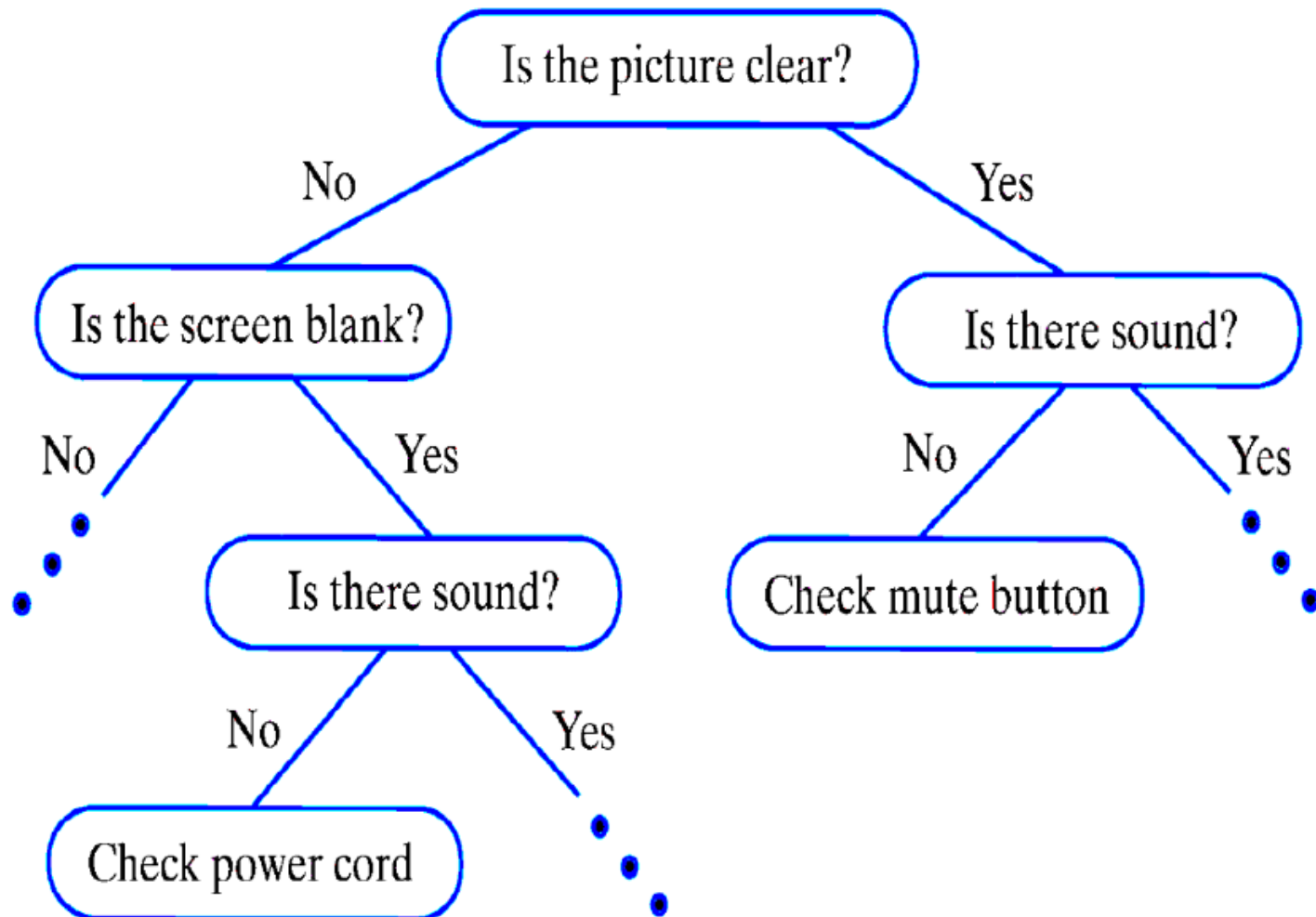


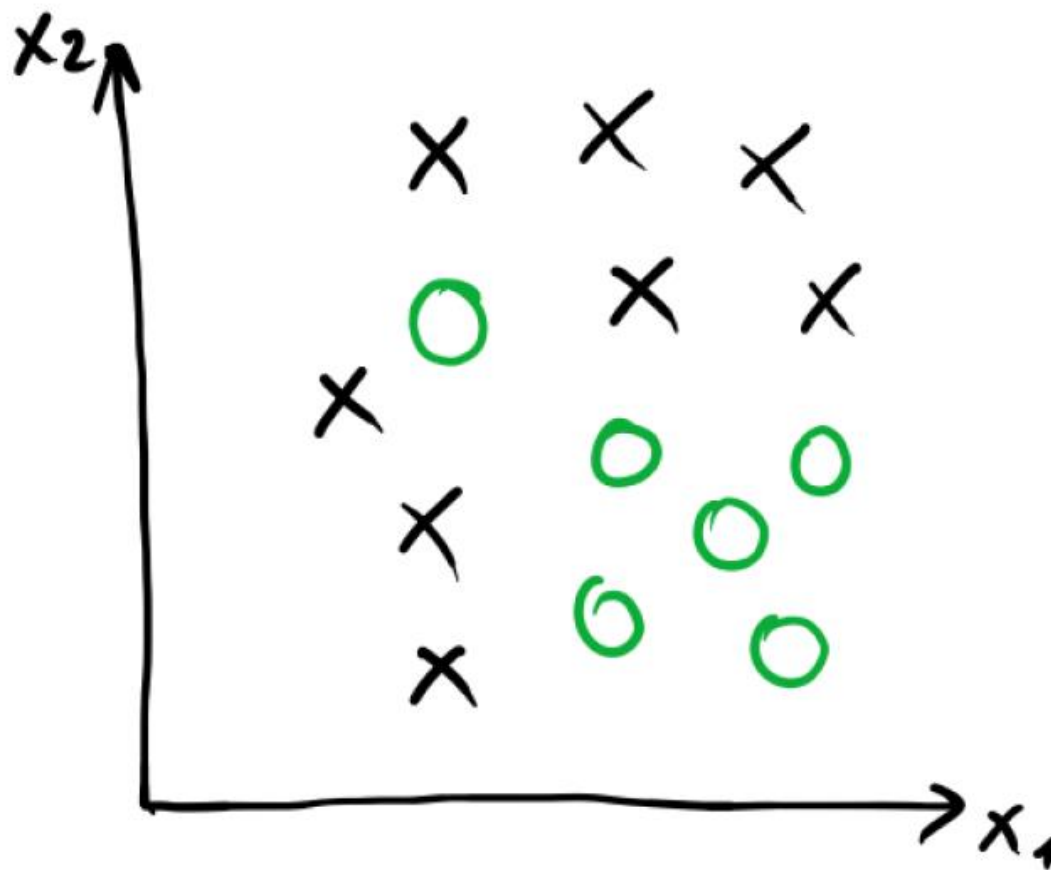
Решающие деревья

Елена Кантонистова

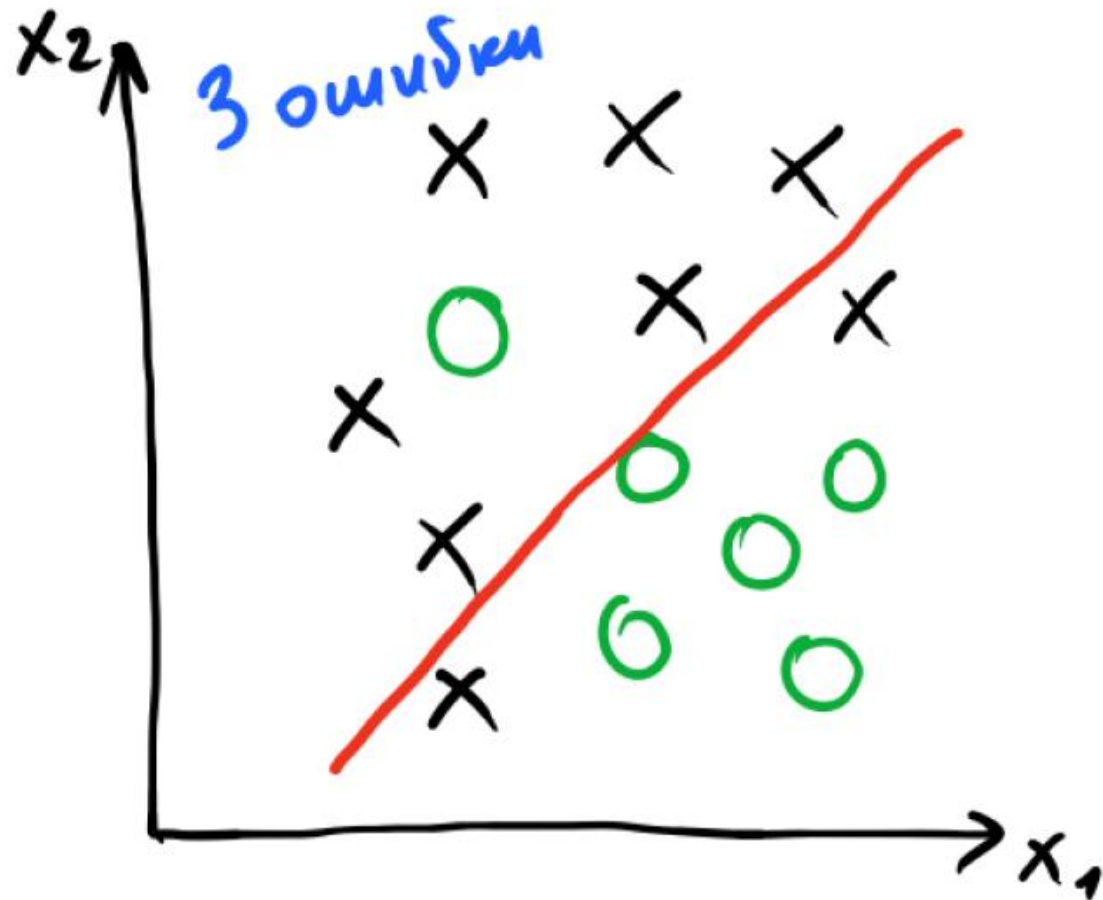
ПРИМЕР РЕШАЮЩЕГО ДЕРЕВА



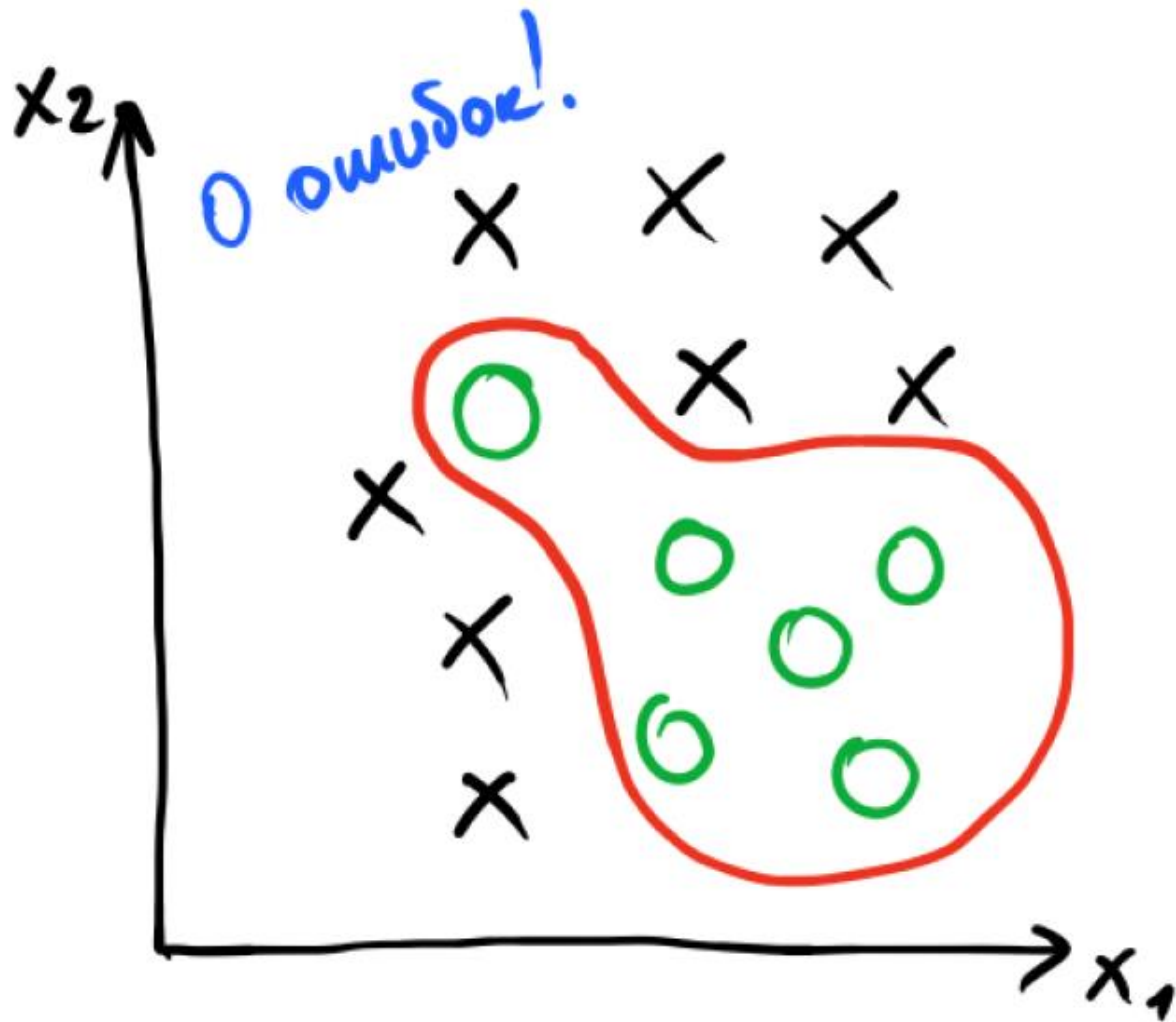
ПРИМЕР



ЛИНЕЙНАЯ МОДЕЛЬ



НЕЛИНЕЙНЫЙ АЛГОРИТМ

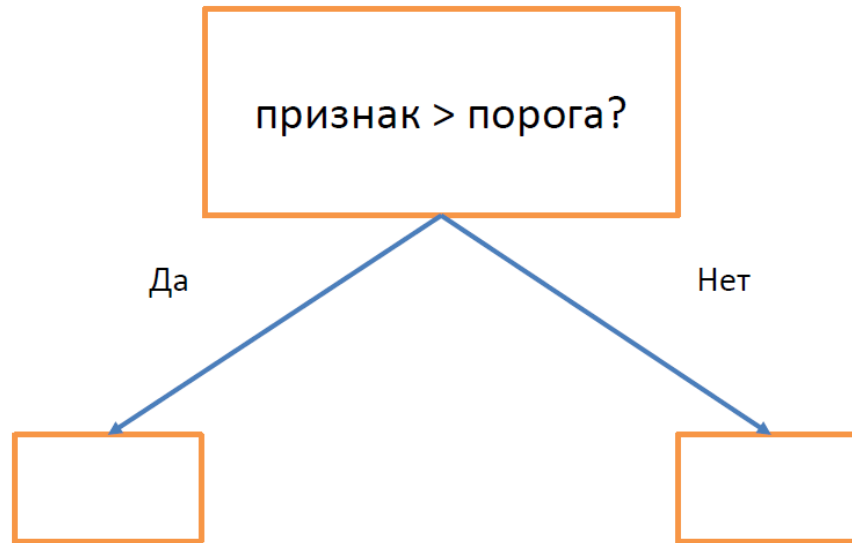


РЕШАЮЩЕЕ ДЕРЕВО

Решающее дерево – это бинарное дерево, в котором:

1) каждой вершине v приписана функция (предикат)

$$\beta_v: X \rightarrow \{0,1\}$$



РЕШАЮЩЕЕ ДЕРЕВО

Решающее дерево – это бинарное дерево, в котором:

1) каждой вершине v приписана функция (предикат)

$$\beta_v: X \rightarrow \{0,1\}$$



