

# Машинное обучение

Лекция 5  
Решающие деревья.

Михаил Гущин  
[mhushchyn@hse.ru](mailto:mhushchyn@hse.ru)

НИУ ВШЭ, 2023



НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
УНИВЕРСИТЕТ

# Эволюция решающих деревьев



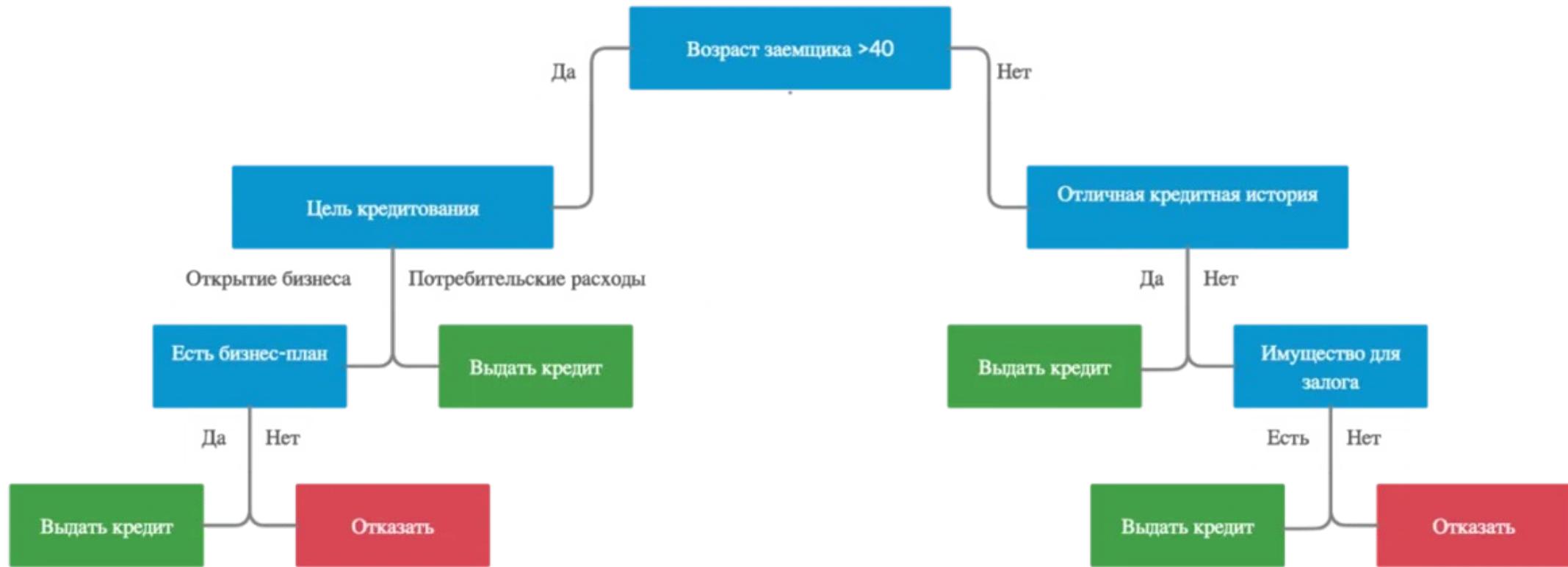
Решающие  
деревья



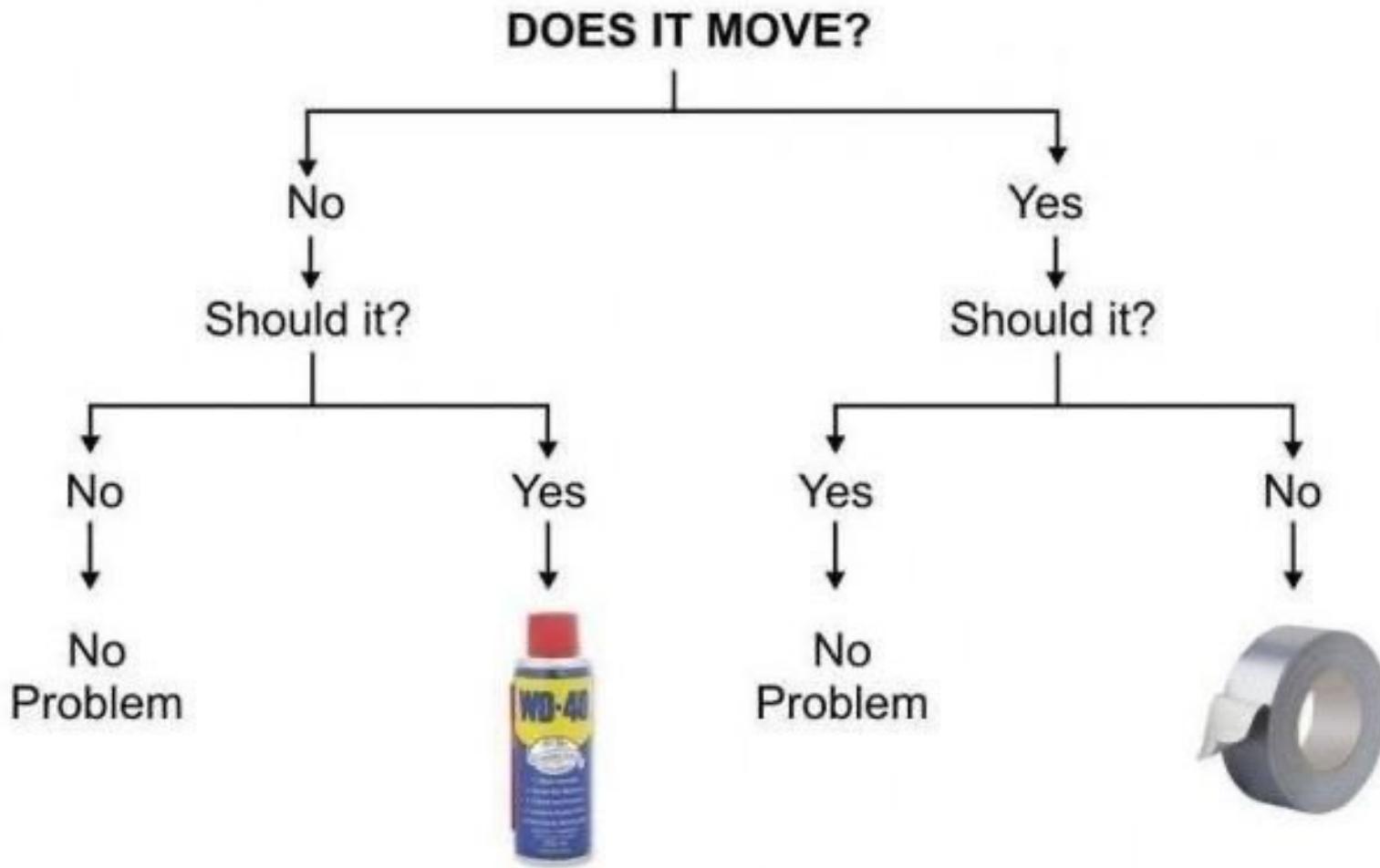
Интуиция

---

# Выдача кредита

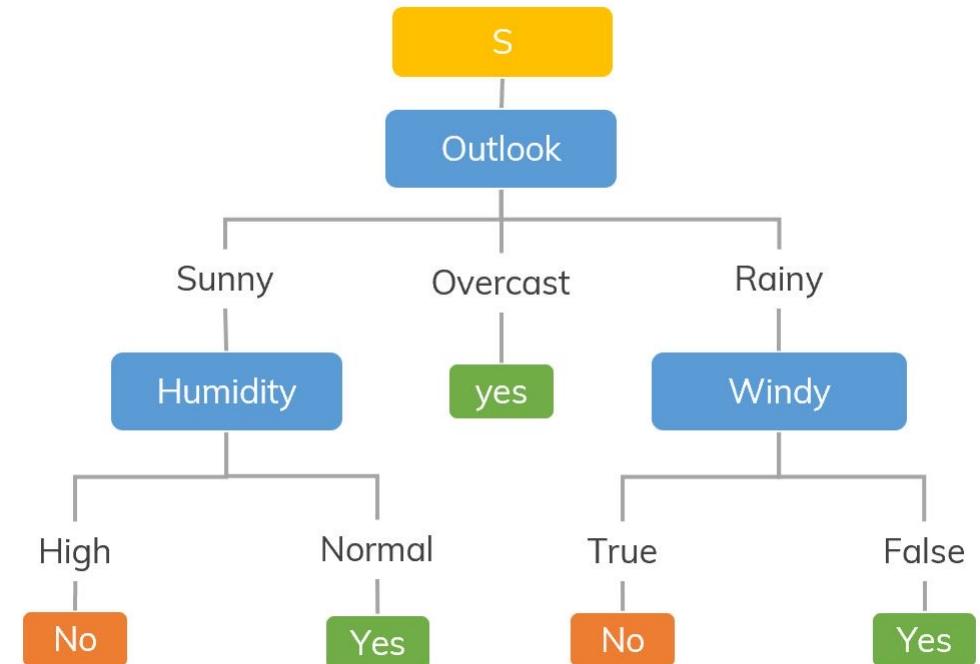


