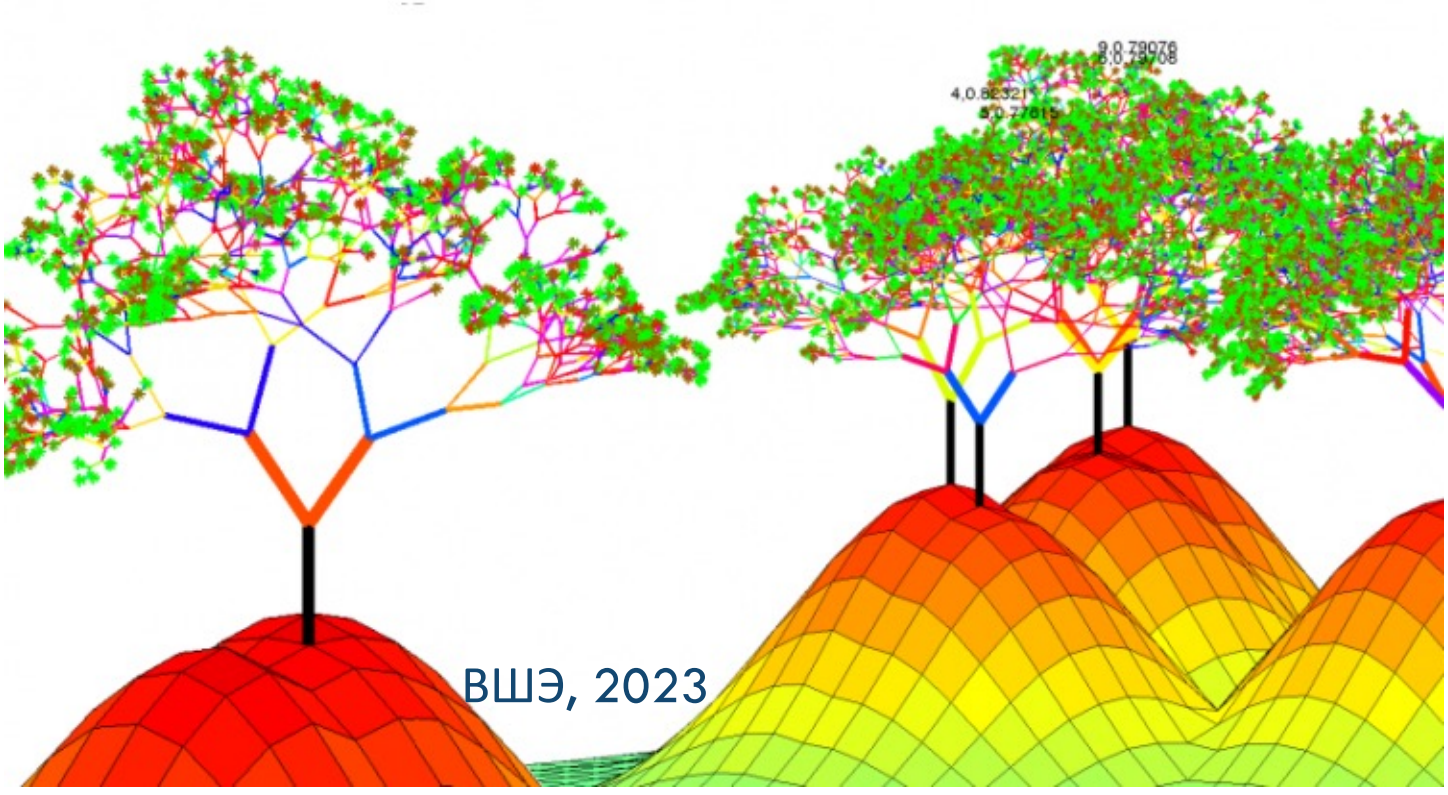
Временные ряды для прогноза криптовалют

Елена Кантонистова



ВШЭ, 2023

СЕГОДНЯ В ПРОГРАММЕ

- Минутка рекламы
- Решающие деревья и их композиции (теория)
- Деревья и их композиции для прогноза временных рядов в Python
- Фреймворки для прогнозирования рядов в Python
- Учет экзогенных признаков

МАГИСТРАТУРА “МАШИННОЕ ОБУЧЕНИЕ И ВЫСОКОНАГРУЖЕННЫЕ СИСТЕМЫ” ФКН ВШЭ

Мы готовим специалистов по анализу данных, которые могут осуществить весь цикл разработки модели

Программа ориентирована на освоение:

- алгоритмов машинного обучения
- подходов к разработке эффективных и отказоустойчивых сервисов

МАГИСТРАТУРА “МАШИННОЕ ОБУЧЕНИЕ И ВЫСОКОНАГРУЖЕННЫЕ СИСТЕМЫ” ФКН ВШЭ

- Длительность обучения: 2 года (4 семестра)
- Обучение – онлайн в одном из двух форматов:
  - синхронный: лекции и семинары проходят в режиме реального времени в Zoom
  - blended: студенты проходят онлайн-курс и в дополнение к нему посещают семинары в Zoom
- В конце обучения: диплом государственного образца

Форма обучения – очная! То есть все привилегии студентов есть: студенческий билет, бесплатные музеи, отсрочка от армии.

МАГИСТРАТУРА “МАШИННОЕ ОБУЧЕНИЕ И ВЫСОКОНАГРУЖЕННЫЕ СИСТЕМЫ” ФКН ВШЭ

Контакты:

- Страница программы **https://www.hse.ru/ma/mlds**
- Почта программы **mlds@hse.ru**
- Телеграм-сообщество абитуриентов и подготовка к поступлению:

БЛИЖАЙШИЕ ИНТЕНСИВЫ СООБЩЕСТВА AI EDU

Линейные модели и их презентация (старт - 9 августа!)

- В курсе вы не только разберете теоретическую сторону ML-моделей, но и узнаете как использовать модели на практике
- Пройдете все этапы задачи ML: от разведочного анализа до получения прогноза и его интерпретации
- Научитесь строить интерактивные дашборды при помощи библиотеки Streamlit и создавать веб-сервисы с использованием фреймворка FastApi