2) каждой листовой вершине v приписан прогноз $c_v \in Y$ (для классификации – класс или вероятность класса, для регрессии – действительное значение целевой переменной)

ЖАДНЫЙ АЛГОРИТМ ПОСТРОЕНИЯ РЕШАЮЩЕГО ДЕРЕВА

1 шаг: найдем наилучшее разбиение всей выборки X на две части: $R_1(j, t) = \{x \mid x_j < t\}$ и $R_2(j, t) = \{x \mid x_j \geq t\}$ с точки зрения некоторого функционала $Q(X, j, t)$:

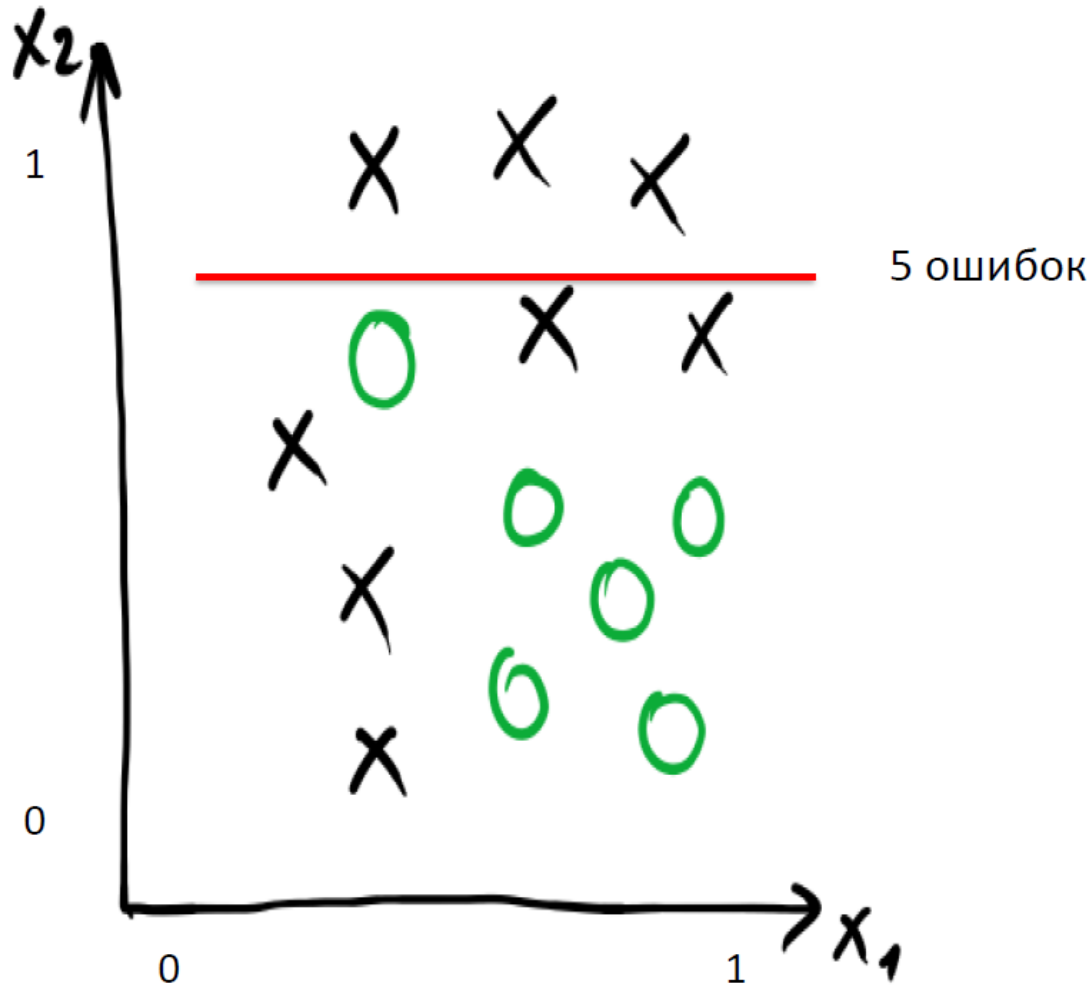
- найдем наилучшие j и t
- создадим корень дерева, поставив в него предикат $[x_j < t]$.

2 шаг: Для каждой из полученных подвыборок R_1 и R_2 рекурсивно применим шаг 1. И т.д.

В каждой вершине на каждом шаге проверяем, не выполнилось ли условие останова. Если выполнилось, то объявляем вершину листом и записываем в него предсказание.