# Ремонт



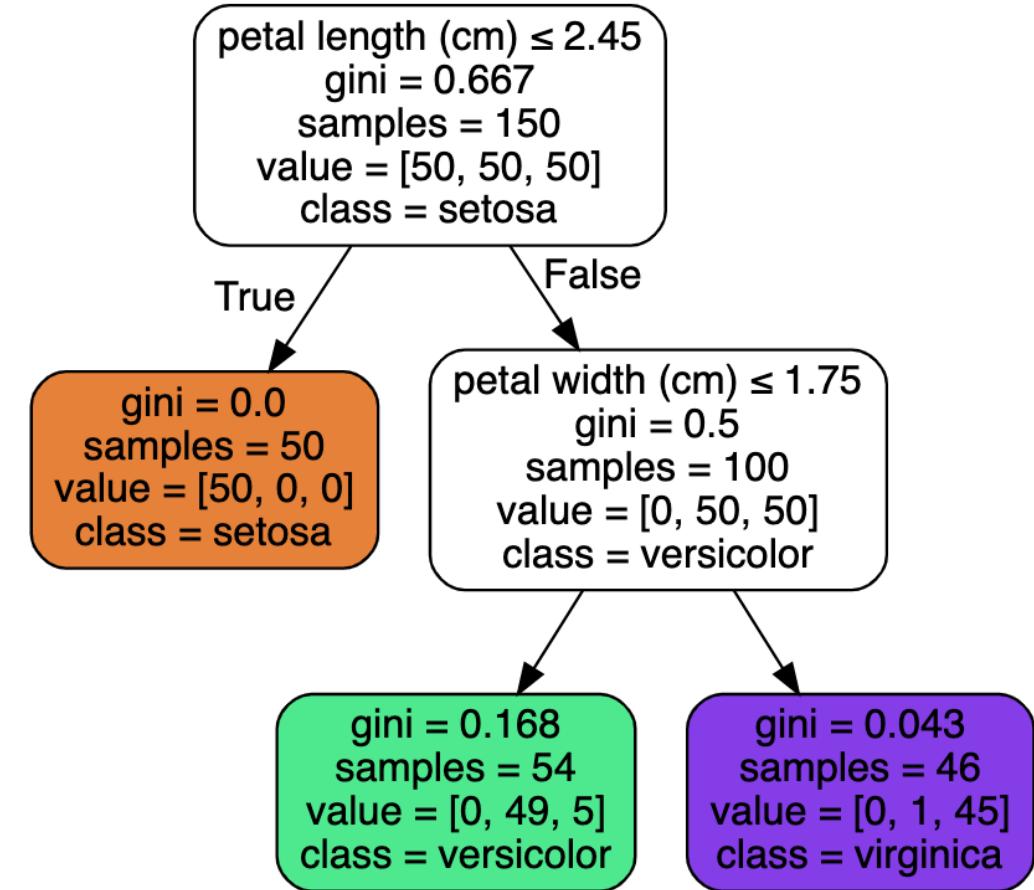
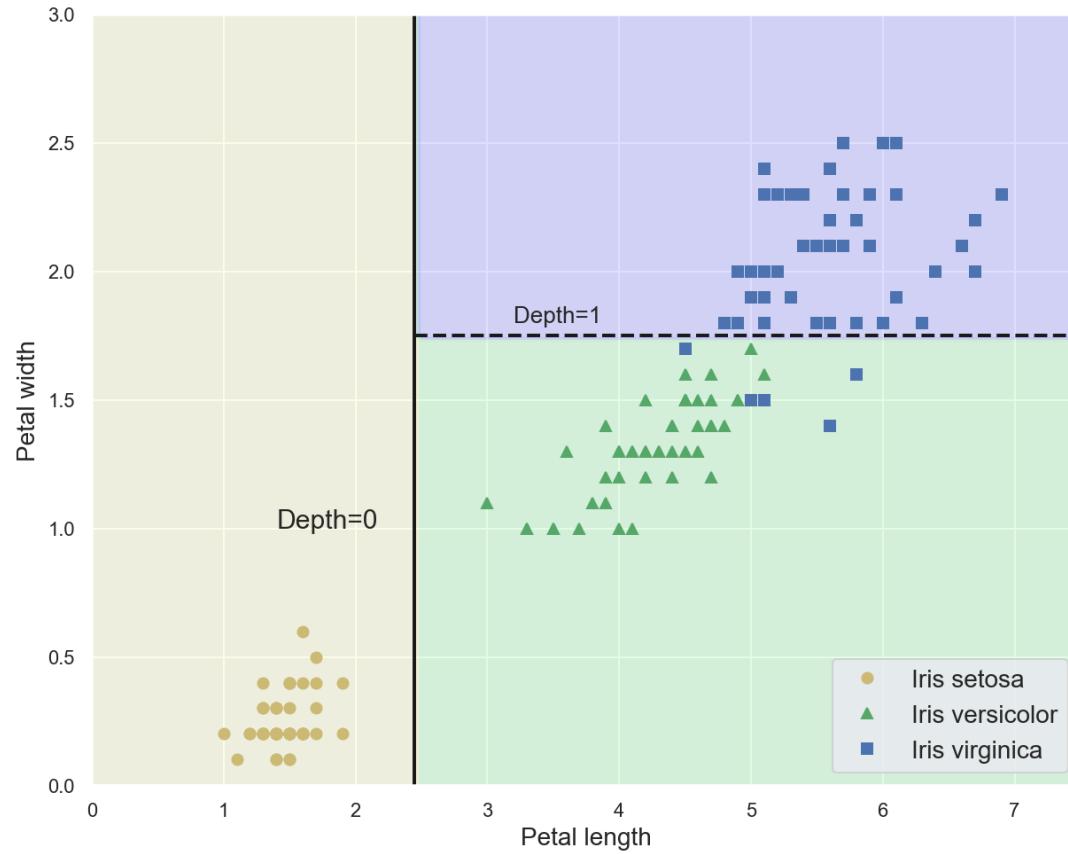
# Игра в гольф

Outlook	Temperature	Humidity	Windy	Play Golf
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No



<https://dinhanhthi.com/decision-tree-classifier/>

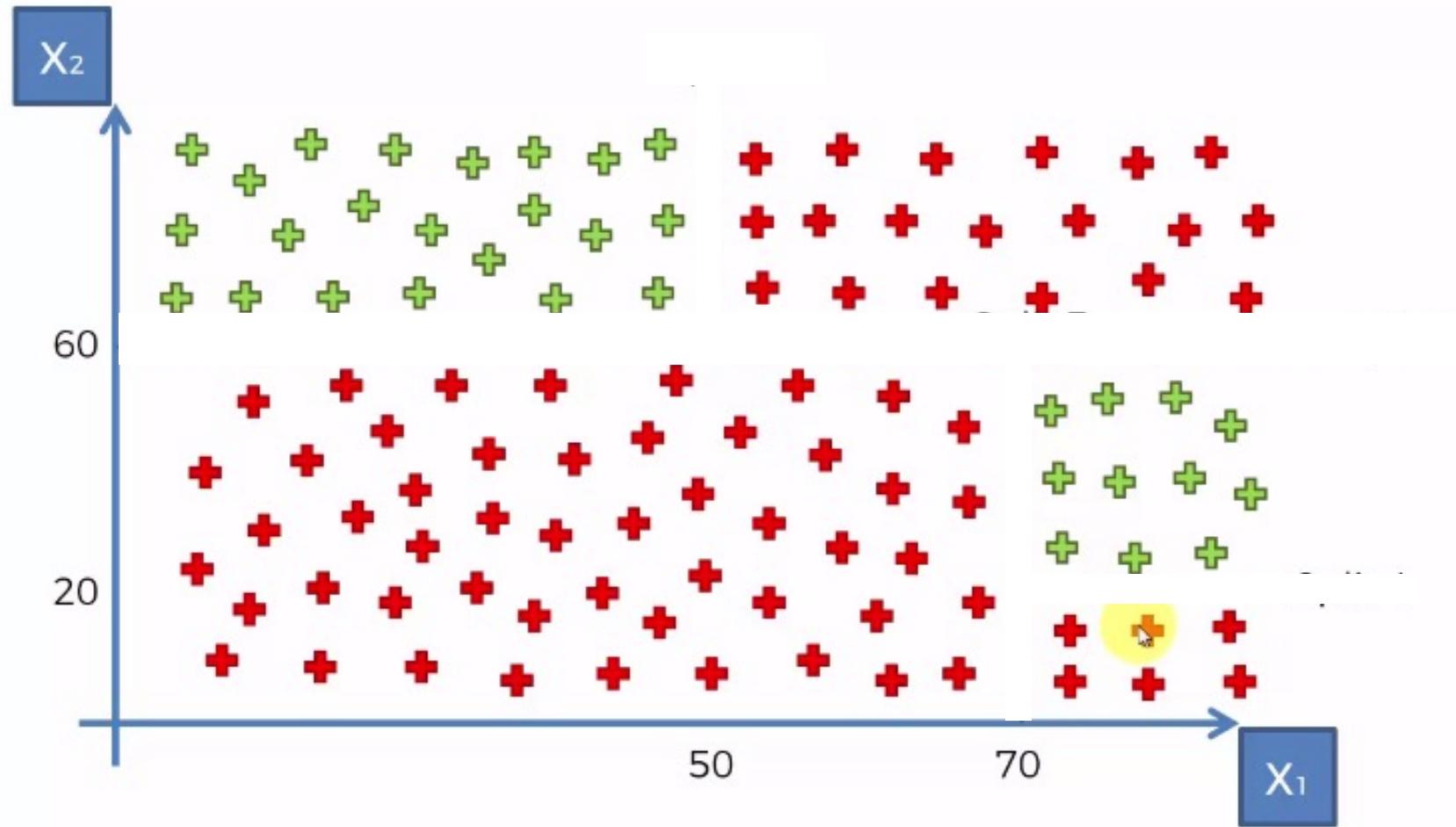
# Ирисы 😊



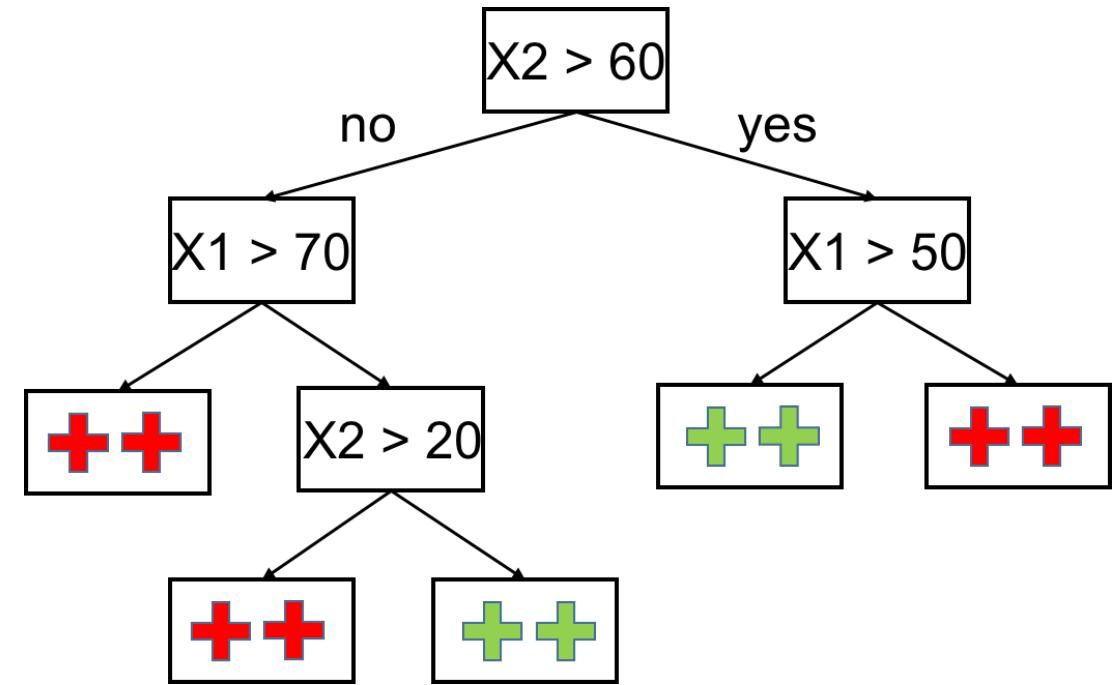
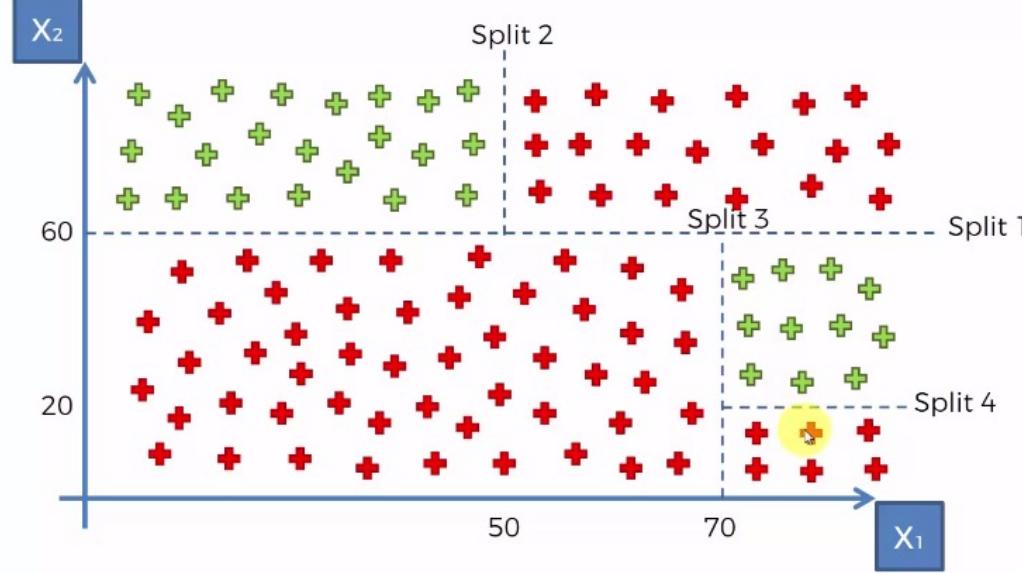
# Двоичные деревья

