Искусственный Интеллект

Лекция 6: Деревья решений. Ансамбли моделей

Мартынюк Полина Антоновна

telegram: @PAMartynyuk

email: pa-martynyuk@yandex.ru

Алгоритмы машинного обучения

- Умеем обучать алгоритм ближайших соседей
- Умеем обучать линейные алгоритмы:
  - Умеем обучать модель для решения задачи классификации
  - Умеем обучать модель для решения задачи регрессии

Проблемы внедрения в бизнес:

1. Сложность в интерпретации работы модели, ее непредсказуемость
   - *А как работает модель?*
   - *Модель предсказуема?*
   - *Где гарантия того, что модель не наделает ошибок?*
2. В жизни мы принимаем решения не так, как это делают известные нам модели

Алгоритмы машинного обучения

Примеры решающих правил

- Если в анкете указан домашний телефон и зарплата клиента > $1000 и размер кредита < $3000, то кредит выдать
- Если возраст пациента > 60 и пациент ранее перенёс инфаркт, то операцию не делать

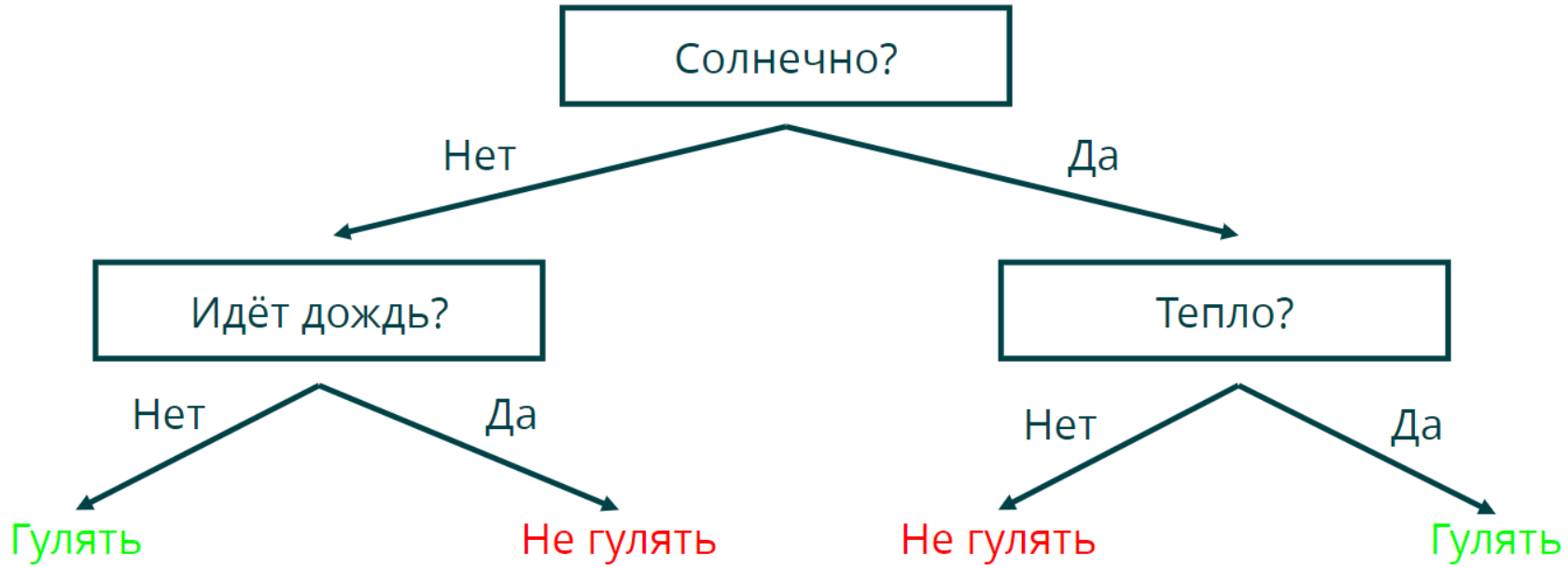
С подобными решающими правилами работают **логические алгоритмы**.

Логический алгоритм — алгоритм, использующий логические закономерности в данных.

Один из видов логических алгоритмов – **дерево решений**

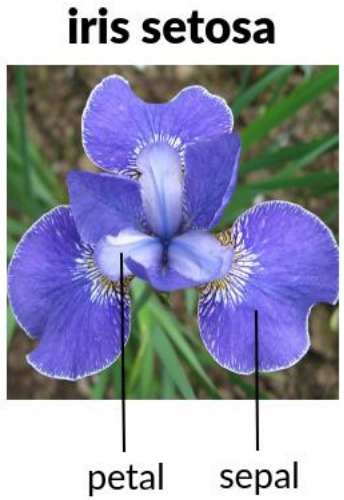
Дерево решений

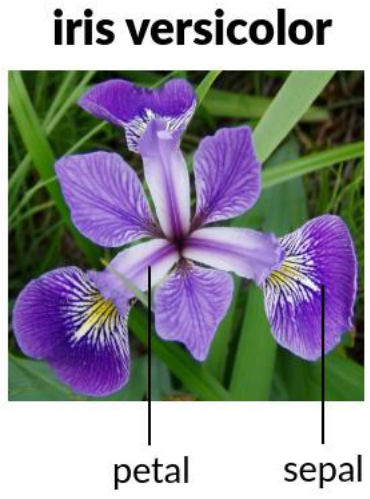
- В каждой вершине дерева находится вопрос
- В зависимости от ответа на вопрос, алгоритм направляется в нужную ветвь дерева
- Листы дерева соответствуют решению алгоритма

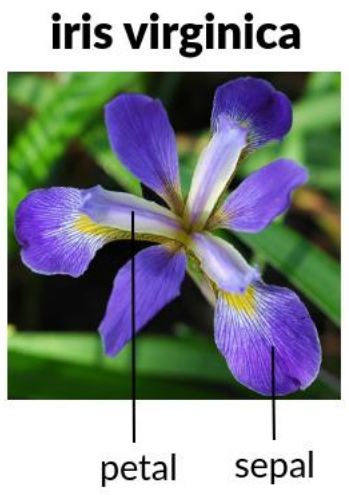


Дерево решений

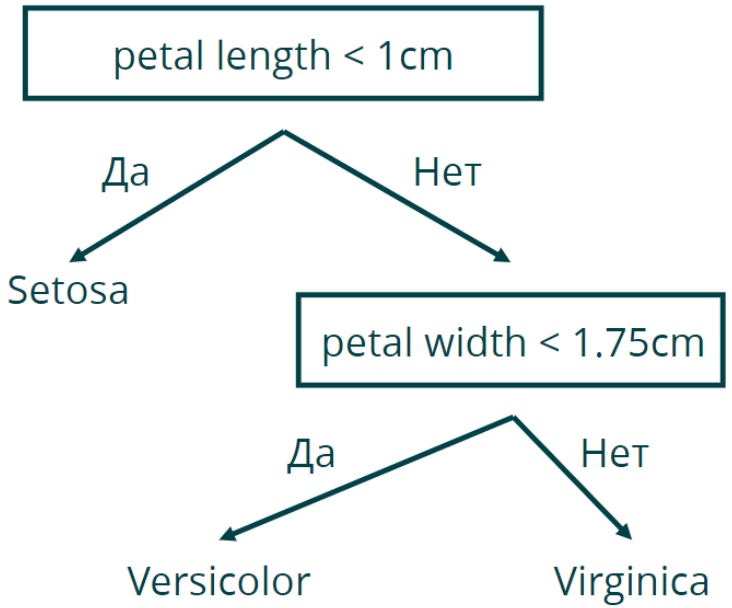
- Ирис







petal length	petal width	sepal length	sepal width	target
				virginica
				versicolor
				setosa
				virginica
				virginica
				setosa
				versicolor

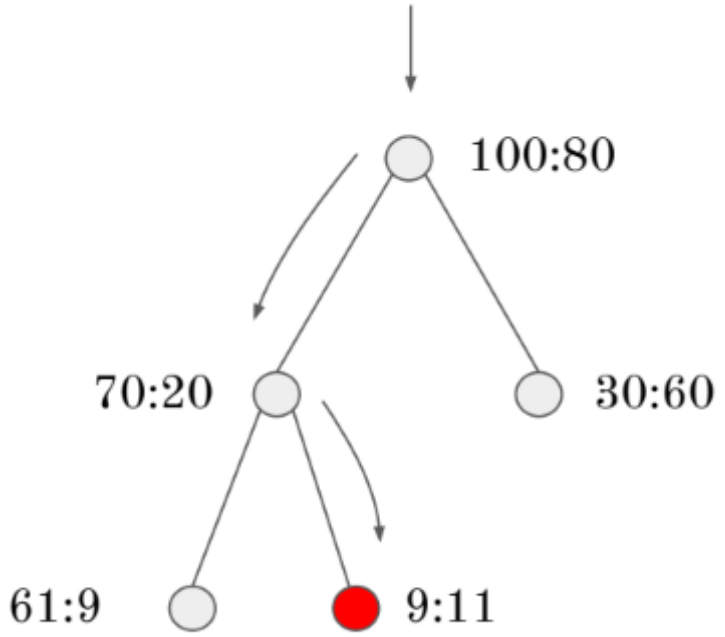


Дерево решений

Как обучать?

Рассмотрим на примере:

- Пусть задача – бинарная классификация
- Находимся в красной вершине
- До красной вершины дошла часть объектов обучающей выборки

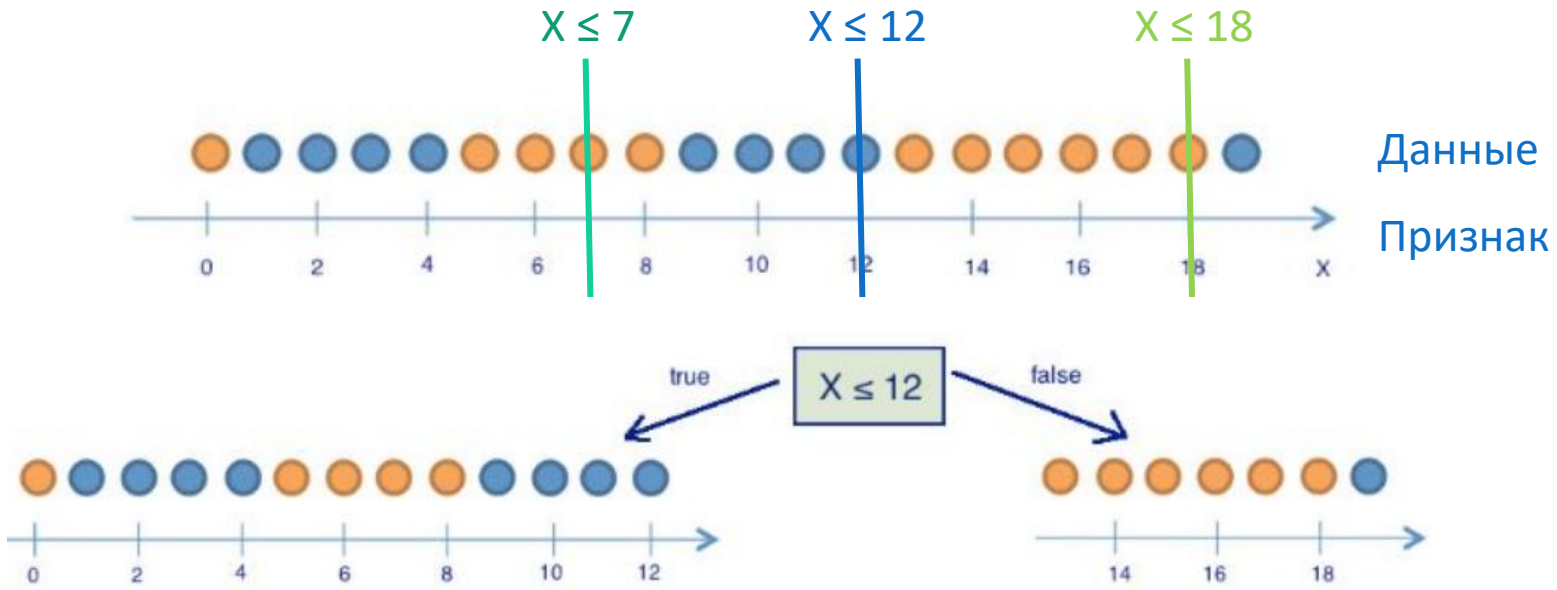


1. 100 объектов класса 1, 80 объектов класса 0
2. 70 объектов класса 1, 20 объектов класса 0
3. 9 объектов класса 1, 11 объектов класса 0

