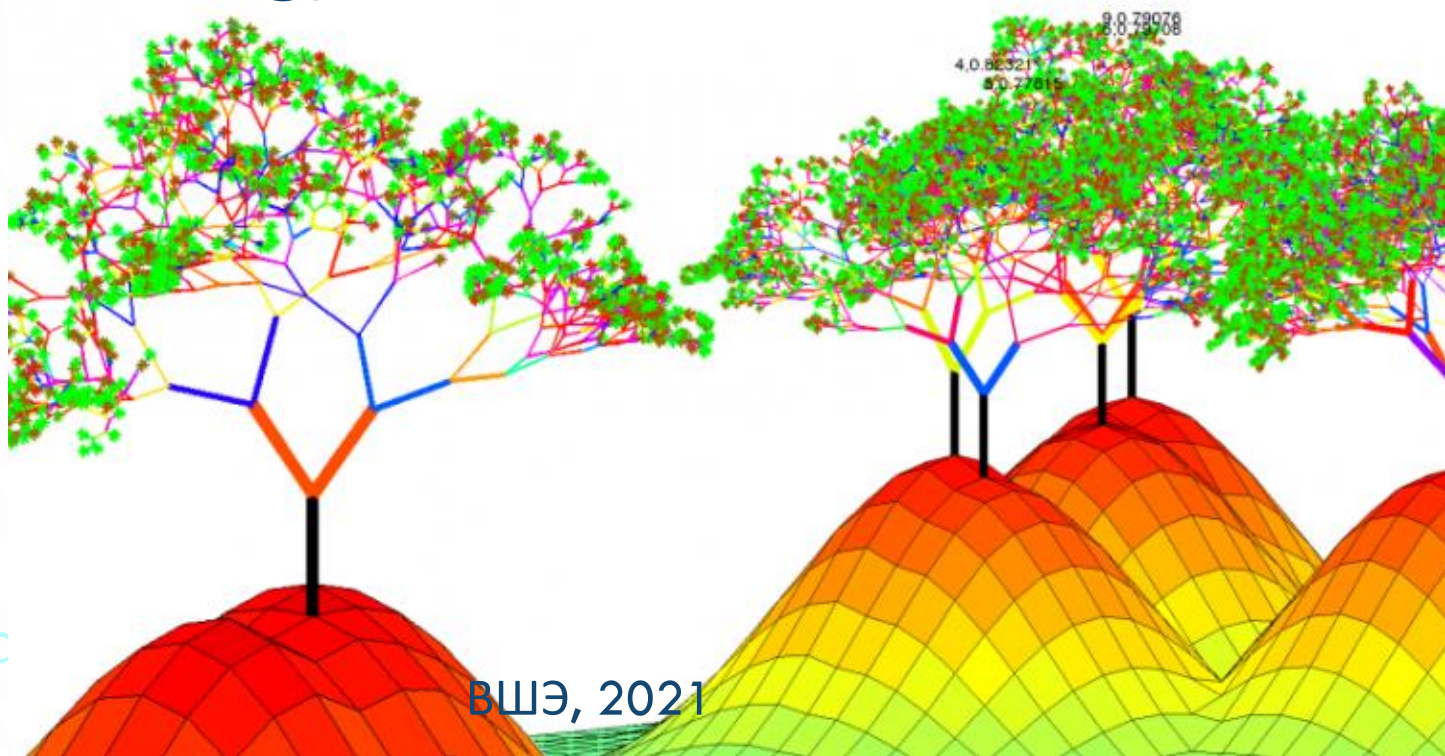


# Занятие 10

## Бустинг.

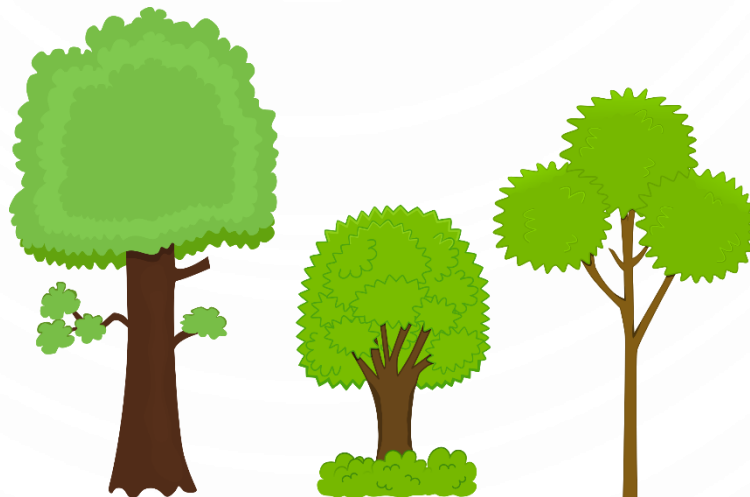
Елена Кантонистова

[elena.kantonistova@yandex.ru](mailto:elena.kantonistova@yandex.ru)



