



# Деревья принятия решений

# Основные элементы

Решающее дерево – ациклический граф, основными элементами которого являются:

- вершины: предикаты – функции, принимающие значения  $\{0,1\}$ ;
- листья (терминальные вершины): содержат значения целевой переменной;
- ребра: значения предикатов, из которых выходит ребро.

Ответ алгоритма: соответствующее предикатам объекта значение целевой переменной в терминальной вершине.

# Пример

Стоит задача предсказать, выиграет ли «Зенит» свой следующий матч.

Список бинарных признаков:

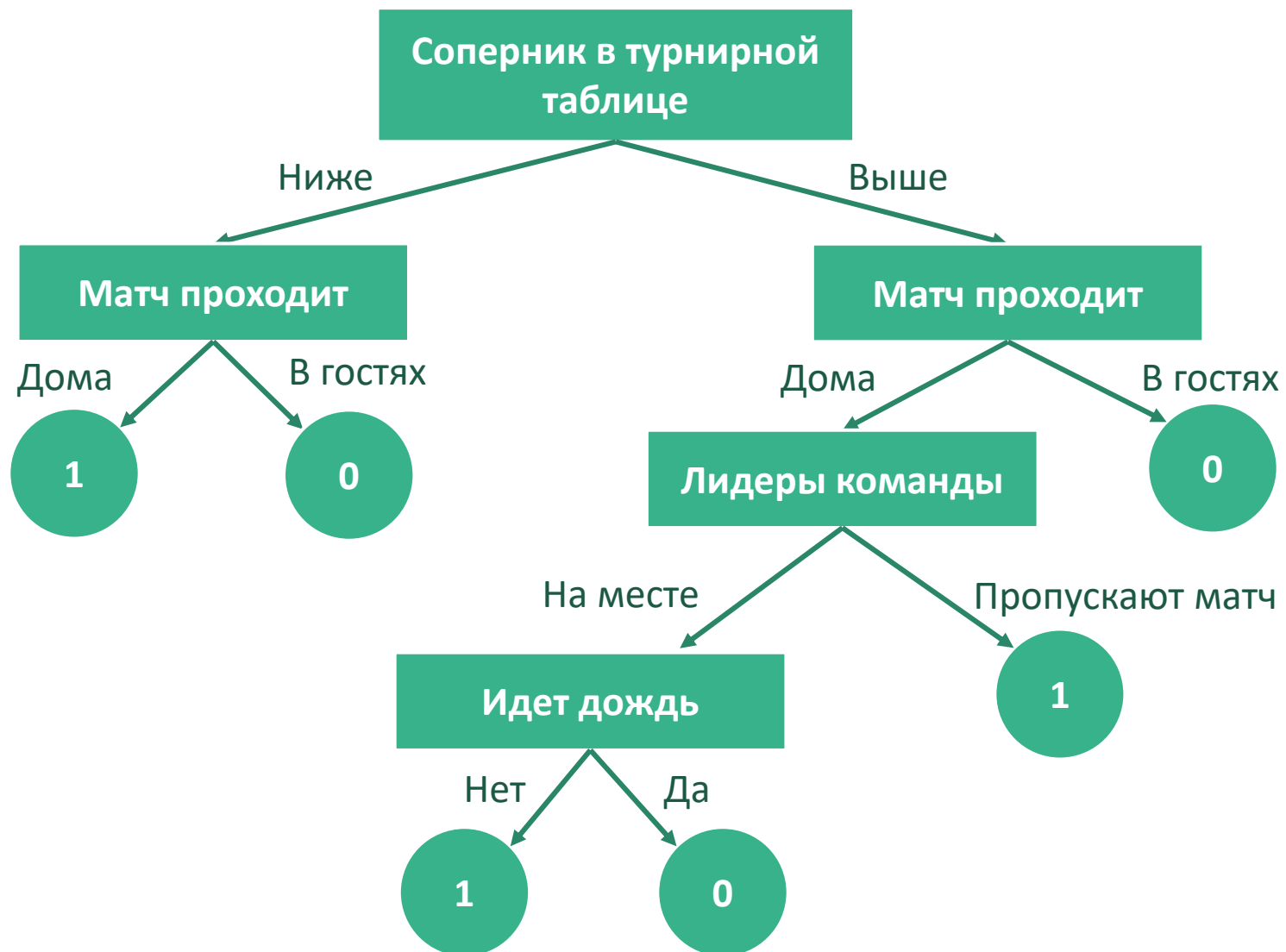
- находится ли соперник выше в турнирной таблице;
- проходит ли матч на домашнем стадионе;
- пропускают ли матч лидеры команды-соперника;
- обещают ли в этот день дождь.

Зная историю проведенных «Зенитом» игр, можно попробовать предсказать результат предстоящего матча.