Задача: найти такое решающее правило, которое обеспечит наилучшее разделение объектов разных классов друг от друга

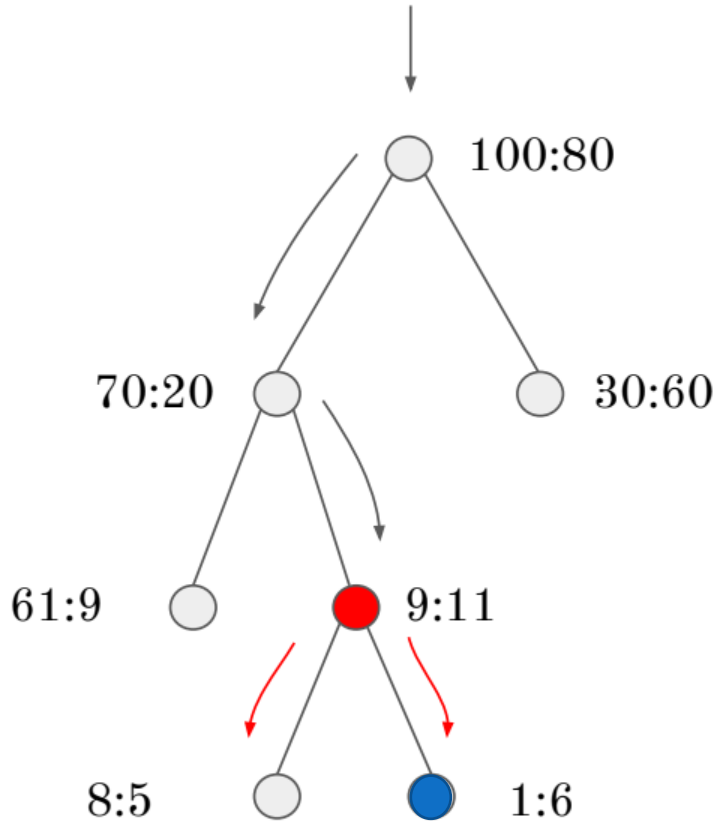
Дерево решений

Решающее правило строится по характеристикам – фичам:



8 объектов класса 1
5 объектов класса 0

1 объект класса 1
6 объектов класса 0

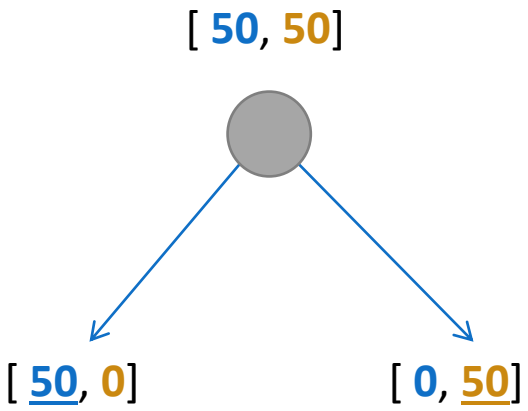


- Одна из вершин стала терминальной
- Повторяем разбиения с другими нетерминальными вершинами

Дерево решений

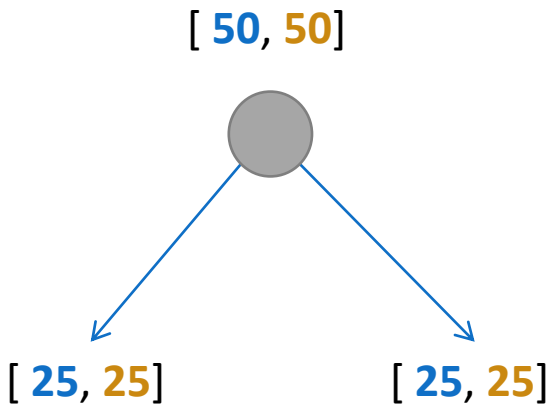
Формализуем выбор критерия ветвления:

Идеальный вариант



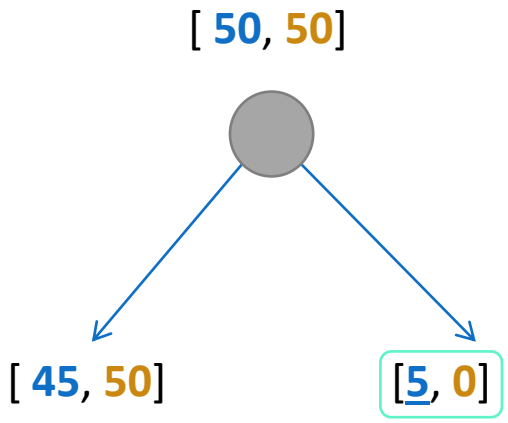
L = 0

Наихудший вариант



L = max

Специфический вариант



Очень специфичное условие – разделение хорошее, но мала доля отделенных элементов класса

→ надо следить, **какую долю наблюдений отсекает условие**

Дерево решений

Формализуем выбор критерия ветвления:

$$L(X_m, i, c) = \frac{X_l}{X_m} I_{left} + \frac{X_r}{X_m} I_{right}$$

Ошибка слева
(критерий информативности)

Ошибка справа
(критерий информативности)

X_m - количество экземпляров данных, дошедших до вершины m

i - i-тый признак (рассматриваем потенциальные разбиения по всем признакам)

c - константа, входящая в условие разбиения (граничное значение признака)

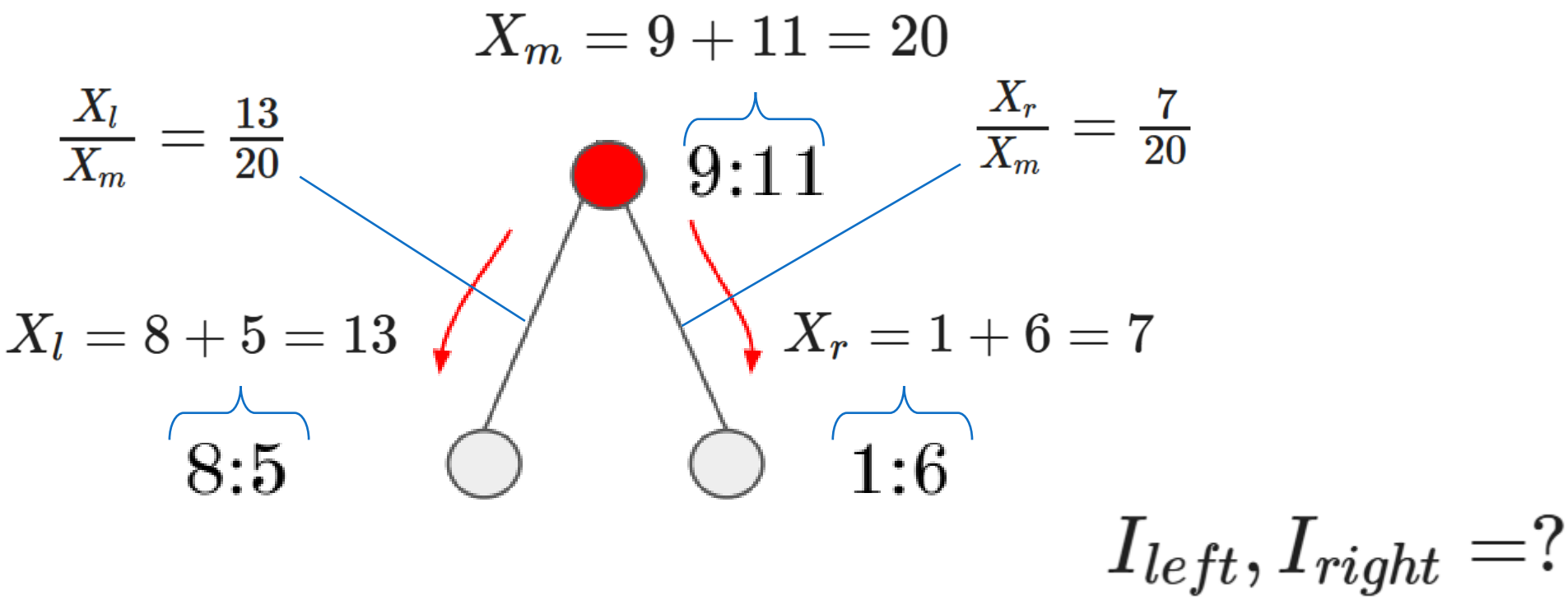
$\frac{X_l}{X_m}$ - доля экземпляров данных, отброшенных в левую ветку

$\frac{X_r}{X_m}$ - доля экземпляров данных, отброшенных в правую ветку

Дерево решений

Формализуем выбор критерия ветвления:

$$L(X_m, i, c) = \frac{X_l}{X_m} I_{left} + \frac{X_r}{X_m} I_{right} = \frac{13}{20} I_{left} + \frac{7}{20} I_{right}$$