---

# Задача классификации



# Решение

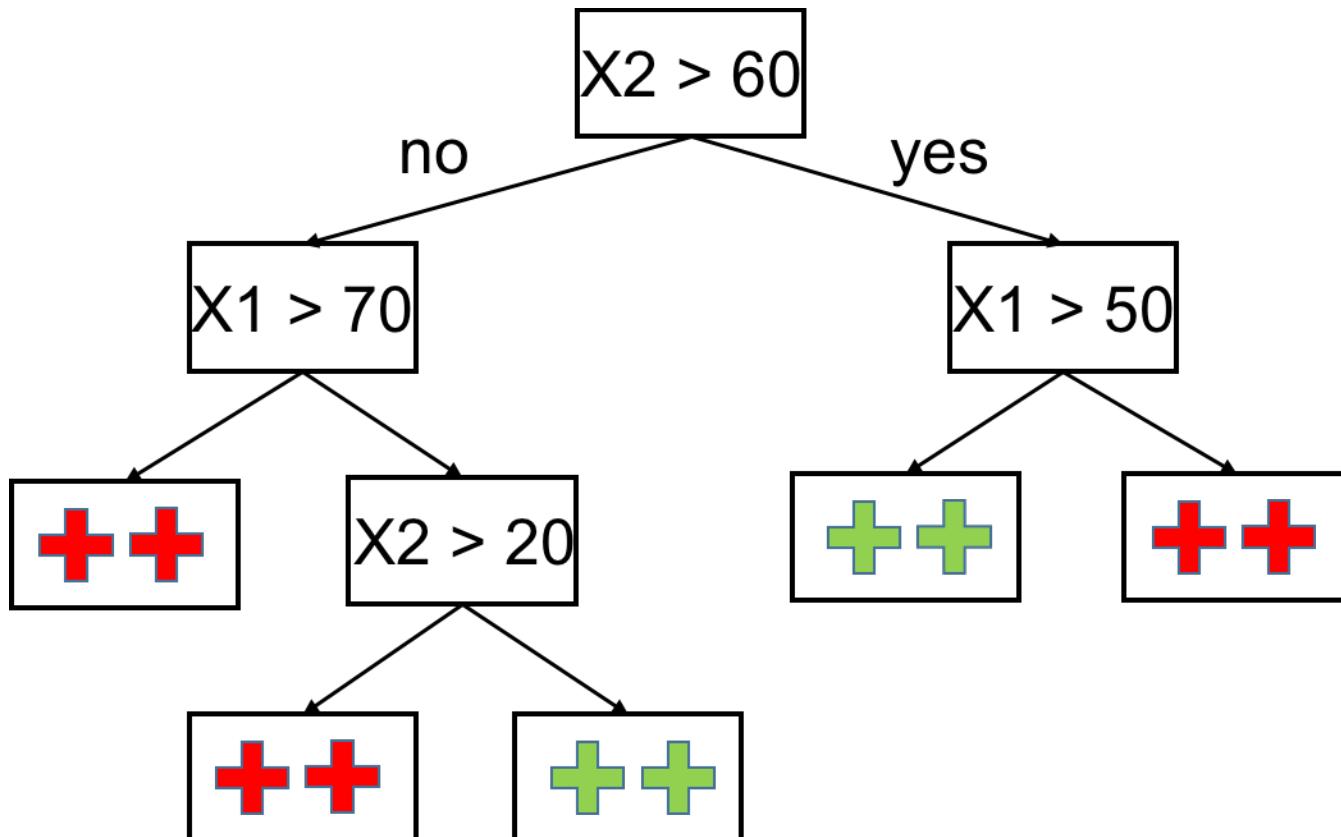


# Двоичное решающее дерево

- ▶ Дерево начинается из **одной** корневой вершины
- ▶ Данные  $X$  внутри вершины разбиваются по **одному** входному признаку. Это условие называют **предикатом**:  $\{x_j > t\}$
- ▶ Каждая вершина дерева имеет **два потомка**: правый и левый

$$R_r(j, t) = \{x | x_j > t\}$$
$$R_l(j, t) = \{x | x_j \leq t\}$$
- ▶ Вершины без потомков – **листья** дерева

# Двоичное дерево



# Жадный алгоритм построения дерева

```
1. def decision_tree(X, y):
2.     if termination_criterion(X, y) == True: ← ?
3.         R = create_leaf_with_prediction(y)
4.     else:
5.         R = create_node()
6.         (X_1, y_1), (X_2, y_2) = best_split(X, y) ← ?
7.         for i in [1, 2]:
8.             C = decision_tree(X_i, y_i)
9.             connect_nodes(R, C)
10.    return R
```

# Вопросы

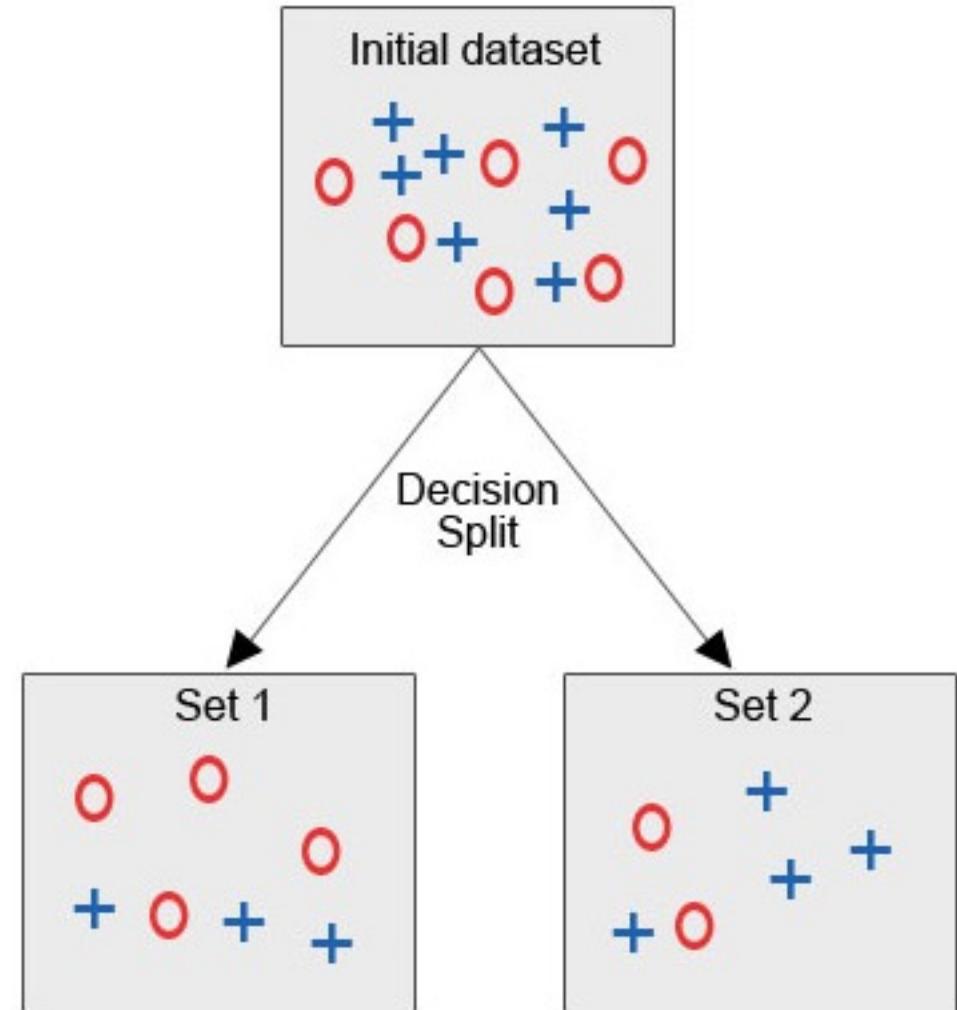
- ▶ Как находить предикаты для каждой вершины дерева?
- ▶ Какие есть критерии останова?

# Критерии информативности для классификации

---

# Выбор предиката

- ▶ Предикат делит вершину на две дочерние: левую и правую.
- ▶ Как понять, что разделение хорошее?



# Функционал качества предиката

- ▶ Для поиска лучшего предиката для вершины дерева зададим его функционал качества:

$$\Delta I_{node} = I_{node} - \left( I_{left} \frac{N_{left}}{N_{node}} + I_{right} \frac{N_{right}}{N_{node}} \right) \rightarrow \max_{j,t}$$

- $I_{node}$  - **критерий информативности** для вершины
- $N_{node}$  - число объектов в левой вершине дерева
- $\Delta I_{node}$  - функционал качества предиката (**information gain**)

# Критерий информативности

- ▶ Чем **меньше разнообразие** целевой переменной внутри вершины, тем **меньше** должно быть **значение** критерия информативности
- ▶ Соответственно, мы максимизируем функционал качества для каждого предиката

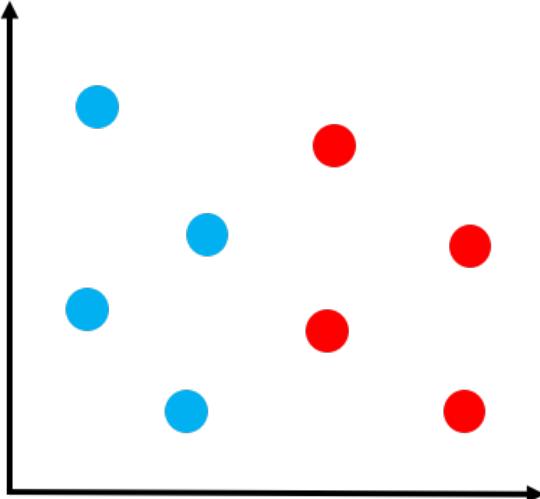
# Критерий Джини (Gini)

$$I_{node} = \sum_{k=1}^K p_k(1 - p_k)$$

$$p_k = \frac{1}{N_{node}} \sum_{i=1}^{N_{node}} [y_i = k]$$

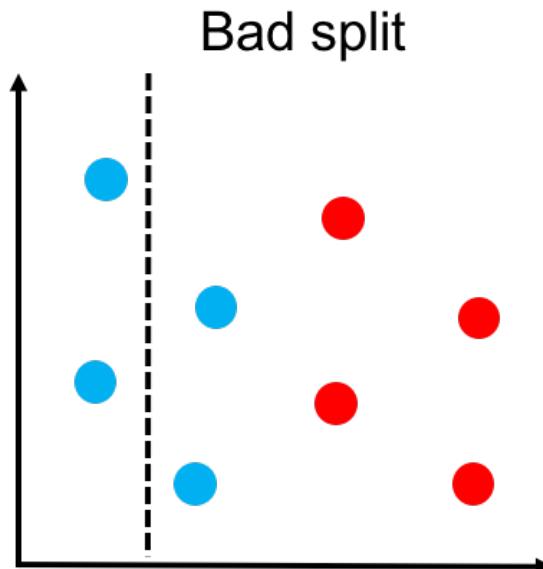
- $p_k$  – вероятность класса для вершины дерева
- К – общее число классов в задаче
- $N_{node}$  – число объектов в вершине дерева

# Пример



$$p = \frac{4}{8}, p = \frac{4}{8}$$

$$I = \frac{4}{8} \left(1 - \frac{4}{8}\right) + \frac{4}{8} \left(1 - \frac{4}{8}\right) = \frac{1}{2}$$