# СЛУЧАЙНЫЙ ЛЕС (RANDOM FOREST)

- Возьмем в качестве базовых алгоритмов для бэггинга **решающие деревья**, т.е. каждое случайное дерево  $b_i(x)$  построено по своей бутстрепной подвыборке  $X_i$ .
- В каждой вершине дерева будем искать **разбиение не по всем признакам, а по подмножеству признаков**.
- Итоговая композиция имеет вид  $a(x) = \frac{1}{N} \sum_{j=1}^N b_j(x)$ .



# СМЕЩЕНИЕ И РАЗБРОС У БЭГГИНГА

**Утверждение.**

1) **Бэггинг не ухудшает смещенность модели, т.е. смещение  $a_N(x)$  равно смещению одного базового алгоритма.**

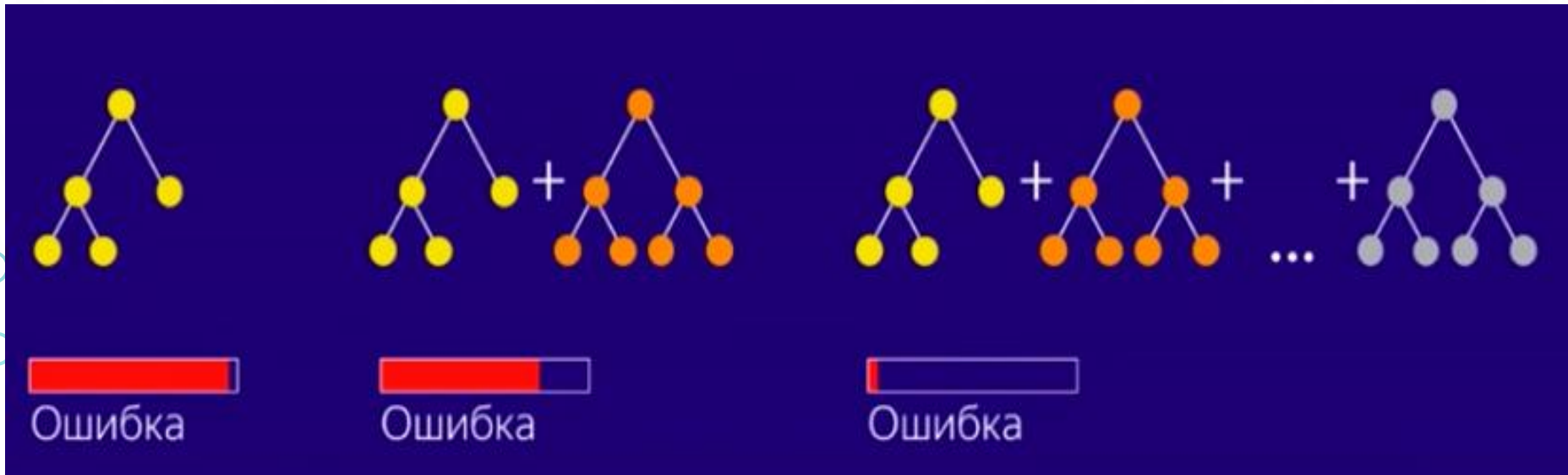
2) **Если базовые алгоритмы некоррелированы, то дисперсия бэггинга  $a_N(x)$  в  $N$  раз меньше дисперсии отдельных базовых алгоритмов.**

# 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:

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

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 В ЗАДАЧЕ РЕГРЕССИИ

Каждый следующий алгоритм настраиваем на ошибку предыдущих.

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

Ищем алгоритм  $b_N(x)$ :

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

**Утверждение.** Ошибка на  $N$ -м шаге – это антиградиент функции потерь по ответу модели, вычисленный в точке ответа уже построенной композиции:

$$s_i^{(N)} = y_i - a_{N-1}(x_i) = - \frac{\partial}{\partial z} \frac{1}{2} (z - y_i)^2 \Big|_{z=a_{N-1}(x_i)}$$

