно уходит в 0. Ошибка на контрольной выборке существенно больше ошибки на обучающей выборке, достигает минимума примерно на 10 итерации, а затем начинает опять возрастать. Имеет место переобучение.

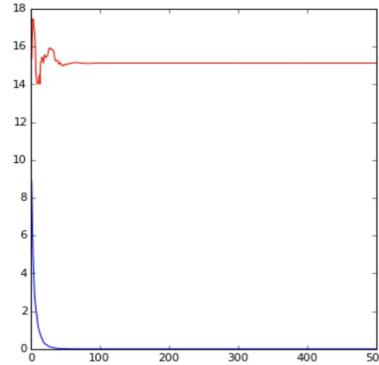
- **Сокращение размера шага**

Чтобы решить эту проблему, нужно «не доверять» направлению, которое построил базовый алгоритм и лишь чуть-чуть смещаться в сторону этого вектора:

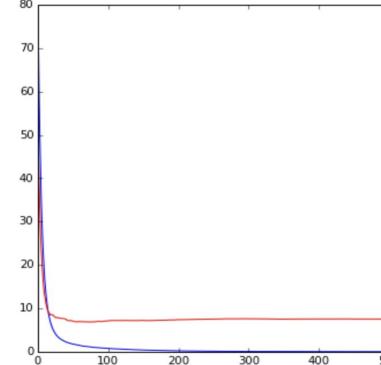
$$a_N(x) = a_{N-1}(x) + \eta b_N(x),$$

где  $\eta \in (0, 1]$  — длина шага. Это обеспечивает очень аккуратное движение в пространстве, что делает возможным нахождение локального минимума.

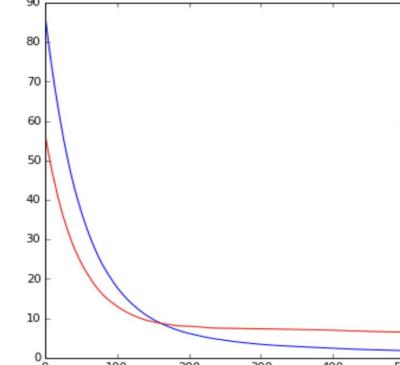
# Недостатки градиентного бустинга



(a) Случай  $\eta = 1$ .



(b) Случай  $\eta = 0.1$ .



(c) Случай  $\eta = 0.01$ .

Как видно по графикам, при  $\eta = 0.1$  качество на контрольной выборке уже существенно лучше, то есть в некотором смысле удалось побороть переобучение. При еще меньшей длине шага  $\eta = 0.01$  градиентному бустингу требуется существенно больше итераций, чтобы достичь чуть-чуть большего качества. Таким образом:

- Чем меньше размер шага, тем больше нужно базовых алгоритмов, чтобы достичь хорошего качества, и тем больше времени занимает процесс.
- Чем меньше размер шага, тем лучше качества можно достичь.

Другими словами, приходится выбирать: или быстро получить достаточно хорошее качество, или получить качество чуть-чуть лучше за большее время.

# Недостатки бустинга

- Идея бустинга обычно плохо применима к построению композиции из достаточно сложных и мощных алгоритмов. Построение такой композиции занимает очень много времени, а качество существенно не увеличивается.
- Результаты работы бустинга сложно интерпретируемы, особенно если в композицию входят десятки алгоритмов.

# Преимущества

- Хорошая обобщающая способность. В реальных задачах (не всегда, но часто) удаётся строить композиции, превосходящие по качеству базовые алгоритмы. Обобщающая способность может улучшаться (в некоторых задачах) по мере увеличения числа базовых алгоритмов.
- Простота реализации.
- Собственные накладные расходы бустинга невелики. Время построения композиции практически полностью определяется временем обучения базовых алгоритмов.
- Возможность идентифицировать объекты, являющиеся шумовыми выбросами.

---

## Вопрос на засыпку

Если бы вы могли воспользоваться многоядерным процессором, вы бы предпочли алгоритм бустинга над деревьями случайному лесу? Почему?