$$I_{left}, I_{right} = ?$$

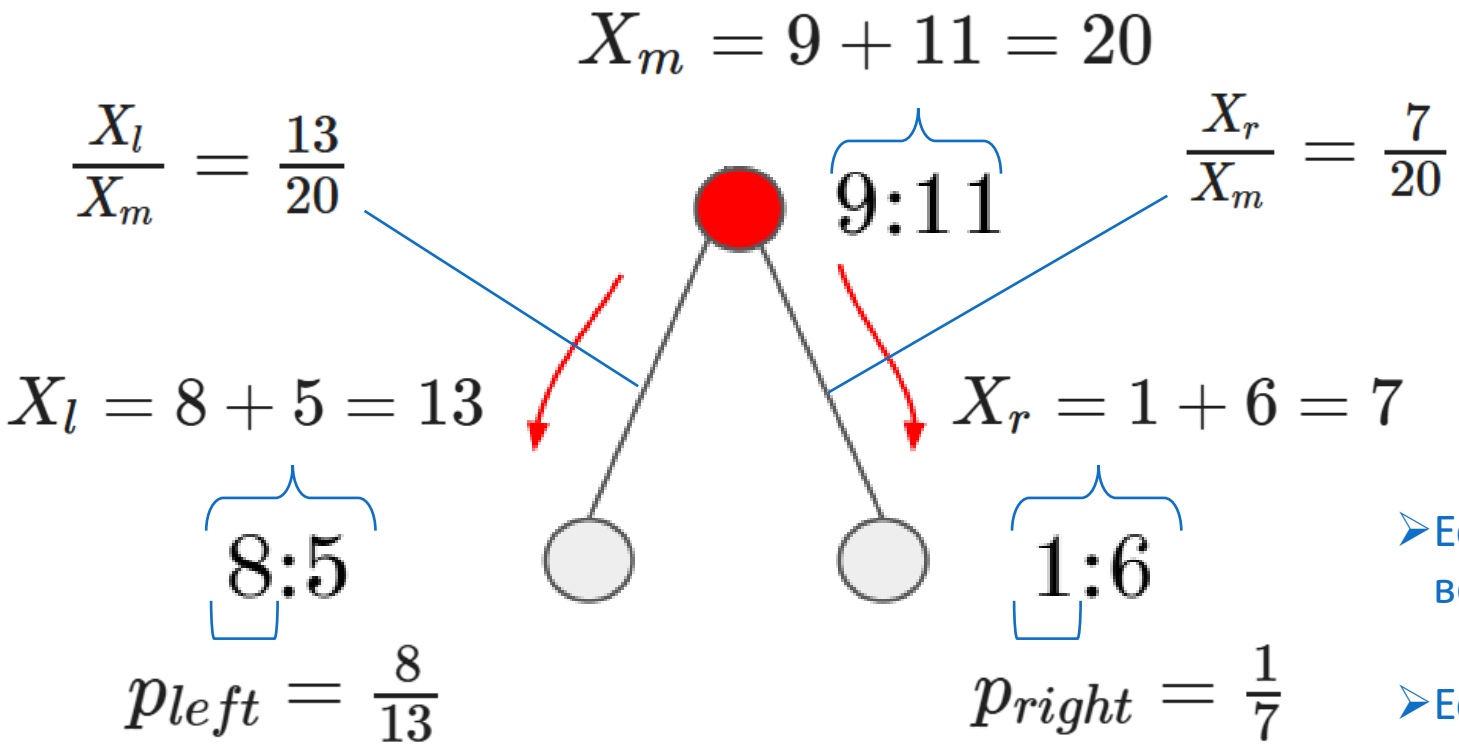
Дерево решений

Задача – бинарная кластеризация

Цель – выявить объекты класса 1

p_{left} - вероятность появления экземпляра класса 1 в левой ветке

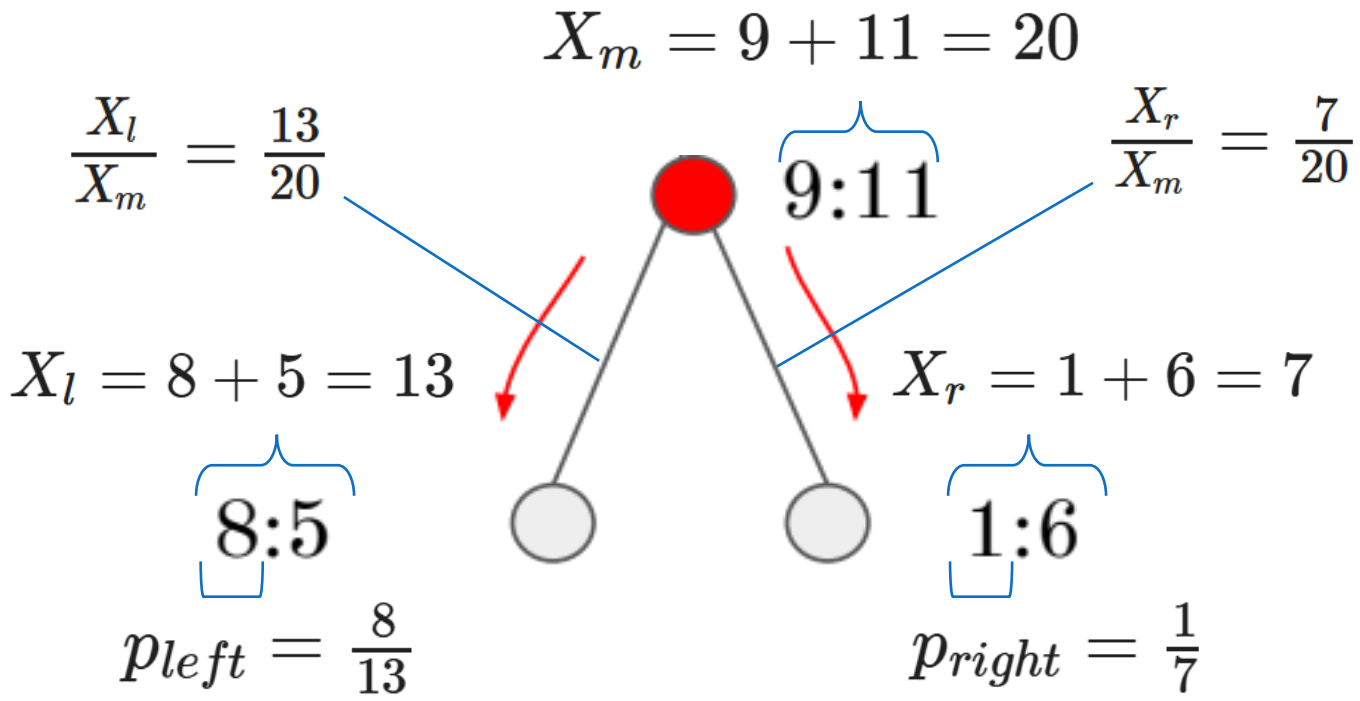
p_{right} - вероятность появления экземпляра класса 1 в правой ветке



- Если *p* близко к 0 или 1 – вершина «чистая», это хорошо
- Если *p* близко к 0.5 – это плохо

Дерево решений

$$I_{left} = H(p_{left})$$
$$I_{right} = H(p_{right})$$



$$L(X_m, i, c) = \frac{X_l}{X_m} H(p_{left}) + \frac{X_r}{X_m} H(p_{right}) \rightarrow min$$

$H(p)$: Значение меньше, если *p* ближе к 0 или 1; больше, если *p* ближе к 0.5

Дерево решений

$$\frac{X_l}{X_m} H(p_{left}) + \frac{X_r}{X_m} H(p_{right}) \to min$$

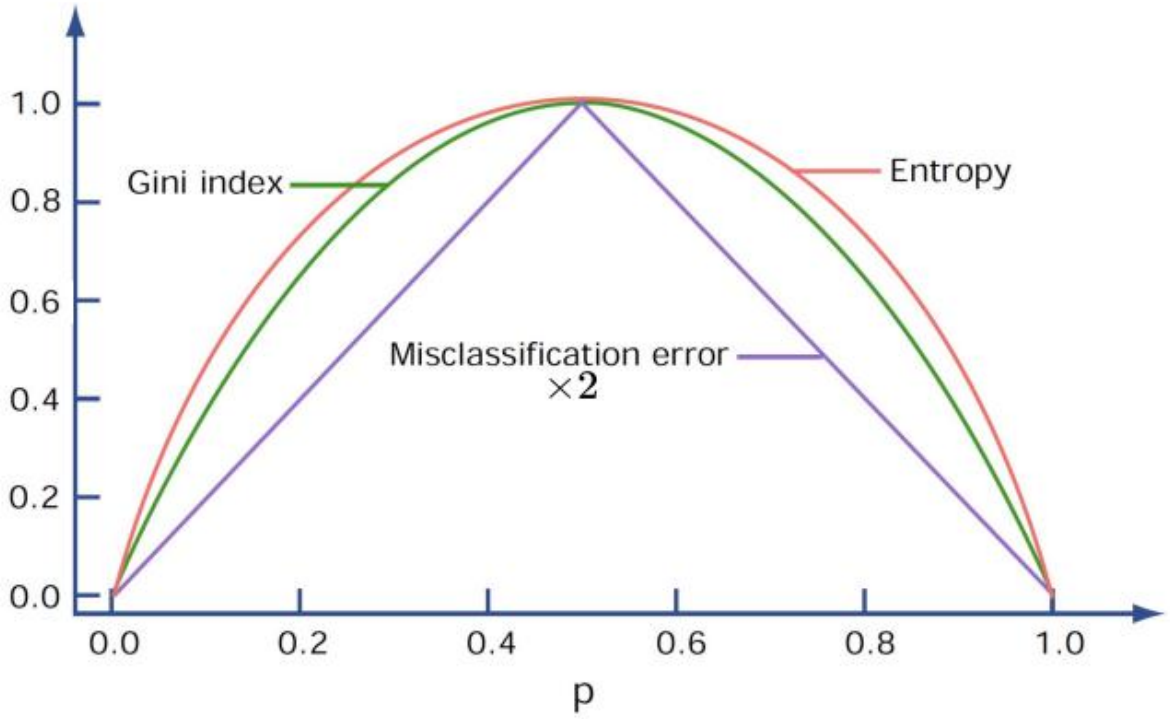
Возможные функции $H(p)$:

- Энтропия

$$H(q) = -q \log q - (1-q) \log(1-q)$$

- Индекс Джини

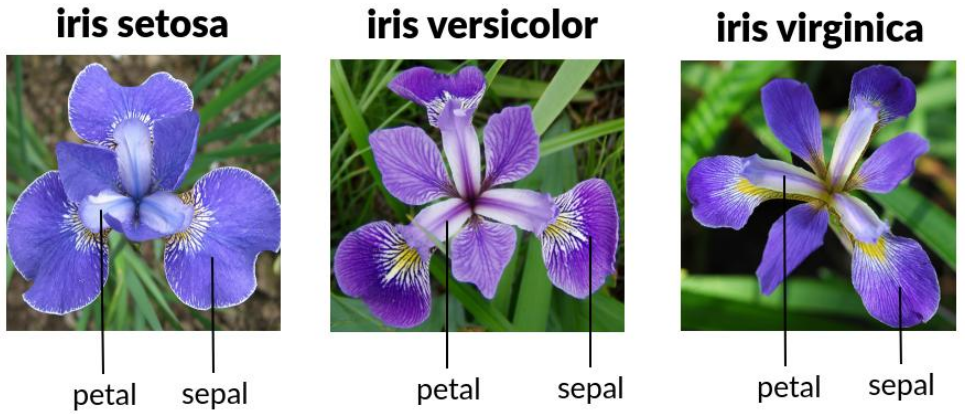
$$H(q) = 4q(1-q)$$

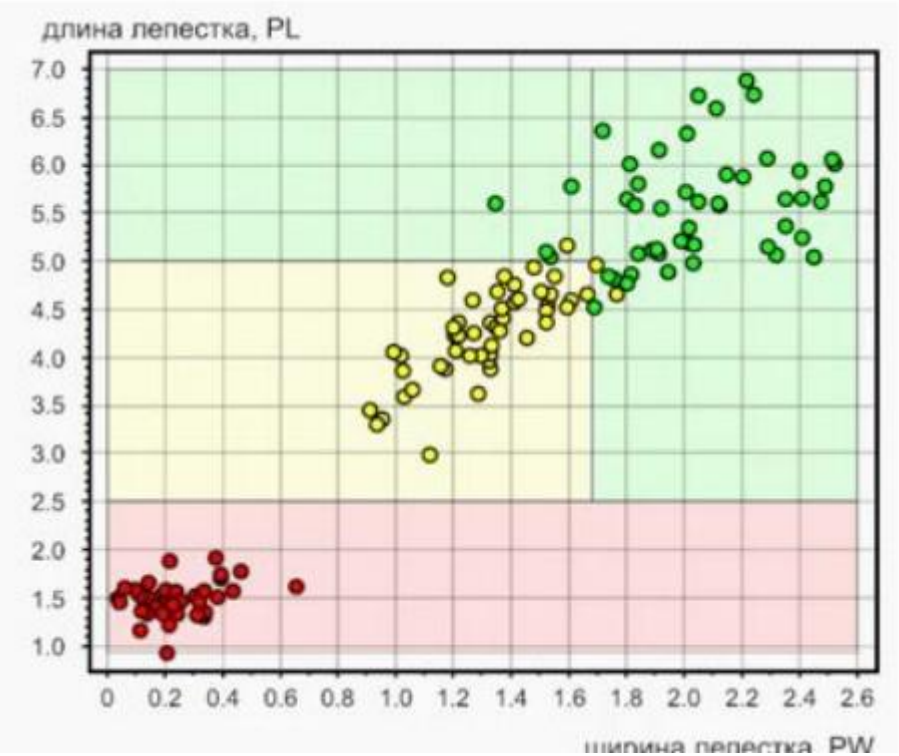


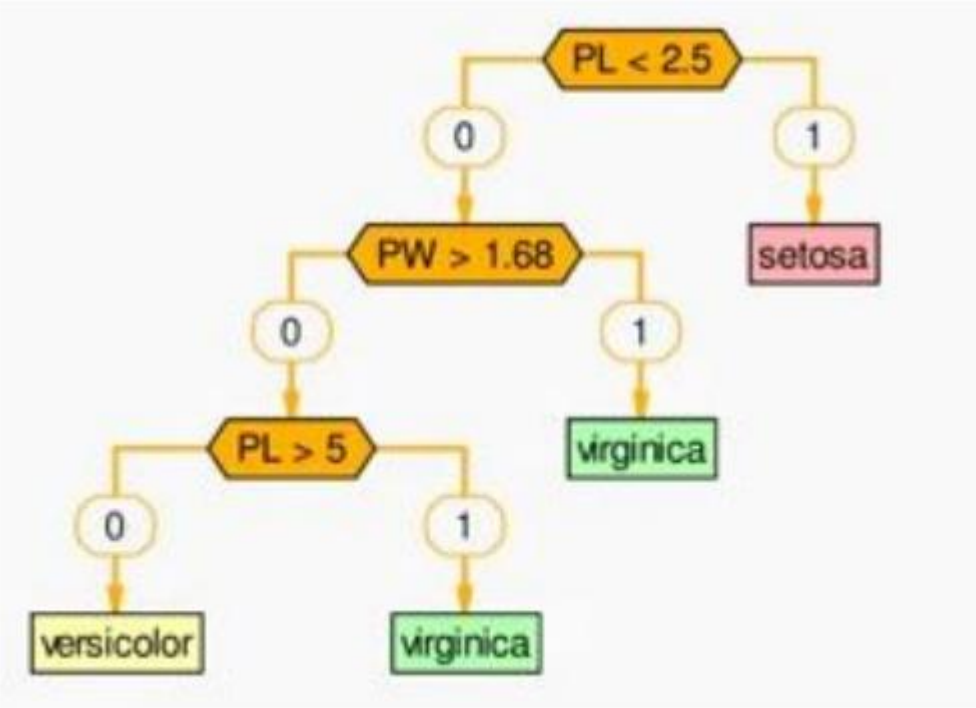
Затем решаем оптимизационную задачу для поиска правила

Дерево решений

- В осях двух самых информативных признаков два класса разделились без ошибок, на третьем — три ошибки.