КУРСЫ СООБЩЕСТВА AI EDU

Практический Machine Learning

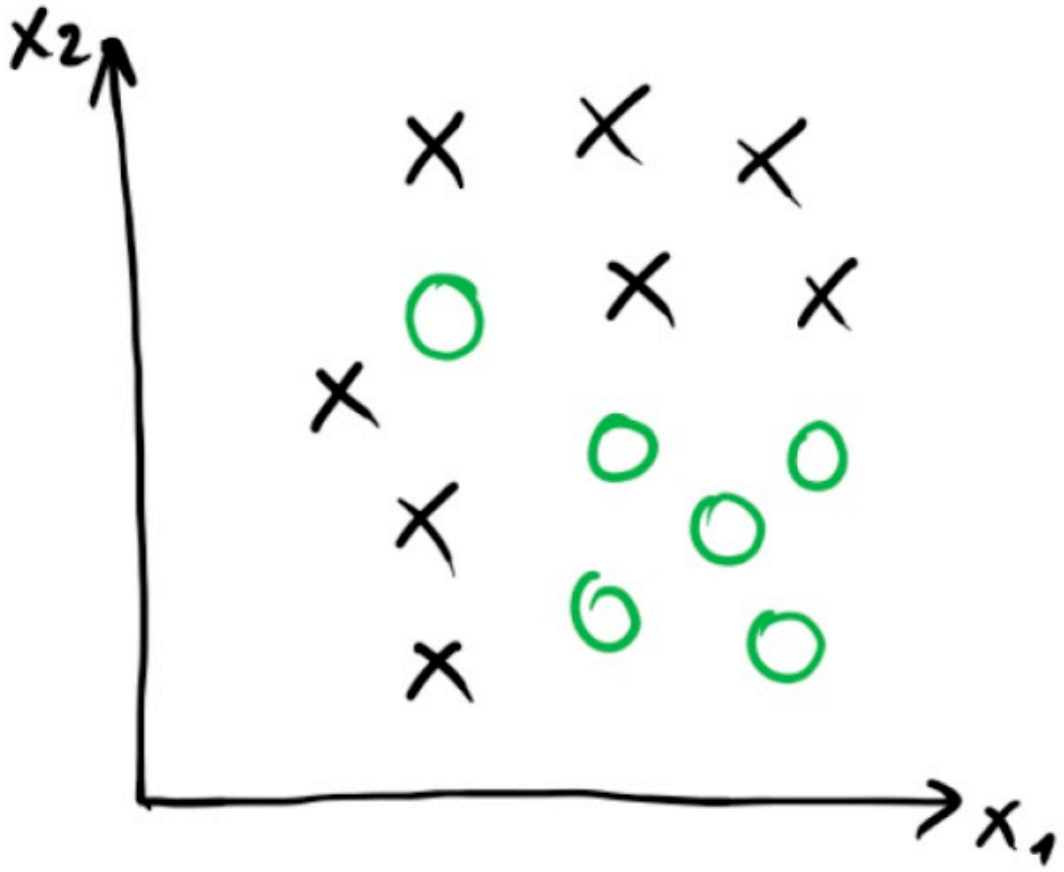
- Курс посвящен изучению всех основных задач и подходов в машинном обучении. Курс дает как теорию методов, так и обширную практику.
- За 16 недель курса вы пройдете путь практически с нуля в машинном обучении до уверенного владения основными алгоритмами и подходами.

Прикладные задачи машинного обучения

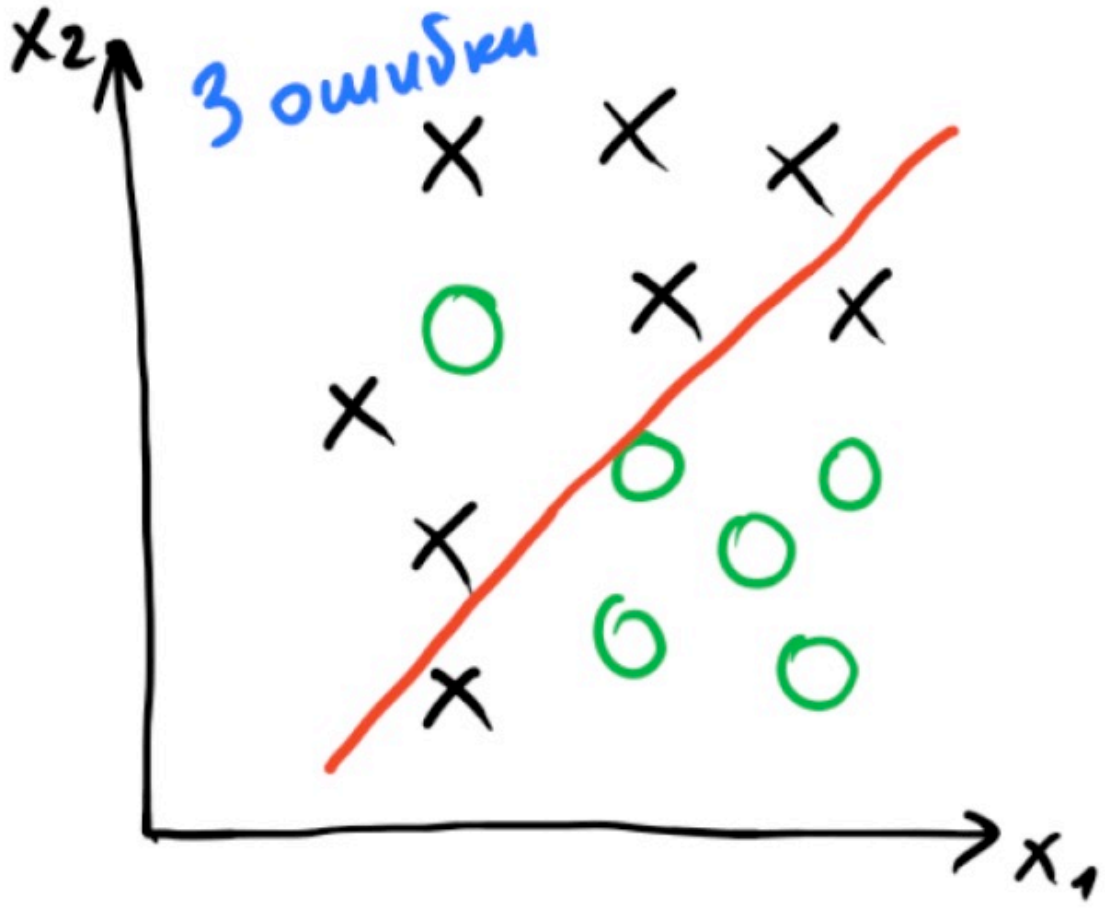
В курсе подробно разобраны следующие темы:

- построение рекомендательных систем
- анализ временных рядов
- продвинутые методы интерпретации ML-моделей
- AutoML