# Открытые реализации градиентного бустинга

*dmlc*  
**XGBoost**



Yandex  
CatBoost

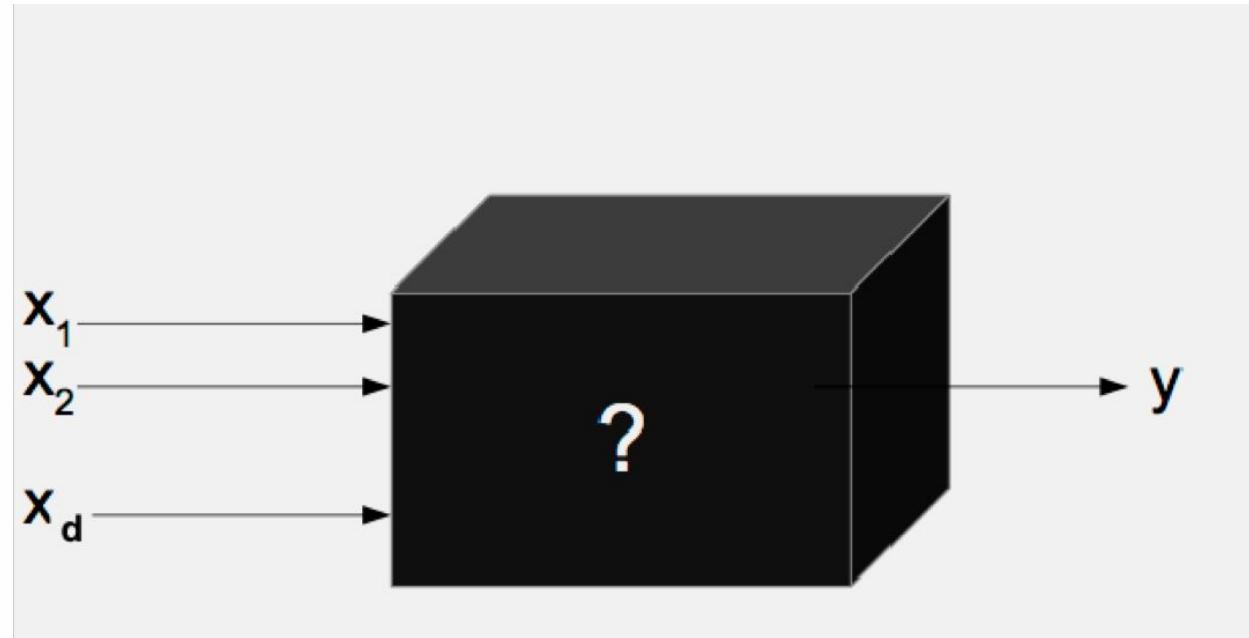


#0111

# Автоматический подбор гиперпараметров моделей

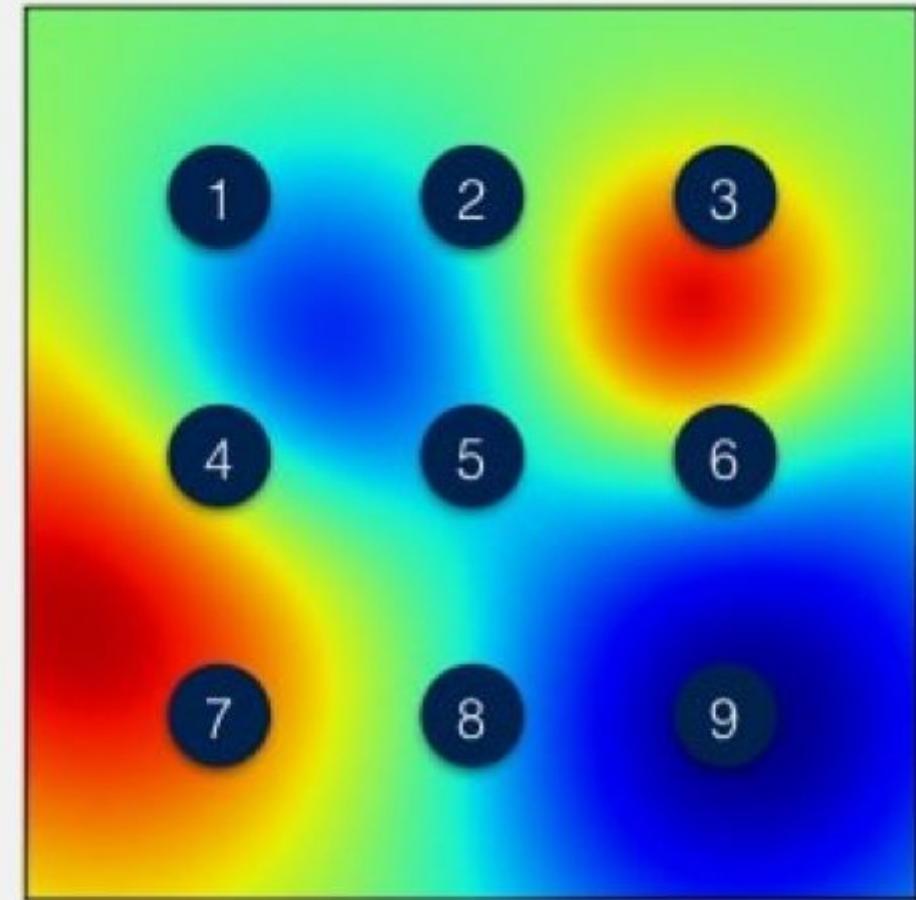