ПРИМЕР

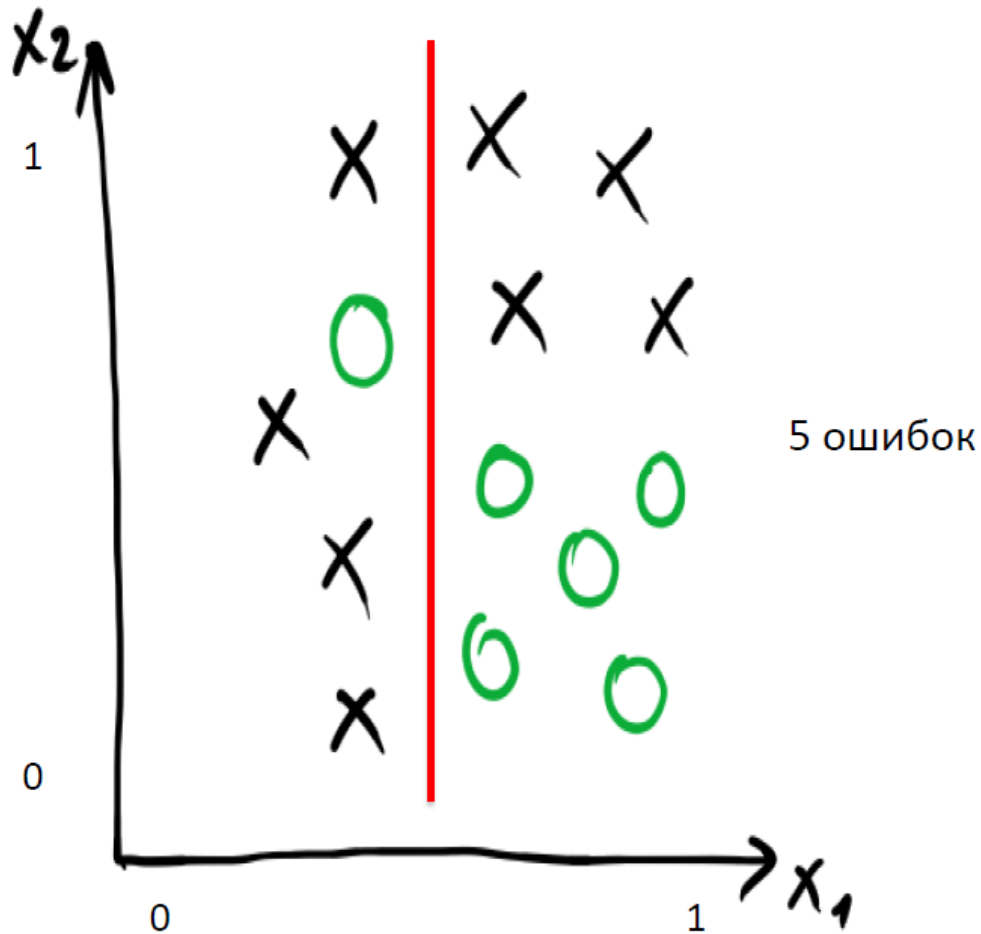
- Жадно найдем наилучший предикат



$$x_2 > 0.8$$

ПРИМЕР

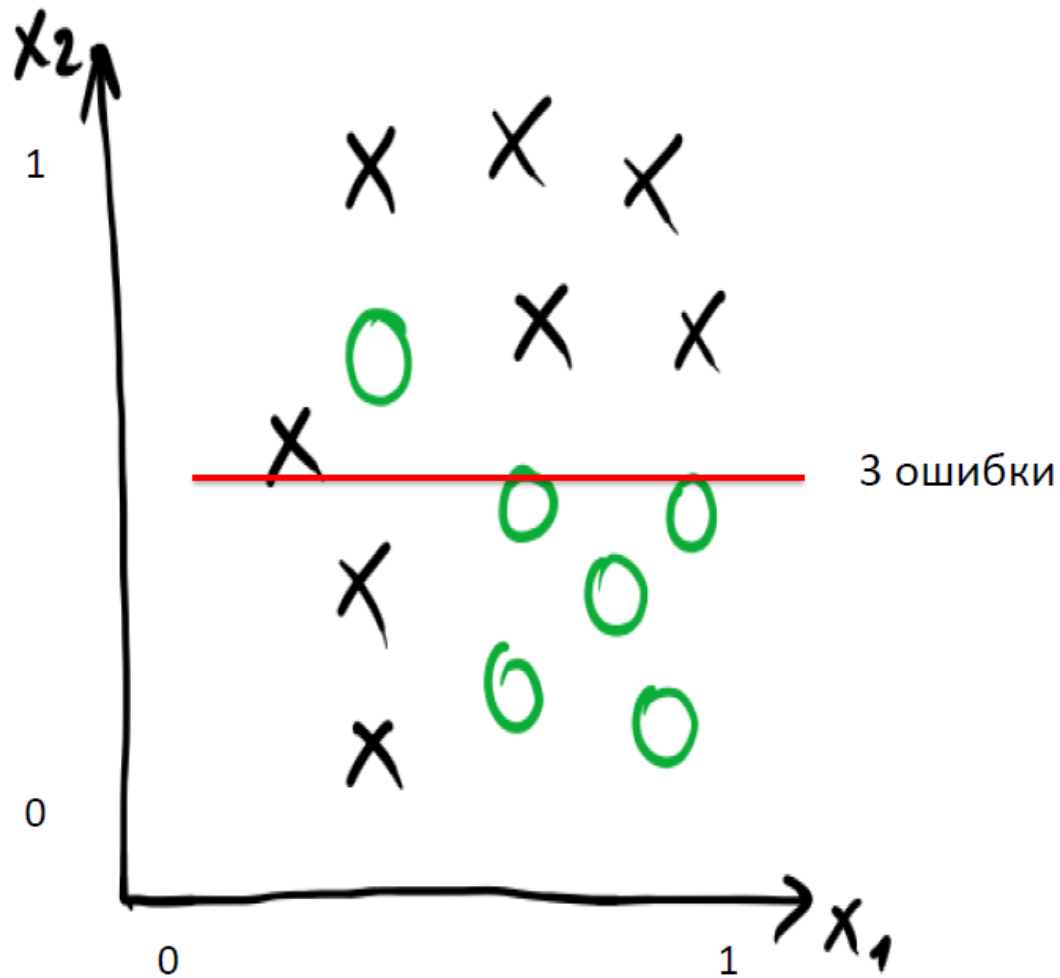
- Жадно найдем наилучший предикат



$$x_1 > 0.5$$

ПРИМЕР

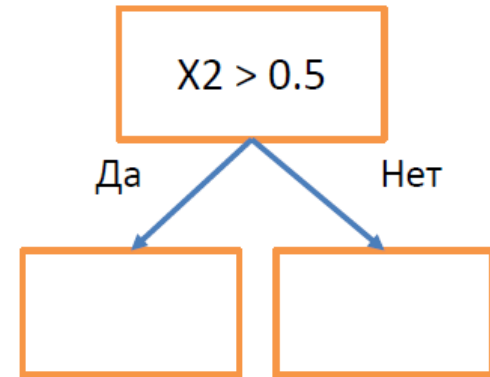
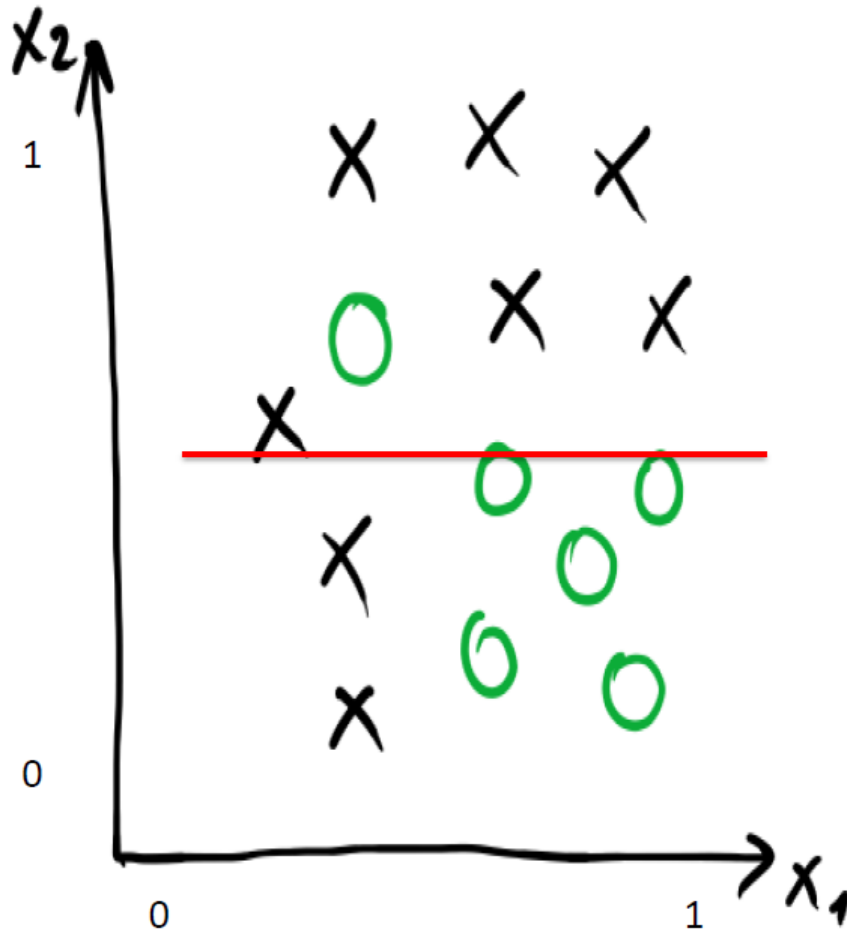
- Жадно найдем наилучший предикат



$$x_2 > 0.5$$

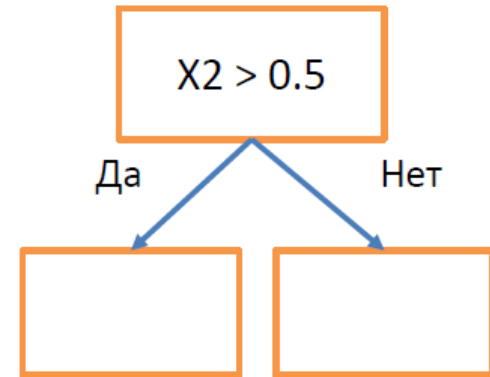
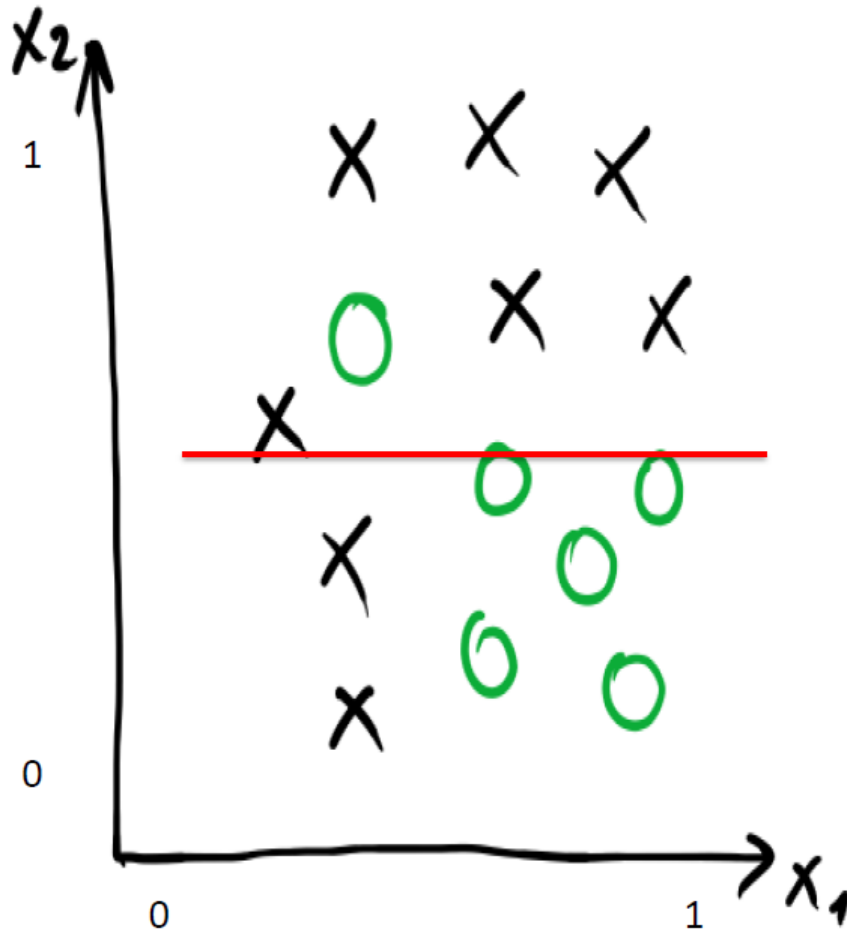
ПРИМЕР

- Нашли лучшее первое ветвление



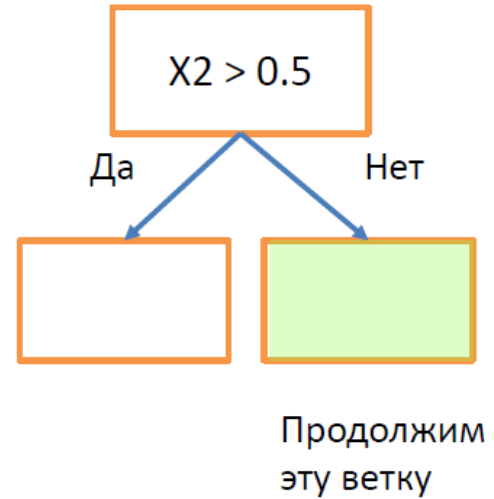
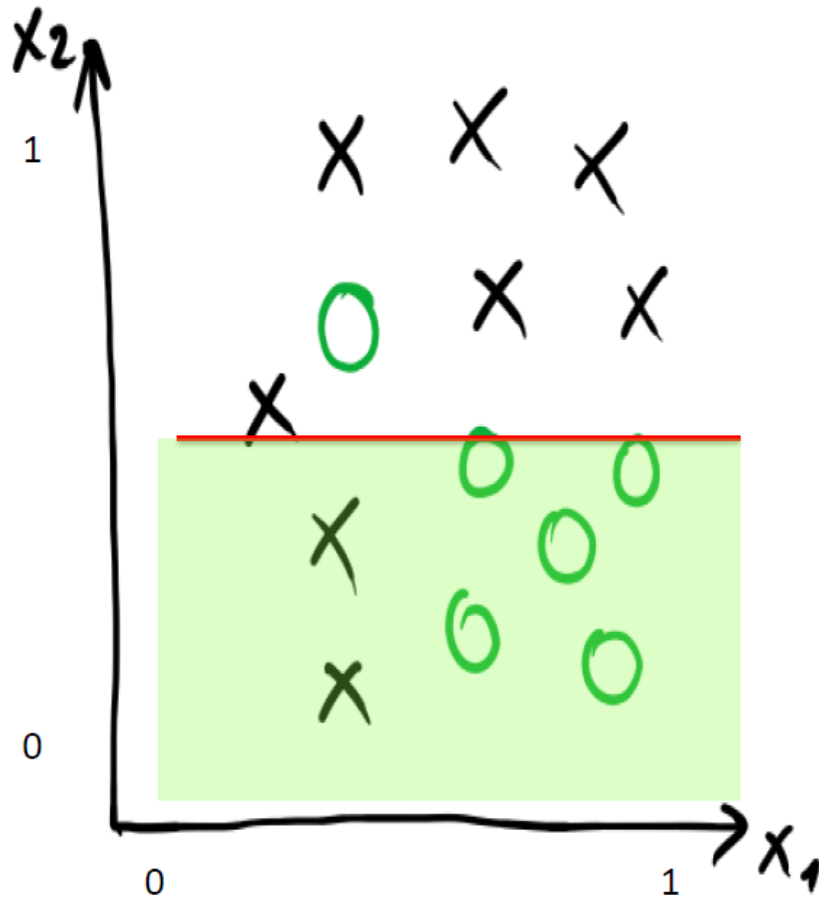
ПРИМЕР

- Нашли лучшее первое ветвление



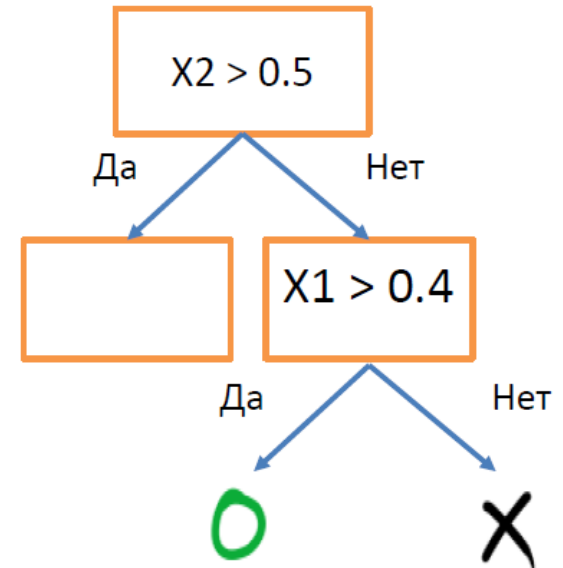
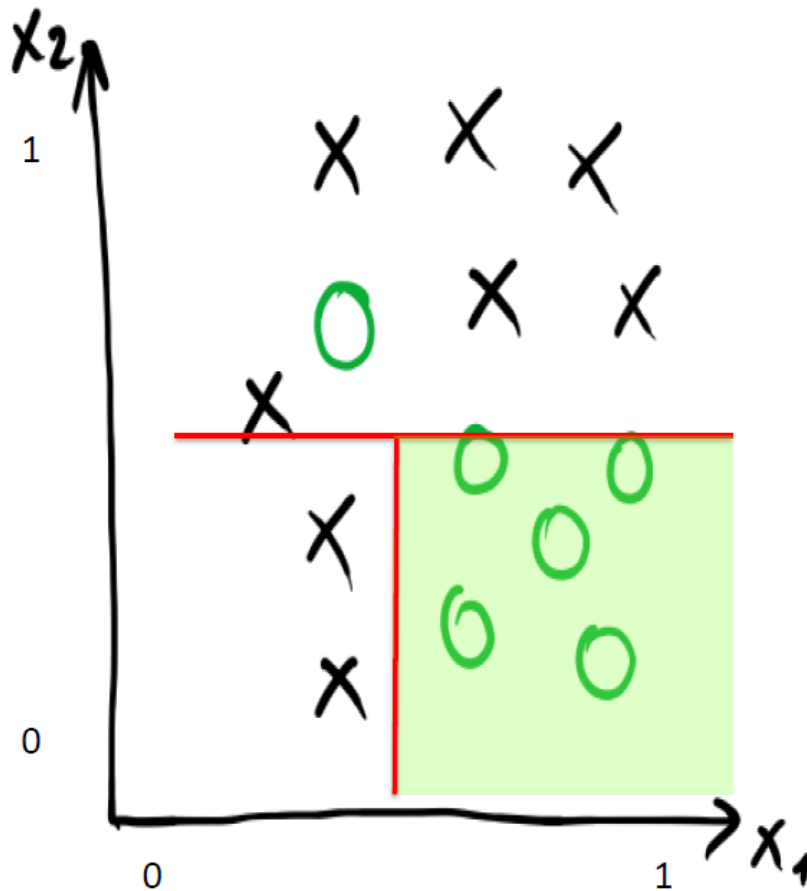
ПРИМЕР