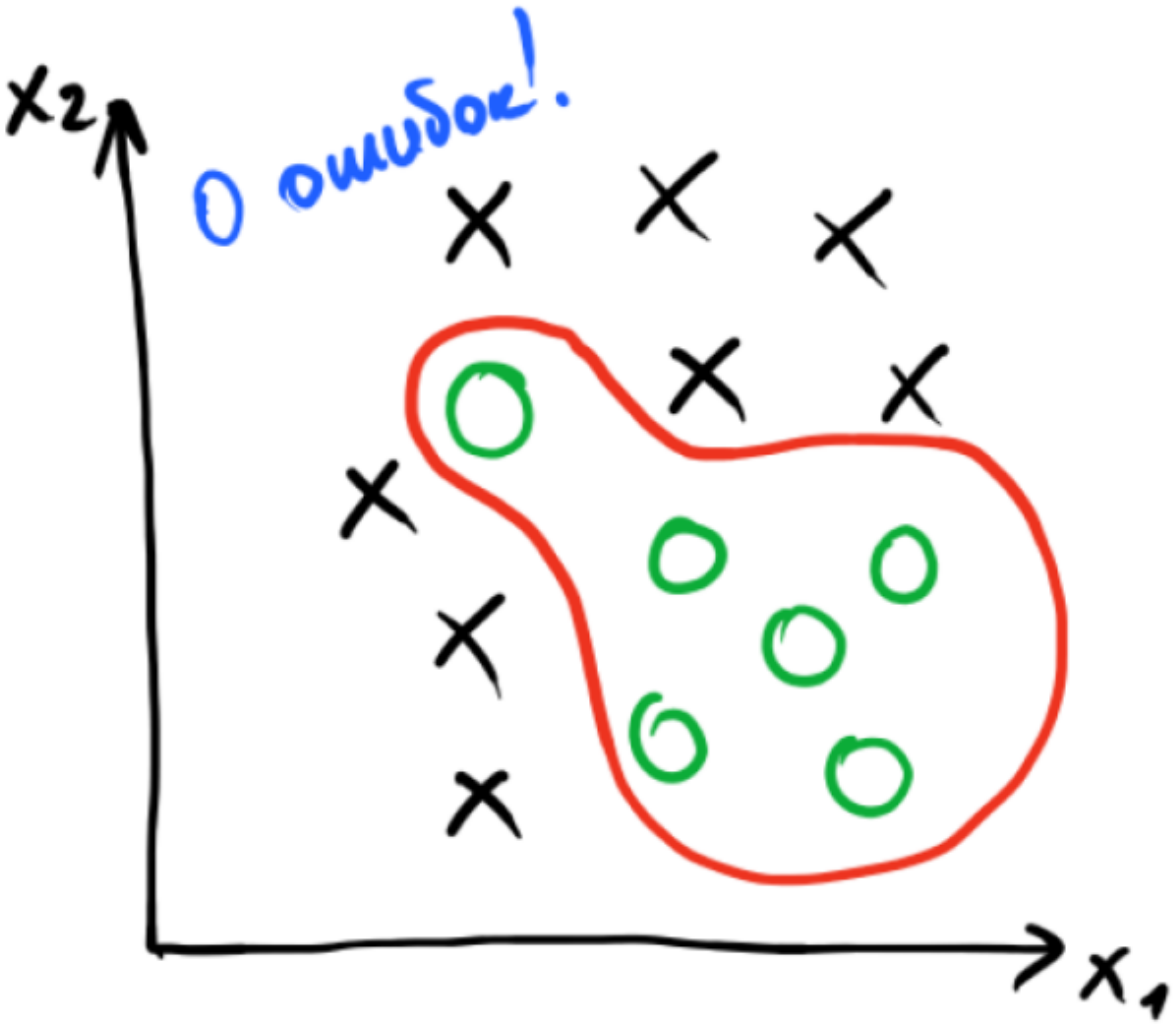
РЕШАЮЩЕЕ ДЕРЕВО



ЛИНЕЙНАЯ МОДЕЛЬ



НЕЛИНЕЙНЫЙ АЛГОРИТМ



РЕШАЮЩЕЕ ДЕРЕВО

Решающее дерево – это бинарное дерево, в котором:

1) каждой вершине v приписана функция (предикат) $\beta_v: X \to \{0,1\}$

РЕШАЮЩЕЕ ДЕРЕВО

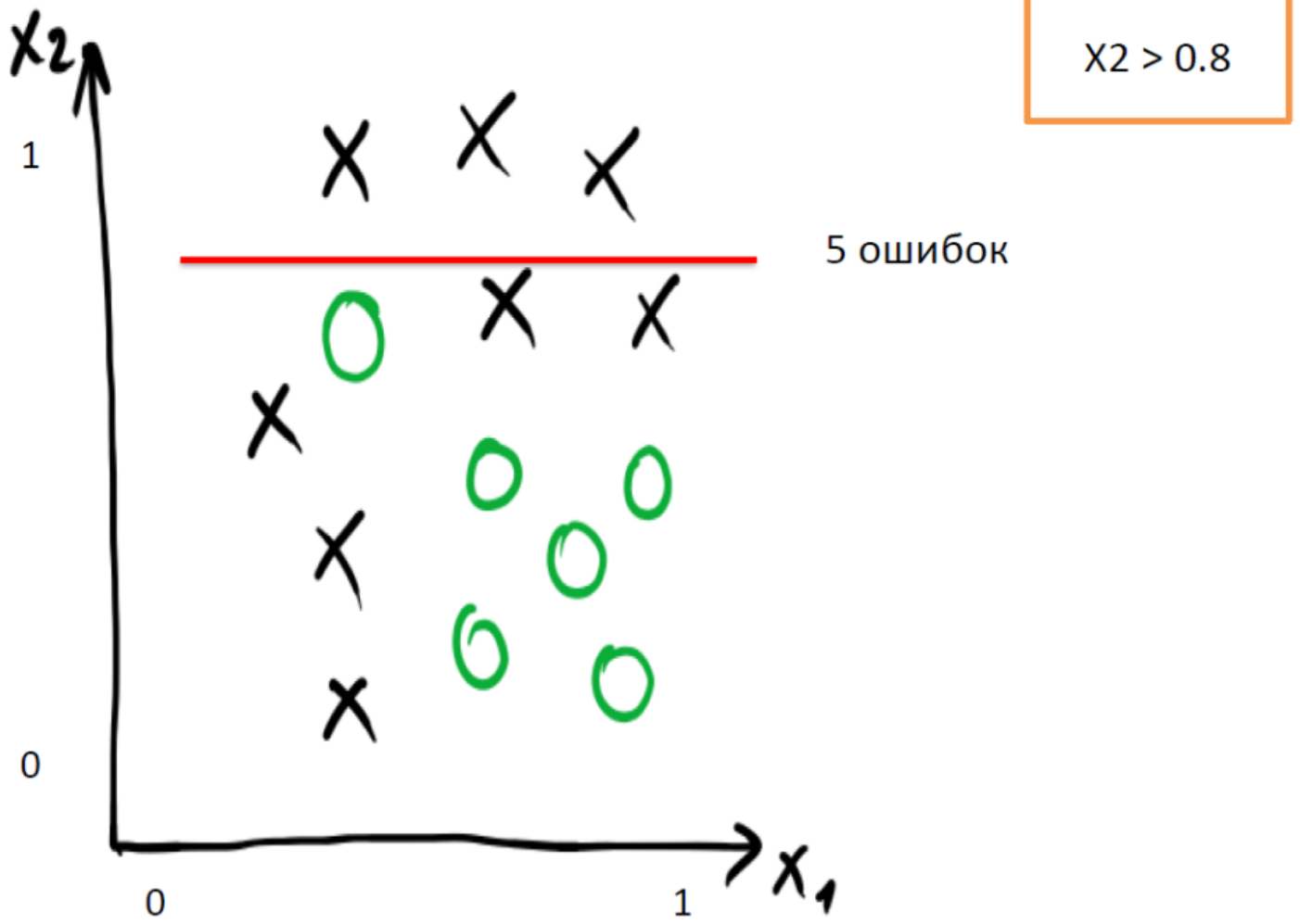
Решающее дерево – это бинарное дерево, в котором:

1) каждой вершине v приписана функция (предикат) $\beta_v: X \to \{0,1\}$

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

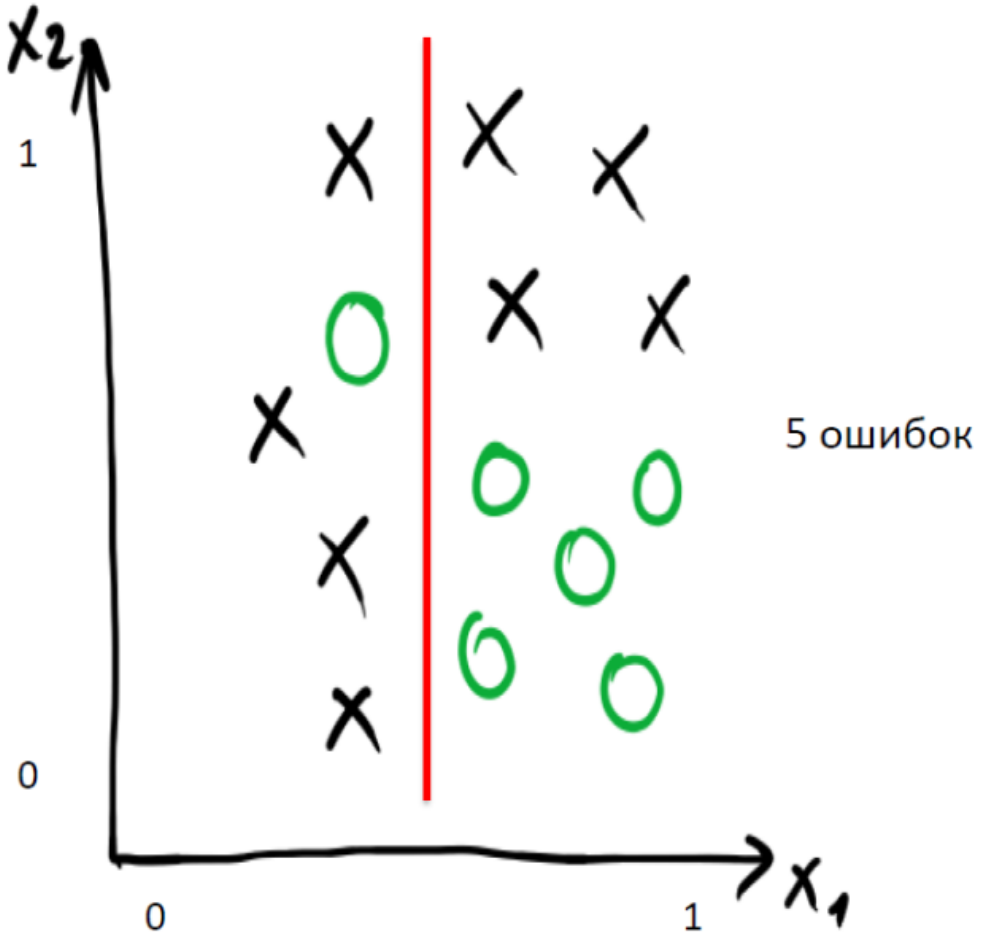
ПРИМЕР

- Жадно найдем наилучший предикат



ПРИМЕР

- Жадно найдем наилучший предикат

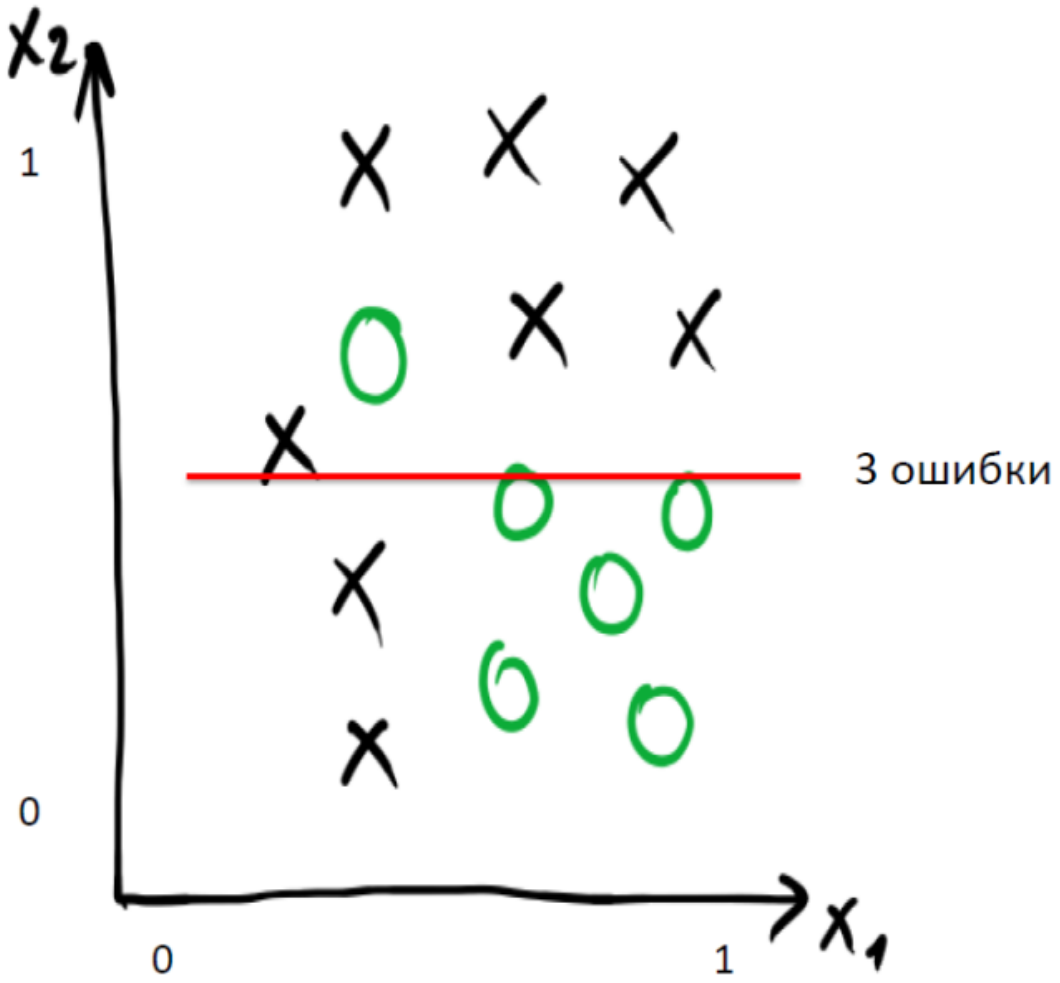


X1 > 0.5

5 ошибок

ПРИМЕР

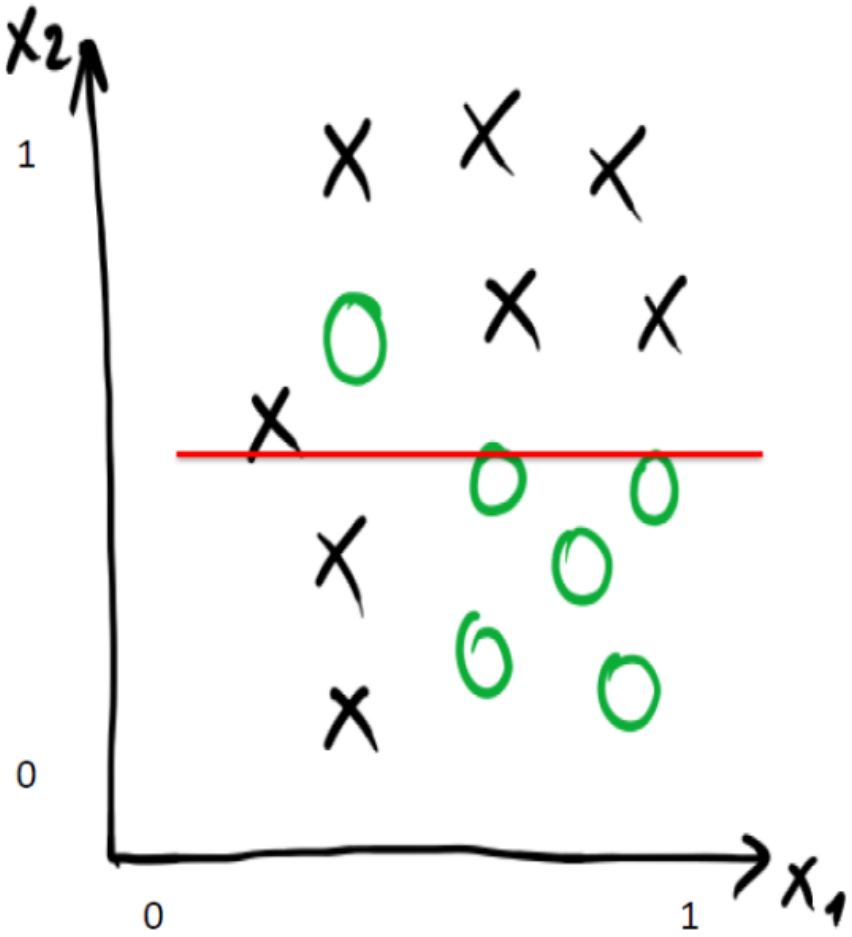
- Жадно найдем наилучший предикат

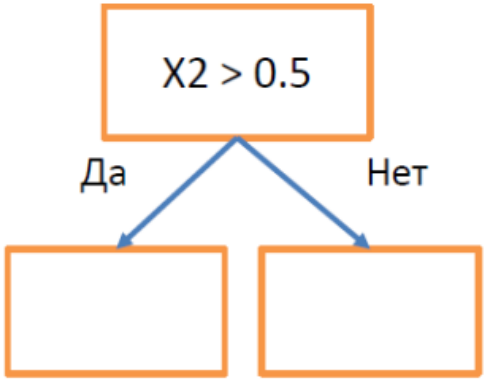


X2 > 0.5

ПРИМЕР