# ГРАДИЕНТНЫЙ БУСТИНГ

Пусть  $L(y, z)$  – произвольная дифференцируемая функция потерь. Строим алгоритм  $a_N(x)$  вида

$$a_L(x) = \sum_{n=1}^L \gamma_n b_n(x)$$

# ГРАДИЕНТНЫЙ БУСТИНГ

Пусть  $L(y, z)$  – произвольная дифференцируемая функция потерь. Строим алгоритм  $a_N(x)$  вида

$$a_L(x) = \sum_{n=1}^L \gamma_n b_n(x),$$

где на  $N$ -м шаге

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

$$s_i^{(N)} = y_i - a_{N-1}(x_i)?$$

# ГРАДИЕНТНЫЙ БУСТИНГ

Пусть  $L(y, z)$  – произвольная дифференцируемая функция потерь. Строим алгоритм  $a_N(x)$  вида

$$a_L(x) = \sum_{n=1}^L \gamma_n b_n(x),$$

где на  $N$ -м шаге

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

$$\cancel{s_i^{(N)} = y_i - a_{N-1}(x_i)} \quad s_i^{(N)} = -\frac{\partial L}{\partial z}$$

# ГРАДИЕНТНЫЙ БУСТИНГ

Пусть  $L(y, z)$  – произвольная дифференцируемая функция потерь.  
Строим алгоритм  $a_N(x)$  вида

$$a_L(x) = \sum_{n=1}^L \gamma_n b_n(x),$$

где на  $N$ -м шаге

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

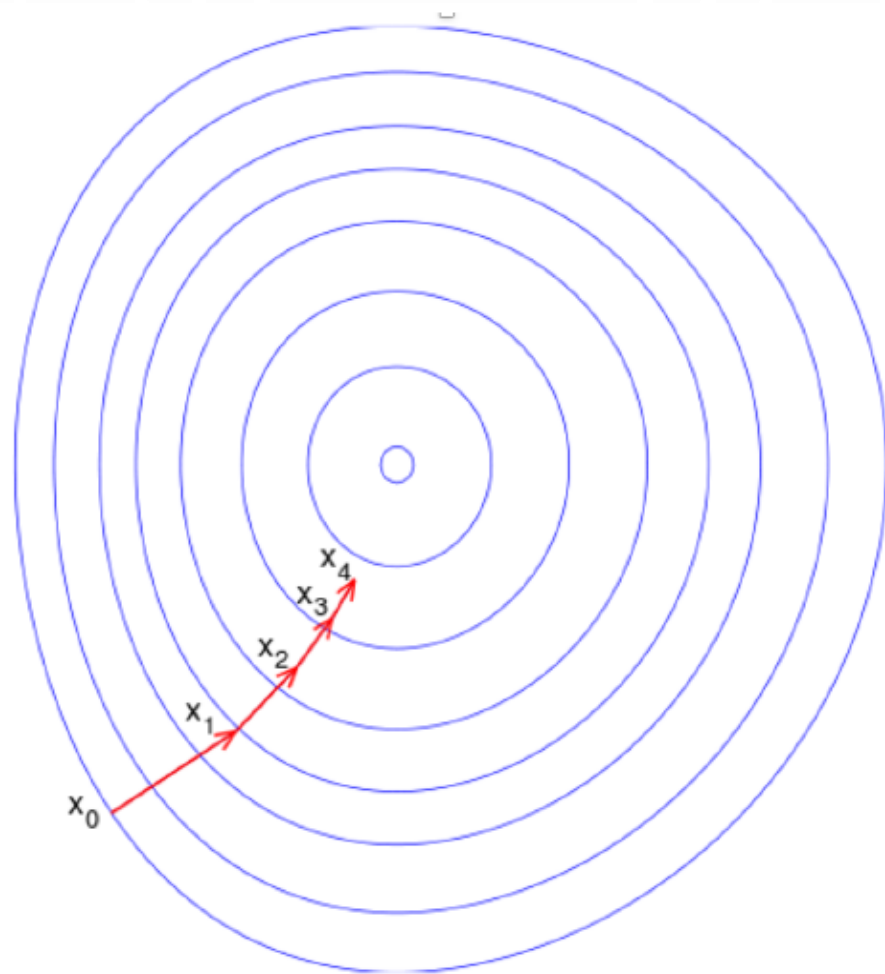
$$s_i^{(N)} = -\frac{\partial L}{\partial z}$$

Коэффициент  $\gamma_N$  должен минимизировать ошибку:

$$\gamma_N = \min_{\gamma \in \mathbb{R}} \sum_{i=1}^l L(y_i, a_{N-1}(x_i) + \gamma_N b_N(x_i))$$



# ГРАДИЕНТНЫЙ СПУСК В ПРОСТРАНСТВЕ ФУНКЦИЙ



# ВЫБОР БАЗОВЫХ АЛГОРИТМОВ

- *Что произойдет с предсказанием бустинга, если базовые алгоритмы слишком простые?*
- *Что будет, если базовые алгоритмы слишком сложные?*

# 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.

# 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.

Чаще всего в качестве базовых алгоритмов используют *решающие деревья*.