- Нашли лучшее первое ветвление



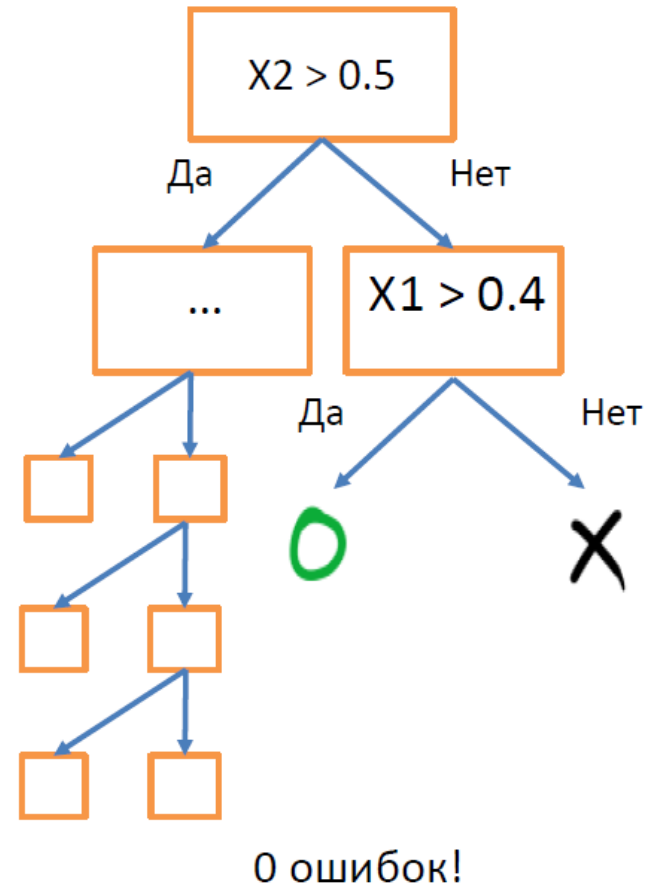
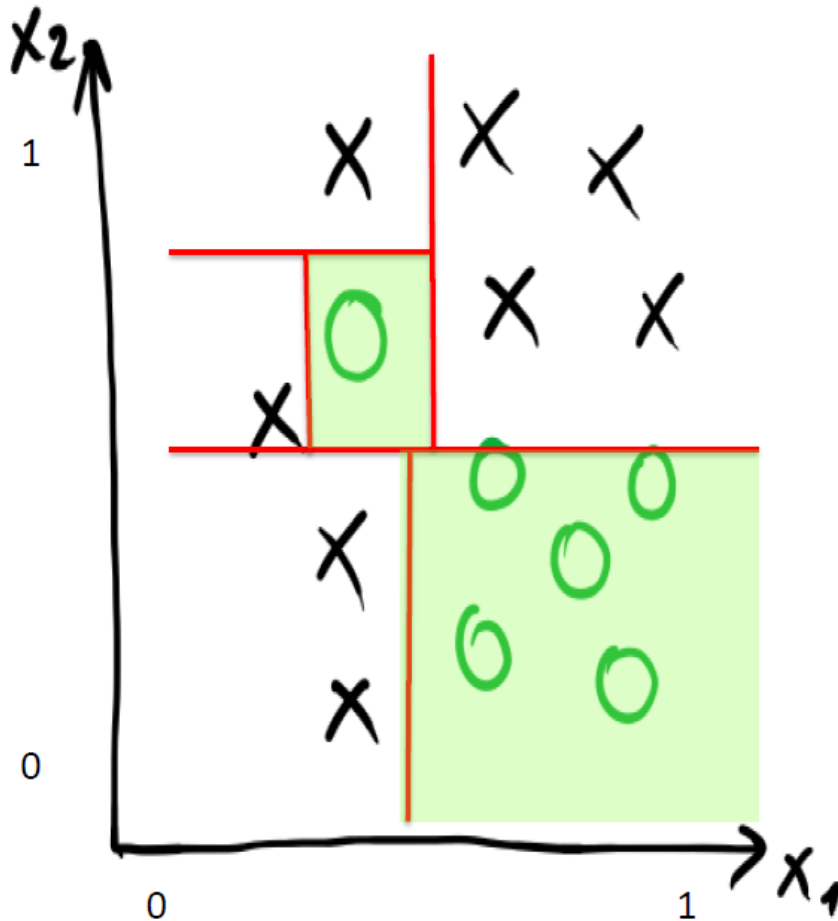
ПРИМЕР

- Нашли лучшее второе ветвление



ПРИМЕР

- Построили всё дерево



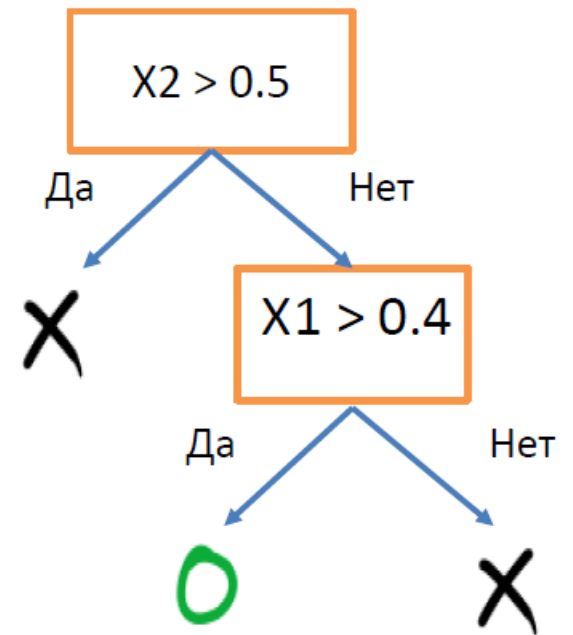
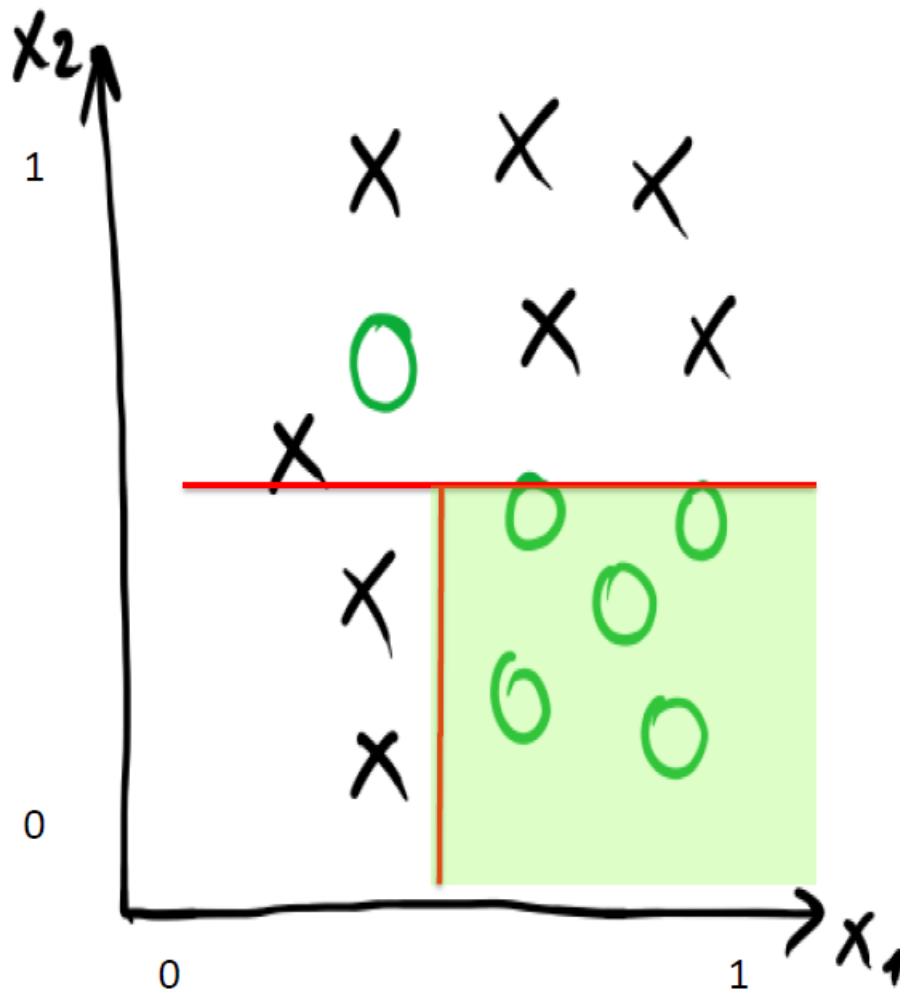
ПЕРЕОБУЧЕНИЕ

Для любой выборки можно построить решающее дерево, не допускающее на ней ни одной ошибки. Такое дерево скорее всего будет переобученным.

ЧТО ВЛИЯЕТ НА ПОСТРОЕНИЕ РЕШАЮЩЕГО ДЕРЕВА

- вид предикатов в вершинах
- функционал качества $Q(X, j, t)$
- критерий останова
- метод обработки пропущенных значений
- метод стрижки

ПРИМЕР



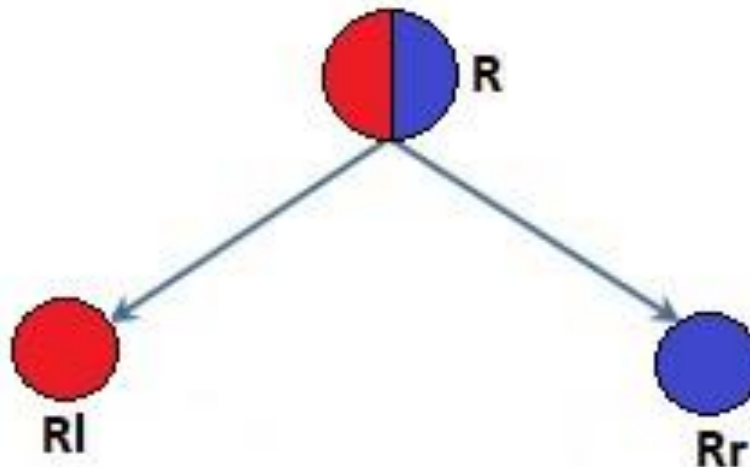
1 ошибка, но
скорее всего будет
лучше на тесте!

КРИТЕРИИ ИНФОРМАТИВНОСТИ

В каждой вершине оптимизируем функционал $Q(X, j, t)$.

- Пусть R – множество объектов, попадающих в вершину на данном шаге, а R_l и R_r - объекты, попадающие в левую и правую ветки после разбиения.

Цель: хотим, чтобы после разбиения объектов на две группы внутри каждой группы как можно больше объектов было одного класса.



КРИТЕРИИ ИНФОРМАТИВНОСТИ

- Пусть R – множество объектов, попадающих в вершину на данном шаге, а R_l и R_r - объекты, попадающие в левую и правую ветки после разбиения.

Цель: хотим, чтобы после разбиения объектов на две группы внутри каждой группы как можно больше объектов было одного класса.

- $H(R)$ - *критерий информативности* - мера неоднородности целевых (разнообразие) переменных внутри группы R .
- Чем меньше разнообразие целевой переменной внутри группы, тем меньше значение $H(R)$. То есть хотим

$$H(R_l) \rightarrow \min, H(R_r) \rightarrow \min$$

КРИТЕРИИ ИНФОРМАТИВНОСТИ

- Пусть R – множество объектов, попадающих в вершину на данном шаге, а R_l и R_r - объекты, попадающие в левую и правую ветки после разбиения.

Цель: хотим, чтобы после разбиения объектов на две группы внутри каждой группы как можно больше объектов было одного класса.

- Чем меньше разнообразие целевой переменной внутри группы, тем меньше значение $H(R)$. То есть

$$H(R_l) \rightarrow \min, H(R_r) \rightarrow \min$$

- Определим функционал Q по формуле:

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

КРИТЕРИИ ИНФОРМАТИВНОСТИ

- Пусть R – множество объектов, попадающих в вершину на данном шаге, а R_l и R_r - объекты, попадающие в левую и правую ветки после разбиения.

Цель: хотим, чтобы после разбиения объектов на две группы внутри каждой группы как можно больше объектов было одного класса.

- Чем меньше разнообразие целевой переменной внутри группы, тем меньше значение $H(R)$. То есть

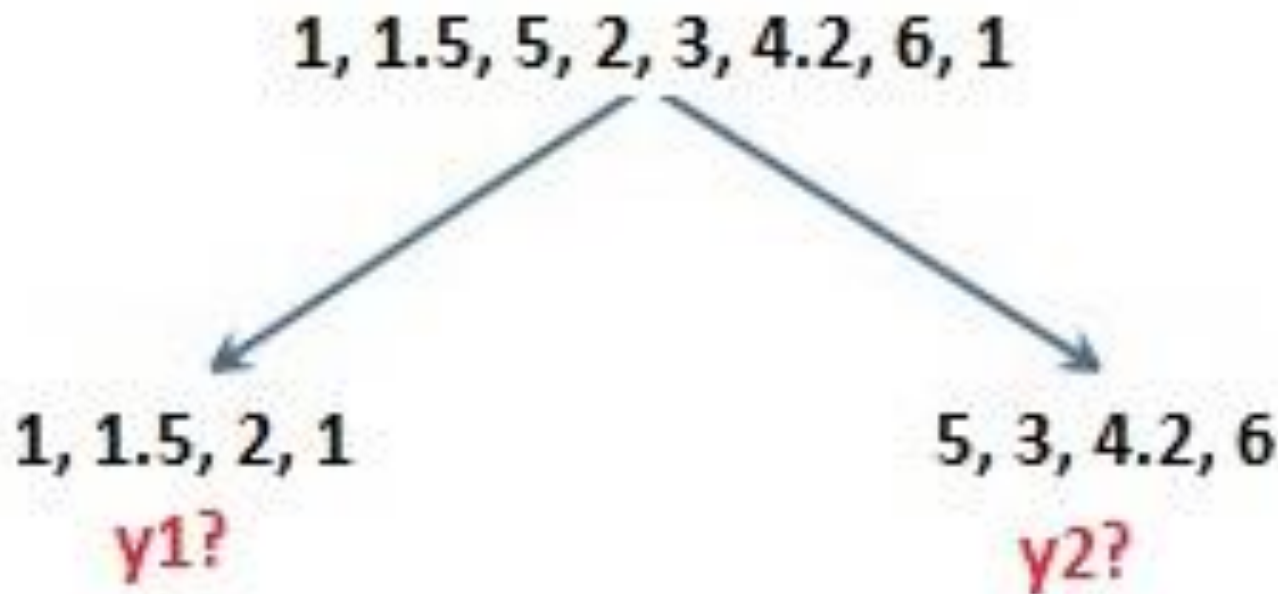
$$H(R_l) \rightarrow \min, H(R_r) \rightarrow \min$$