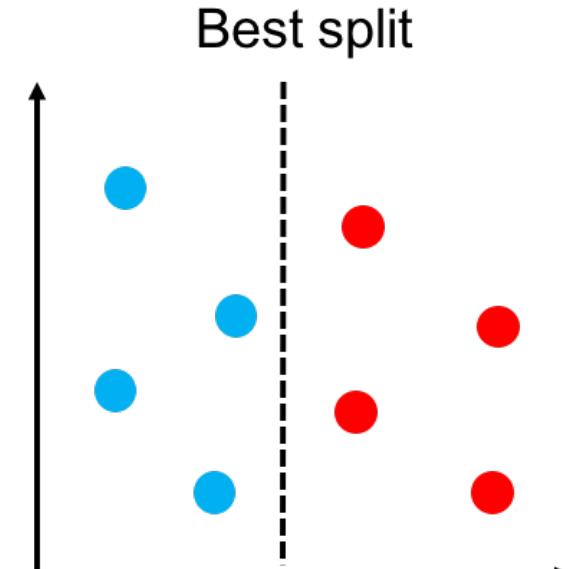
$$p = \frac{2}{2}, p = \frac{0}{2}$$

$$I_{left} = \frac{2}{2} \left(1 - \frac{2}{2}\right) + \frac{0}{2} \left(1 - \frac{0}{2}\right) = 0$$

$$p = \frac{2}{6}, p = \frac{4}{6}$$

$$I_{right} = \frac{2}{6} \left(1 - \frac{2}{6}\right) + \frac{4}{6} \left(1 - \frac{4}{6}\right) = \frac{4}{9}$$

$$\Delta I = \frac{1}{2} - 0 \frac{2}{8} - \frac{4}{9} \frac{6}{8} = \frac{1}{6}$$



$$p = \frac{4}{4}, p = \frac{0}{4}$$

$$I_{left} = \frac{4}{4} \left(1 - \frac{4}{4}\right) + \frac{0}{4} \left(1 - \frac{0}{4}\right) = 0$$

$$p = \frac{0}{4}, p = \frac{4}{4}$$

$$I_{right} = \frac{0}{4} \left(1 - \frac{0}{4}\right) + \frac{4}{4} \left(1 - \frac{4}{4}\right) = 0$$

$$\Delta I = \frac{1}{2} - 0 \frac{4}{8} - 0 \frac{4}{8} = \frac{1}{2}$$

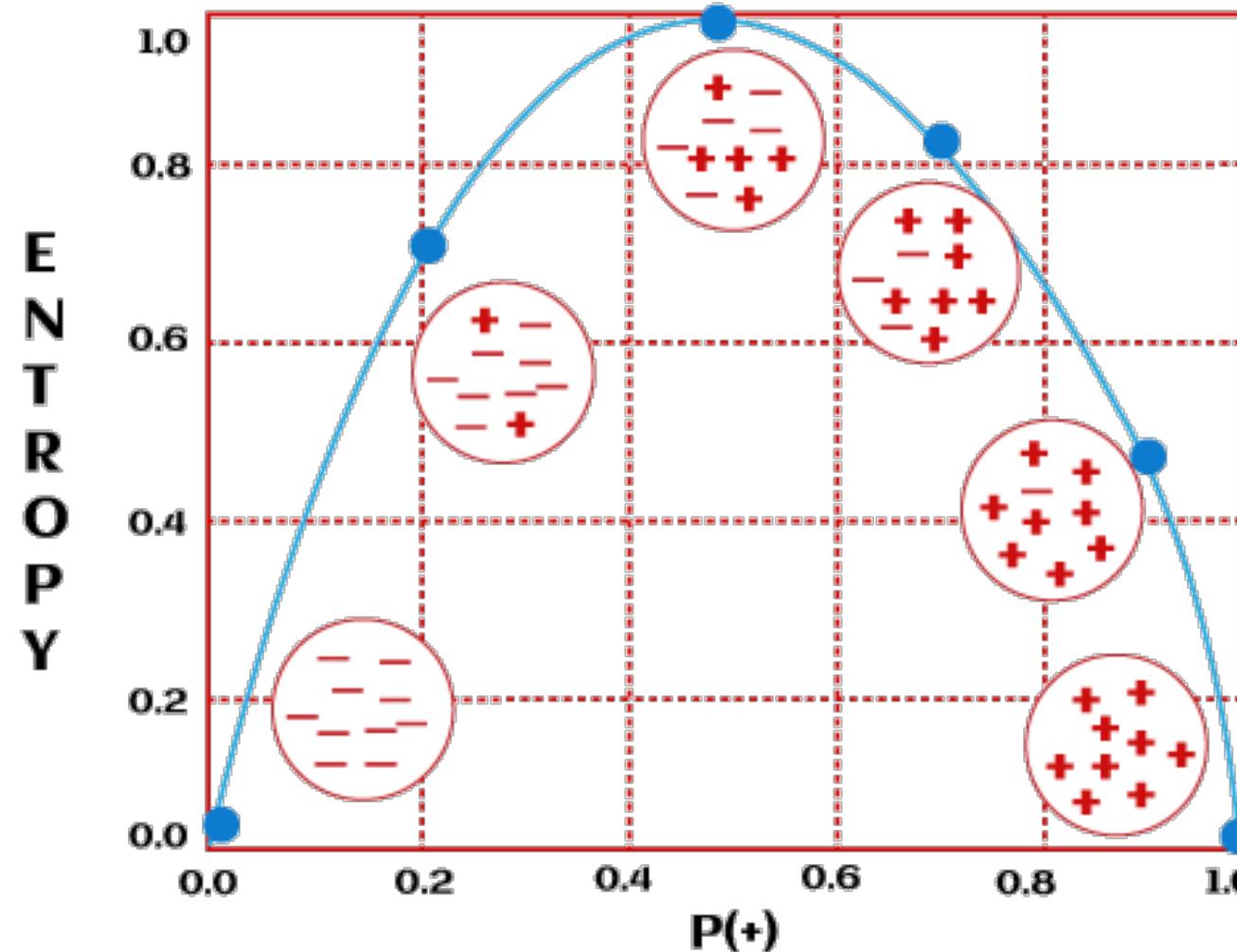
# Энтропийный критерий (Entropy)

$$I_{node} = - \sum_{k=1}^K p_k \log p_k$$

$$p_k = \frac{1}{N_{node}} \sum_{i=1}^{N_{node}} [y_i = k]$$

- $p_k$  – вероятность класса для вершины дерева
- К – общее число классов в задаче
- $N_{node}$  – число объектов в вершине дерева