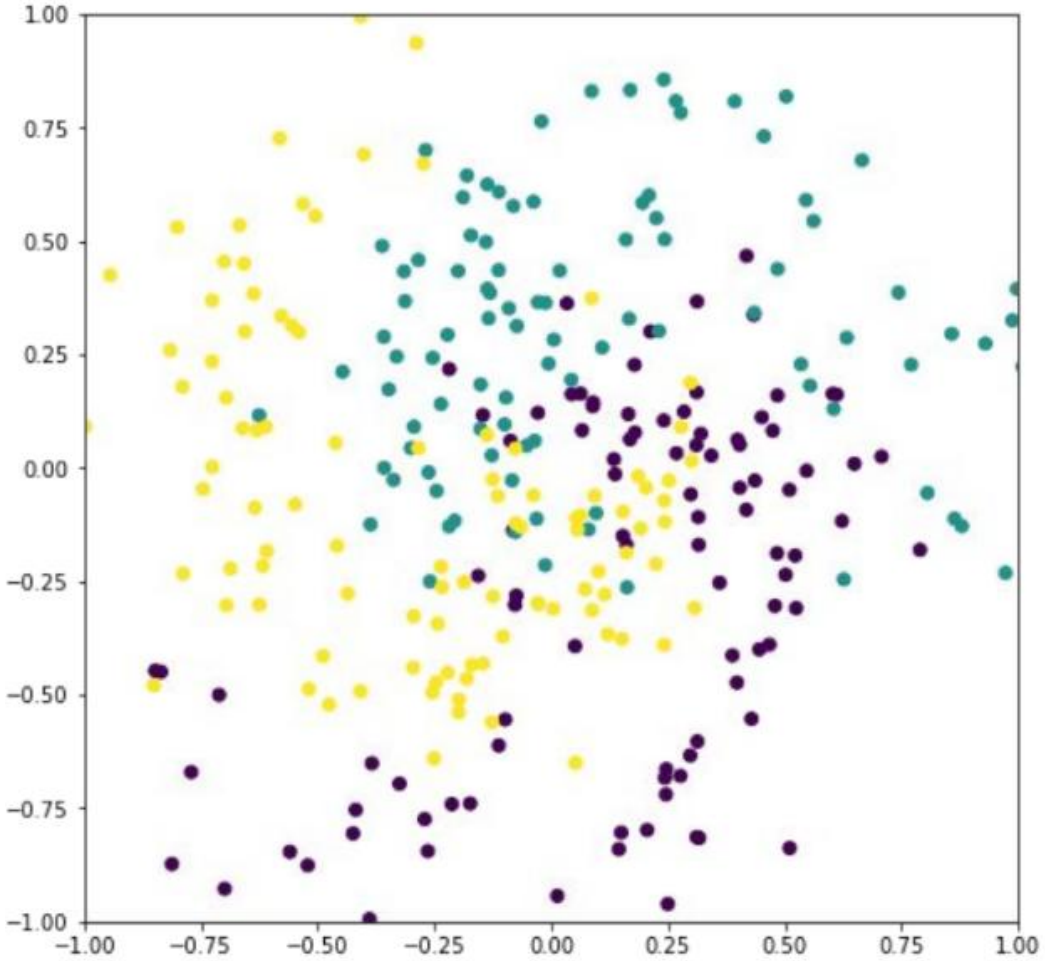






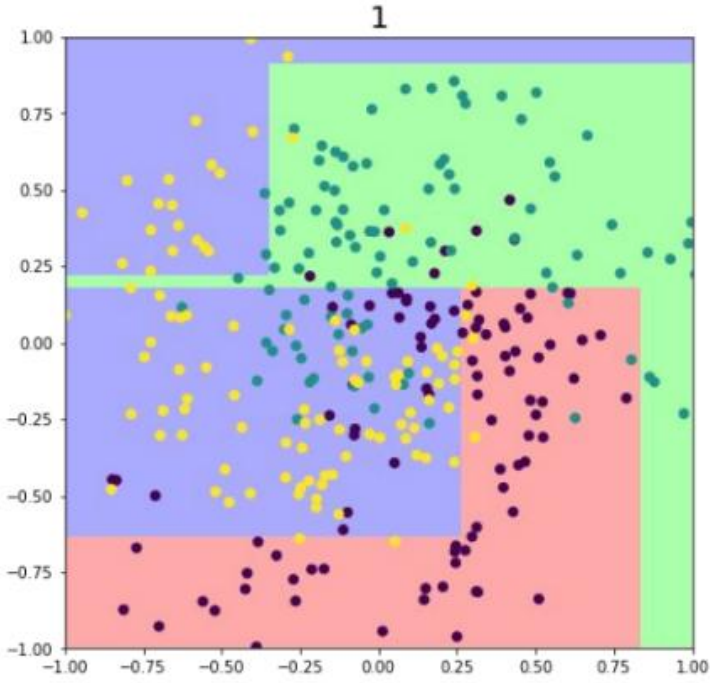
Дерево решений

- Недообучение и переобучение

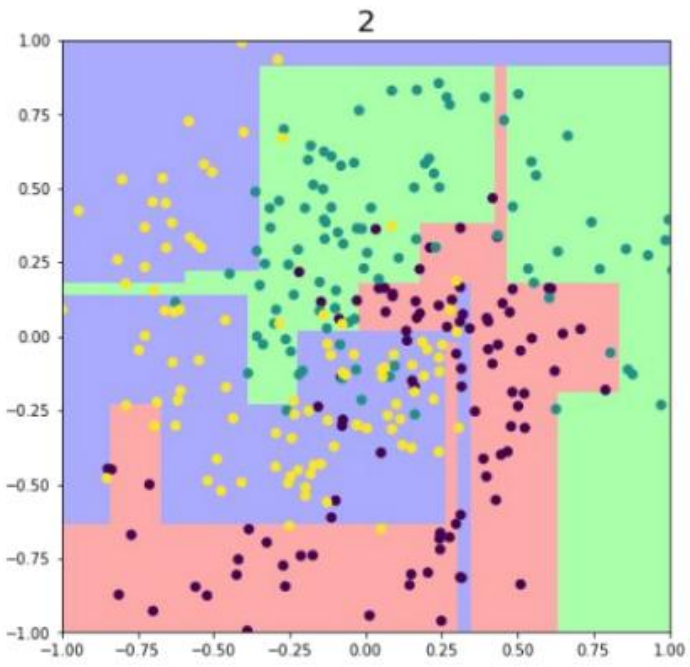


Дерево решений

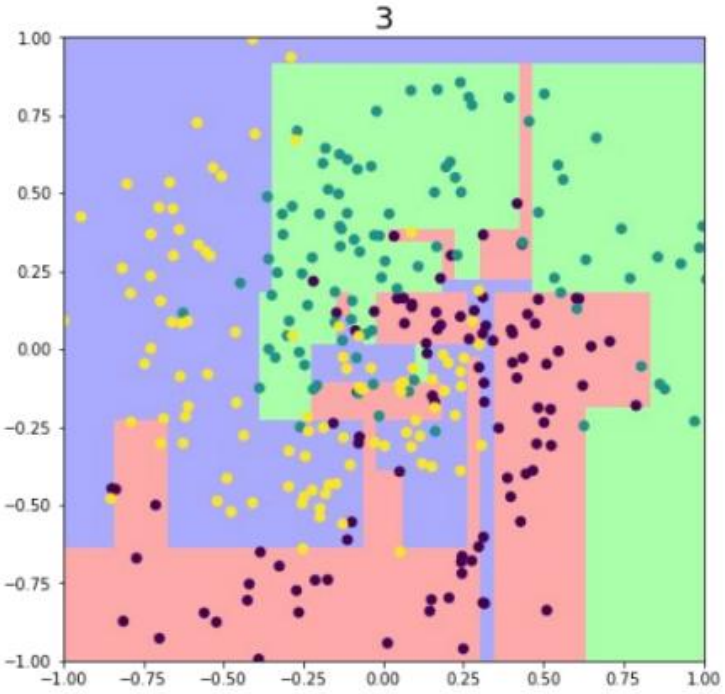
- Недообучение и переобучение



depth = 3



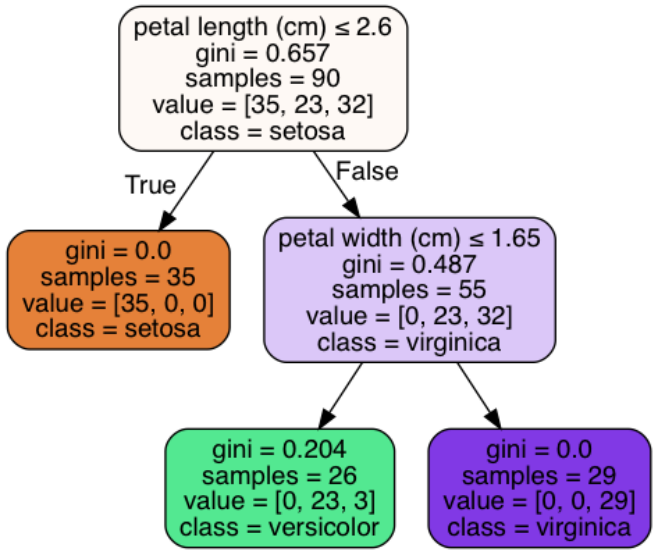
depth = 7



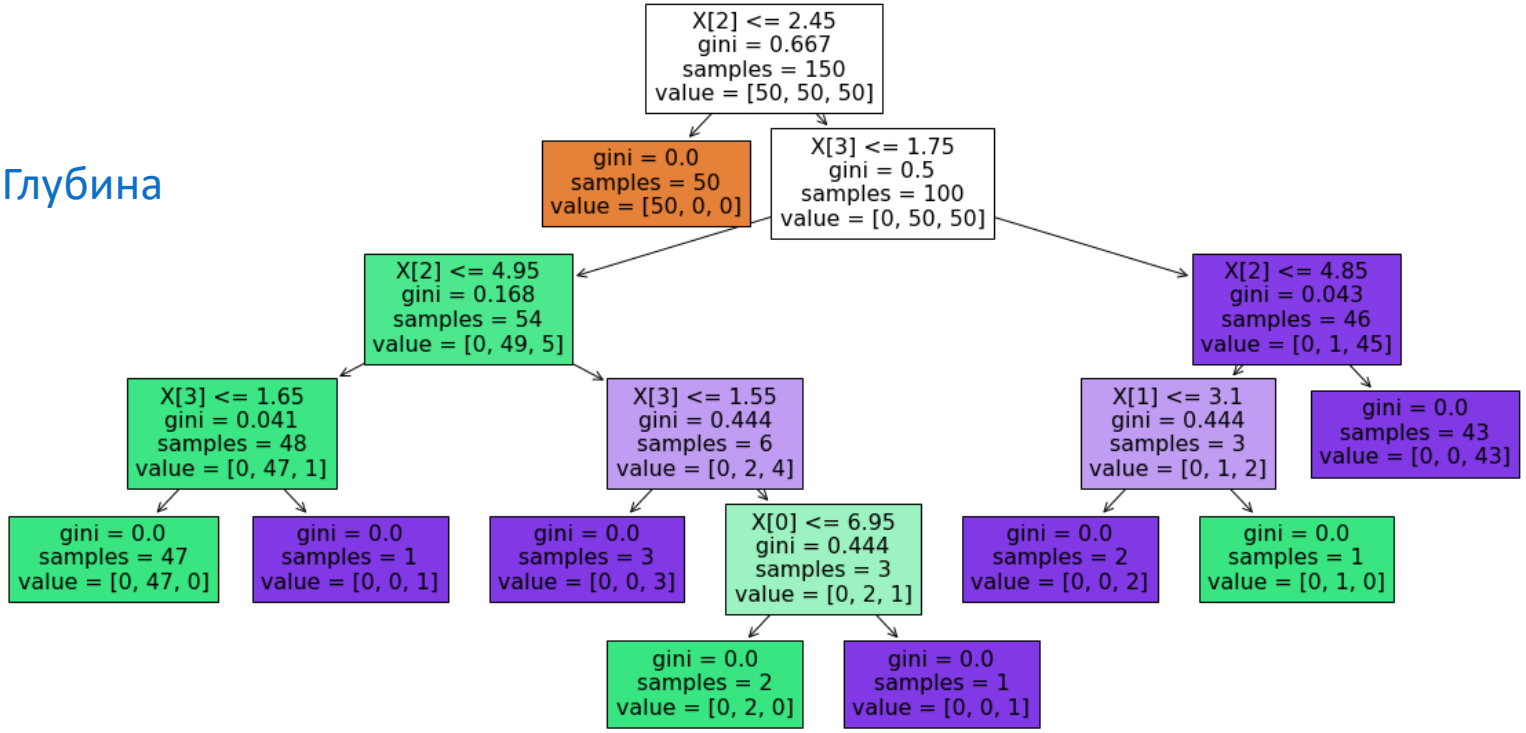
depth = 12

Дерево решений

- Параметры дерева



Глубина



Глубина = 2

Глубина = 5

Дерево решений

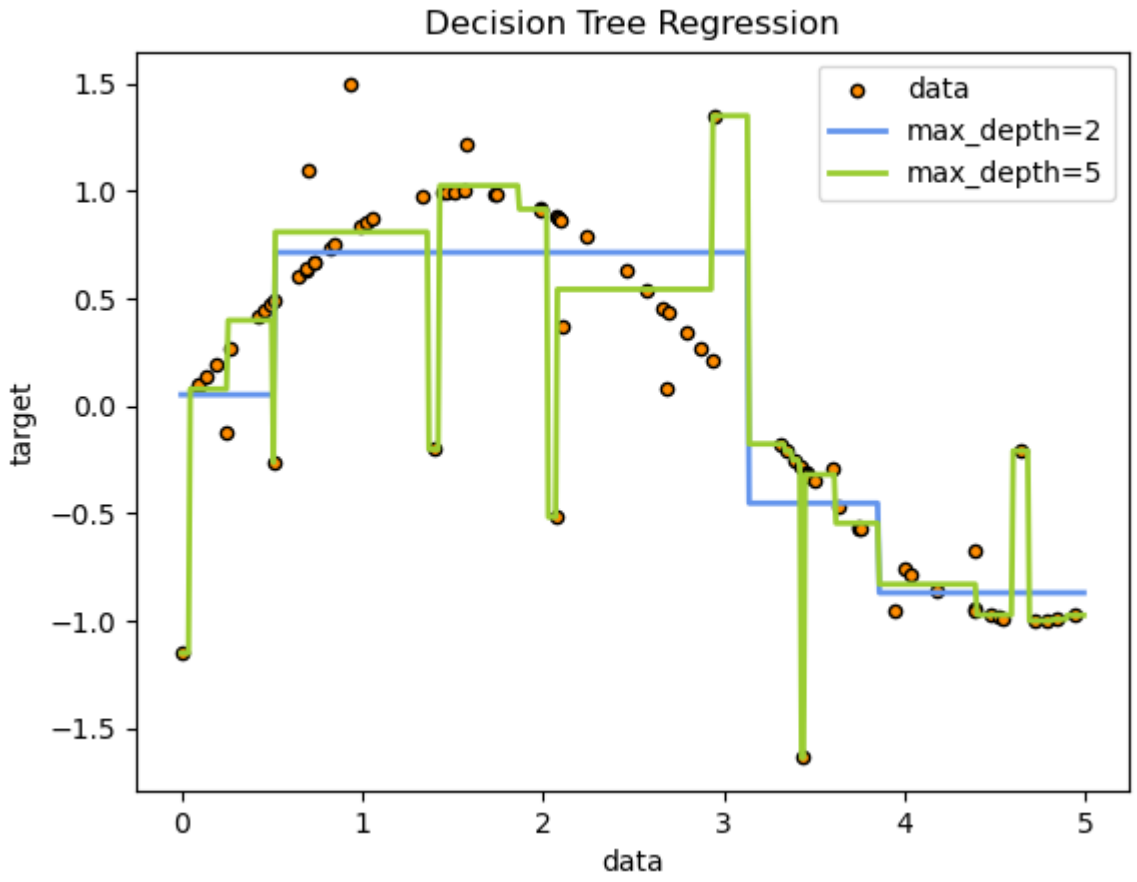
- **Параметры дерева**

  - Критерий ветвления (‘gini’, ‘entropy’)
  - Максимальная глубина
  - Минимальный размер листа
  - Стратегия сплита

Дерево решений

- **Решающее дерево для регрессии**

  - В листах дерева вместо классов объектов находятся действительные числа
  - Количество ответов решающего дерева ограничено количеством листовых вершин



Дерево решений : резюме

Преимущества:

- простая идея и алгоритм обучения
- интерпретируемое
- возможна регуляризация
- обработка пропущенных значений

Недостатки:

- переобучается
- проблемы с регрессией
- сильно меняется в зависимости от параметров

Как улучшить?

Композиции алгоритмов

- Умеем обучать алгоритм **ближайших соседей**
- Умеем обучать **линейные алгоритмы**:
  - Умеем обучать модель для решения задачи классификации