- Определим функционал Q по формуле:

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

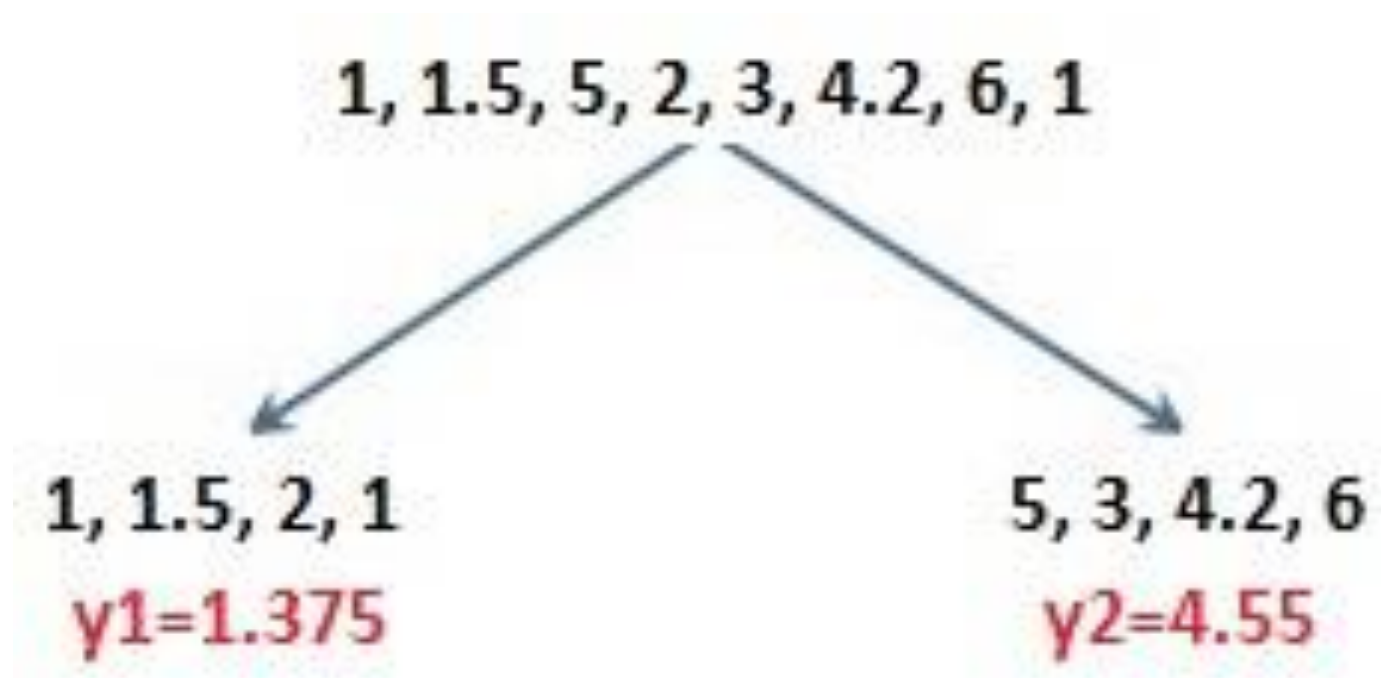
ПРИМЕР: РЕШАЮЩЕЕ ДЕРЕВО В ЗАДАЧЕ РЕГРЕССИИ

Предположим, что в лист дерева попало несколько объектов. В каждом листе дерево предсказывает константу. Какую константу выгоднее всего выдать в качестве ответа?



ПРИМЕР: РЕШАЮЩЕЕ ДЕРЕВО В ЗАДАЧЕ РЕГРЕССИИ

Если в качестве функционала ошибки в листе использовать среднеквадратичную ошибку, то в качестве ответа надо выдавать среднее значение целевых переменных всех объектов, попавших в лист.



ВИД КРИТЕРИЯ ИНФОРМАТИВНОСТИ

- В каждом листе дерево выдает константу c (вещественное число – в регрессии, класс или вероятность класса – в классификации).
- Чем лучше объекты в листе предсказываются этой константой, тем меньше средняя ошибка на объектах:

$$H(R) = \min_{c \in \mathbb{R}} \frac{1}{|R|} \sum_{(x_i, y_i) \in R} L(y_i, c),$$

где $L(y, c)$ – некоторая функция потерь.

$H(R)$ В ЗАДАЧЕ РЕГРЕССИИ

Если в качестве функции потерь мы берём квадратичную ошибку, то

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

H(R) В ЗАДАЧЕ РЕГРЕССИИ

Если в качестве функции потерь мы берём квадратичную ошибку, то

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

Её минимум достигается при

$$c = \frac{1}{|R|} \sum_{(x_i, y_i) \in R} y_i,$$

то есть в листе предсказывается среднее значение целевой переменной на объектах, попавших в лист.

$H(R)$ В ЗАДАЧЕ РЕГРЕССИИ

Если в качестве функции потерь мы берём квадратичную ошибку, то

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

Её минимум достигается при

$$c = \frac{1}{|R|} \sum_{(x_i, y_i) \in R} y_i,$$

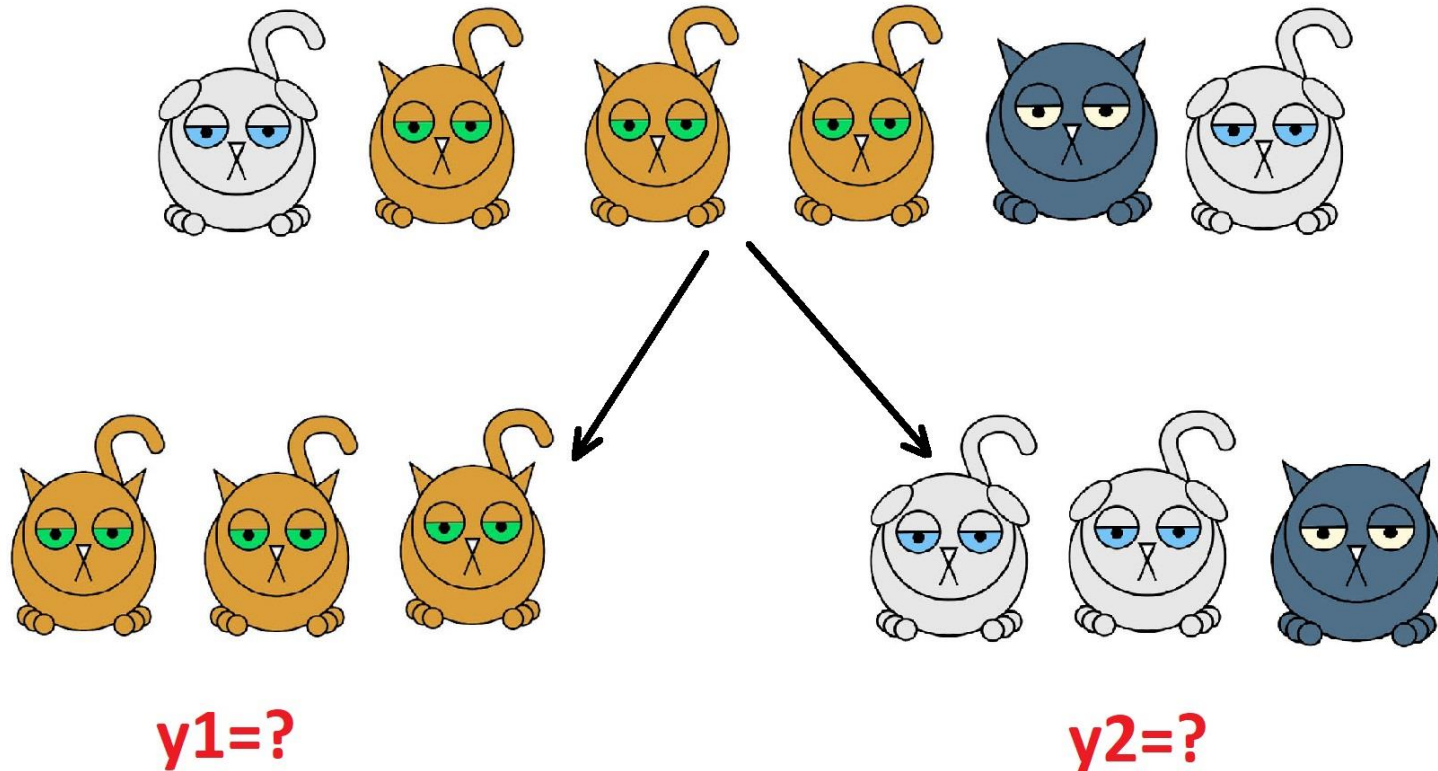
то есть в листе предсказывается среднее значение целевой переменной на объектах, попавших в лист.

Значит, *информативность $H(R)$ в листе – это дисперсия целевой переменной (для объектов, попавших в этот лист).*

Чем меньше дисперсия, тем меньше разброс целевой переменной объектов, попавших в лист.

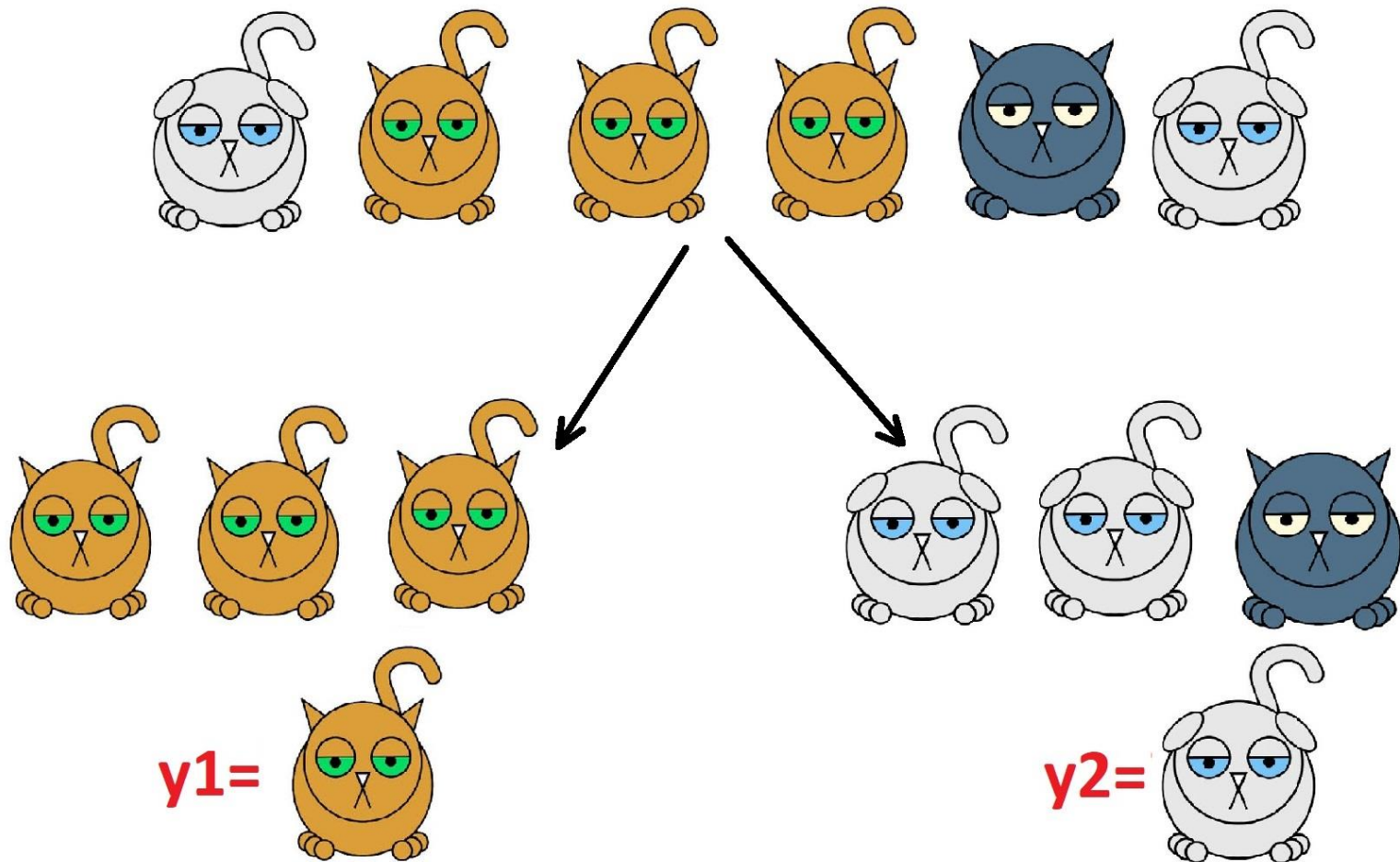
ПРИМЕР: РЕШАЮЩЕЕ ДЕРЕВО В ЗАДАЧЕ КЛАССИФИКАЦИИ

Предположим, что в лист дерева попало несколько объектов. В каждом листе дерево предсказывает класс объекта. Какой класс выгоднее всего выдать в качестве ответа?



ПРИМЕР: РЕШАЮЩЕЕ ДЕРЕВО В ЗАДАЧЕ КЛАССИФИКАЦИИ

Разумнее всего в качестве ответа в листе выдавать самый представительный класс.