Модель(параметры) -> качество

Задача - максимизировать качество



# Grid Search

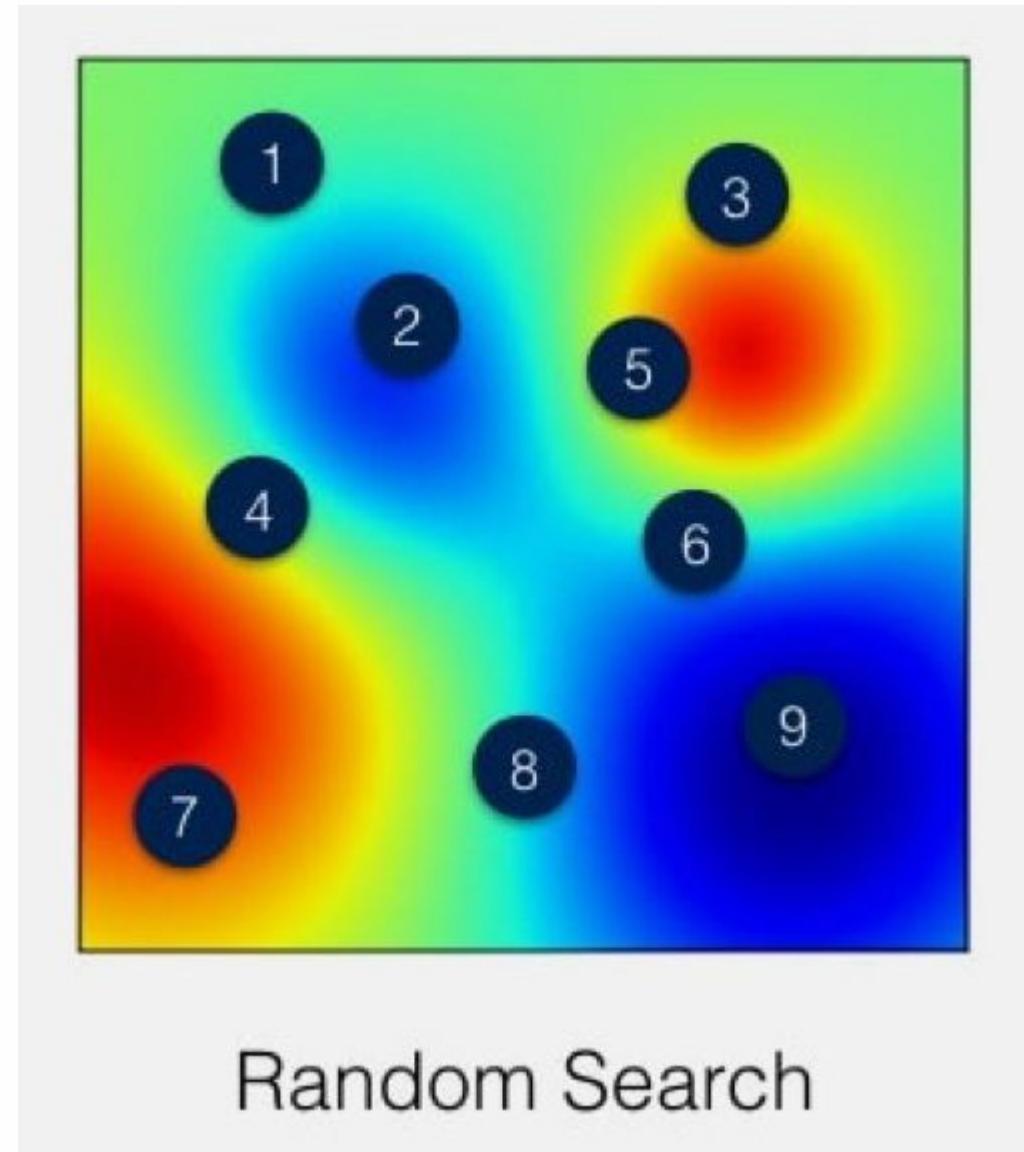
1. Перебираем параметры модели по решетке
2. Выбираем параметры, которые дают самое высокое качество