# 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.

Чаще всего в качестве базовых алгоритмов используют *решающие деревья*.

В таком случае *решающие деревья не должны быть очень маленькими, а также очень глубокими.*

Оптимальная глубина – от 3 до 6 (зависит от задачи).

# СОКРАЩЕНИЕ ШАГА (РЕГУЛЯРИЗАЦИЯ)

- 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.

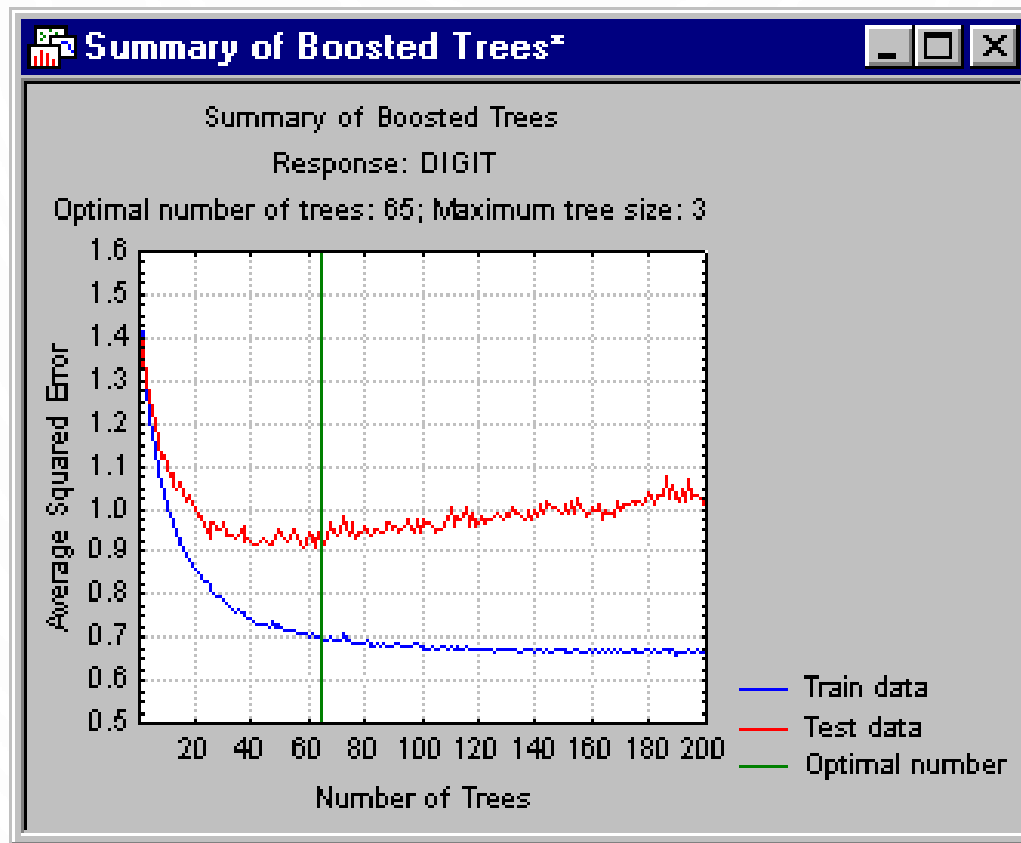
Возможное решение – сокращение шага:

$$a_N(x) = a_{N-1}(x) + \eta \gamma_N b_N(x), \eta \in (0; 1]$$

Чем меньше темп обучения  $\eta$ , тем меньше степень доверия к каждому базовому алгоритму, и тем лучше качество итоговой композиции.

# КОЛИЧЕСТВО ИТЕРАЦИЙ БУСТИНГА

*Так как на каждом шаге бустинга целенаправленно уменьшается ошибка на тренировочной выборке, то если процесс не остановить, то мы достигнем нулевой ошибки, а значит, переобучимся!*



# СТОХАСТИЧЕСКИЙ ГРАДИЕНТНЫЙ БУСТИНГ

- Будем обучать базовый алгоритм  $b_N$  не по всей выборке  $X$ , а по случайной подвыборке  $X^k \subset X$ .

+: снижается уровень шума в данных

+: вычисления становятся быстрее

Обычно берут  $|X^k| = \frac{1}{2} |X|$ .



# 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).*