# Энтропийный критерий (Entropy)



Источник: <https://www.javatpoint.com/entropy-in-machine-learning>

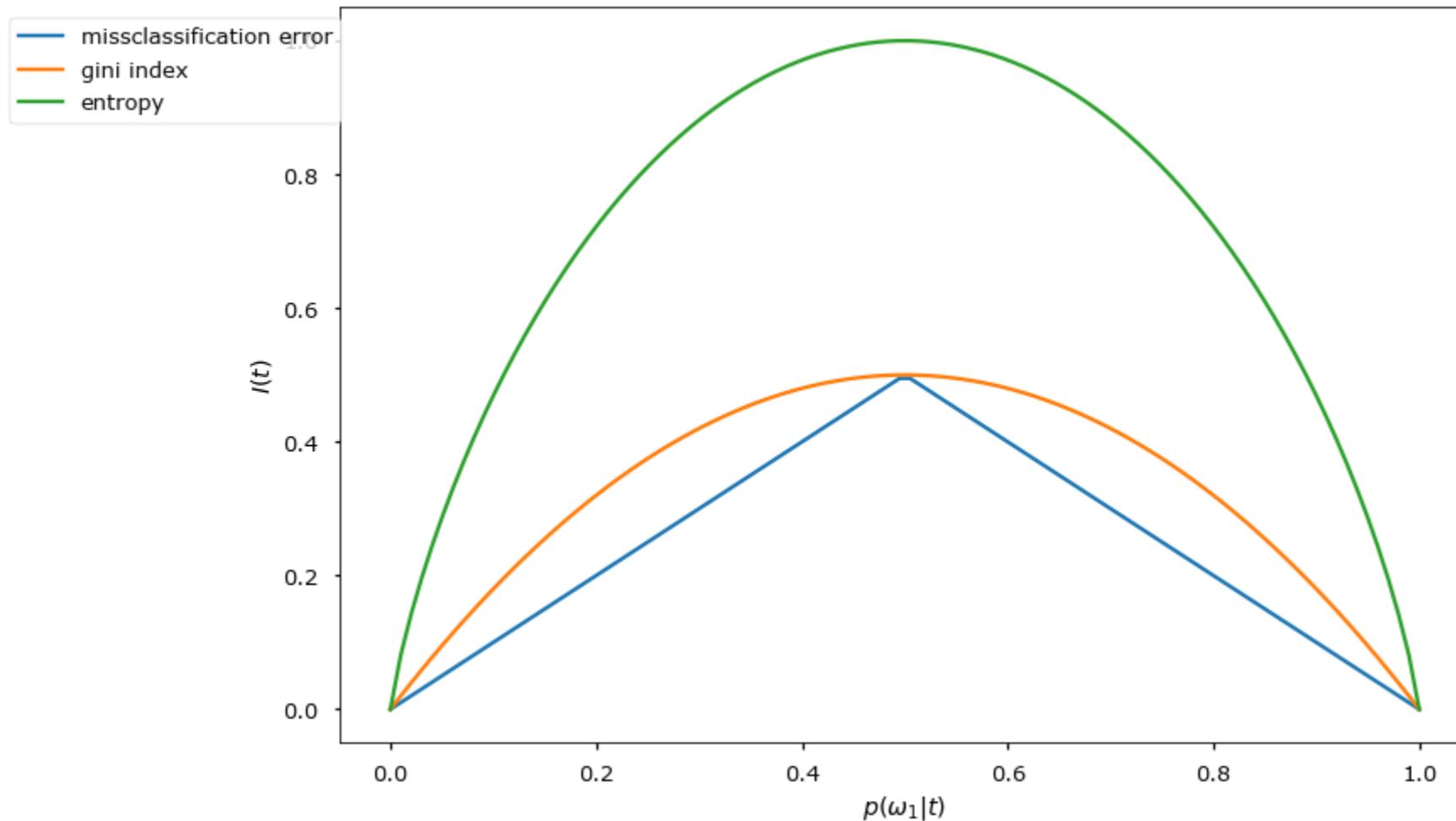
# Ошибка классификации

$$I_{node} = 1 - \max_k p_k$$

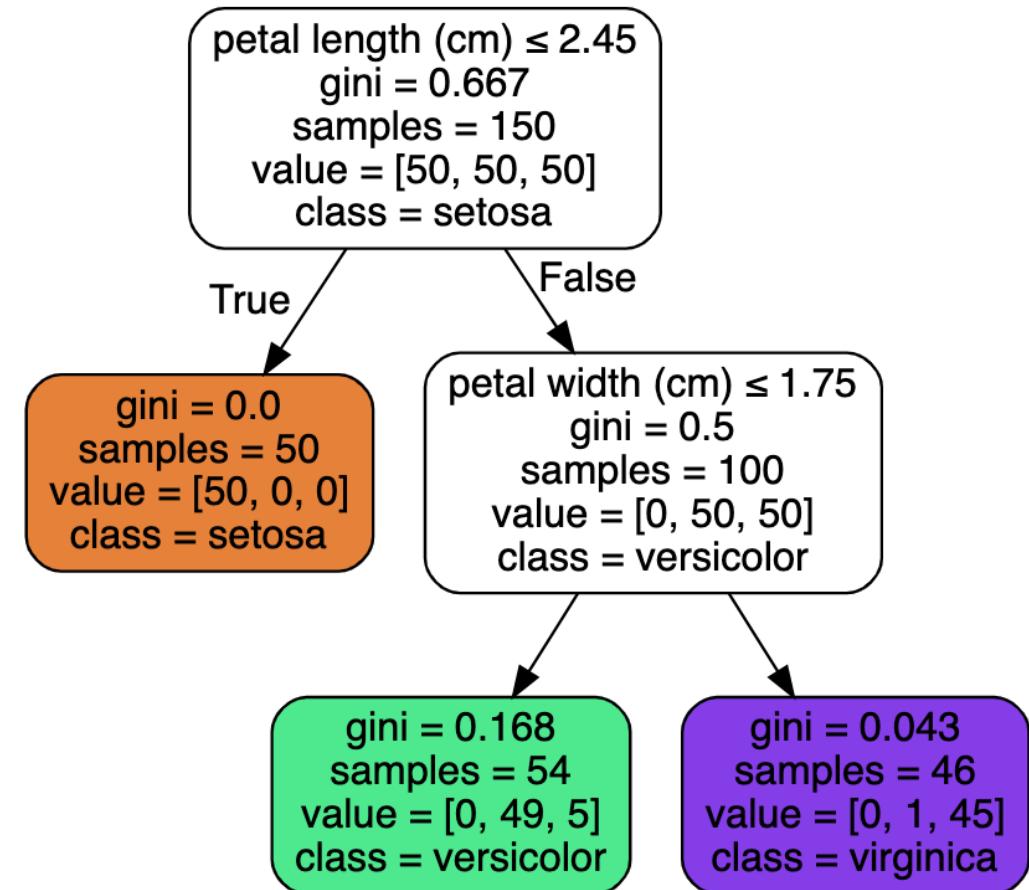
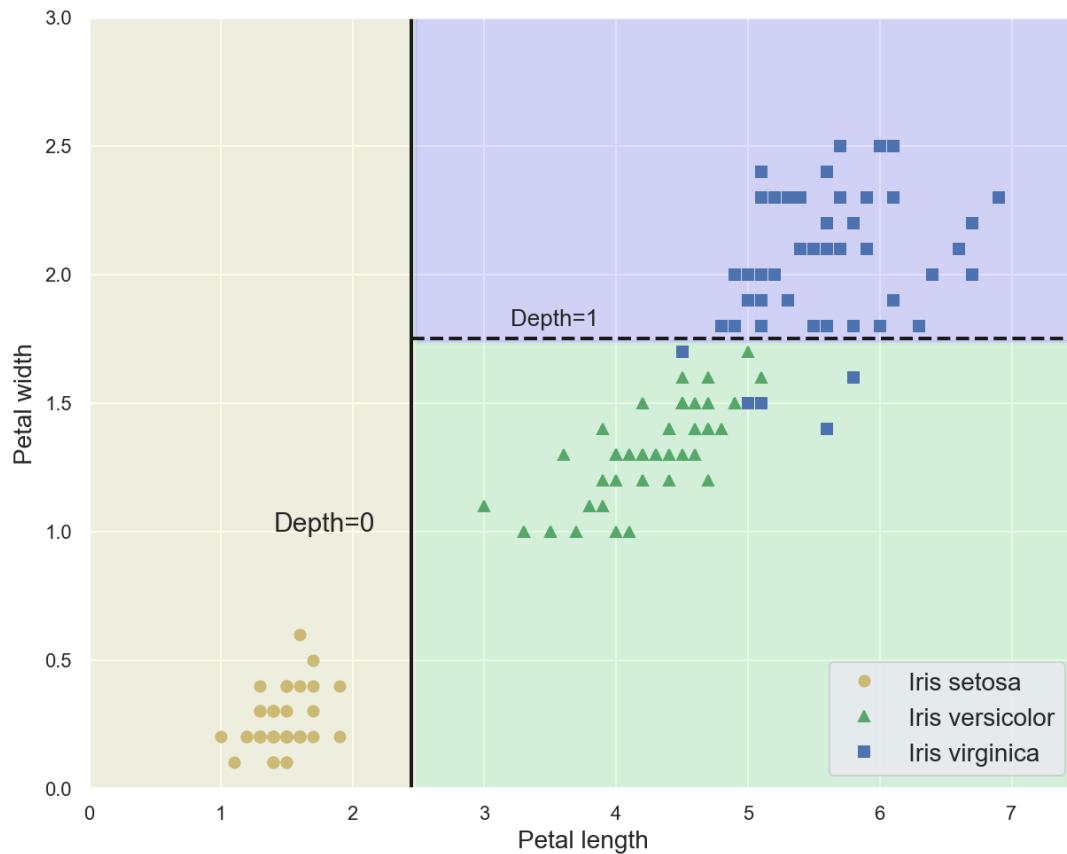
$$p_k = \frac{1}{N_{node}} \sum_{i=1}^{N_{node}} [y_i = k]$$

- $p_k$  – вероятность класса для вершины дерева
- К – общее число классов в задаче
- $N_{node}$  – число объектов в вершине дерева

# Критерии

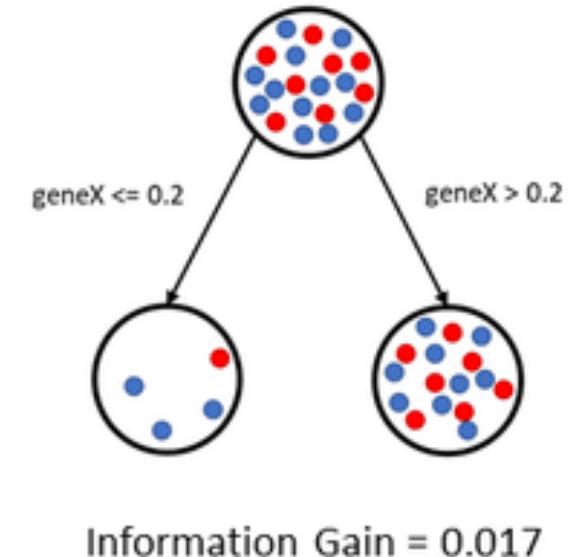
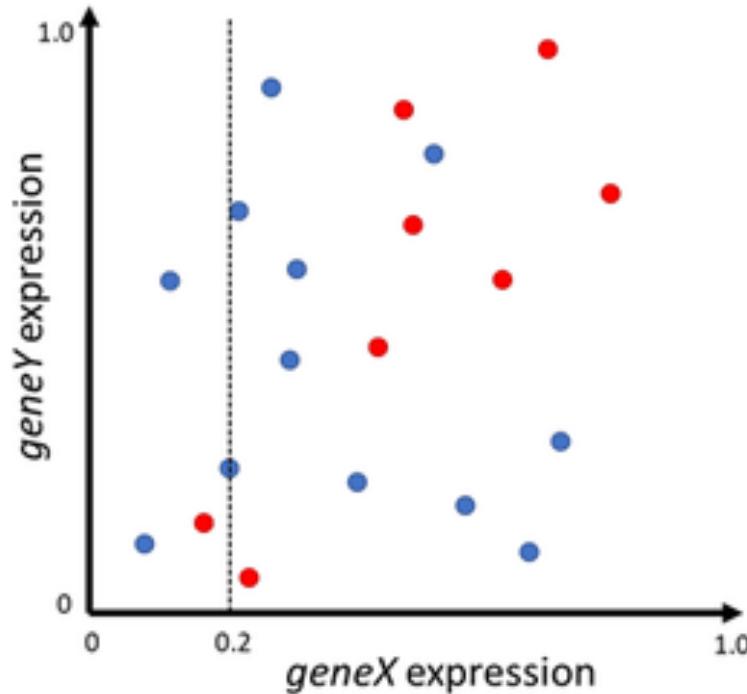
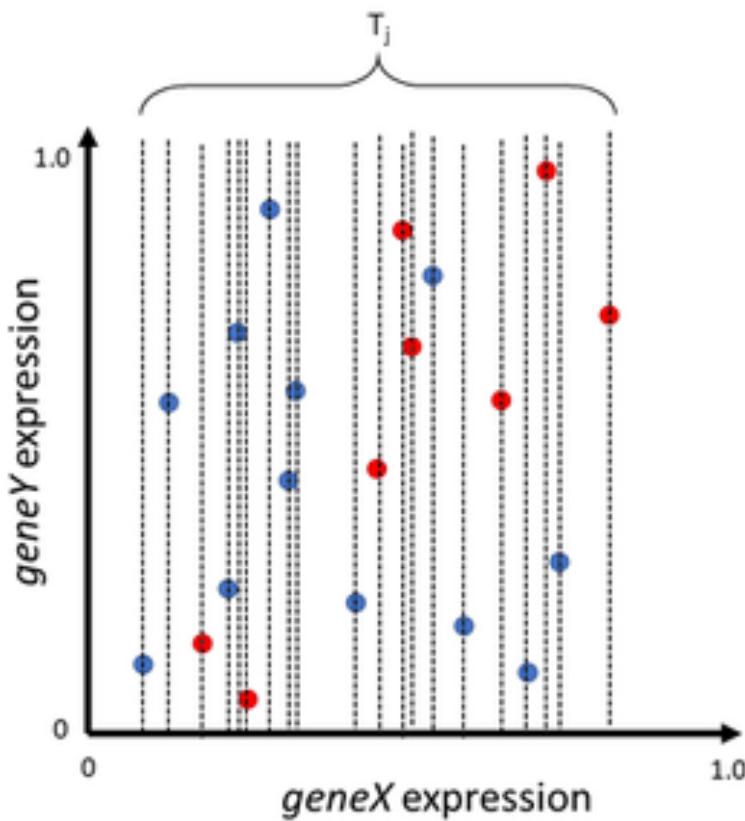


# Пример



# Строим дерево. Шаг 1

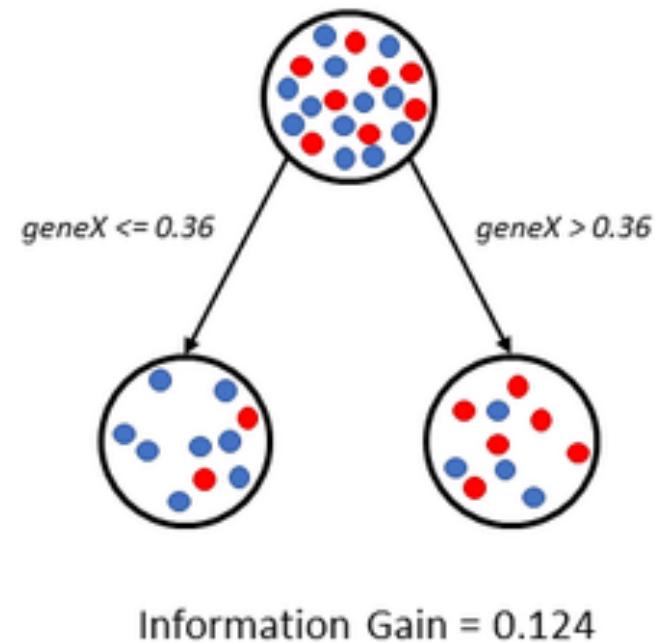
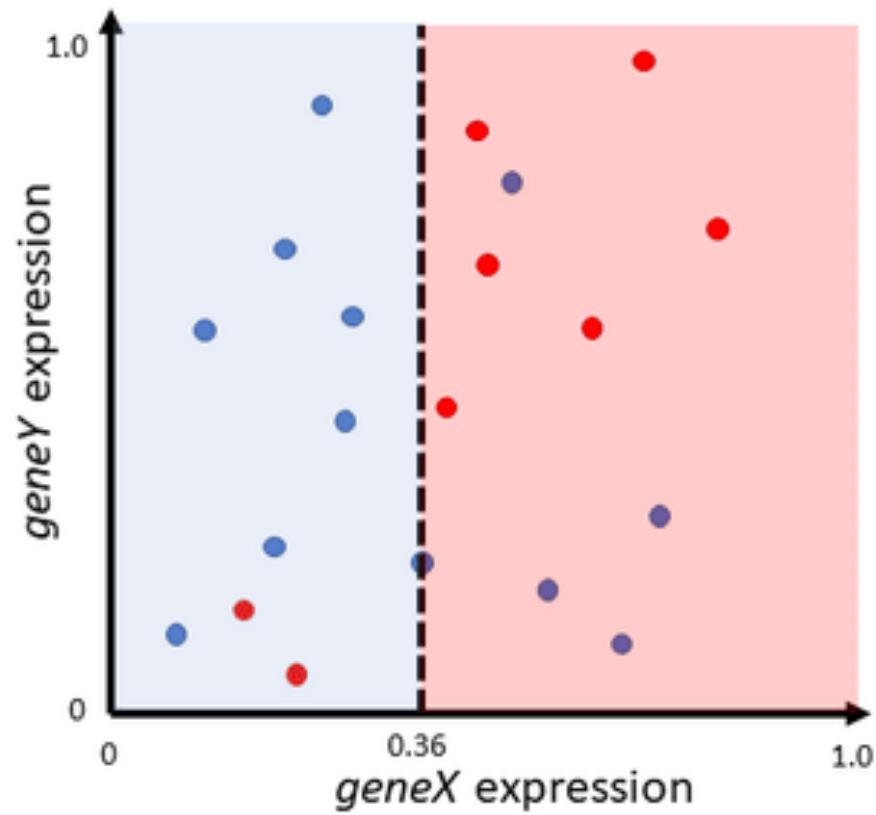
- i. For each feature,  $j$ , consider all possible thresholds  $T_j$  and calculate the cost function for each split