  - Умеем обучать модель для решения задачи регрессии
- Умеем обучать **решающее дерево**

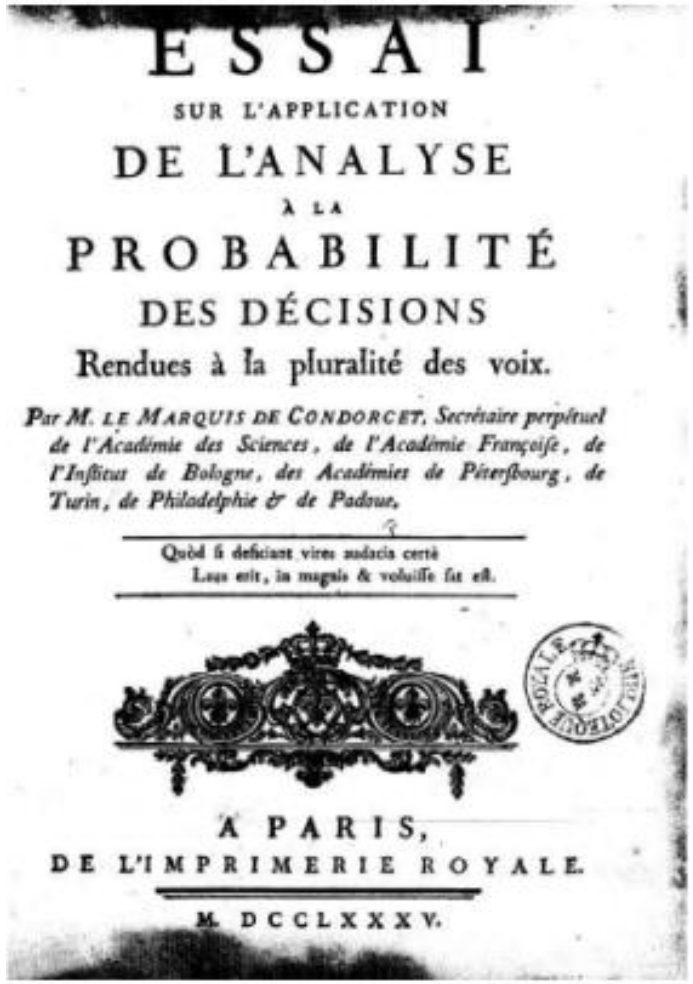
➢ **У всех – свои недостатки**

➢ **Идея – построения композиции алгоритмов**

Композиции алгоритмов

Принцип Кондорсе

- Если вероятность правильного решения члена жюри больше 0.5, то вероятность правильного решения присяжных в целом возрастает с увеличением количества членов жюри и стремится к единице.
- Если же вероятность быть правым меньше 0.5, то вероятность принятия правильного решения присяжными монотонно уменьшается и стремится к нулю с увеличением количества присяжных.



ESSAI

SUR L'APPLICATION

DE L'ANALYSE

À LA

PROBABILITÉ

DES DÉCISIONS

Rendues à la pluralité des voix.

Par M. LE MARQUIS DE CONDORCET, Secrétaire perpétuel de l'Académie des Sciences, de l'Académie Françoise, de l'Institut de Bologne, des Académies de Péterſbourg, de Turin, de Philadelphie & de Padoue,

Quòd ſi deficiunt vires audacia certè
Laus erit, in magnis & voluiſſe ſat eſt.

A PARIS,

DE L'IMPRIMERIE ROYALE.

M. DCCLXXXV.

Принцип Кондорсе, 1784

Композиции алгоритмов

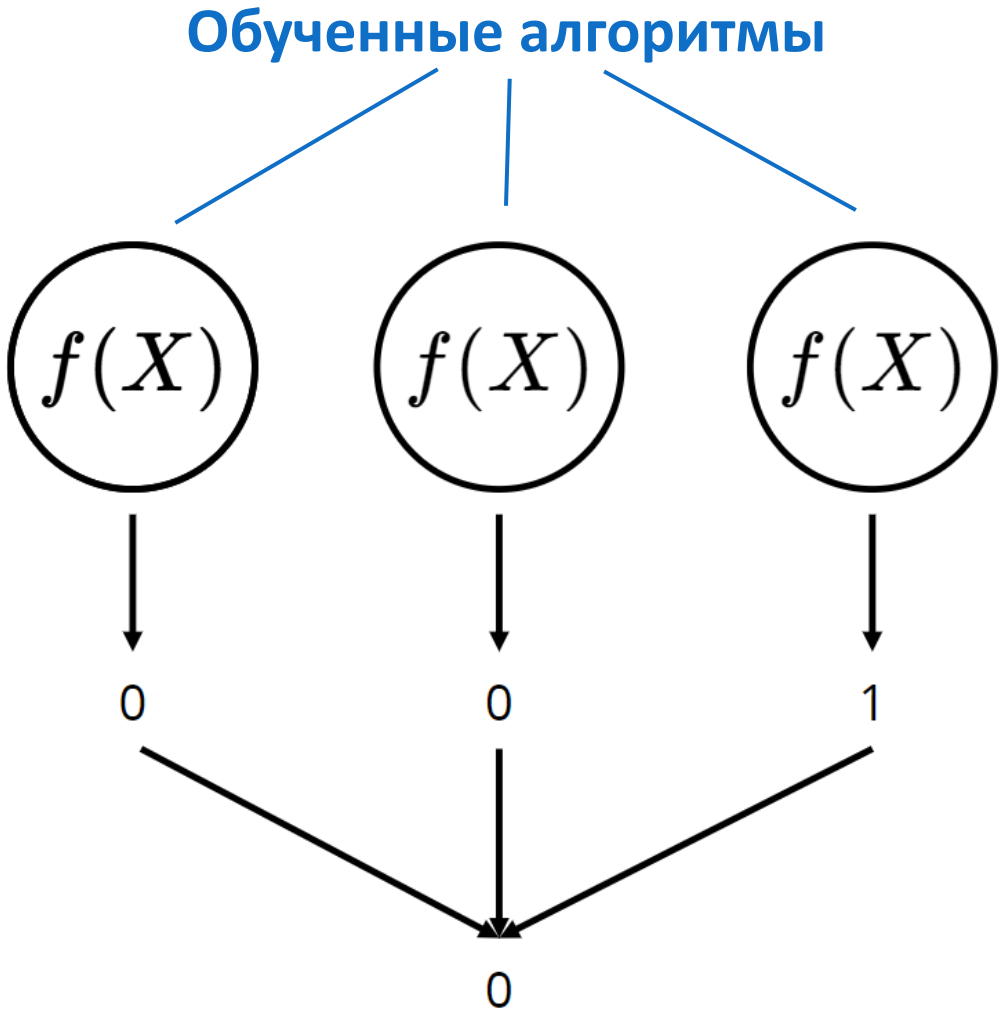
Эксперимент Френсиса Гальтона

Собралось около 800 человек, которые попытались угадать вес быка на ярмарке. Бык весил 1198 фунтов.