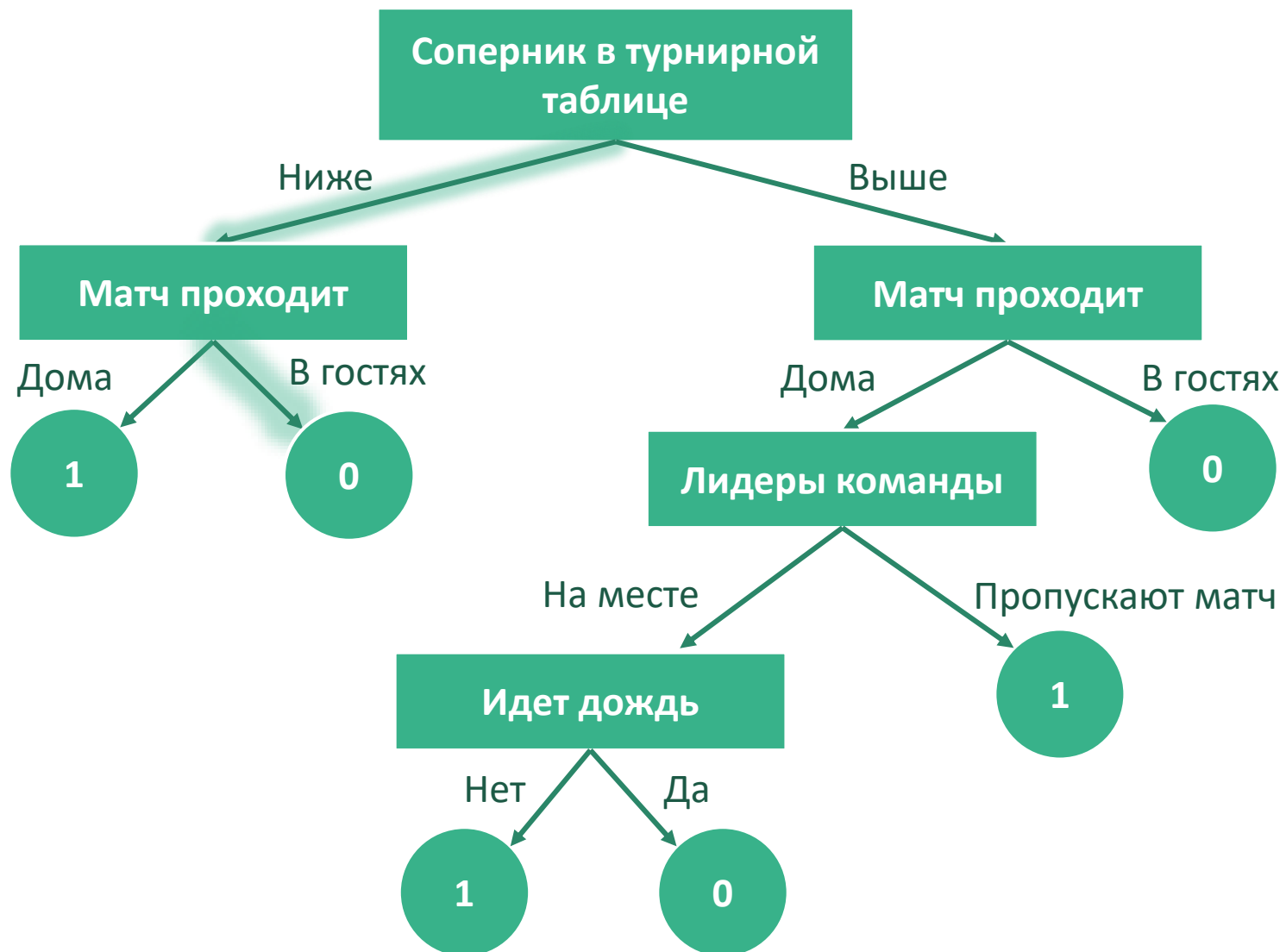
# История матчей «Зенита»

Соперник	Играем	Лидеры	Дождь	Победа
Выше	Дома	На месте	Да	Нет
Выше	Дома	На месте	Нет	Да
Выше	Дома	Пропускают	Нет	Да
Ниже	Дома	Пропускают	Нет	Да
Ниже	В гостях	Пропускают	Нет	Нет
Ниже	Дома	Пропускают	Да	Да
Выше	В гостях	На месте	Да	Нет
Ниже	В гостях	На месте	Нет	?