Источник: [https://www.researchgate.net/figure/Decision-tree-construction-protocol-with-the-Classification-and-Regression-tree-CART\\_fig2\\_347868061](https://www.researchgate.net/figure/Decision-tree-construction-protocol-with-the-Classification-and-Regression-tree-CART_fig2_347868061)

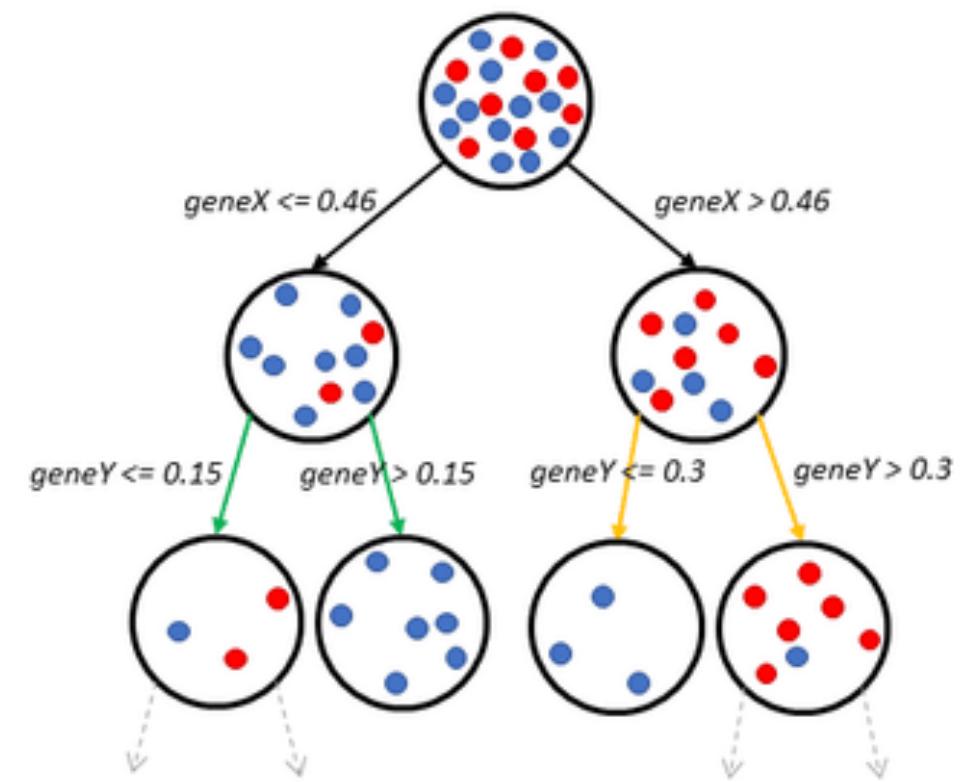
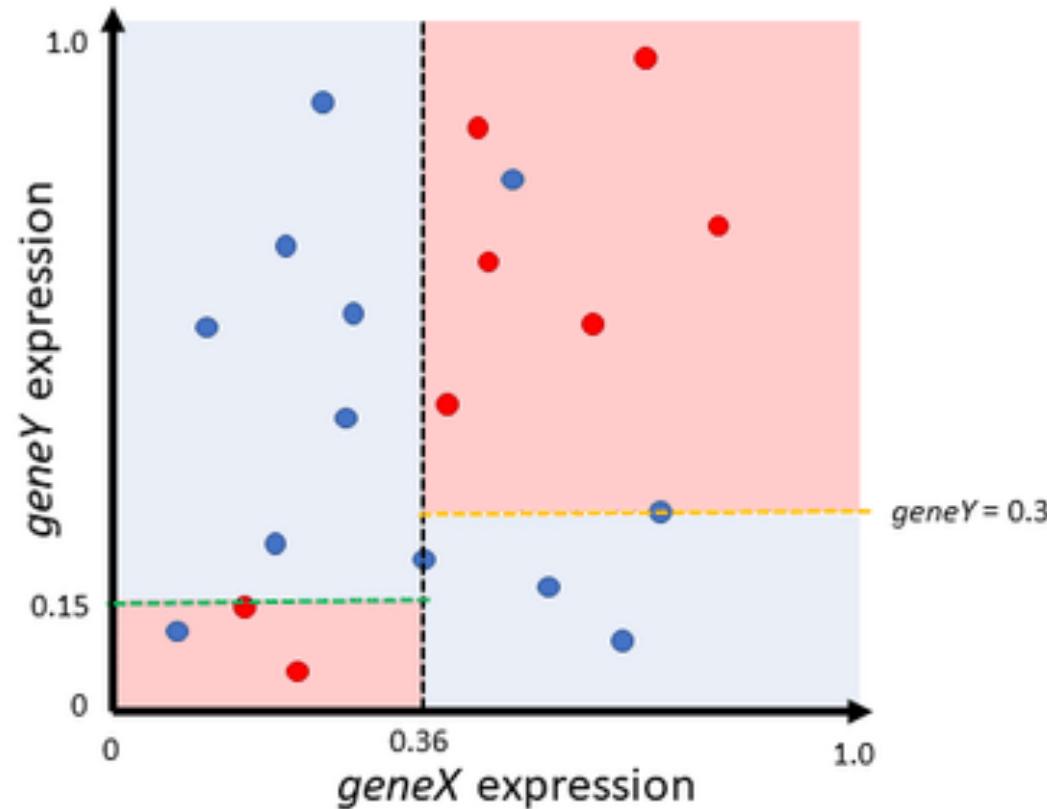
# Строим дерево. Шаг 2

- ii. Identify the best feature and threshold pair  $(j^*, t^*)$ , that returns the purest daughter nodes



# Строим дерево. Шаг 3

iii. Recurse on the daughter nodes to grow the tree, considering the next best feature-threshold pair



# Критерии информативности для регрессии

---

# Функционал качества предиката

- ▶ Для поиска лучшего предиката для вершины дерева зададим его функционал качества:

$$\Delta I_{node} = I_{node} - \left( I_{left} \frac{N_{left}}{N_{node}} + I_{right} \frac{N_{right}}{N_{node}} \right) \rightarrow \min_{j,t}$$

- $I_{node}$  - **критерий информативности** для вершины
- $N_{node}$  - число объектов в левой вершине дерева
- $\Delta I_{node}$  - функционал качества предиката (**information gain**)

# Среднеквадратичная ошибка (MSE)

$$I_{node} = \frac{1}{N_{node}} \sum_{i=1}^{N_{node}} (y_i - \mu)^2$$

$$\mu = \frac{1}{N_{node}} \sum_{i=1}^{N_{node}} y_i$$

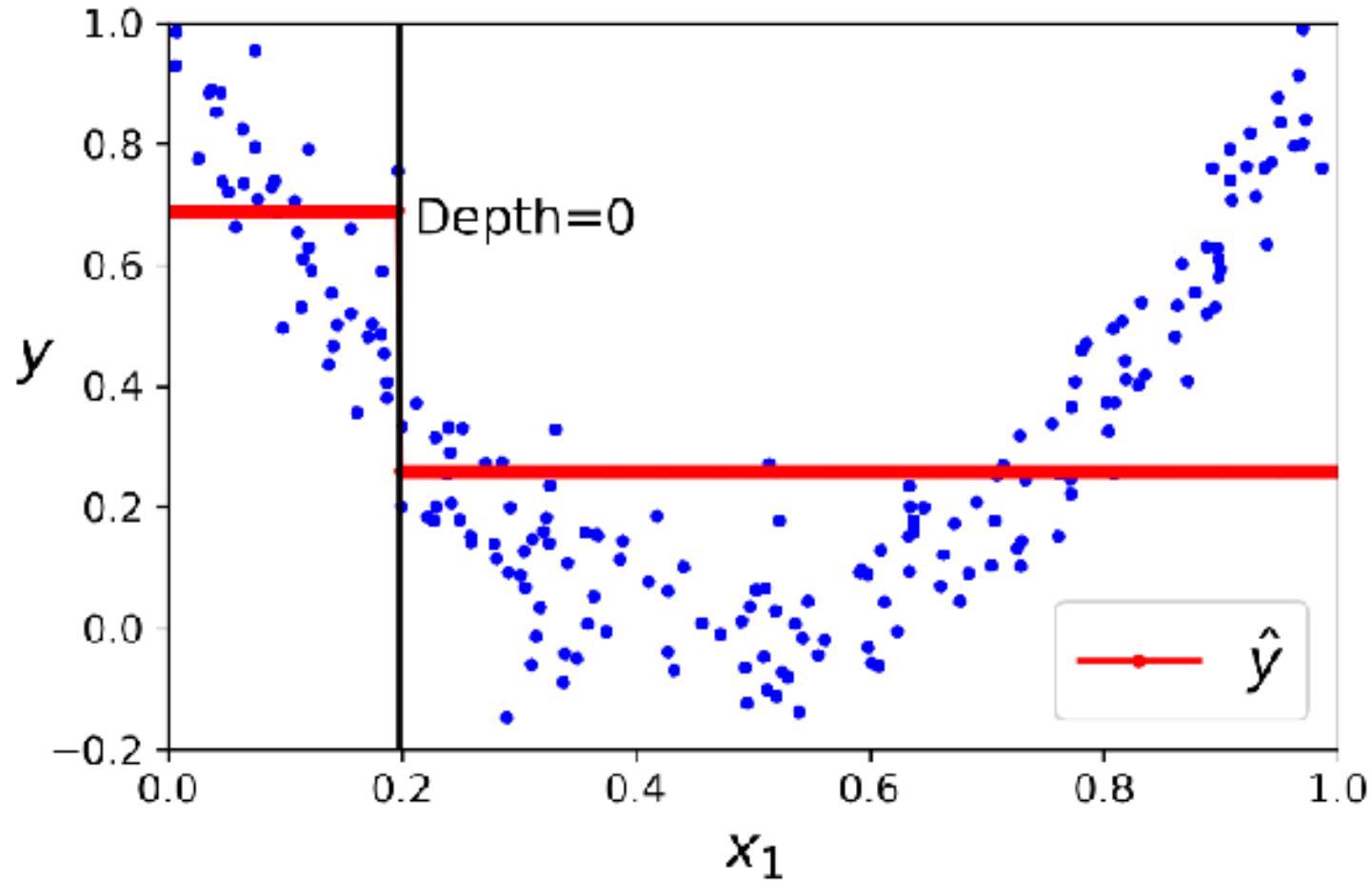
- $\mu$  – **среднее** значения целевой переменной в вершине дерева
- $N_{node}$  – число объектов в вершине дерева