- Ни один крестьянин не угадал точный вес быка
- Среднее предсказание оказалось равным 1197 фунтов.

Композиции алгоритмов

1. **Метод простого голосования (Max voting)**



Минус – нет степени уверенности в ответе

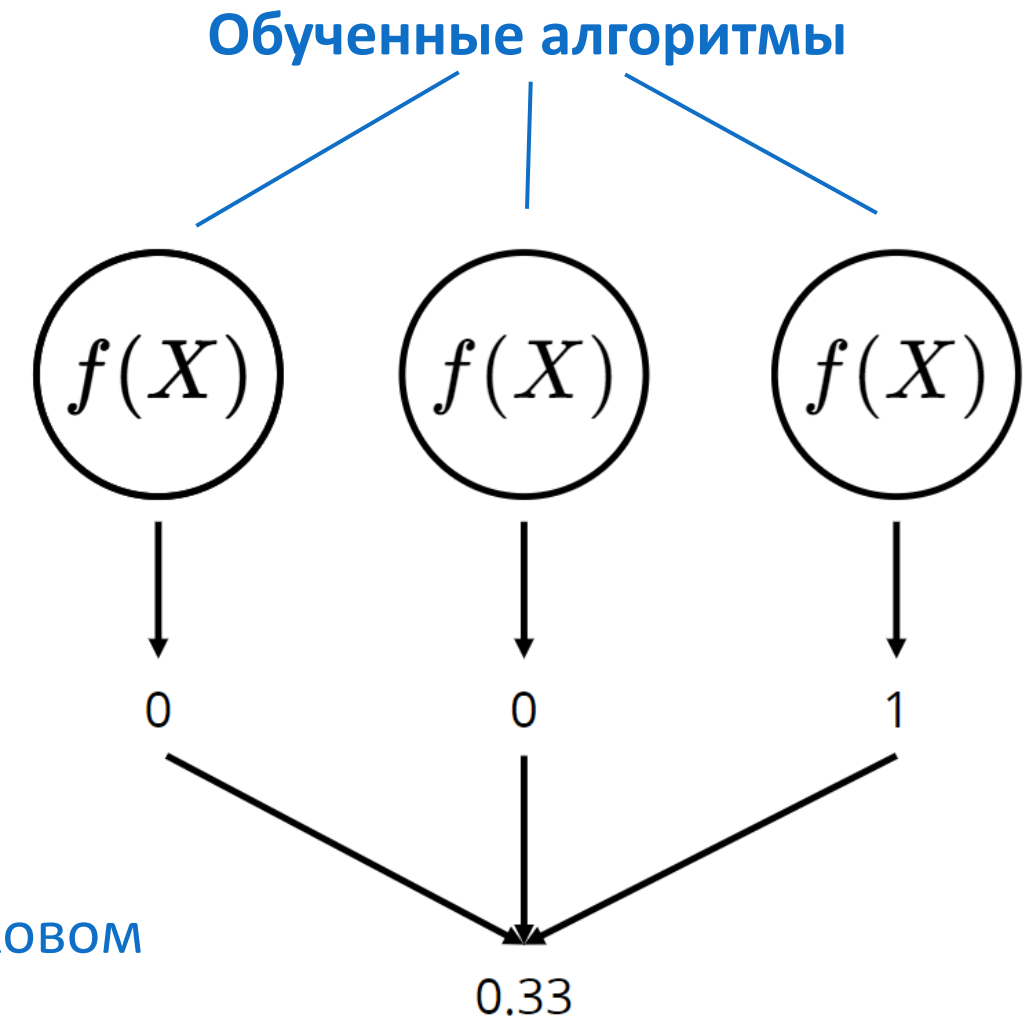
Композиции алгоритмов

1. **Метод простого голосования (Max voting)**

   Усреднение (Averaging)

$$f_1(X), f_2(X), \ldots, f_n(X)$$

$$f(X) = \frac{\sum_i f_i(X)}{n}$$



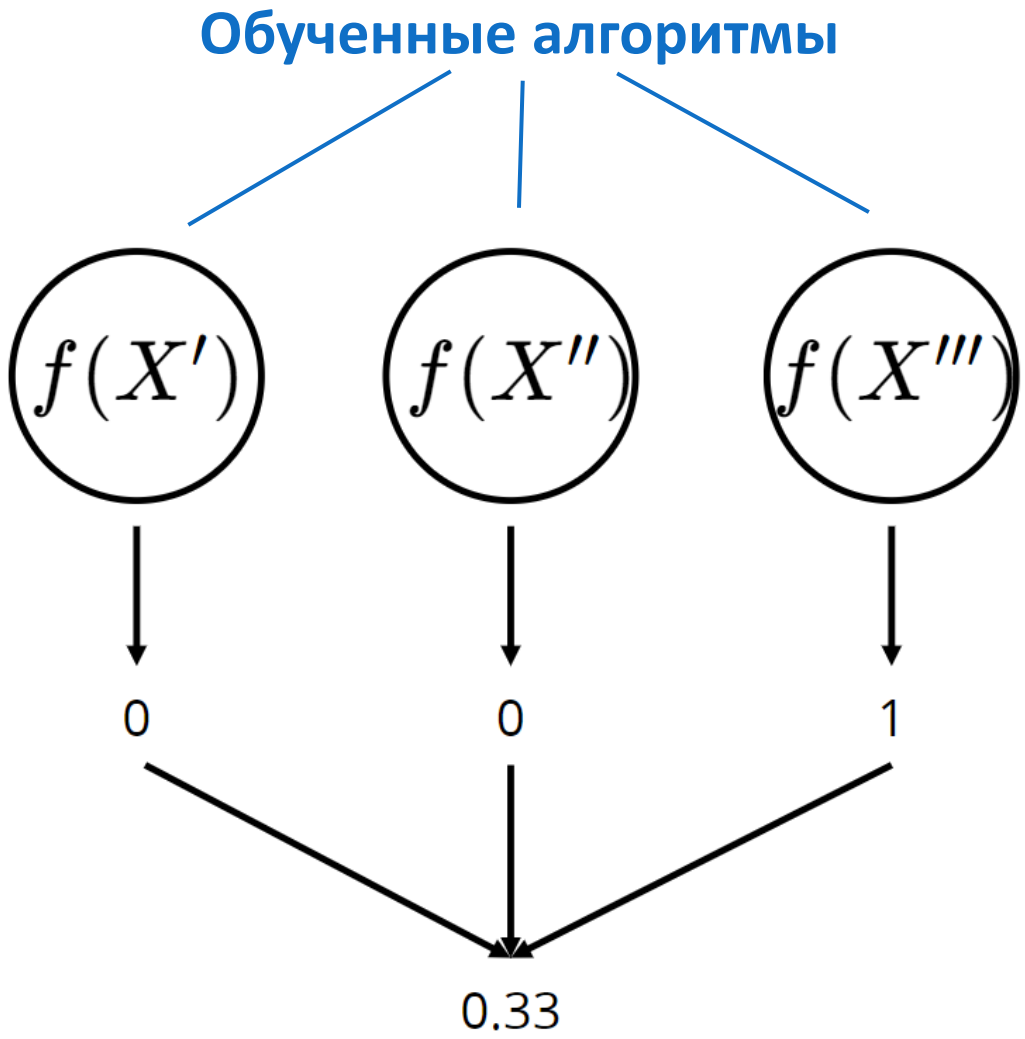
Минус – если все модели обучены на одинаковом датасете, они дадут одинаковый результат

Композиции алгоритмов

2. Бэггинг (Bagging)

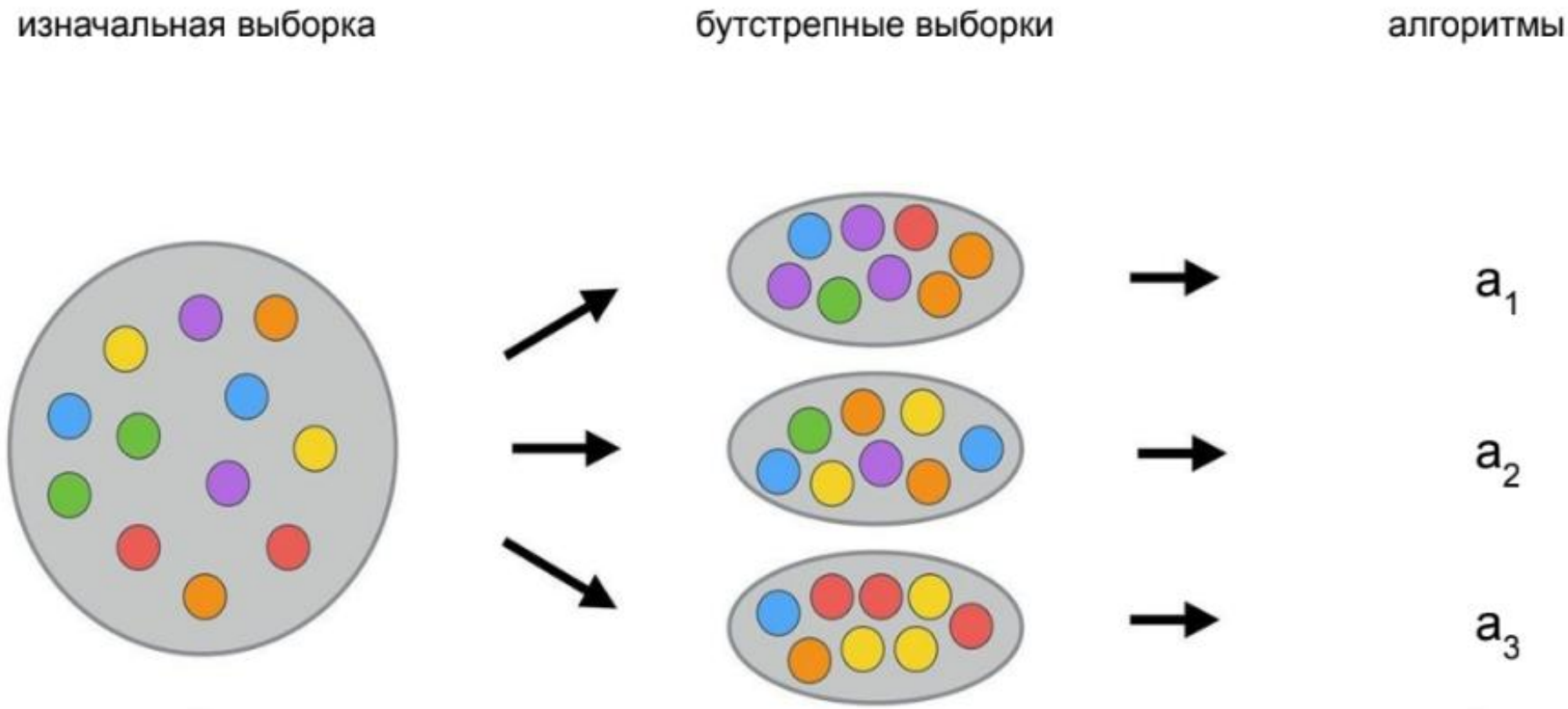
Bootstrap AGGregation

Наиболее известная модель – **Случайный лес** (Random Forest)