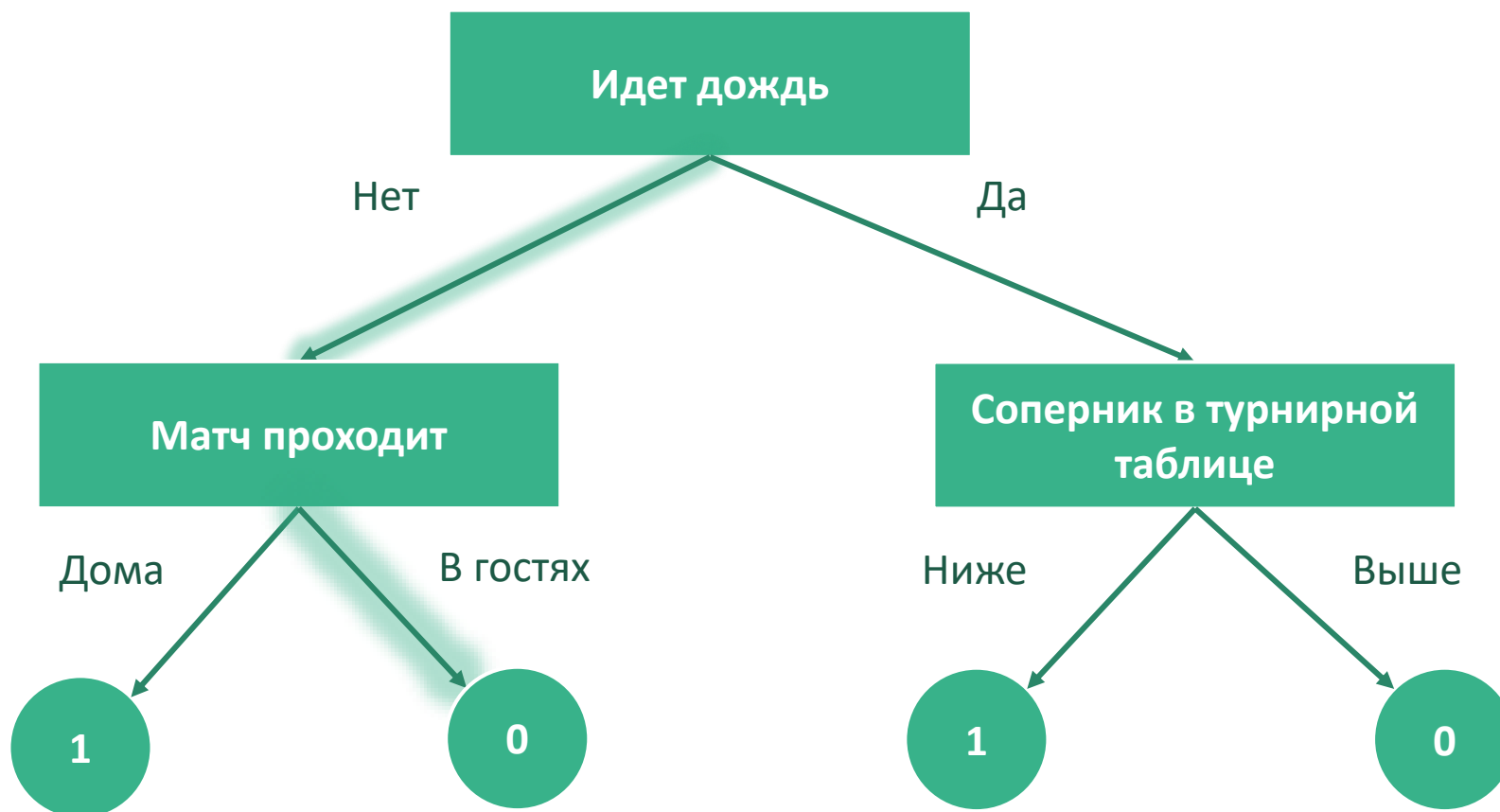
# Пример решающего дерева



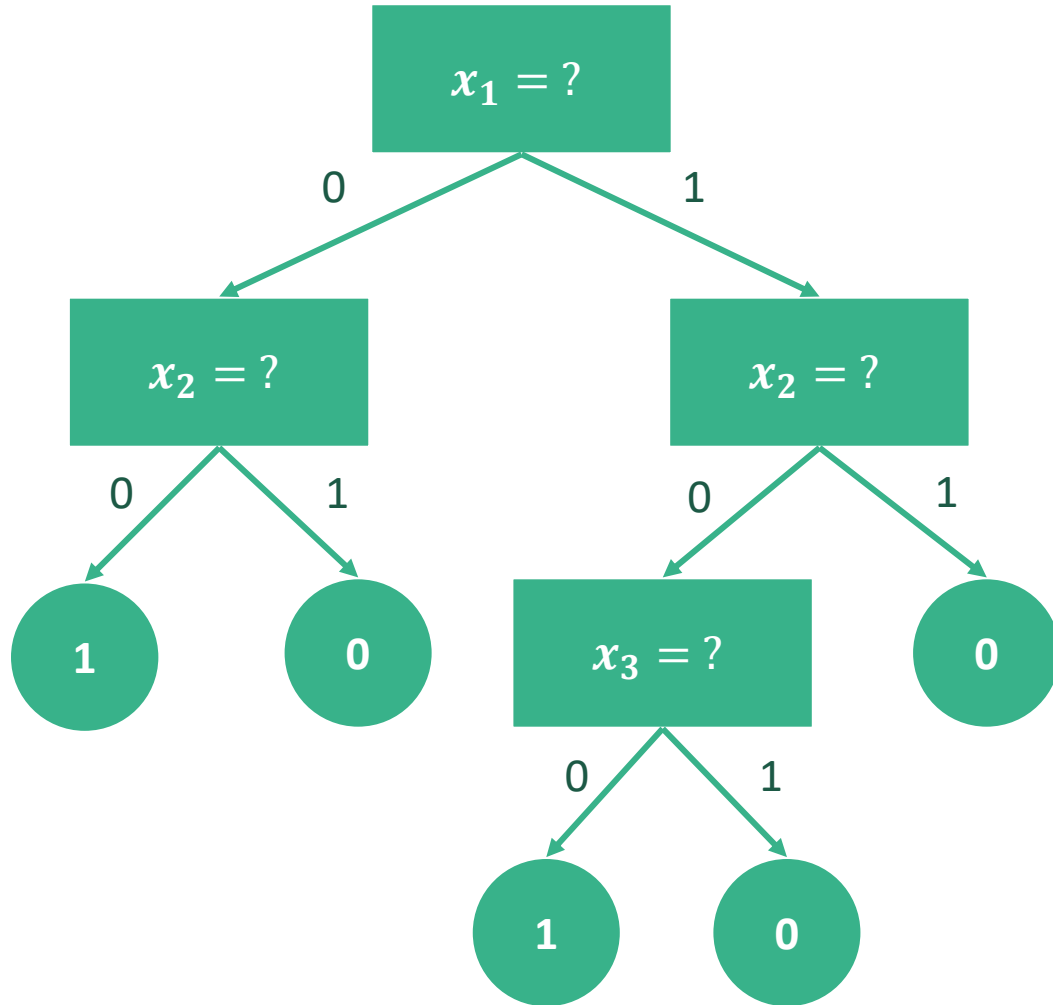
# Пример решающего дерева



# Оптимальное