# Средняя абсолютная ошибка (MAE)

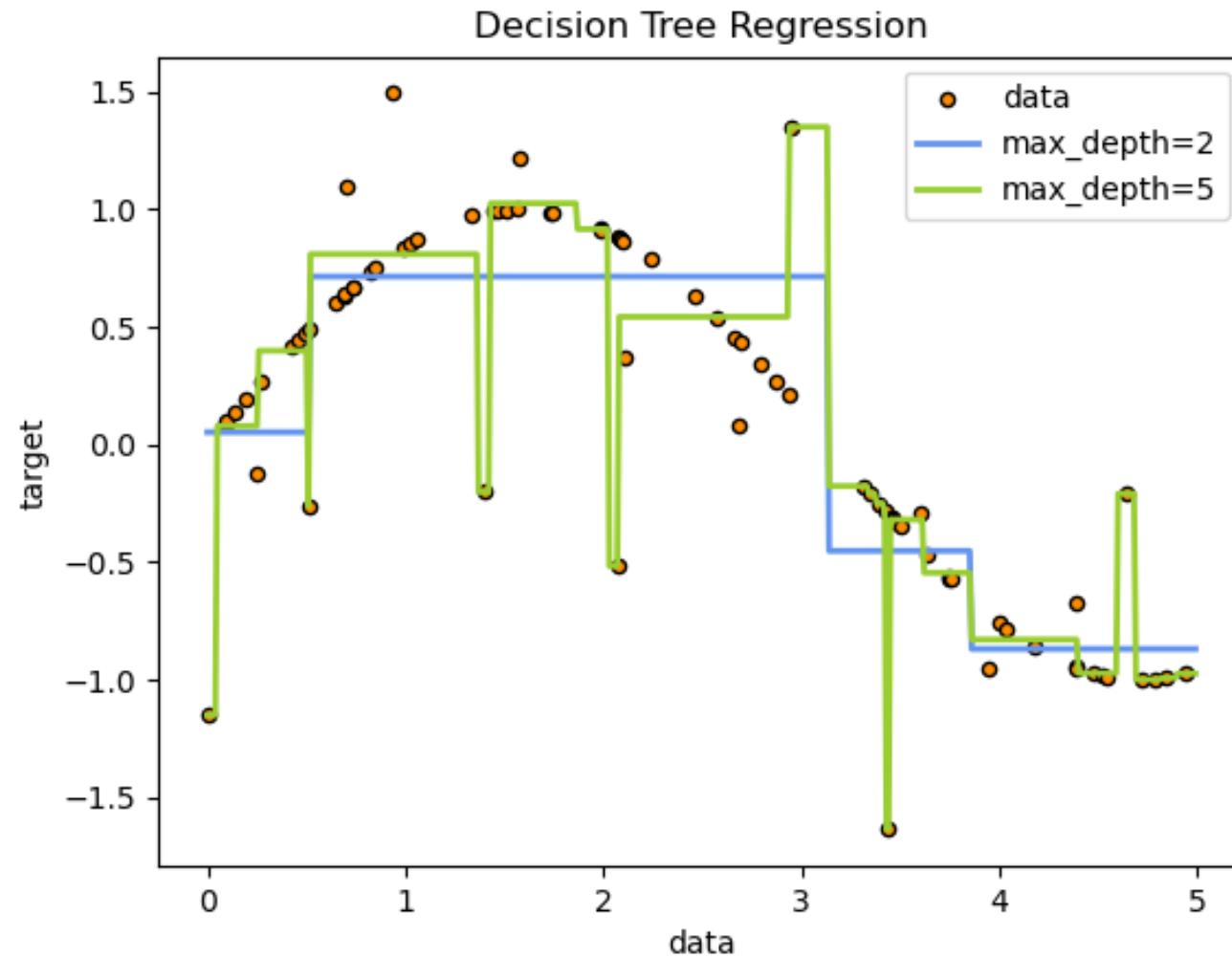
$$I_{node} = \frac{1}{N_{node}} \sum_{i=1}^{N_{node}} |y_i - \mu|$$

- $\mu$  – **медианное** значения целевой переменной в вершине дерева
- $N_{node}$  – число объектов в вершине дерева

# Пример



# Пример



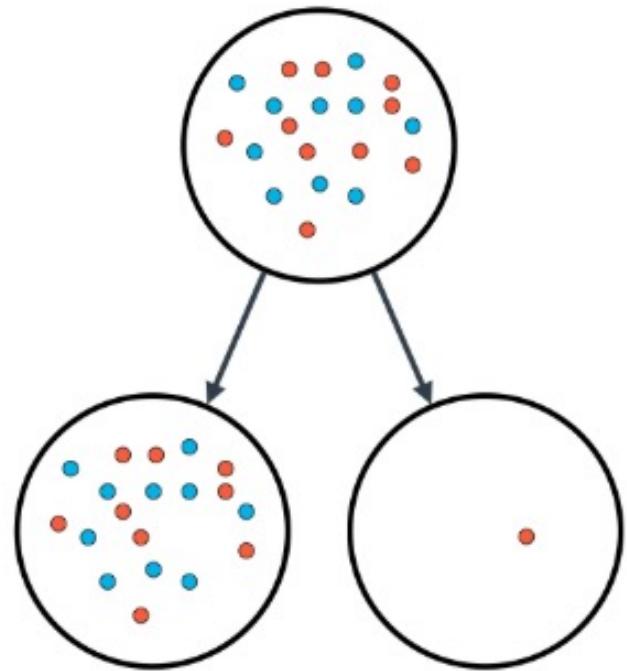


Критерии останова

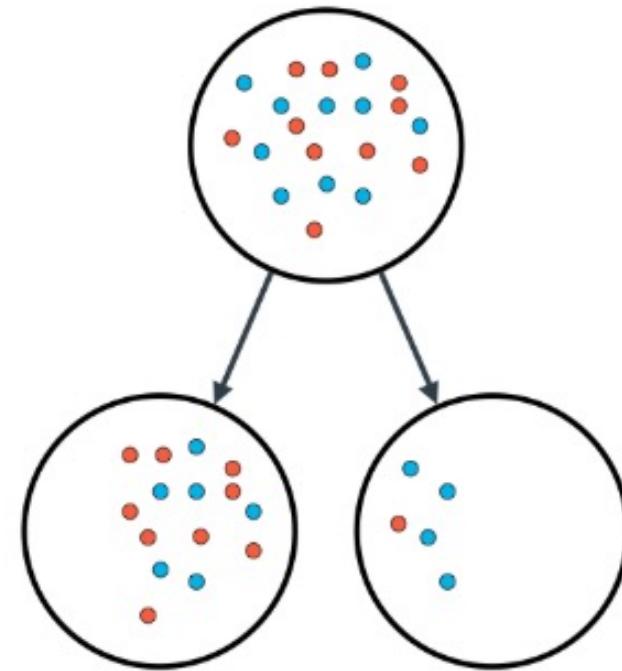
# Критерии останова

- ▶ Мы строим дерево до тех пор, пока не выполнится некоторые условия – критерии останова
- ▶ **Примеры критериев:**
  - Максимальная глубина дерева
  - Минимальное число объектов внутри листа дерева
  - Максимальное число листьев дерева
  - Останов, если все объекты в вершине относятся к одному классу
  - Требование, чтобы  $\Delta I_{node} > h$

# Минимальное число объектов внутри листа



Minimum samples per leaf = 1



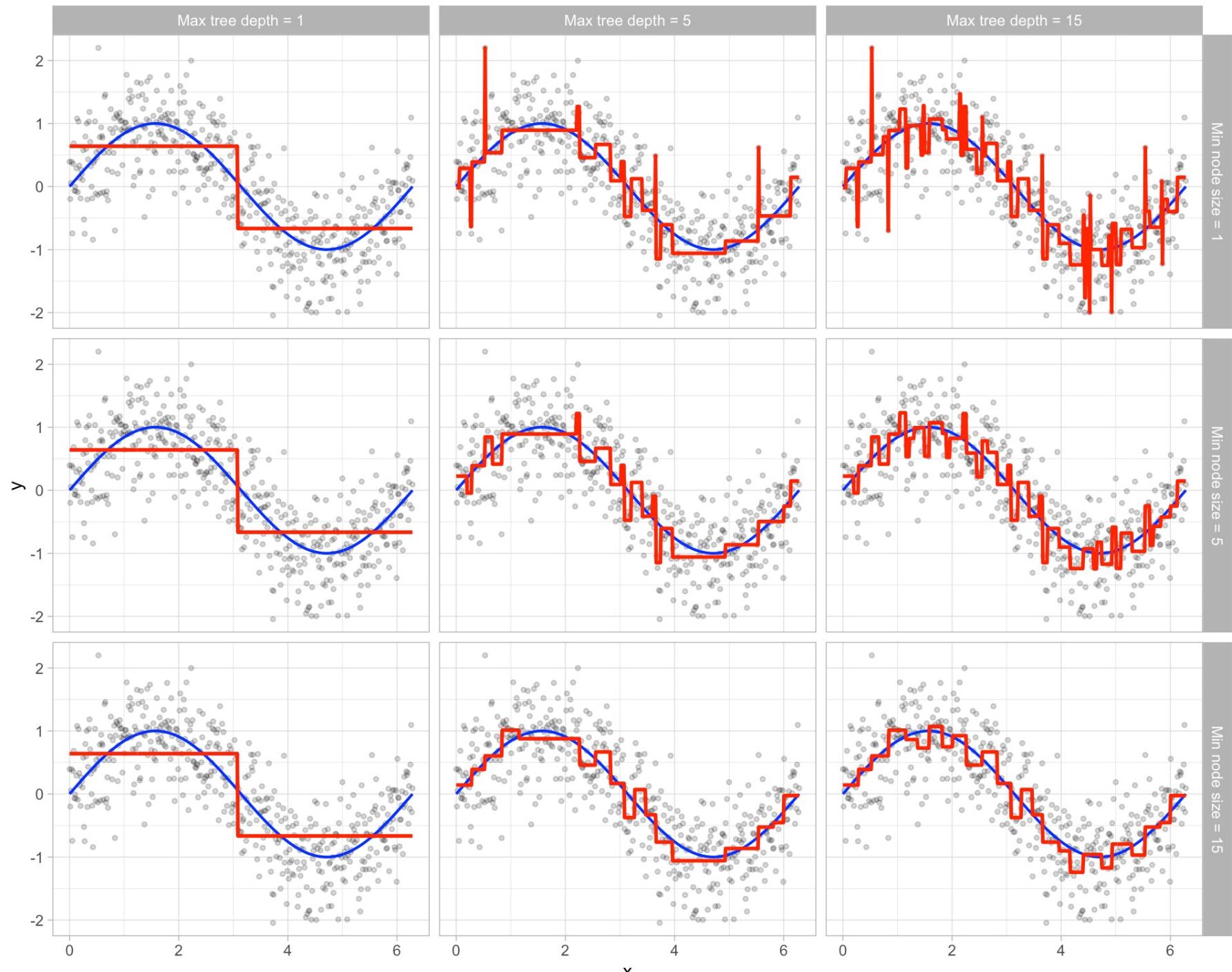
Minimum samples per leaf = 5

Minimum number of samples per leaf

# Глубина дерева

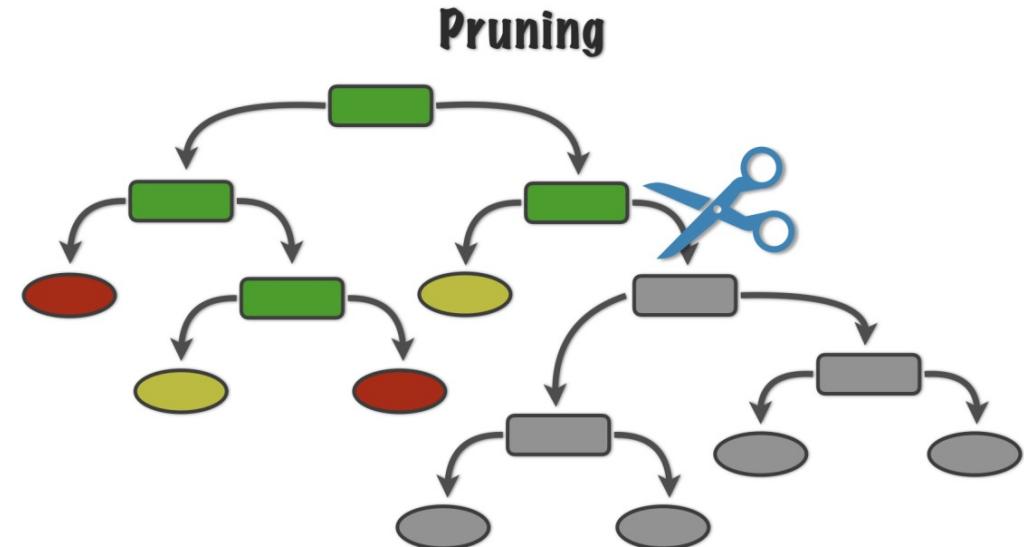
- ▶ Критерии останова влияют на качество прогнозов дерева
- ▶ Критерии останова влияют на глубину решающего дерева
- ▶ Очень глубокие деревья **переобучаются**
- ▶ Неглубокие деревья **недообучаются**

