Введение в анализ данных

Градиентный бустинг

Слайды в основном Евгения Соколова

НИУ ВШЭ, 2020

Случайный лес (Random forest)

1. Для $n = 1, \dots, N$:
2. Сгенерировать выборку $\tilde{X}$ с помощью бутстрапа
3. Построить решающее дерево $b_n(x)$ по выборке $\tilde{X}$
4. Дерево строится, пока в каждом листе не окажется не более n_{min} объектов
5. Оптимальное разбиение ищется среди q случайных признаков

Чем плох случайный лес?

- Нужны глубокие деревья, могут очень долго обучаться
- Если одно дерево не справляется с задачей, то усреднение вряд ли поможет

Bias-variance decomposition

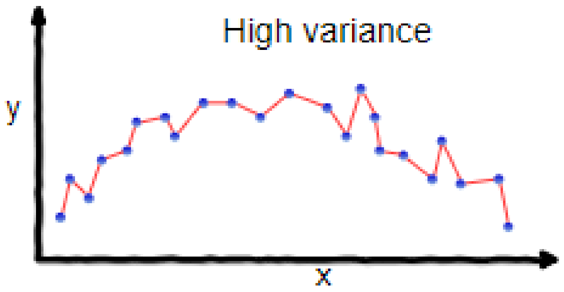
$$L(\mu) = \underbrace{\mathbb{E}_{x,y}\Big[\big(y - \mathbb{E}[y \mid x]\big)^2\Big]}_{\text{шум}} +$$

$$+ \underbrace{\mathbb{E}_x\Big[\big(\mathbb{E}_X[\mu(X)] - \mathbb{E}[y \mid x]\big)^2\Big]}_{\text{смещение}} + \underbrace{\mathbb{E}_x\Big[\mathbb{E}_X\Big[\big(\mu(X) - \mathbb{E}_X[\mu(X)]\big)^2\Big]\Big]}_{\text{разброс}}$$

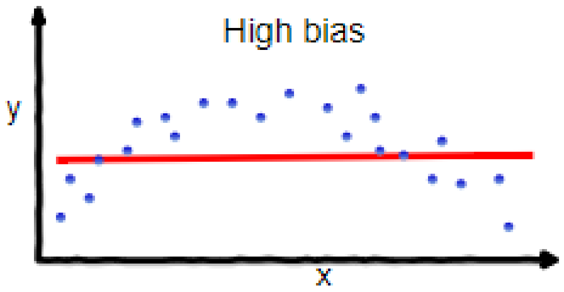
Bias-variance decomposition

- Можно показать, что ошибка метода обучение раскладывается на три слагаемых: шум, смещение, разброс
- Шум — как сильно ошибается лучшая модель
- **Смещение** (bias) — как сильно в среднем отклоняется наша модель от лучшей модели (недообучение)
- **Разброс** (variance) — как сильно может меняться модель, если немного поменять обучающую выборку (переобучение)

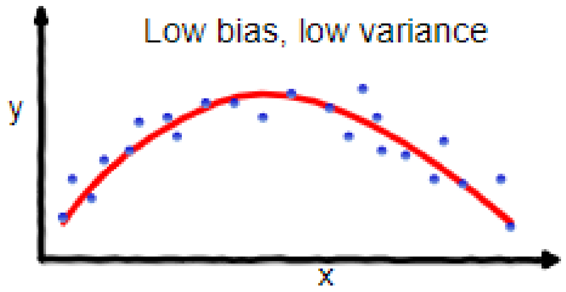
Bias-variance decomposition



overfitting



underfitting



Good balance

Смещение и разброс в беггинге

Можно показать, что в беггинге:

- Смещение композиции такое же, как у одной модели
- Разброс уменьшается тем сильнее, чем меньше корреляция между базовыми моделями
  - Поэтому в случайном лесе мы придумывали способы повышения разнообразия моделей

- Вывод: если дерево имеет высокое смещение, то беггинг не даст хороший результат

Градиентный бустинг

Идея бустинга

- Будем обучать каждую следующую модель в композиции так, чтобы она исправляла ошибки предыдущих моделей

Бустинг для MSE

- Композиция:

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

- Обучим первый базовый алгоритм как обычно (например, стандартная процедура обучения дерева для регрессии):

$$\frac{1}{\ell} \sum_{i=1}^{\ell} (b_1(x_i) - y_i)^2 \to \min_{b_1}$$

Бустинг для MSE

- Вторая базовая модель должна корректировать ошибки первой:
$$b_2(x_i) \approx y_i - b_1(x_i)$$
- Если получится этого добиться, то
$$b_1(x_i) + b_2(x_i) \approx y_i$$
- Значит, вторую модель обучаем так:
$$\frac{1}{\ell}\sum_{i=1}^{\ell}\Big(b_2(x_i) - \big(y_i - b_1(x_i)\big)\Big)^2 \to \min_{b_2}$$
- b_1 тут уже фиксирован!

Бустинг для MSE

- И так далее:

$$\frac{1}{\ell}\sum_{i=1}^{\ell}\Big(b_3(x_i)\ -\big(y_i - b_1(x_i) + b_2(x_i)\big)\Big)^2 \to \min_{b_3}$$

$$\frac{1}{\ell}\sum_{i=1}^{\ell}\Big(b_4(x_i)\ -\big(y_i - b_1(x_i) + b_2(x_i) + b_3(x_i)\big)\Big)^2 \to \min_{b_4}$$

Бустинг для MSE

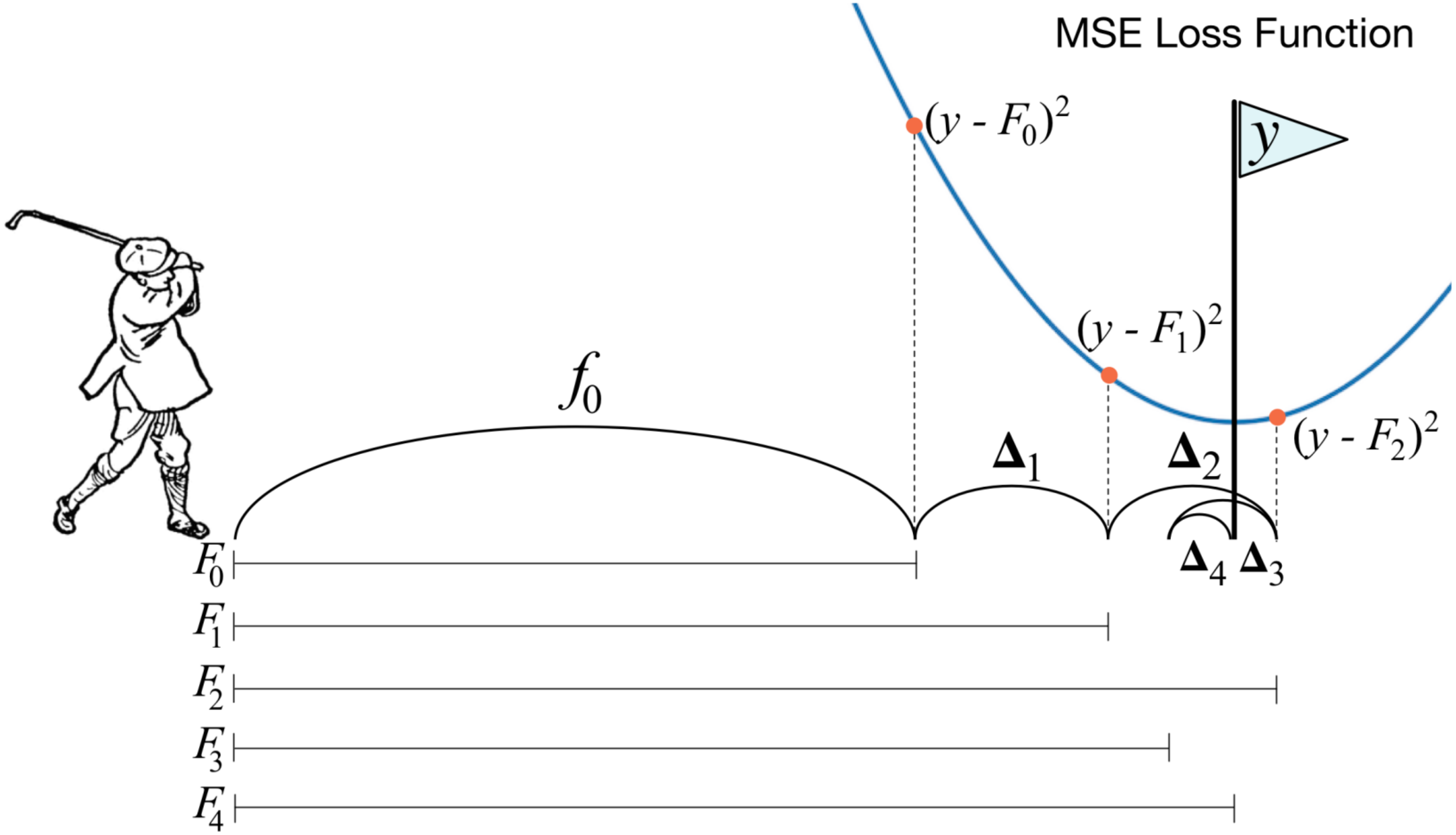
- Стараемся прийти к минимуму квадратичной функции ошибки
- Каждый новый алгоритм должен уменьшать ошибку композиции предыдущих
- Считаем антиградиент и идем по нему в сторону минимума