Grid Search

# Random Search

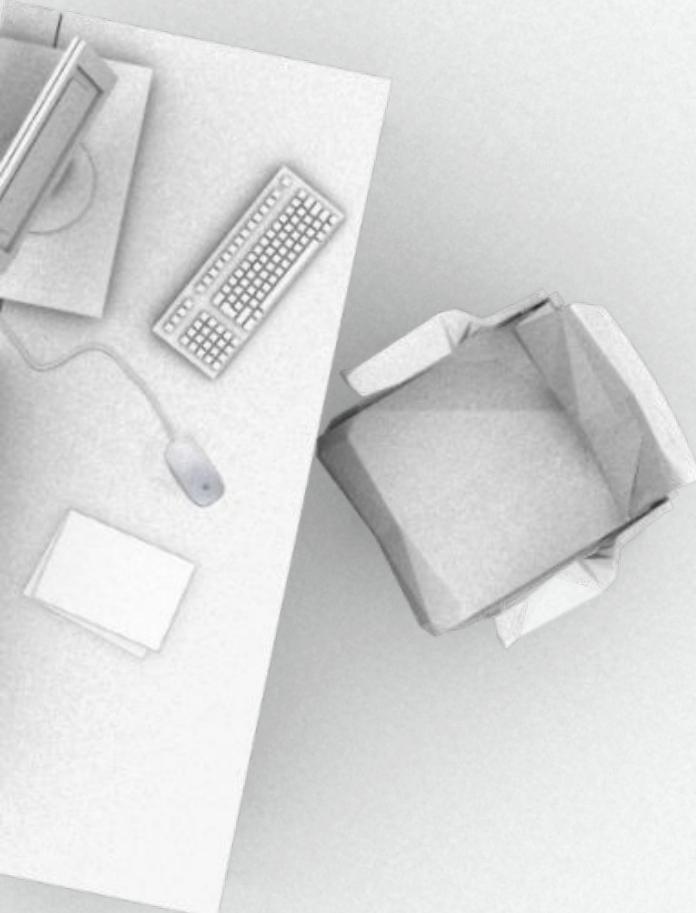
1. Сэмплируем параметры модели
2. Выбираем параметры, которые дают самое высокое качество



---

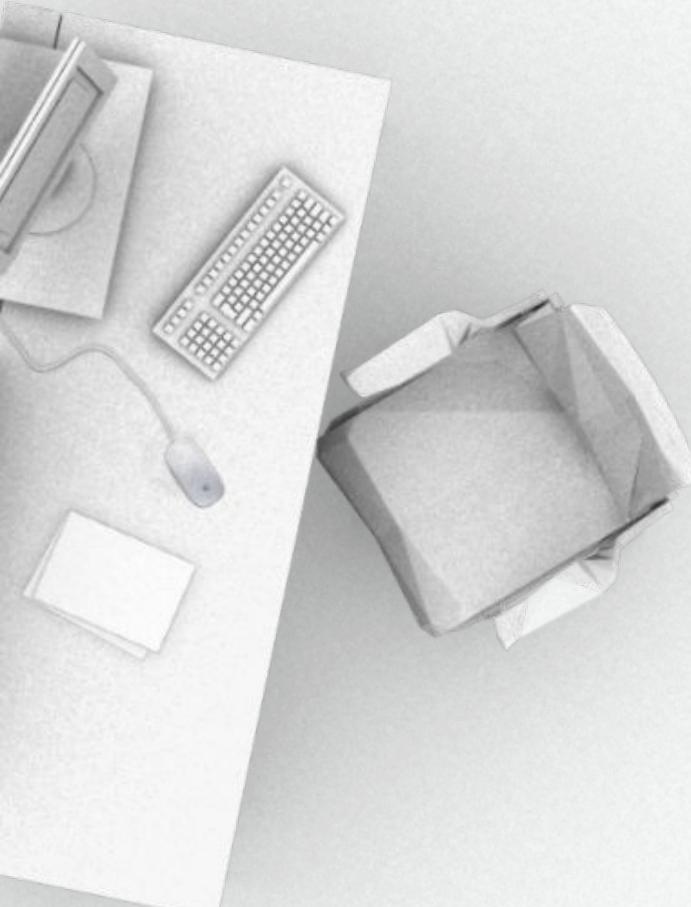
# Инструменты для оптимизации гиперпараметров