дерево



# Деревья и булевы функции



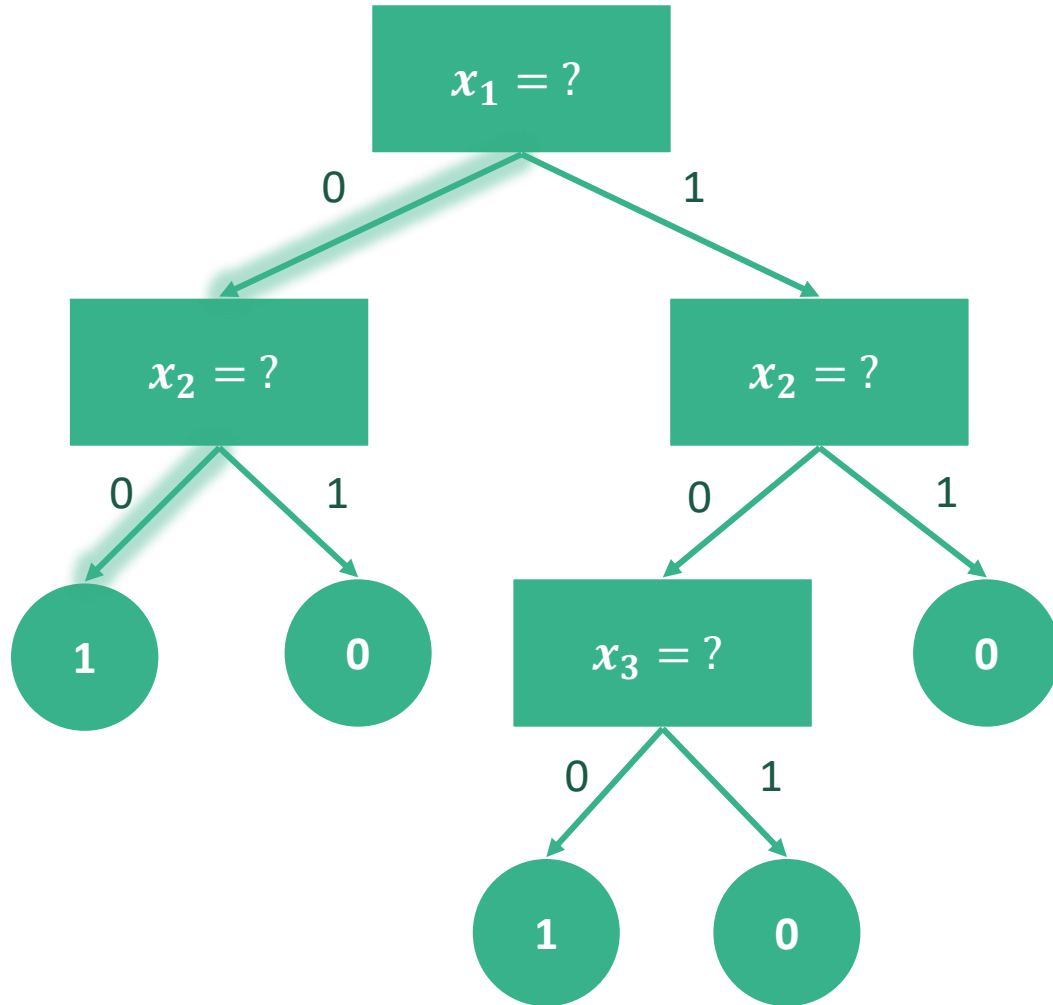
Дерево решений может быть представлено в виде булевой функции в ДНФ.