Композиции алгоритмов

Bootstrap



Композиции алгоритмов

Bootstrap

$$\{x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}, x^{(5)}\} \to \{x^{(1)}, x^{(3)}, x^{(4)}, x^{(1)}, x^{(4)}\}$$

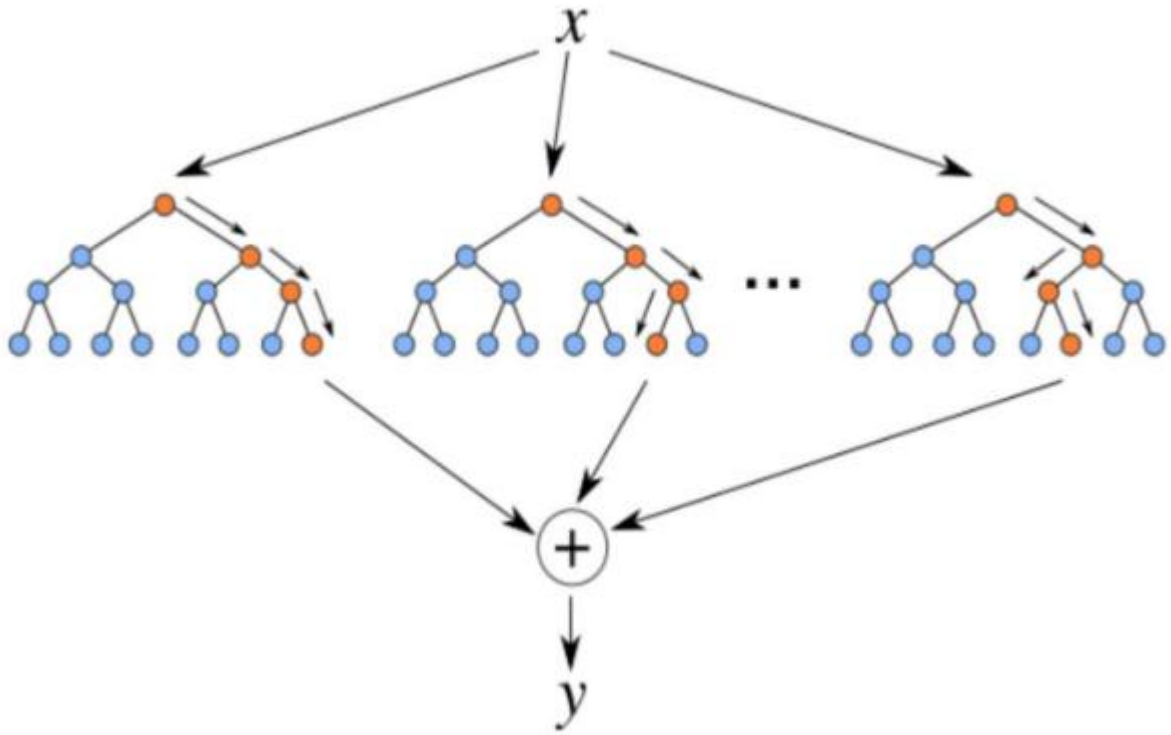
height	color	gender	weight
1.6	blue	m	88
1.6	green	f	76
1.5	blue	f	56
1.8	red	m	73
1.5	green	m	77
1.4	blue	f	57

height	color	gender	weight
1.6	blue	m	88
1.8	red	m	73
1.5	blue	f	56
1.8	red	m	73
1.5	green	m	77
1.5	blue	f	56

Композиции алгоритмов

Случайный лес (Random forest)

- Много деревьев
- Бутстрепная выборка для каждого дерева
- Финальное решение принимается простым голосованием



Композиции алгоритмов

Случайный лес (Random forest)

- Всё, что относится к деревьям, можно сказать и про случайный лес
- **Позволяет преодолеть склонность деревьев переобучаться поодиночке**

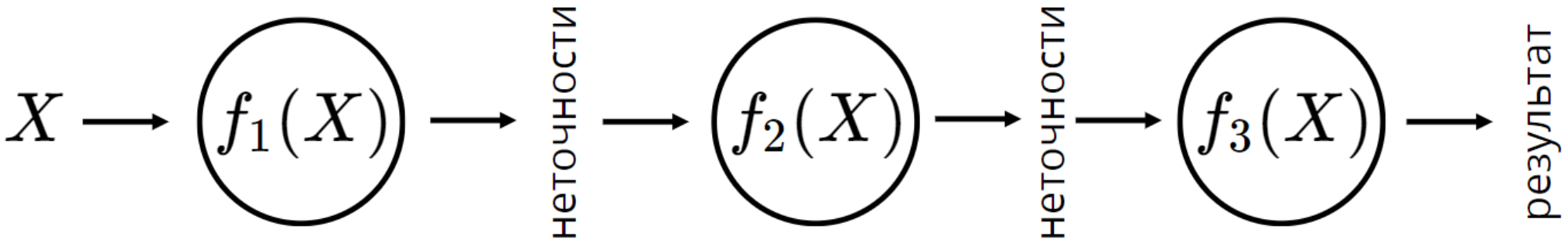
Преимущества:

- Можно распараллелить
- Показывает важность каждой фичи

Композиции алгоритмов

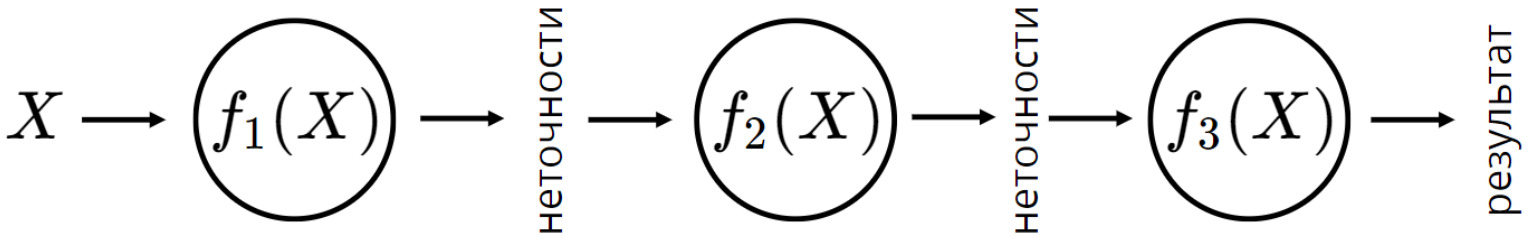
3. Бустинг (Boosting)

Последовательный набор моделей, последующие работают с ошибками предыдущей модели



Композиции алгоритмов

3. Бустинг (Boosting)



target

height	color	gender	weight
1.6	blue	m	88
1.6	green	f	76
1.5	blue	f	56
1.8	red	m	73
1.5	green	m	77
1.4	blue	f	57

Средний вес: 71.2

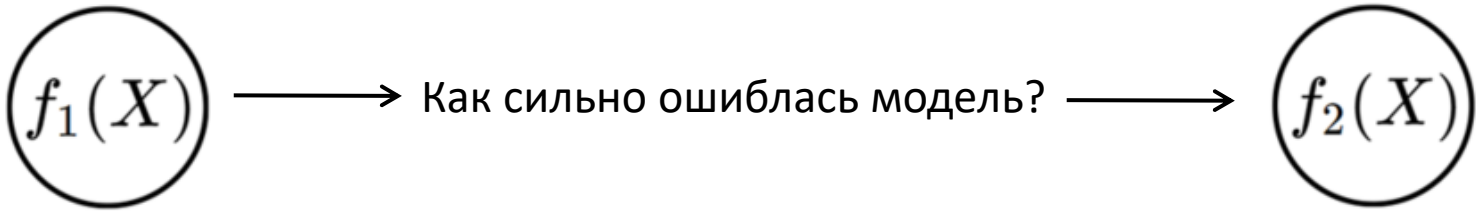
Композиции алгоритмов

3. **Бустинг (Boosting) –** последующие модели исправляют ошибки предыдущих

target

height	color	gender	weight	residuals
1.6	blue	m	88	16.8
1.6	green	f	76	4.8
1.5	blue	f	56	-15.2
1.8	red	m	73	1.8
1.5	green	m	77	5.8
1.4	blue	f	57	-14.2

Средний вес: 71.2



Композиции алгоритмов

3. Бустинг (Boosting)

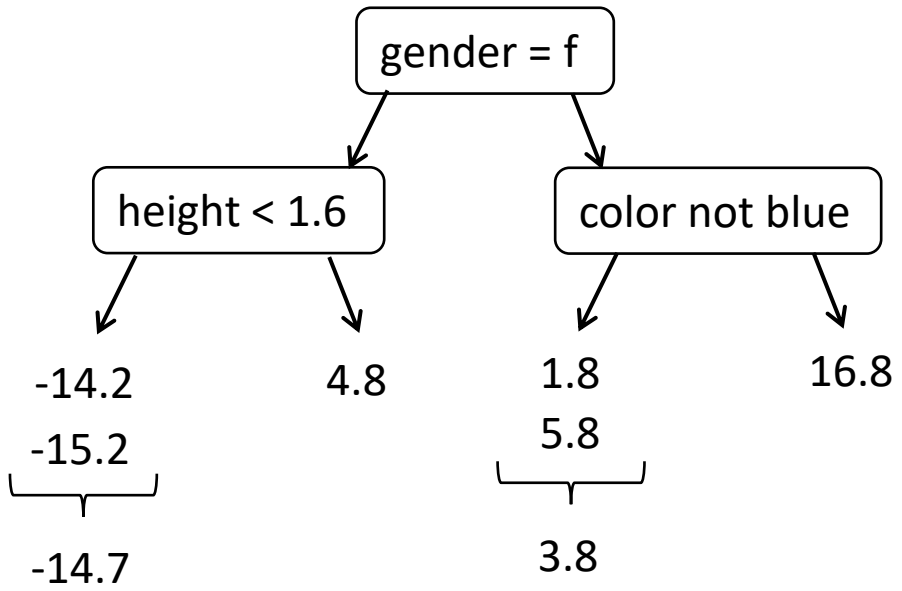
target

height	color	gender	weight	residuals
1.6	blue	m	88	16.8
1.6	green	f	76	4.8
1.5	blue	f	56	-15.2
1.8	red	m	73	1.8
1.5	green	m	77	5.8
1.4	blue	f	57	-14.2