Н(R) В ЗАДАЧАХ КЛАССИФИКАЦИИ

Решаем задачу классификации с K классами: $1, 2, \dots, K$.

- Пусть p_k доля объектов класса k , попавших в вершину:

$$p_k = \frac{1}{|R|} \sum_{(x_i, y_i) \in R} [y_i = k]$$

- Пусть k_* - самый представительный класс в данной вершине:

$$k_* = \operatorname{argmax}_k p_k$$

Ошибка классификации:

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

Н(R) В ЗАДАЧАХ КЛАССИФИКАЦИИ

Критерий Джини

- Будем в каждой вершине в качестве ответа выдавать не класс, а распределение вероятностей классов:
 $c = (c_1, \dots, c_K), \sum_i c_i = 1.$
- Качество распределения можно измерить с помощью критерия Бриера:

$$H(R) = \min_c \frac{1}{|R|} \sum_{(x_i, y_i) \in R} \sum_{k=1}^K (c_k - [y_i = k])^2$$

Утверждение.

- 1) Минимальное значение функционала $H(R)$ достигается на векторе, состоящем из долей классов: $c_* = (p_1, \dots, p_K)$
- 2) На векторе c_* функционал (*) переписывается в виде

$$H(R) = \sum_{k=1}^K p_k(1 - p_k) \text{ (критерий Джини).}$$

Н(R) В ЗАДАЧАХ КЛАССИФИКАЦИИ

Энтропийный критерий

Запишем логарифм правдоподобия:

$$H(R) = \min_c \left(-\frac{1}{|R|} \sum_{(x_i, y_i) \in R} \sum_{k=1}^K [y_i = k] \log c_k \right) (*)$$

На векторе $c_* = (p_1, \dots, p_K)$ функционал (*) записывается в виде

$$H(R) = -\sum_{k=1}^K p_k \log p_k \text{ (энтропия)}$$

Н(R) В ЗАДАЧАХ КЛАССИФИКАЦИИ

Энтропийный критерий

Запишем логарифм правдоподобия:

$$H(R) = \min_c \left(-\frac{1}{|R|} \sum_{(x_i, y_i) \in R} \sum_{k=1}^K [y_i = k] \log c_k \right) (*)$$

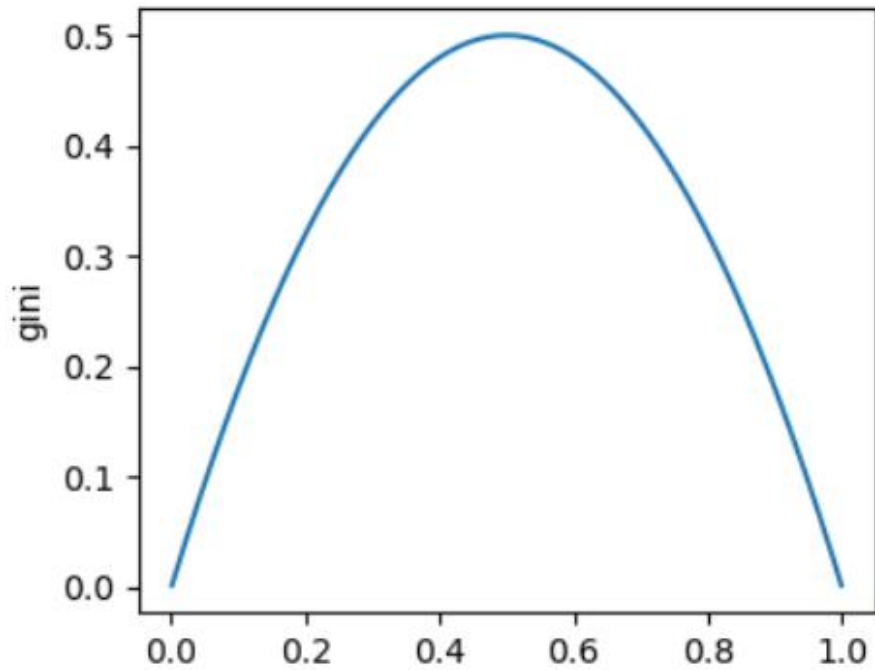
На векторе $c_* = (p_1, \dots, p_K)$ функционал (*) записывается в виде

$$H(R) = -\sum_{k=1}^K p_k \log p_k \text{ (энтропия)}$$

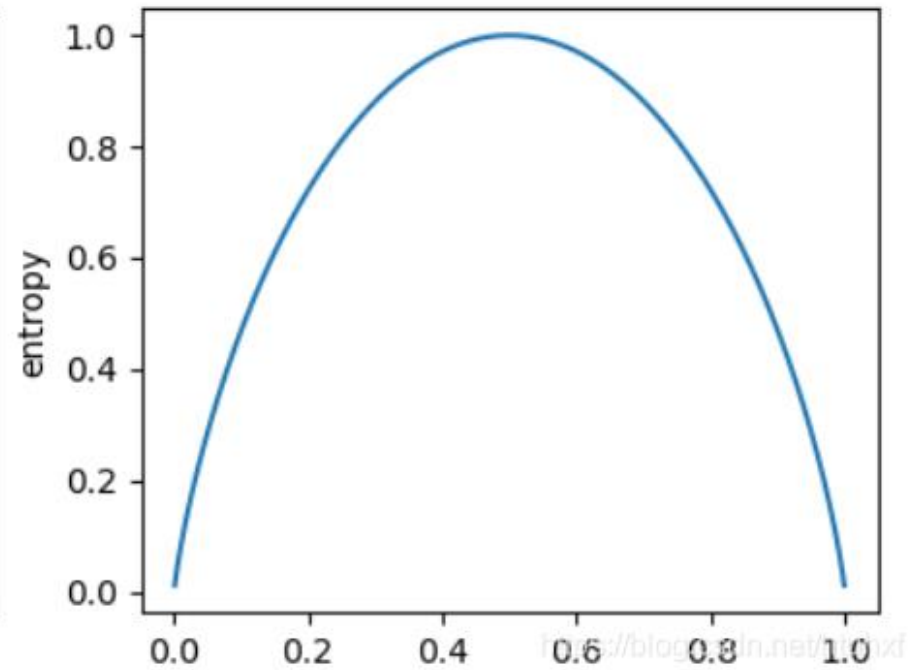
- Энтропия $H(R) \geq 0$ (минимум на распределении $p_i = 1, p_j = 0, j \neq i$)
- $\max H(R)$ достигается на равномерном распределении $p_1 = \dots = p_K = \frac{1}{K}$.

КРИТЕРИИ GINI И ENTROPY

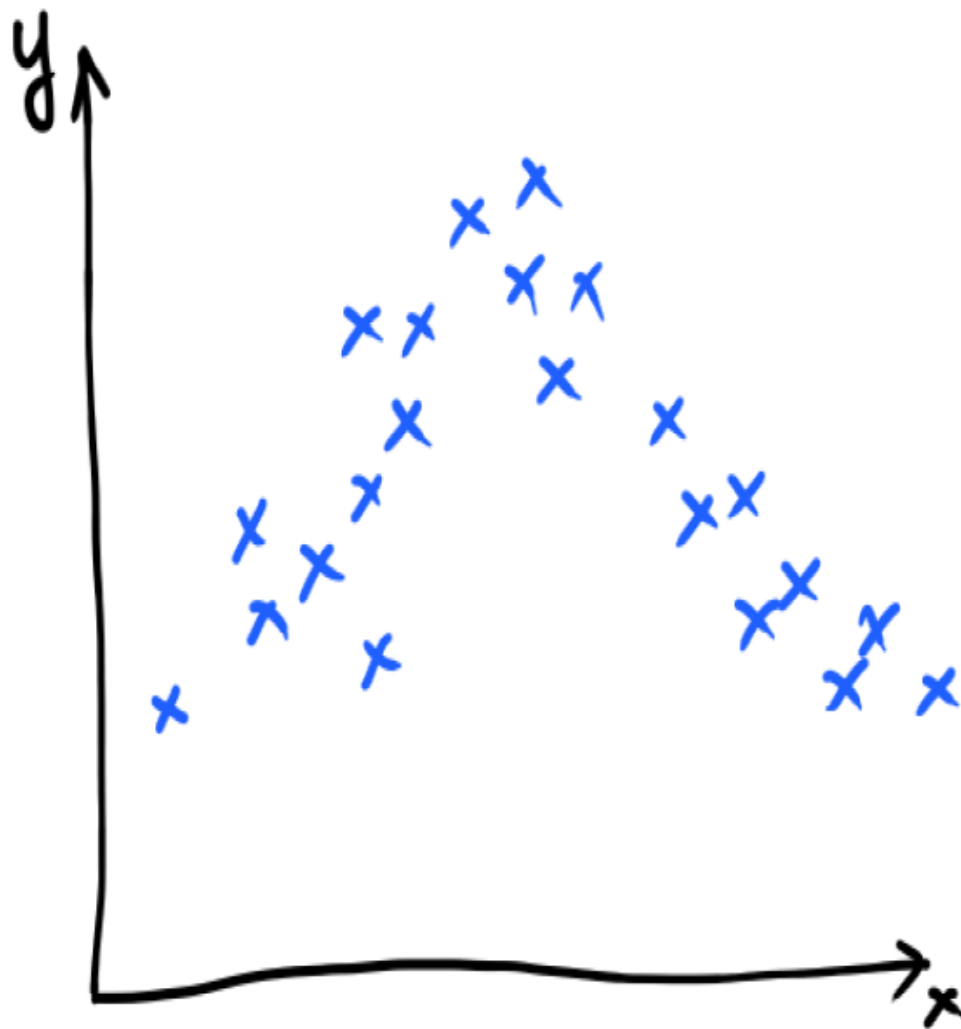
GINI



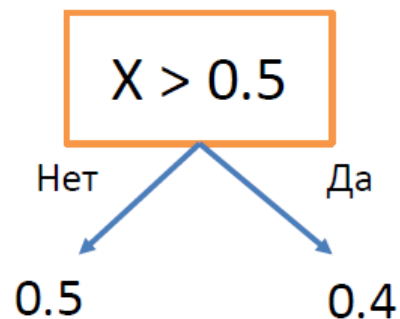
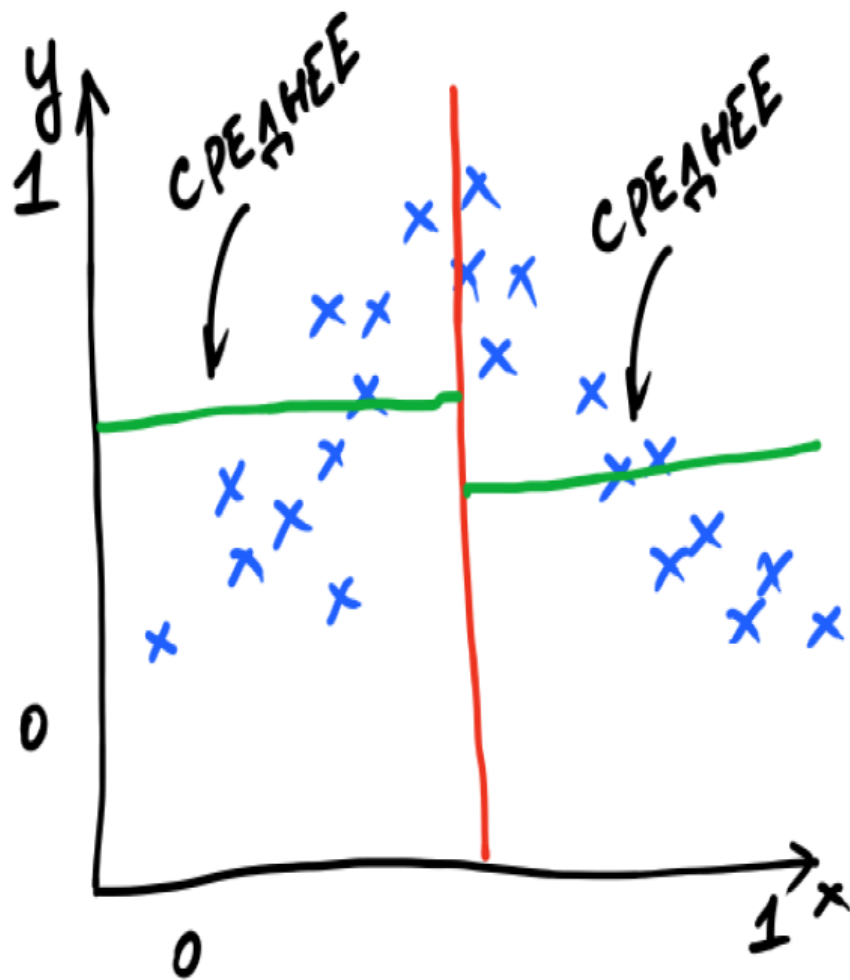
ENTROPY