# Пример



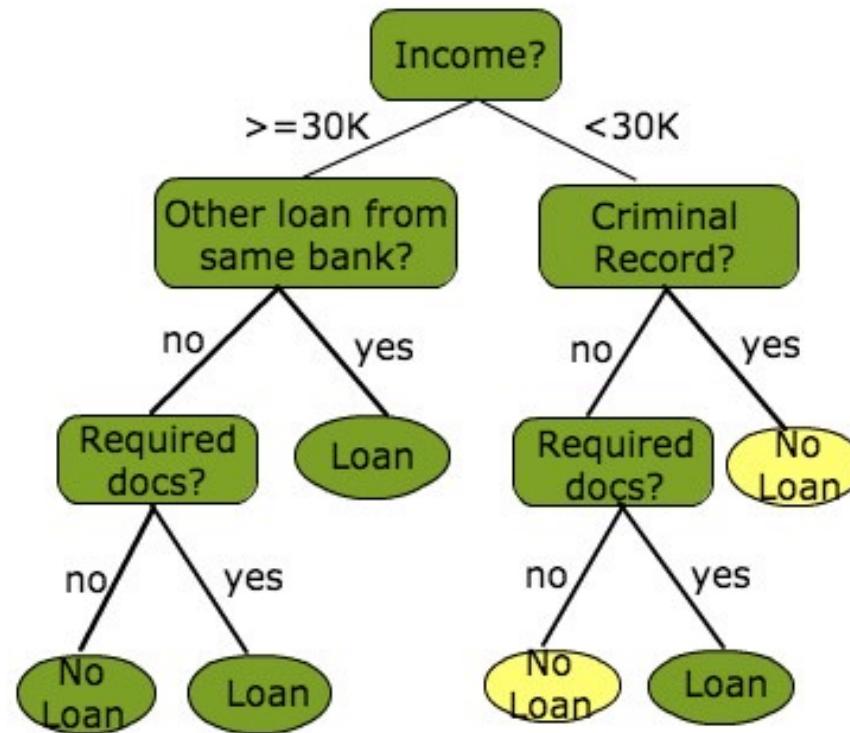
# Стрижка (pruning) деревьев

- ▶ Строим переобученное дерево
- ▶ Отрезаем его «ветки» так, чтобы на тестовой выборке было наилучшее качество
- ▶ Есть несколько методов стрижки деревьев
- ▶ На практике применяется редко

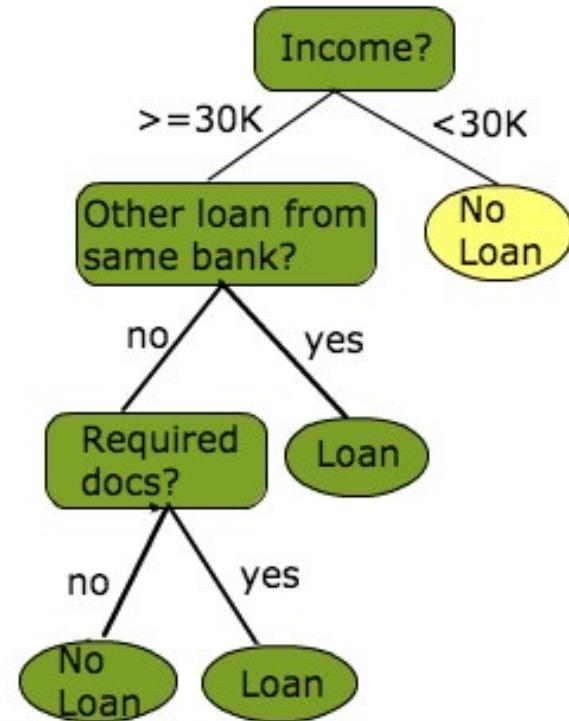


# Пример

An Unpruned Decision Tree



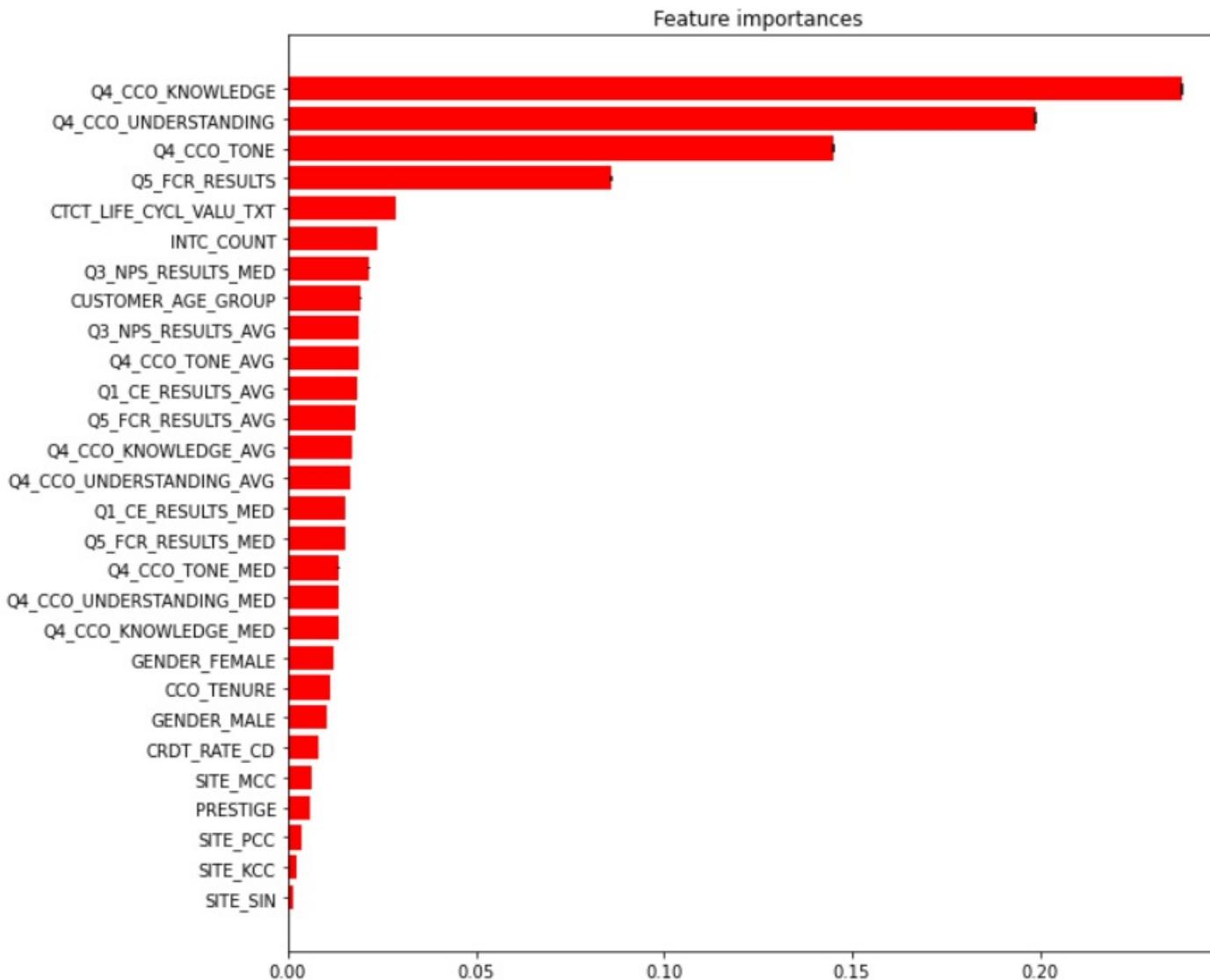
A Pruned Decision Tree



# Важность признаков

---

# Важность признаков



# Важность признаков

- ▶ Решающее дерево позволяет оценить вклад каждого входного признака в прогноз
- ▶ **Важность признака  $j$**  (feature importance):

$$FIM_j = \sum_{node \in T(j)} N_{node} \Delta I_{node}$$

- $T(j)$  – **все вершины дерева, где** предикаты использовали признак  $j$ :  $\{x_j > t\}$

# Обработка пропусков

- ▶ **Обучение** решающего дерева:
  - Пусть для признака  $j$  некоторые значения пропущены
  - При поиске лучшего предиката  $\{x_j > t\}$  не используем пропуски  $x_j$ . Считаем  $\Delta I_{node}$  только на известных  $x_j$ .
- ▶ **Применение** решающего дерева:
  - Если пропущено значение  $x_j = None$  для объекта, то идем и в лево, и в право.  
Делаем прогноз в обоих поддеревьях
  - Итоговый прогноз  $\hat{y}$  для объекта усредняем:

$$\hat{y} = \frac{N_{left}}{N_{node}} \hat{y}_l + \frac{N_{right}}{N_{node}} \hat{y}_r$$

# Методы построения деревьев

- ▶ ID3
  - Энтропийный критерий
  - Только категориальные признаки
- ▶ C4.5
  - Критерий Gain Ratio
  - Error-Based Prunning
  - Обработка пропусков как описали выше
- ▶ CART (sklearn)
  - Критерий Джини
  - Cost-Complexity Pruning
  - Метод суррогатный предикатов для обработки пропусков

# Заключение



# Резюме

- ▶ Даны данные  $X, y$
- ▶ Находим **предикат**  $\{x_j > t\}$  для вершины дерева:

$$\Delta I_{node} = I_{node} - \left( I_{left} \frac{N_{left}}{N_{node}} + I_{right} \frac{N_{right}}{N_{node}} \right) \rightarrow \max_{j,t}$$

- ▶ Строим две **дочерние вершины**: правую и левую

$$R_r(j, t) = \{x | x_j > t\}$$

$$R_l(j, t) = \{x | x_j \leq t\}$$

- ▶ **Рекурсивно** находим предикаты для всех дочерних вершин
- ▶ Если выполняются **критерии останова**, останавливаемся

# Вопросы

- ▶ Что такое решающее дерево? Запишите формулу предсказания решающего дерева (через разбиение признакового пространства на области).
- ▶ Опишите жадный алгоритм обучения решающего дерева. Запишите функционал, который оптимизируется при построении вершины решающего дерева.
- ▶ Какими основными свойствами должен обладать критерий информативности? Как он используется для выбора предиката во внутренней вершине решающего дерева?
- ▶ Запишите критерий Джини и энтропийный критерий информативности.