Средний вес: 71.2

$f_1(X)$

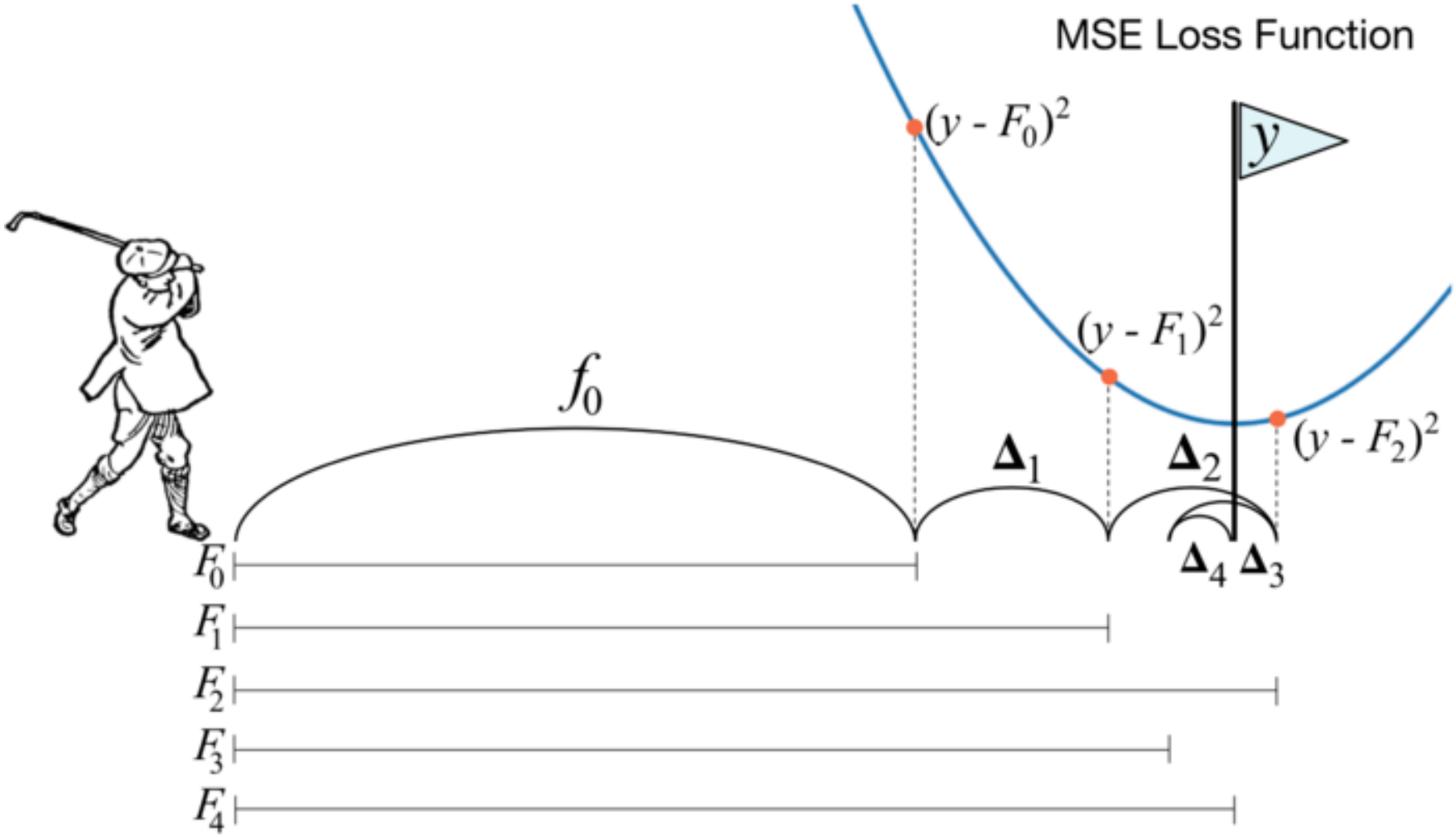
$f_2(X)$



f(x) = (((c1*f1) + c2*f2)+c3*f3)…

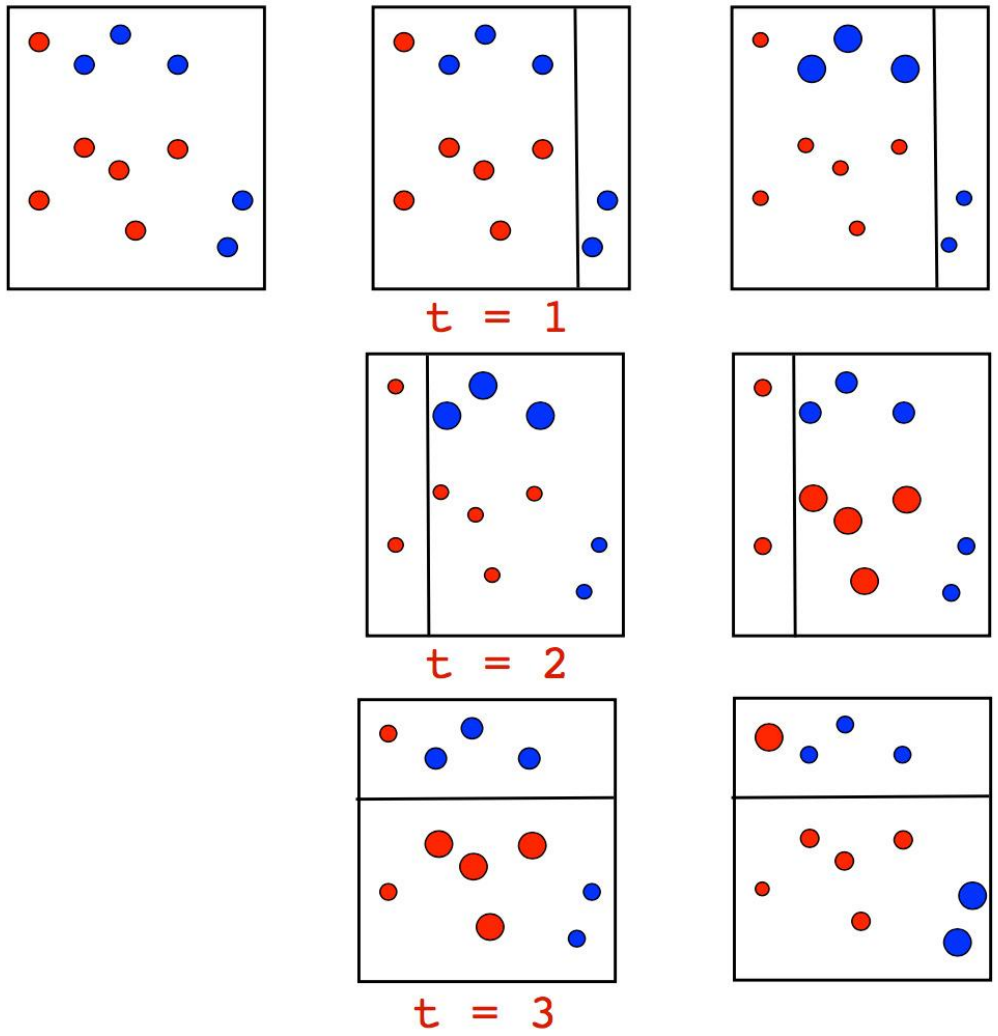
Композиции алгоритмов

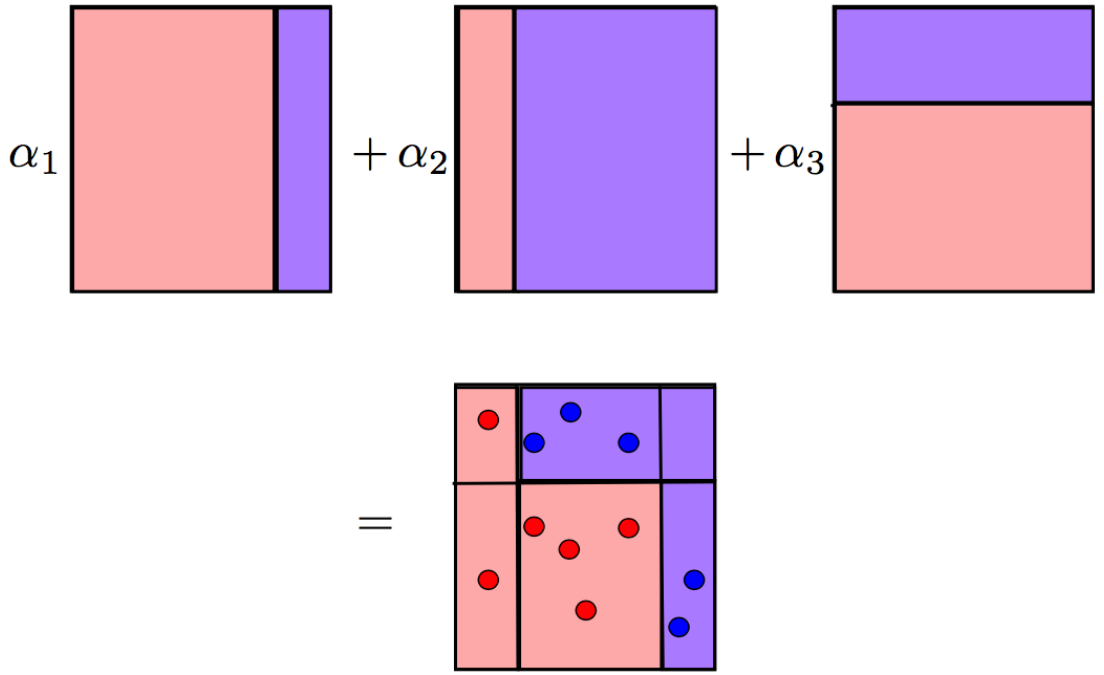
3. Бустинг (Boosting)



Композиции алгоритмов

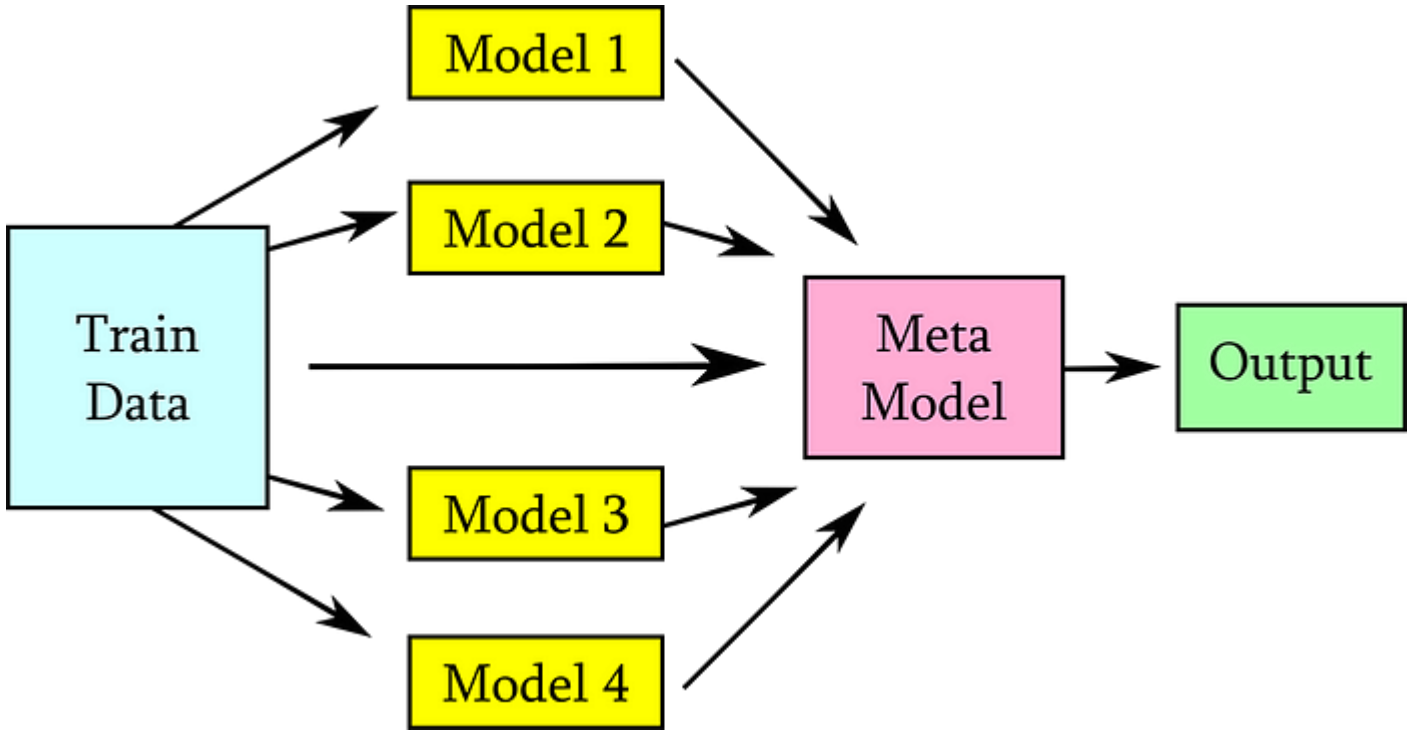
Адаптивный Бустинг (AdaBoost) – изменение весов ошибок





Композиции алгоритмов

4. Стекинг (Stacking)



Мета-модель обучается на выходных данных предыдущих моделей

Выводы

- Дерево решений обучается последовательно (разветвление за разветвлением) с помощью индекса Джини или энтропии и позволяет визуализировать данные
- Ансамбли моделей позволяют с помощью большого количества простых моделей находить сложные нелинейные зависимости
- Случайный лес параллельно объединяет деревья решений с помощью механизма бутстрапа
- Бустинг объединяет модели в цепочку последовательных исправлений ошибок предыдущей моделей

Спасибо за внимание!