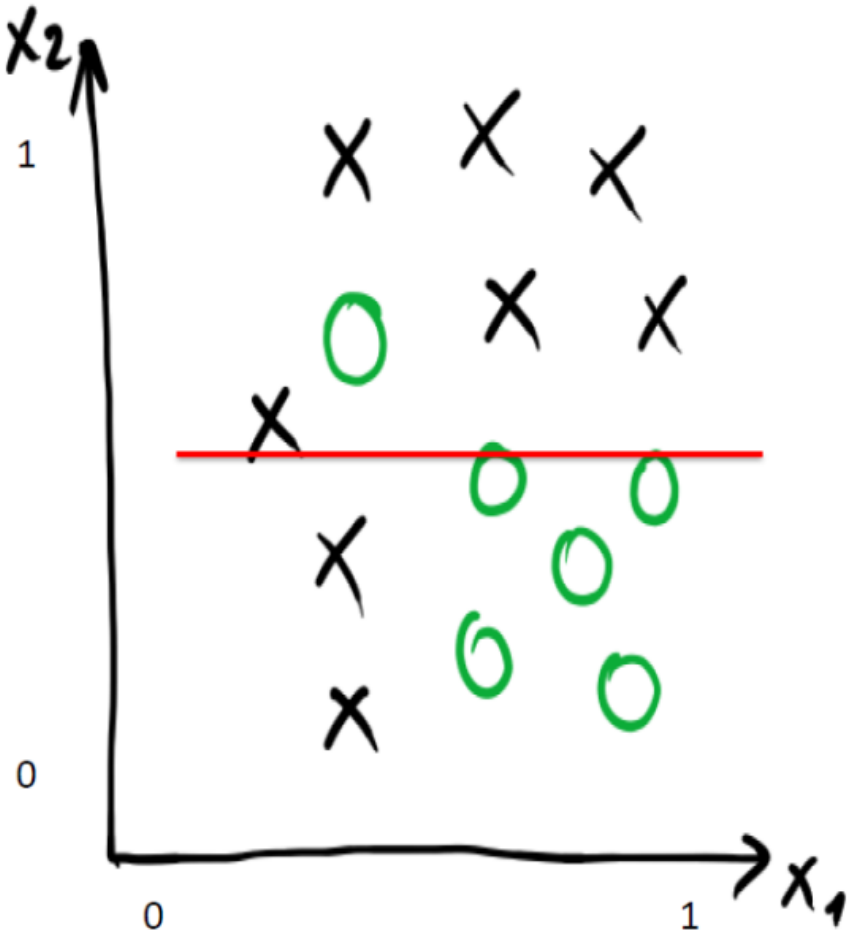
- Нашли лучшее первое ветвление

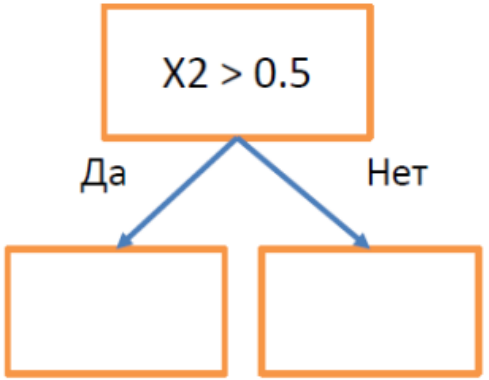




ПРИМЕР

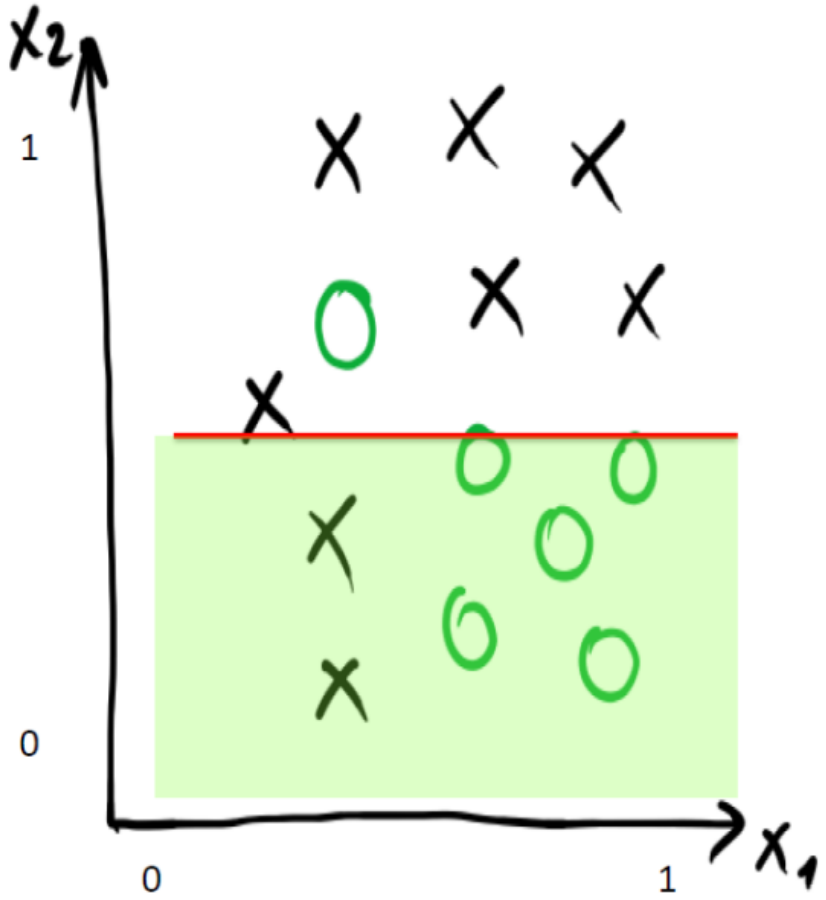
- Нашли лучшее первое ветвление

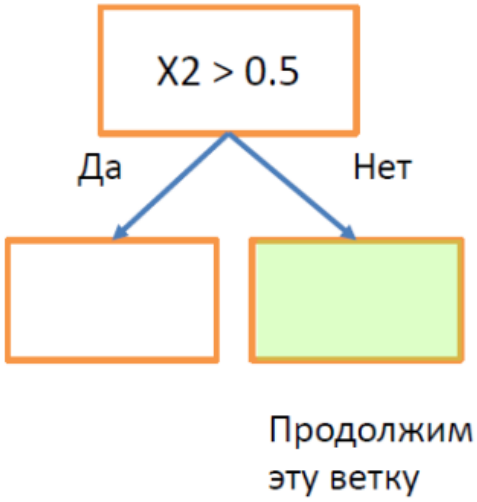




ПРИМЕР

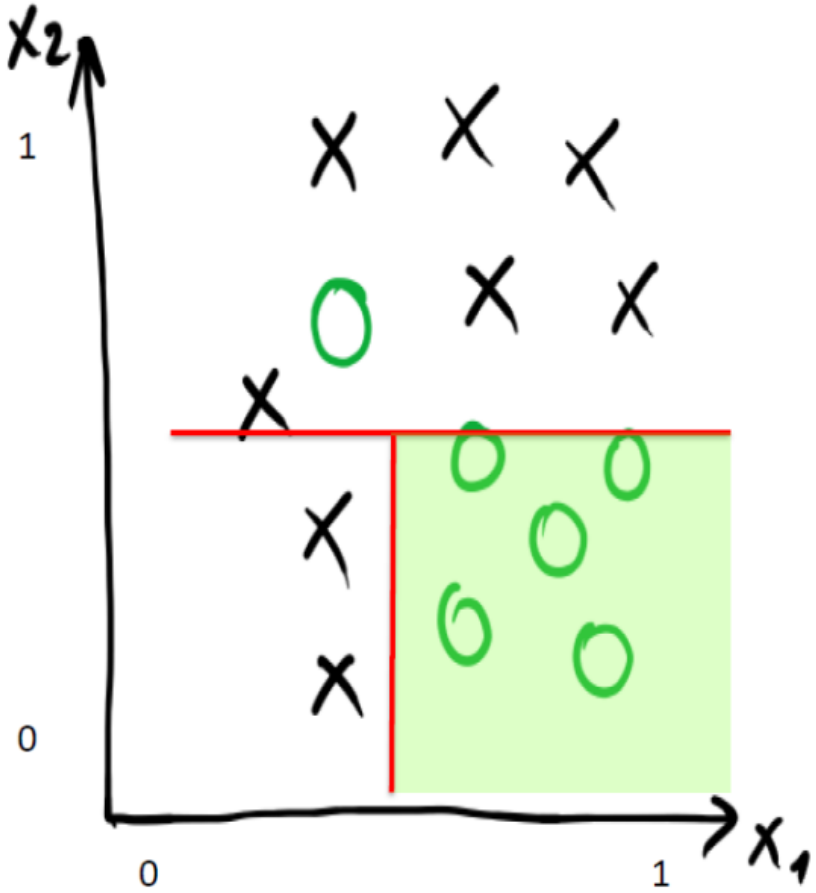
- Нашли лучшее первое ветвление

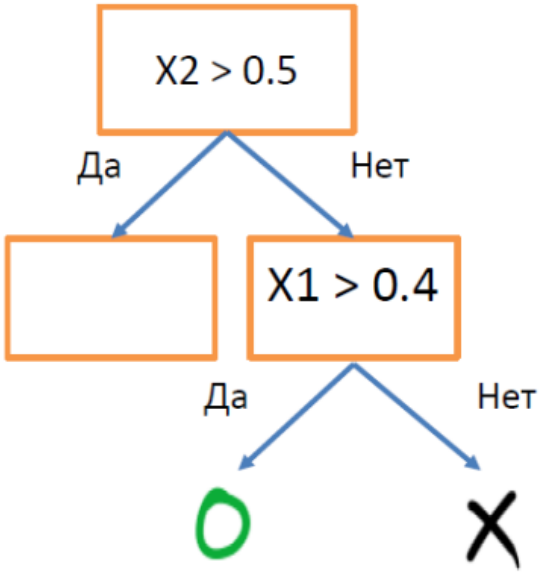




ПРИМЕР

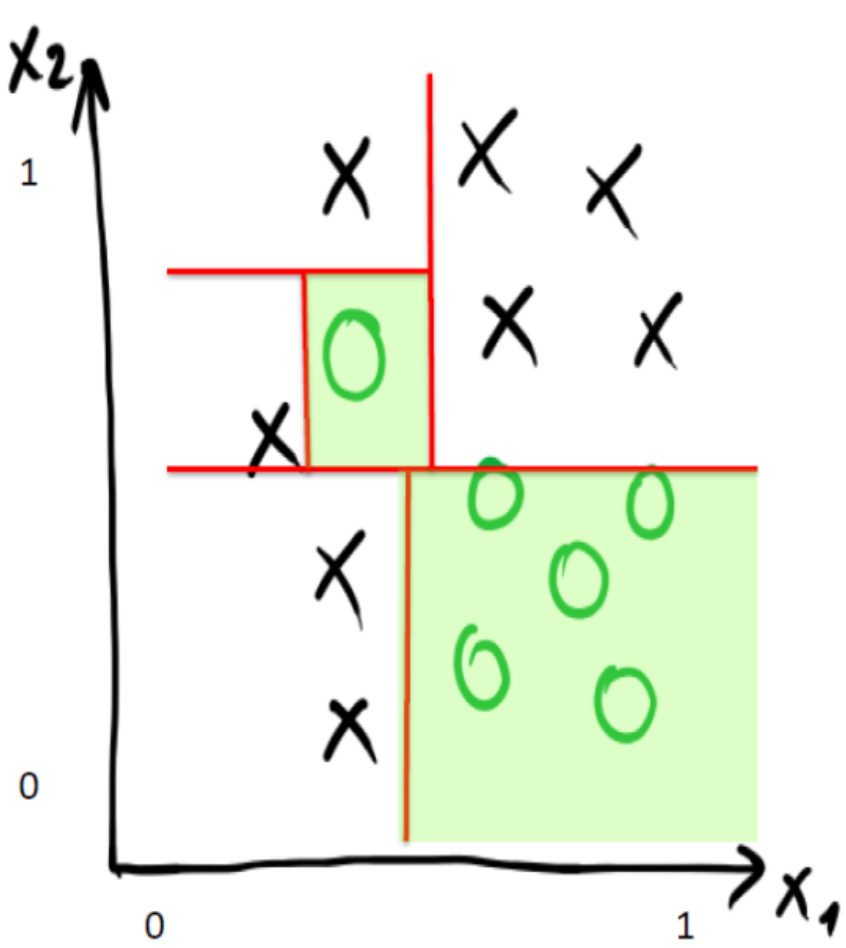
- Нашли лучшее второе ветвление