Например, дерево на рисунке соответствует функции:

$$f(x_1, x_2, x_3) = (\overline{x_1} \wedge \overline{x_2}) \vee (x_1 \wedge \overline{x_2} \wedge \overline{x_3})$$



# Деревья и булевы функции

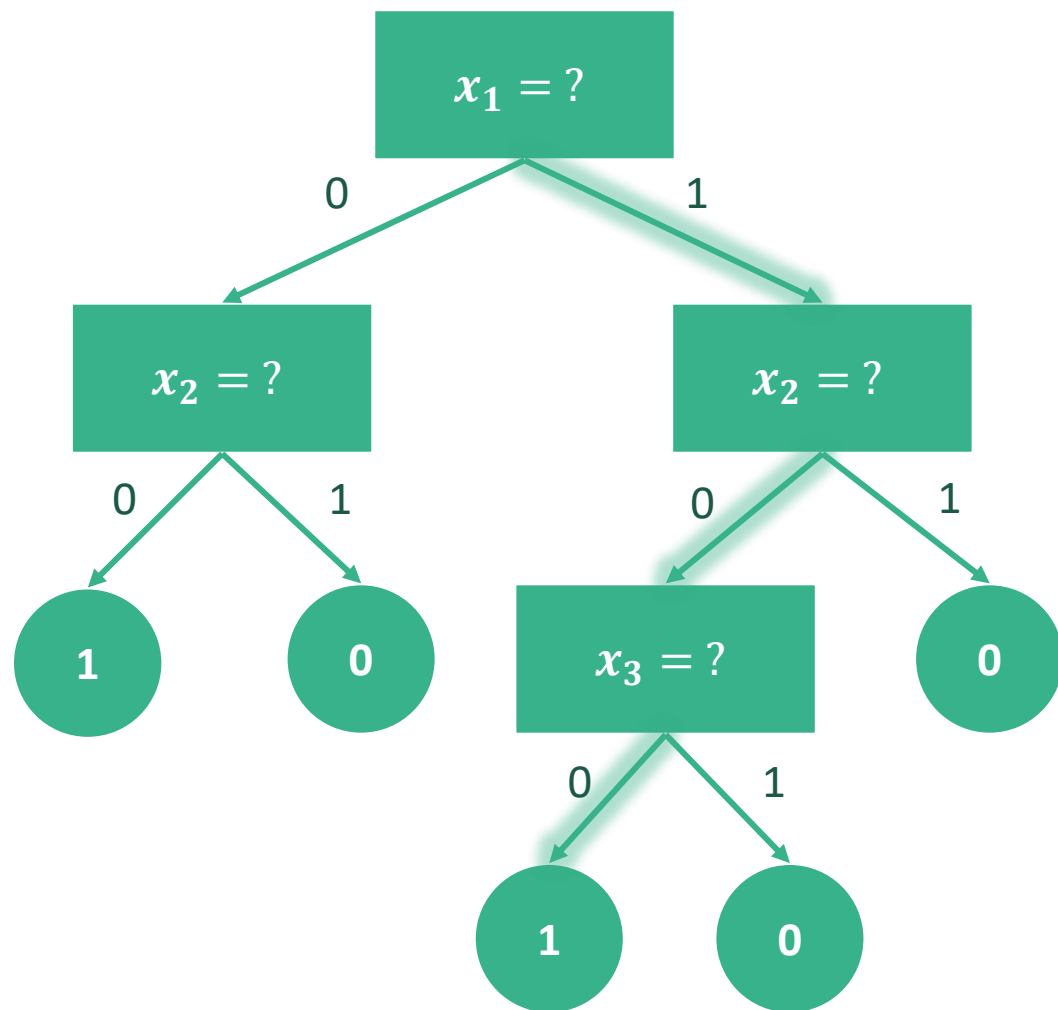


Дерево решений может быть представлено в виде булевой функции в ДНФ.

Например, дерево на рисунке соответствует функции:

$$f(x_1, x_2, x_3) = (\overline{x_1} \wedge \overline{x_2}) \vee (x_1 \wedge \overline{x_2} \wedge \overline{x_3})$$

# Деревья и булевы функции



Дерево решений может быть представлено в виде булевой функции в ДНФ.

Например, дерево на рисунке соответствует функции:

$$f(x_1, x_2, x_3) = (\overline{x_1} \wedge \overline{x_2}) \vee (x_1 \wedge \overline{x_2} \wedge \overline{x_3})$$

# Алгоритм ID3

Вход: LearnID3 ( $U \subseteq X^I$ )

если все объекты из  $U$  лежат в одном классе  $c \in Y$  то

    вернуть новый лист  $v, c_v := c$ ;

найти предикат с максимальной информативностью (*Information Gain*):

$$\beta := \arg \max_{\beta \in \mathcal{B}} IG(\beta, U);$$

разбить выборку на две части  $U = U_0 \sqcup U_1$  по предикату  $\beta$ :

$$U_0 := \{x \in U: \beta(x) = 0\};$$

$$U_1 := \{x \in U: \beta(x) = 1\};$$

если  $U_0 = \emptyset$  или  $U_1 = \emptyset$  то

    вернуть новый лист  $v, c_v := \text{Мажоритарный класс}(U)$ ;

создать новую внутреннюю вершину  $v: \beta_v := \beta$ ;

построить левое поддереву  $L_v := \text{LearnID3}(U_0)$ ;

построить правое поддереву  $R_v := \text{LearnID3}(U_1)$ ;

вернуть  $v$

# Критерии информативности

## Энтропия Шеннона

При  $N$  возможных классах:

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

где  $p_i$  – вероятность принадлежности к классу  $i$ . В случае бинарной классификации:

$$S = -p \log_2 p - (1 - p) \log_2 (1 - p).$$

Уменьшение энтропии называют приростом информации (*Information gain*):

$$IG(\beta) = S_0 - \sum_i \frac{N_i}{N} S_i .$$

# История матчей «Зенита»

Соперник	Играем	Лидеры	Дождь	Победа
Выше	Дома	На месте	Да	Нет
Выше	Дома	На месте	Нет	Да
Выше	Дома	Пропускают	Нет	Да
Ниже	Дома	Пропускают	Нет	Да
Ниже	В гостях	Пропускают	Нет	Нет
Ниже	Дома	Пропускают	Да	Да
Выше	В гостях	На месте	Да	Нет
Ниже	В гостях	На месте	Нет	?

# Энтропия

Из 7 матчей «Зенит» три проиграл и четыре выиграл. Исходная энтропия:

$$S_0 = -\frac{4}{7}\log_2 \frac{4}{7} - \frac{3}{7}\log_2 \frac{3}{7} \approx 0.9852$$

Вычислим прирост информации по признаку «Соперник в турнирной таблице»:

$$\begin{aligned} IG(\text{Соперник}) &= S_0 - \frac{4}{7}S_1 - \frac{3}{7}S_2 \\ &\approx 0.9852 - \frac{4}{7}\left(-\frac{1}{2}\log_2 \frac{1}{2} - \frac{1}{2}\log_2 \frac{1}{2}\right) - \frac{3}{7}\left(-\frac{2}{3}\log_2 \frac{2}{3} - \frac{1}{3}\log_2 \frac{1}{3}\right) \approx \mathbf{0.0202} \end{aligned}$$

# Энтропия

Для определения оптимального предиката вычислим значения энтропийного критерия для всех возможных предикатов:

$$IG(\text{Соперник}) \approx 0.0202$$

$$IG(\text{Играем}) \approx 0.4696$$

$$IG(\text{Лидеры}) \approx 0.1281$$

$$IG(\text{Дождь}) \approx 0.1281$$

Согласно энтропийному критерию в корень дерева необходимо поместить предикат «домашний матч или гостевой».

# Критерии информативности

## Критерий Джини

Максимизирует число пар объектов одного класса, оказавшихся в одном поддереве:

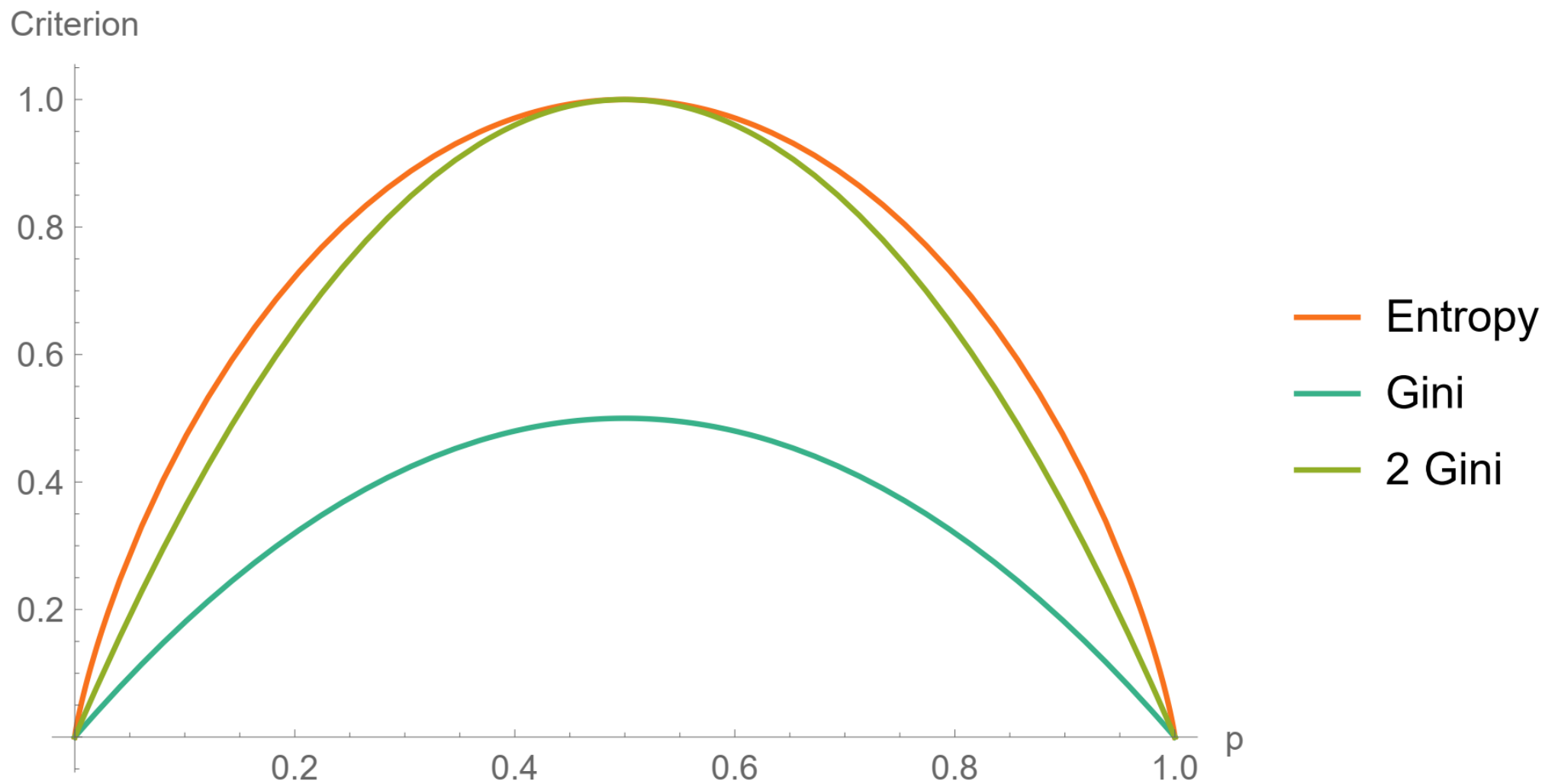
$$G = 1 - \sum_{i=1}^N p_i^2,$$

В случае бинарной классификации:

$$G = 1 - p^2 - (1 - p)^2 = 2p(1 - p).$$



# Сравнение критериев информативности



# Критерии в задачах регрессии

## Дисперсионный критерий

$$D(U) = \frac{1}{|U|} \sum_{x_j \in X} \left( y_j - \frac{1}{|U|} \sum_{x_i \in X} y_i \right)^2,$$

где  $|U|$  – число объектов в листе,  $y_i$  – значение целевой переменной.

$$IG(\beta, U) = D(U) - \frac{|U_0|}{|U|} D(U_0) - \frac{|U_1|}{|U|} D(U_1).$$

Необходимо выбрать такой критерий, при котором дисперсия как в левом, так и в правом поддеревьях значительно уменьшится.

# Алгоритм ID3

Вход:  $\text{LearnID3 } (U \subseteq X^l)$

если все объекты из  $U$  лежат в одном классе  $c \in Y$  то

    вернуть новый лист  $v, c_v := c$ ;

найти предикат с максимальной информативностью (*Information Gain*):

$$\beta := \arg \max_{\beta \in \mathcal{B}} IG(\beta, U);$$

разбить выборку на две части  $U = U_0 \sqcup U_1$  по предикату  $\beta$ :

$$U_0 := \{x \in U: \beta(x) = 0\};$$

$$U_1 := \{x \in U: \beta(x) = 1\};$$

если  $U_0 = \emptyset$  или  $U_1 = \emptyset$  то

    вернуть новый лист  $v, c_v := \text{Мажоритарный класс } (U)$ ;

создать новую внутреннюю вершину  $v: \beta_v := \beta$ ;

построить левое поддерево  $L_v := \text{LearnID3 } (U_0)$ ;

построить правое поддерево  $R_v := \text{LearnID3 } (U_1)$ ;

вернуть  $v$

## Алгоритм С4.5. Усечение дерева решений

Вход:  $X^k$  – независимая контрольная выборка,  $k \approx 0.5 l$ .

для всех  $v \in V_{\text{внутр}}$

$S_v :=$  подмножество объектов  $X^k$ , дошедших до  $v$ ;

если  $S_v = \emptyset$  то

вернуть новый лист  $v$ ,  $c_v := \text{Мажоритарный класс}(U)$ ;


в зависимости от числа ошибок классификации в  $S_v$  заменить одним из способов:

$r(v)$  – поддеревом, растущим из вершины  $v$ ;

$r_L(v)$  – поддеревом левой дочерней вершины  $L_v$ ;

$r_R(v)$  – поддеревом правой дочерней вершины  $R_v$ ;

$r_C(v)$  – терминальной вершиной с классом  $c \in Y$ .



# **Композиции алгоритмов**

# Преимущества и недостатки решающих деревьев

## Преимущества:

- порождение четких правил классификации, понятных человеку;
- быстрые процессы обучения и прогнозирования;
- малое число параметров модели;
- поддержка и числовых, и категориальных признаков.

## Недостатки:

- чувствительны к шумам;
- разделяющая граница, построенная деревом решений, имеет свои ограничения;
- проблема поиска оптимального дерева решений.

# Формула Бернулли

Вероятность того, что в  $n$  независимых испытаниях, в каждом из которых вероятность появления события равна  $p$ , событие наступит ровно  $k$  раз (безразлично, в какой последовательности), равна

$$P_n(k) = C_n^k p^k q^{n-k}.$$

## Упражнение:

Устройство состоит из пяти независимо работающих элементов. Вероятность отказа каждого из них равна 0.2. Найти вероятность того, что откажут:

- а) три элемента;
- б) не менее четырех элементов;
- в) хотя бы один элемент.

## Теорема Кондорсе о жюри присяжных

Если каждый член жюри присяжных имеет независимое мнение, и если вероятность правильного решения члена жюри **больше 0.5**, то тогда вероятность правильного решения присяжных в целом **возрастает с увеличением количества членов жюри**, и стремится к единице. Если же вероятность быть правым у каждого из членов жюри меньше 0.5, то вероятность принятия правильного решения присяжными в целом монотонно уменьшается и стремится к нулю с увеличением количества присяжных.