- sklearn
- Hyperopt
- BayesianOptimization
- Hypropy
- Optunity
- Optuna

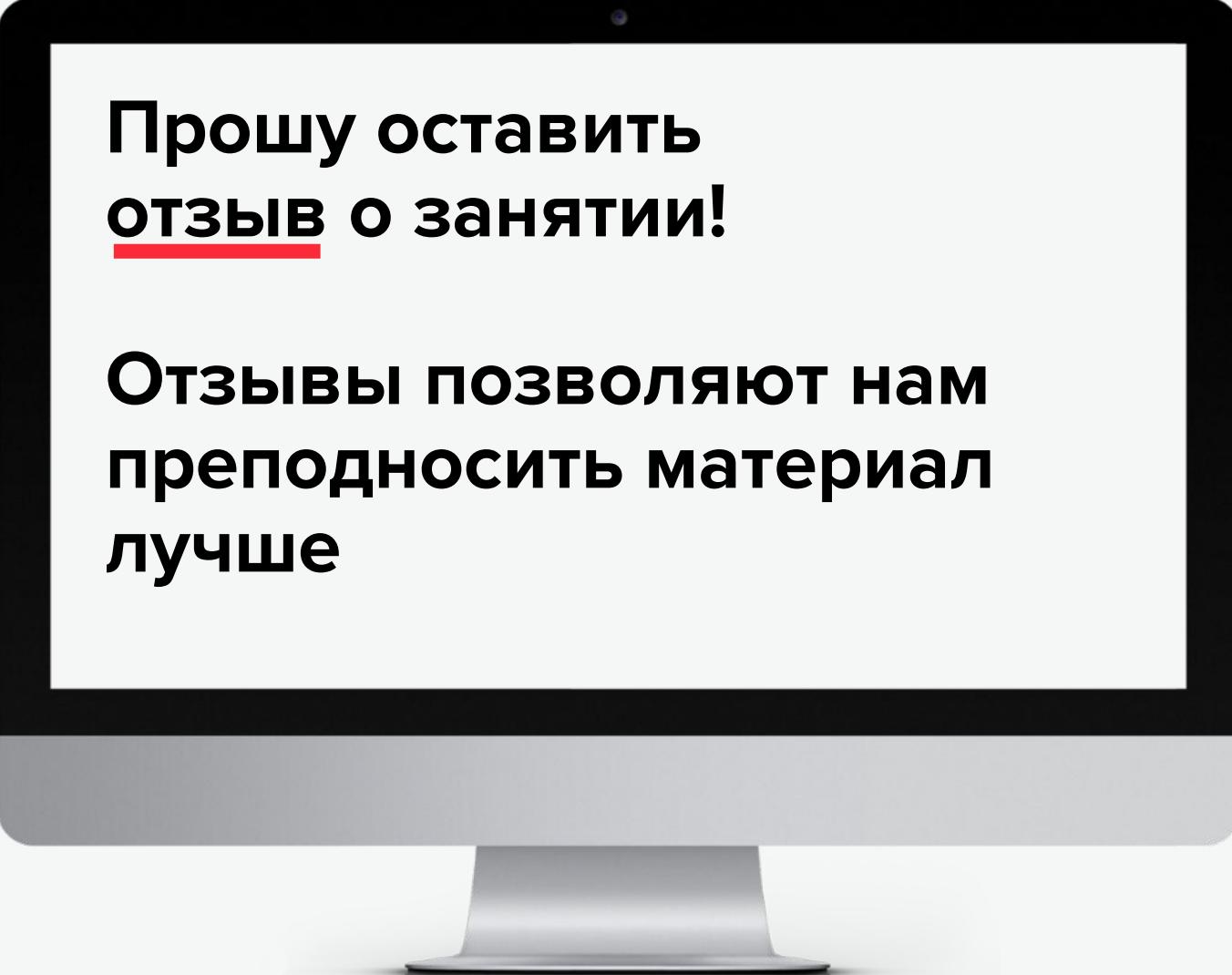


# Домашнее задание: соревнование

- Сделать сабмит решения
- Выложить решение на [github.com](https://github.com) или в collab.  
Прислать свой ник kaggle



Конкурс Тут



**Прошу оставить  
отзыв о занятии!**

**Отзывы позволяют нам  
преподносить материал  
лучше**

**СПАСИБО  
ЗА ВНИМАНИЕ**