ПРИМЕР ПОСТРОЕНИЯ РЕШАЮЩЕГО ДЕРЕВА В ЗАДАЧЕ РЕГРЕССИИ

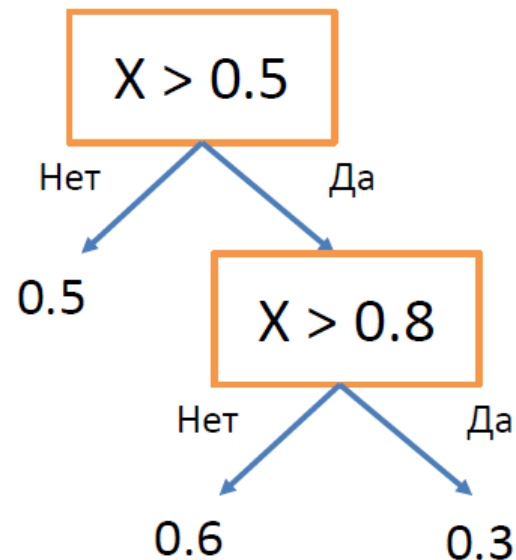
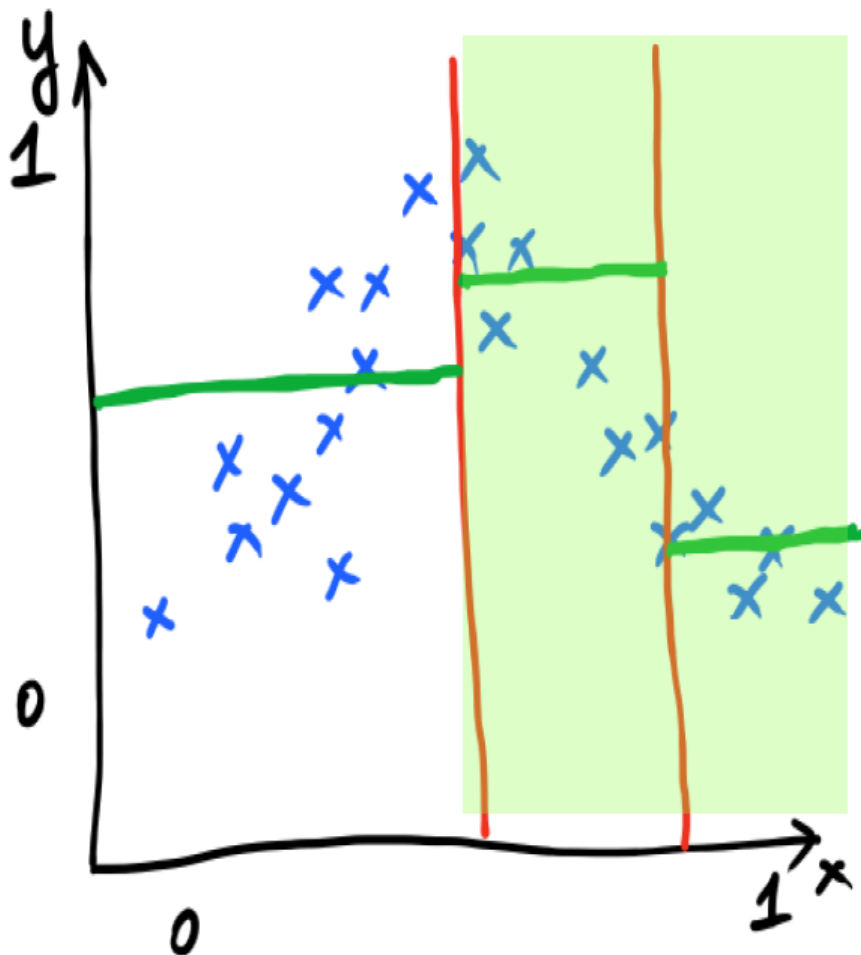


ПРИМЕР ПОСТРОЕНИЯ РЕШАЮЩЕГО ДЕРЕВА В ЗАДАЧЕ РЕГРЕССИИ

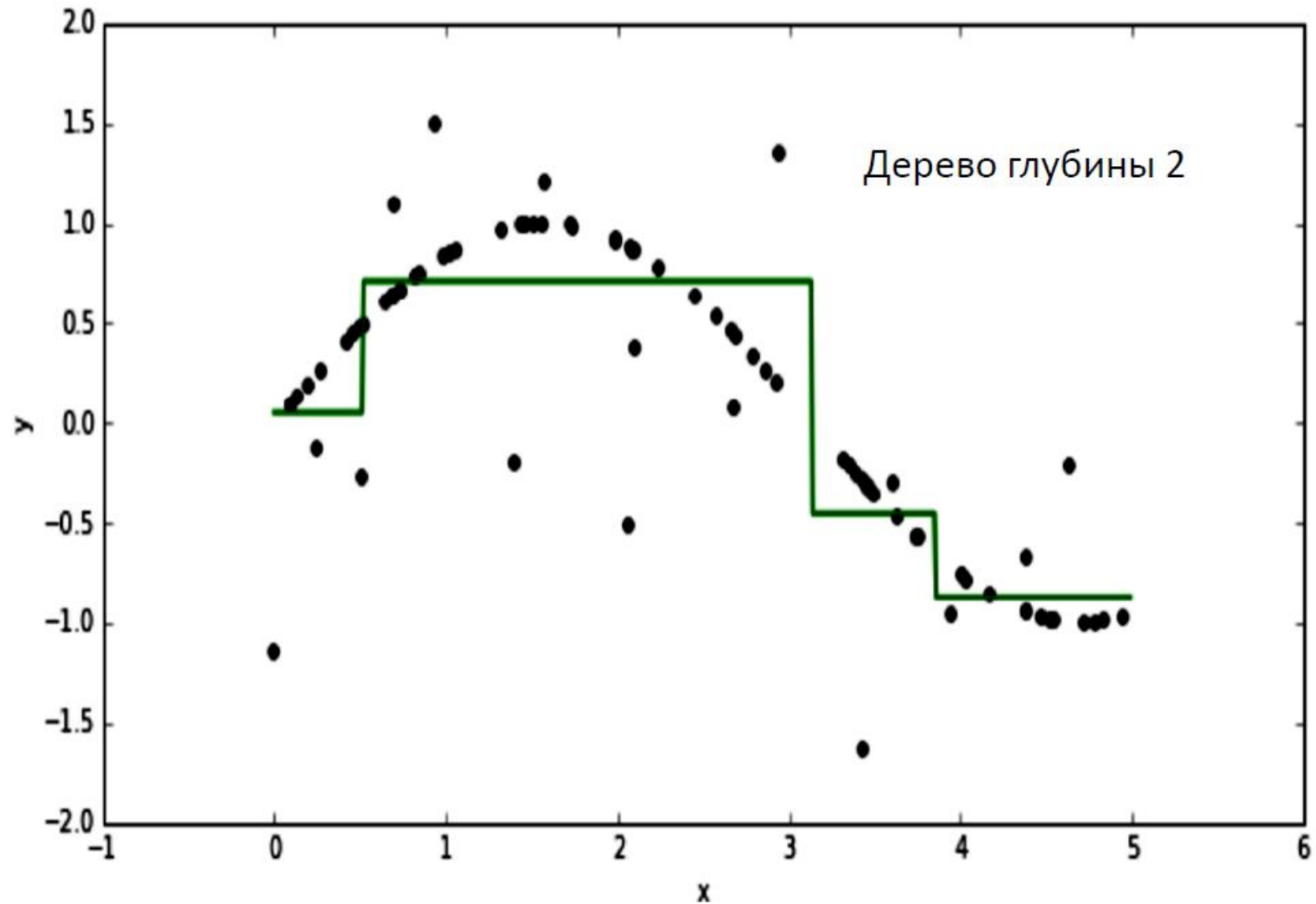


Продолжим
эту ветку

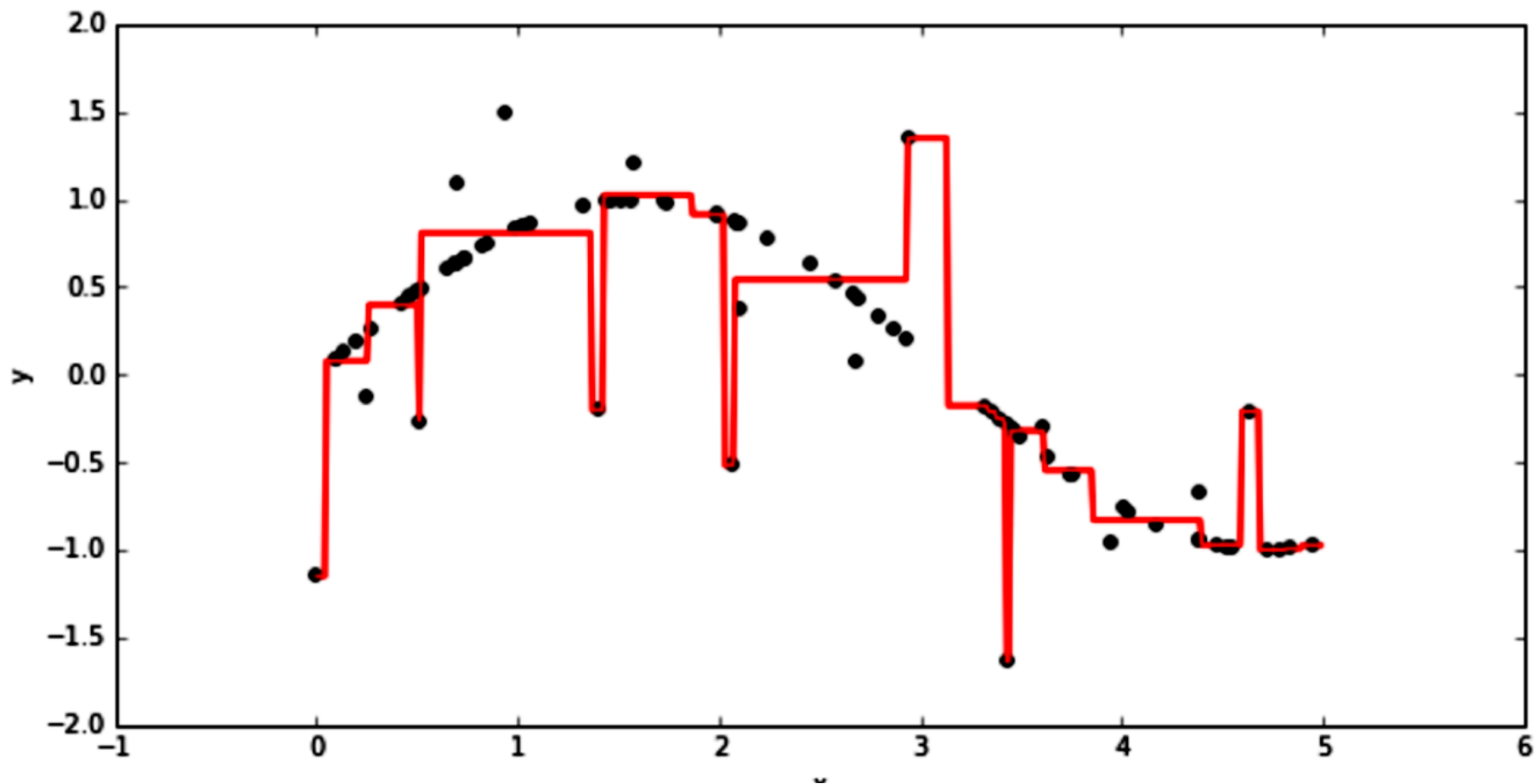
ПРИМЕР ПОСТРОЕНИЯ РЕШАЮЩЕГО ДЕРЕВА В ЗАДАЧЕ РЕГРЕССИИ



ПРИМЕР: ДЕРЕВО ГЛУБИНЫ 2



ПРИМЕР: ДЕРЕВО ГЛУБИНЫ 5



ПЛЮСЫ РЕШАЮЩИХ ДЕРЕВЬЕВ

- Четкие правила классификации (интерпретируемые предикаты, например, “возраст > 25 ”)
- Деревья решений легко визуализируются, то есть хорошо интерпретируются
- Быстро обучаются и выдают прогноз
- Малое число параметров

МИНУСЫ РЕШАЮЩИХ ДЕРЕВЬЕВ

- Очень чувствительны к шумам в данных, модель сильно меняется при небольшом изменении обучающей выборки
- Разделяющая граница имеет свои ограничения (состоит из гиперплоскостей)
- Необходимость борьбы с переобучением (стрижка или какой-либо из критериев останова)
- Проблема поиска оптимального дерева (NP-полная задача, поэтому на практике используется жадное построение дерева)

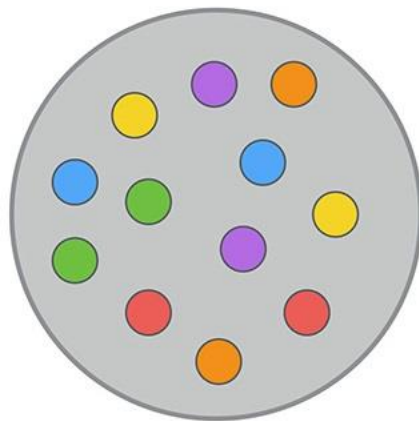
БУТСТРЭП

Дана выборка X .

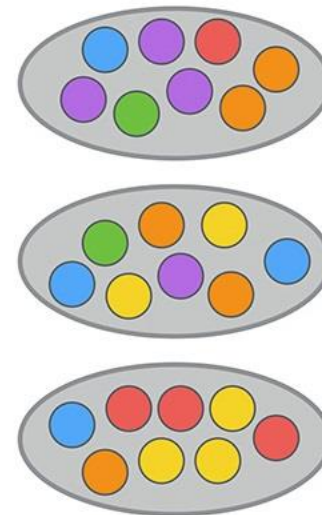
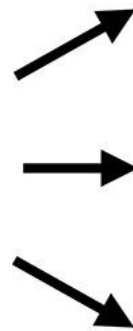
Бутстрэп: равномерно возьмем из выборки X l объектов с возвращением (т.е. в новой выборке будут повторяющиеся объекты). Получим выборку X_1 .

- Повторяем процедуру N раз, получаем выборки X_1, \dots, X_N .