# Теорема Кондорсе о жюри присяжных

$N$  – число членов жюри;

$p$  – вероятность правильного решения одного члена жюри;

$\mu$  – вероятность правильного решения всех членов жюри:

$$\mu = \sum_{i=m}^N C_N^i p^i (1-p)^{N-i}.$$

## Упражнение:

В зале суда есть 7 присяжных, каждый из них по отдельности с вероятностью 0.8 может определить, виновен подсудимый или нет. С какой вероятностью присяжные все вместе вынесут правильный вердикт, если решение принимается большинством голосов?

# Простое голосование классификаторов

Обучающая выборка:  $X^l = (x_i, y_i)_{i=1}^l$ ,  $x_i \in X, y_i \in \{-1, +1\}$

Базовые классификаторы:  $b_1(x), \dots, b_T(x)$ ,  $b_t: X \rightarrow \{-1, +1\}$

Простое голосование базовых классификаторов:

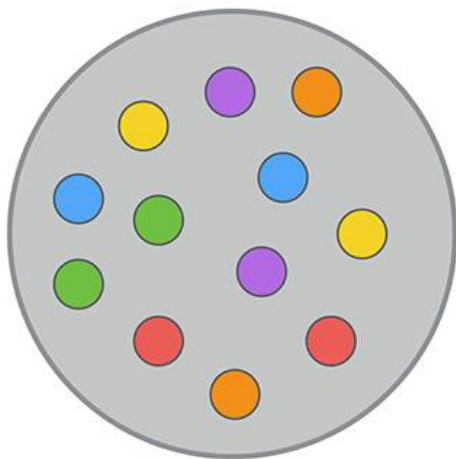
$$a(x) = \text{sign} \sum_{t=1}^T b_t(x).$$

Способы повышения различности базовых алгоритмов:

- обучение по случайным подвыборкам;
- обучение по выборке со случайными весами объектов;
- обучение по случайным подмножествам признаков;
- использование различных моделей классификации.

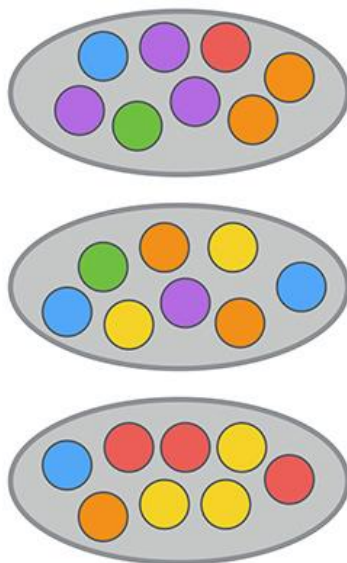
# Бутстрэп

Исходная выборка



Статистика по  
выборке

Бутстрэп выборки



Статистики по  
бутстрэп выборкам

Статистика 1

Статистика 2

Статистика 3

Бутстрэп  
распределение

## Бэггинг и метод случайных подпространств

**Бэггинг** (bagging, bootstrap aggregation) [Breiman, 1996]:

Каждый базовый алгоритм  $b_t(x)$  обучается независимо на случайных подвыборках с повторениями (как в методе bootstrap). При этом не все объекты попадают в обучающую выборку базового алгоритма.

*Out-of-Bag* оценка – это усредненная оценка базовых алгоритмов на тех данных, на которых они не обучались.

**Метод случайных подпространств** (RSM, random subsample method) [Ho, 1998]:

Базовые алгоритмы  $b_t(x)$  обучаются на случайных подмножествах признаков.

# Бэггинг и метод случайных подпространств

**Вход:** обучающая выборка  $X^l$ ; параметры:  $T$

$l'$  – длина обучающих подвыборок;

$n'$  – длина признакового описания;

$\varepsilon_1$  – порог качества базовых алгоритмов на обучении;

$\varepsilon_2$  – порог качества базовых алгоритмов на контроле;

**Выход:** базовые алгоритмы  $b_t, t = 1, \dots, T$ .

1: для всех  $t = 1, \dots, T$

2:  $U_t :=$  случайное подмножество  $X^l$  длины  $l'$ ;

3:  $F_t :=$  случайное подмножество  $F$  длины  $n'$ ;

4:  $b_t := \mu(F_t, U_t)$ ;

5: если  $Q(b_t, U_t) > \varepsilon_1$  или  $Q(b_t, X^l \setminus U_t) > \varepsilon_2$  то

6: не включать  $b_t$  в композицию.

**Композиция** – простое голосование:  $a(x) = \text{sign } \sum_{t=1}^T b_t(x)$

# Случайный лес

Случайный лес – это композиция алгоритмов, в которой:

- используется бэггинг над решающими деревьями;
- усечение дерева (pruning) не производится;
- признак в каждой вершине дерева выбирается из случайного подмножества  $k$  из  $n$  признаков.

То есть, случайный лес – это комбинация метода бэггинга и метода случайных подпространств над решающими деревьями.