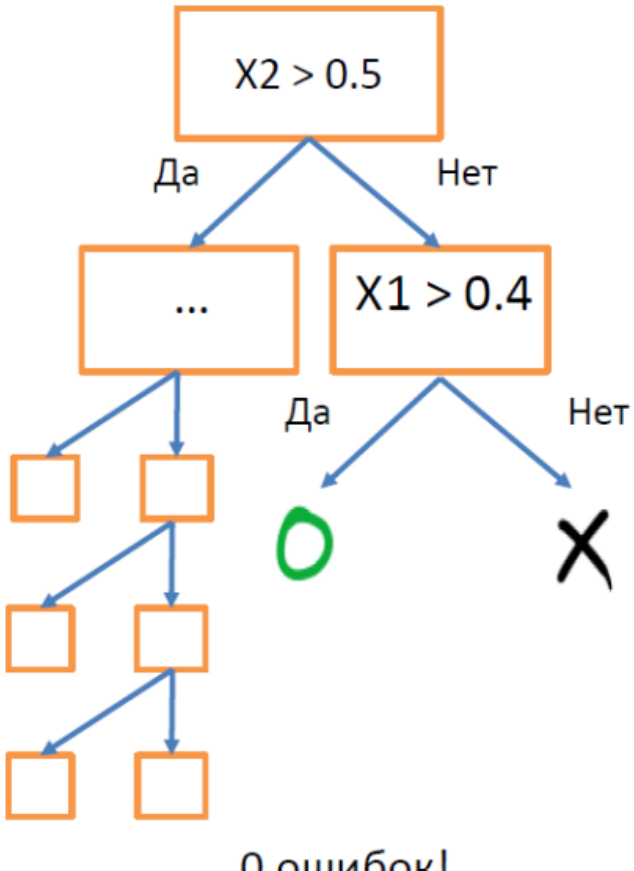




ПРИМЕР

- Построили всё дерево





ПЕРЕОБУЧЕНИЕ

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

РАЗЛОЖЕНИЕ ОШИБКИ (BIAS-VARIANCE DECOMPOSITION)

Зачастую для улучшения качества модели необходимо понять, из-за чего возникает ошибка в предсказаниях.

Утверждение: *ошибку модели* $a(x)$ *можно представить в виде*

$$\mathbf{Err(x) = Bias^2\big(a(x)\big) + Var\big(a(x)\big) + \sigma^2.}$$

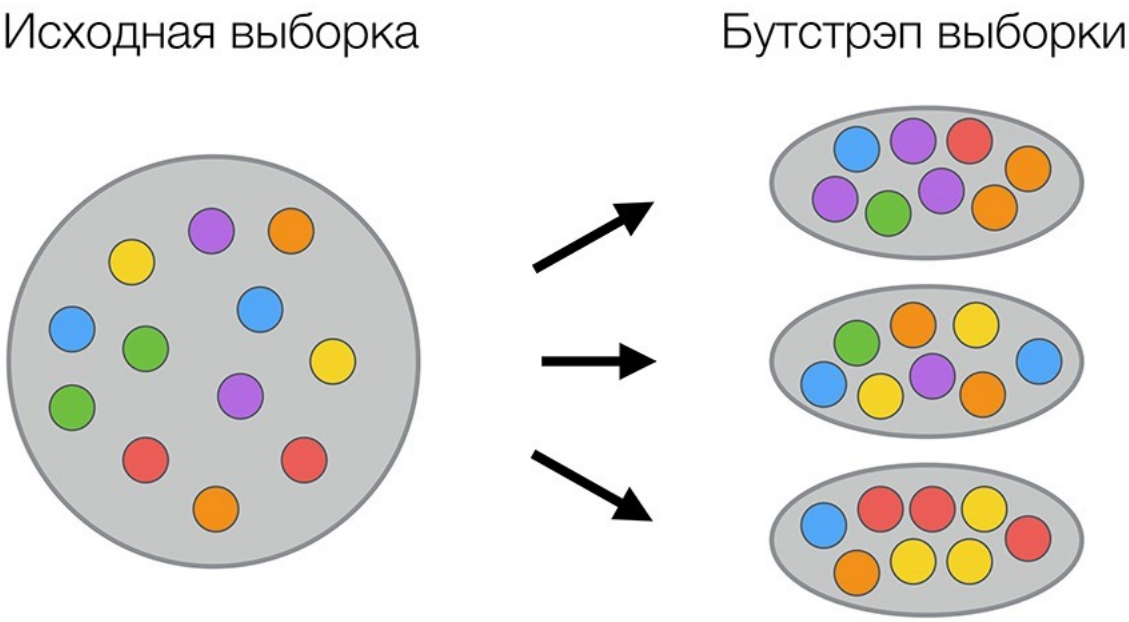
- **Bias** - средняя ошибка по всем возможным наборам данных – **смещение**. Показывает как хорошо модель предсказывает целевую переменную.
- **Var** - дисперсия ошибки, т.е. как сильно различается ошибка при обучении на различных наборах данных – **разброс**. Большой разброс = сильное переобучение.
- $\boldsymbol{\sigma^2}$ - неустранимая ошибка – **шум**.

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 $X_1, \ldots, X_N$.



БЭГГИНГ (BOOTSTRAP AGGREGATION)