Исходная выборка



Бутстрэп выборки



БЭГГИНГ (BOOTSTRAP AGGREGATION)

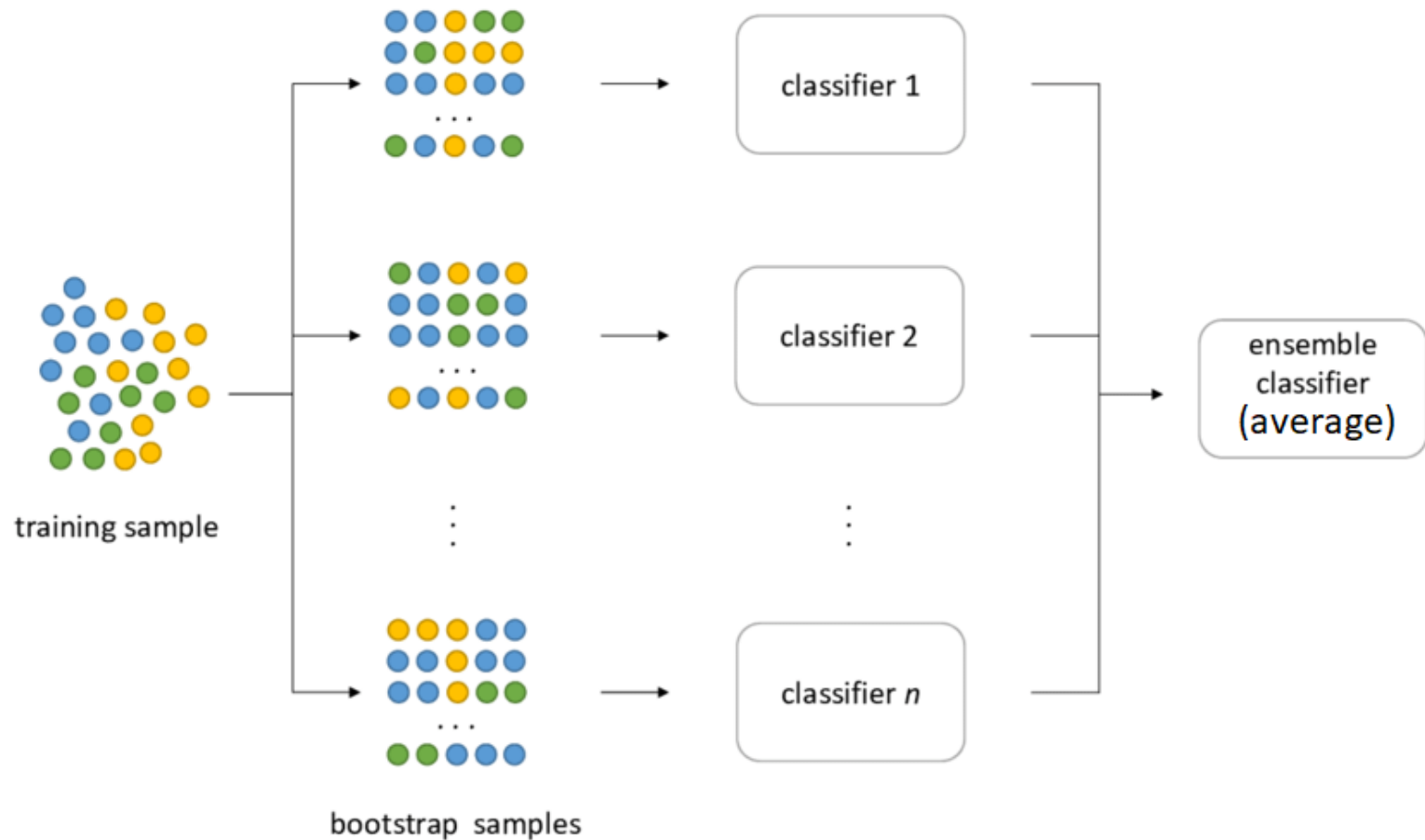
С помощью бутстрэпа мы получили выборки X_1, \dots, X_N .

- Обучим по каждой из них модель – получим базовые алгоритмы $b_1(x), \dots, b_N(x)$.
- Построим новую функцию регрессии:

$$a(x) = \frac{1}{N} \sum_{j=1}^N b_j(x)$$

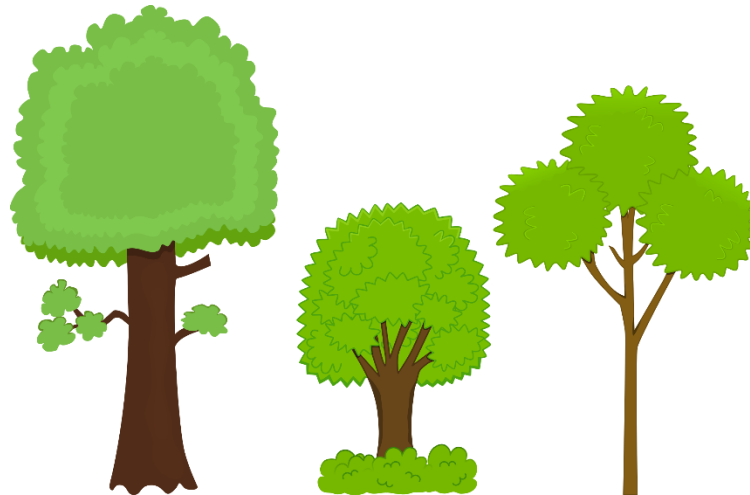
БЭГГИНГ (BOOTSTRAP AGGREGATION)

$$a(x) = \frac{1}{N} \sum_{j=1}^N b_j(x)$$

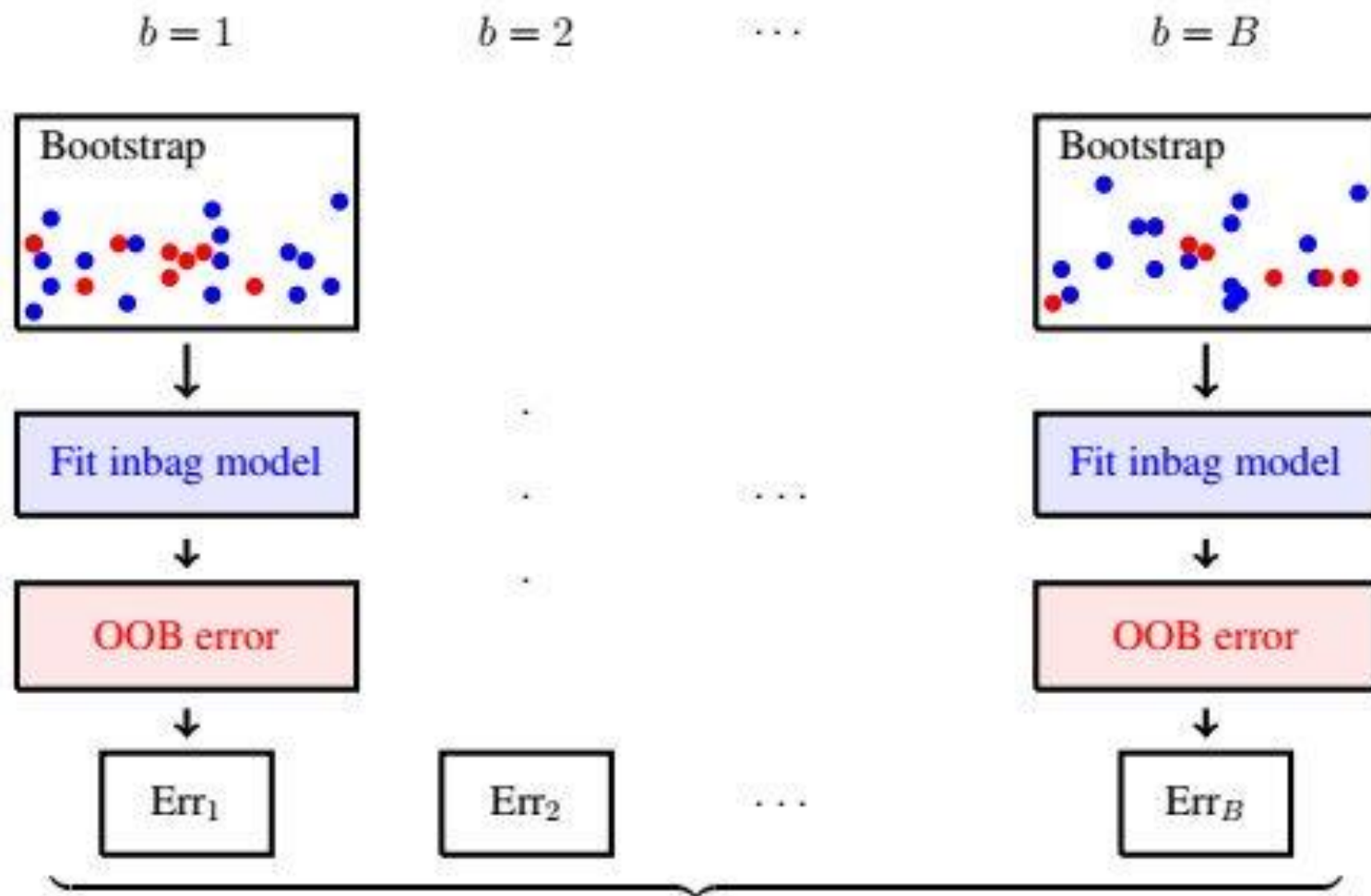


СЛУЧАЙНЫЙ ЛЕС (RANDOM FOREST)

- Возьмем в качестве базовых алгоритмов для бэггинга **решающие деревья**, т.е. каждое случайное дерево $b_i(x)$ построено по своей подвыборке X_i .
- В каждой вершине дерева будем искать **разбиение не по всем признакам, а по подмножеству признаков**.
- Дерево строится до тех пор, пока в листе не окажется n_{min} объектов.



OUT-OF-BAG ОШИБКА

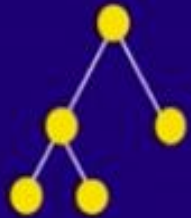


БУСТИНГ

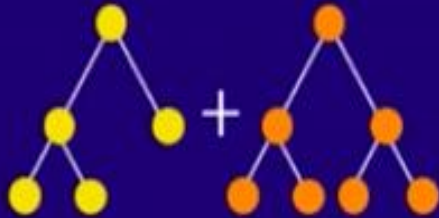
Идея: строим набор алгоритмов, каждый из которых исправляет ошибку предыдущих.

БУСТИНГ

Идея: строим набор алгоритмов, каждый из которых исправляет ошибку предыдущих.



Ошибка



Ошибка



Ошибка

БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

Решаем задачу регрессии с минимизацией квадратичной ошибки:

$$\frac{1}{2} \sum_{i=1}^l (a(x_i) - y_i)^2 \rightarrow \min_a$$

Ищем алгоритм $a(x)$ в виде суммы N базовых алгоритмов:

$$a(x) = \sum_{n=1}^N b_n(x),$$

где базовые алгоритмы $b_n(x)$ принадлежат некоторому семейству A .

БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

Шаг 1: Ищем алгоритм $b_1(x)$, минимизирующий ошибку:

$$b_1(x) = \operatorname{argmin}_{b \in A} \frac{1}{2} \sum_{i=1}^l (b(x_i) - \mathbf{y}_i)^2$$

- Ошибка на объекте x :

$$s = y - b_1(x)$$

БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

Шаг 1: Ищем алгоритм $b_1(x)$, минимизирующий ошибку:

$$b_1(x) = \operatorname{argmin}_{b \in A} \frac{1}{2} \sum_{i=1}^l (b(x_i) - y_i)^2$$

- Ошибка на объекте x :

$$\mathbf{s} = y - b_1(x)$$

Следующий алгоритм должен настраиваться на эту ошибку, т.е. *целевая переменная для следующего алгоритма – это вектор ошибок \mathbf{s}* (а не исходный вектор y)

БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

Шаг 1: Ищем алгоритм $b_1(x)$, минимизирующий ошибку:

$$b_1(x) = \operatorname{argmin}_{b \in A} \frac{1}{2} \sum_{i=1}^l (b(x_i) - y_i)^2$$

Шаг 2: Ищем алгоритм $b_2(x)$, настраивающийся на ошибки s первого алгоритма:

$$b_2(x) = \operatorname{argmin}_{b \in A} \frac{1}{2} \sum_{i=1}^l (b(x_i) - \mathbf{s}_i)^2$$

БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

Шаг 1: Ищем алгоритм $b_1(x)$, минимизирующий ошибку:

$$b_1(x) = \operatorname{argmin}_{b \in A} \frac{1}{2} \sum_{i=1}^l (b(x_i) - y_i)^2$$

Шаг 2: Ищем алгоритм $b_2(x)$, настраивающийся на ошибки s первого алгоритма:

$$b_2(x) = \operatorname{argmin}_{b \in A} \frac{1}{2} \sum_{i=1}^l (b(x_i) - s_i)^2$$

Следующий алгоритм $b_3(x)$ будем выбирать так, чтобы он минимизировал ошибку предыдущей композиции (т.е. $b_1(x) + b_2(x)$) и т.д.

БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

Каждый следующий алгоритм настраиваем на ошибку предыдущих.

Шаг N: Ошибка: $\mathbf{s}_i^{(N)} = y_i - \sum_{n=1}^{N-1} b_n(x_i) = y_i - a_{N-1}(x_i)$

Ищем алгоритм $b_N(x)$:

$$b_N(x) = \operatorname{argmin}_{b \in A} \frac{1}{2} \sum_{i=1}^l \left(b(x_i) - \mathbf{s}_i^{(N)} \right)^2$$

БУСТИНГ: ВЫБОР БАЗОВЫХ АЛГОРИТМОВ

- Если базовые алгоритмы очень простые, то они плохо приближают антиградиент функции потерь, т.е. градиентный бустинг может свестись к случайному блужданию.
- Если базовые алгоритмы сложные, то за несколько шагов бустинг подгонится под обучающую выборку, и получим переобученный алгоритм.