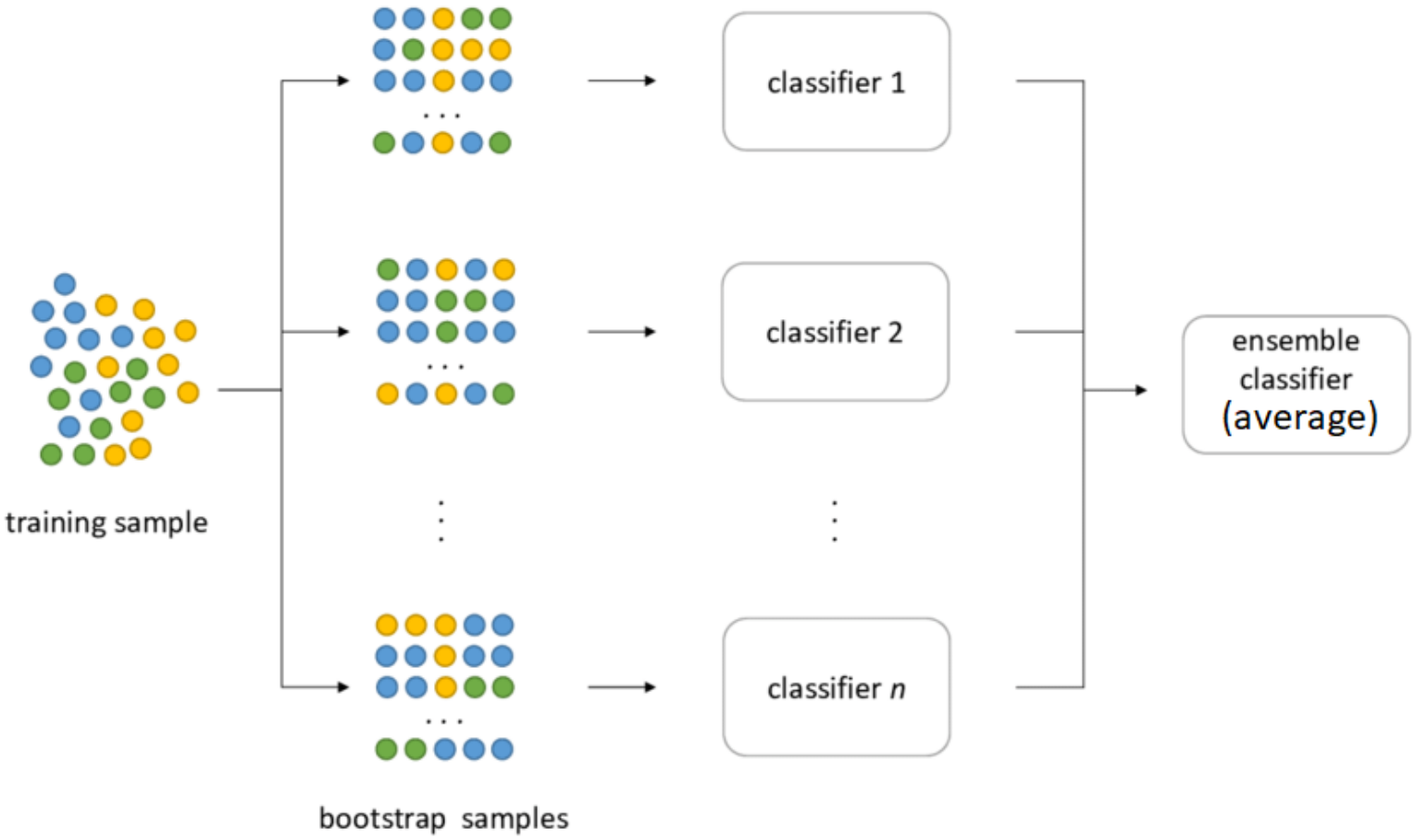
С помощью бутстрэпа мы получили выборки $X_1, \dots, X_N$.

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

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

БЭГГИНГ (BOOTSTRAP AGGREGATION)

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



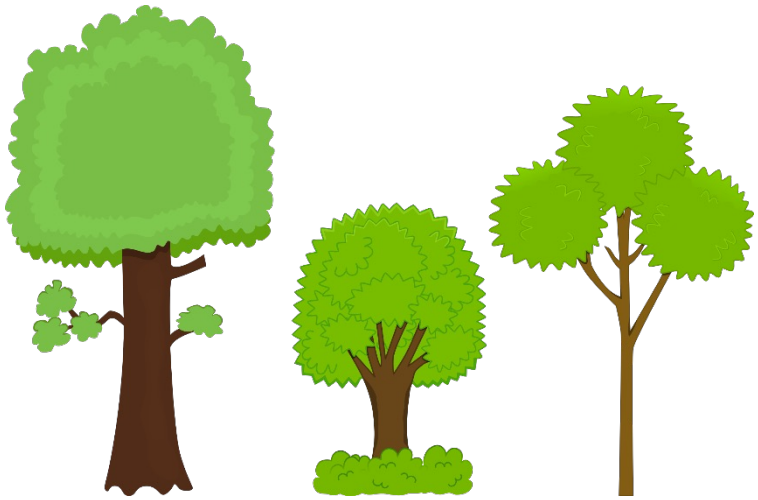
СМЕЩЕНИЕ И РАЗБРОС У БЭГГИНГА

1) ***Смещение у бэггинга равно смещению одного базового алгоритма.***

2) Если базовые алгоритмы некоррелированы, то ***разброс бэггинга $a_N(x)$ в N раз меньше разброса отдельных базовых алгоритмов.***

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***.

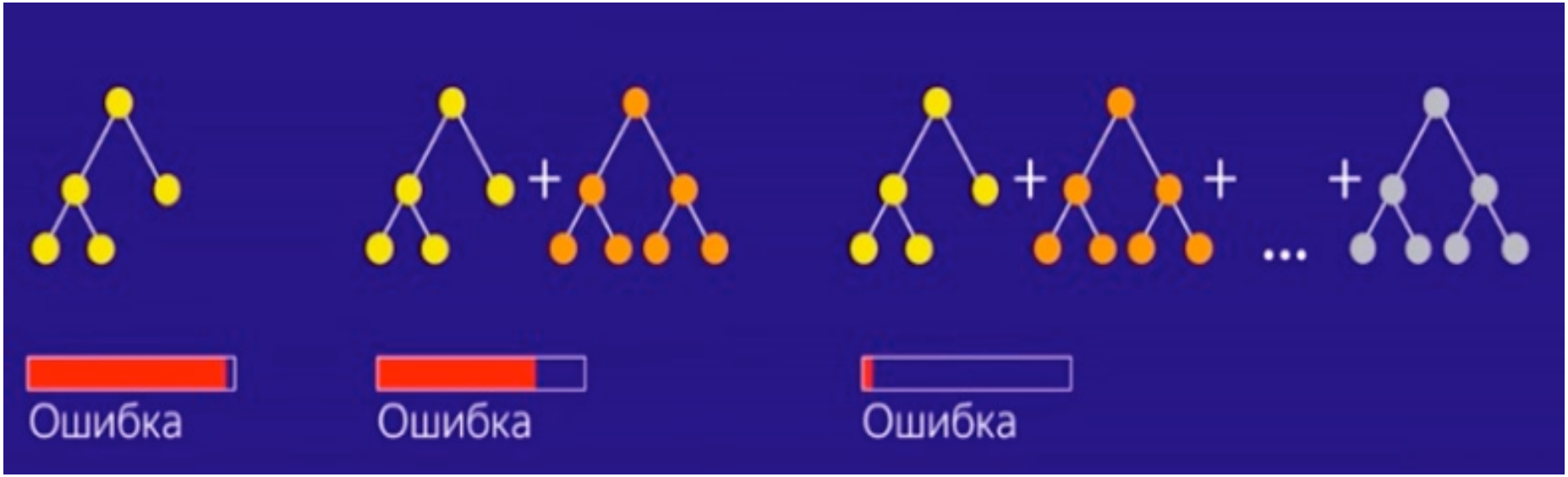


БУСТИНГ

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

БУСТИНГ

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



БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

Решаем задачу регрессии с минимизацией квадратичной ошибки:

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

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:

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

- Ошибка на объекте x:

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

БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

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

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

- Ошибка на объекте x:

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

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

БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

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

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

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

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

БУСТИНГ В ЗАДАЧЕ РЕГРЕССИИ

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

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

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

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

Следующий алгоритм $\mathrm{b}_3(\mathrm{x})$ будем выбирать так, чтобы он минимизировал ошибку предыдущей композиции (т.е. $\mathrm{b}_1(\mathrm{x}) + \mathrm{b}_2(\mathrm{x})$) и т.д.

СМЕЩЕНИЕ И РАЗБРОС БУСТИНГА

- Бустинг целенаправленно уменьшает ошибку, т.е. ***смещение у него маленькое***.
- Алгоритм получается сложным, поэтому ***разброс может быть большим***.

Значит, чтобы не переобучиться, в качестве базовых алгоритмов надо брать неглубокие деревья (глубины 3-6).