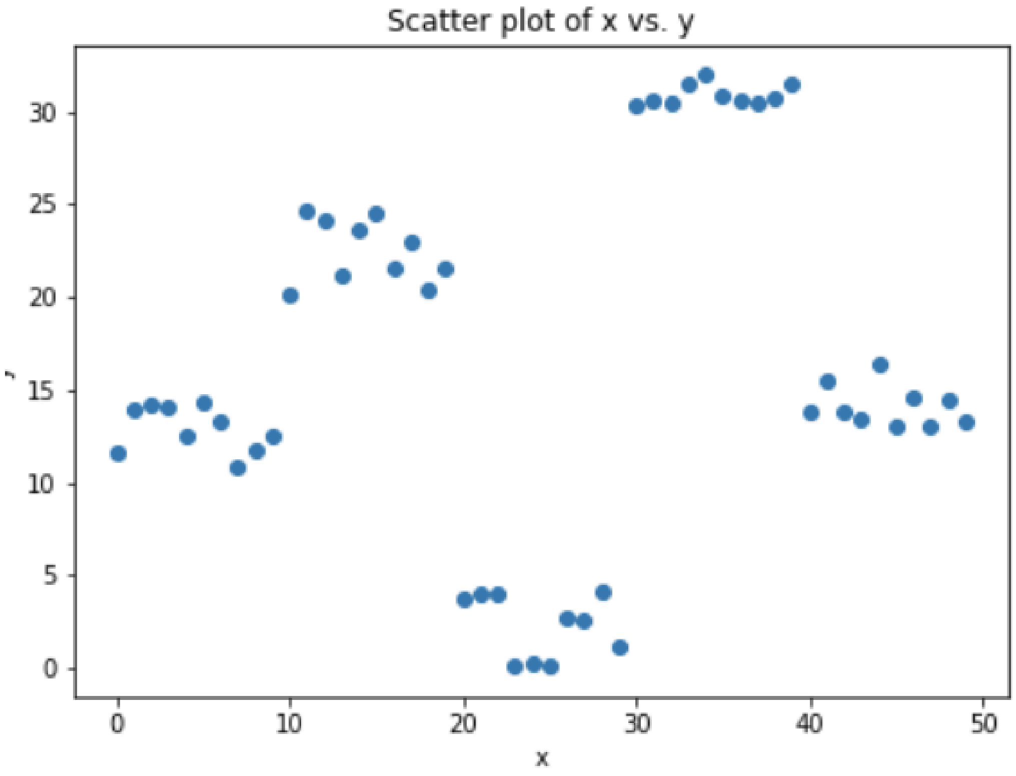
$mse(y, predict) = (y - prediction)^2$ - ошибка

$-\nabla_{predict} mse(y, prediction) = -(prediction - y) = y - prediction$ - антиградиент (остатки)

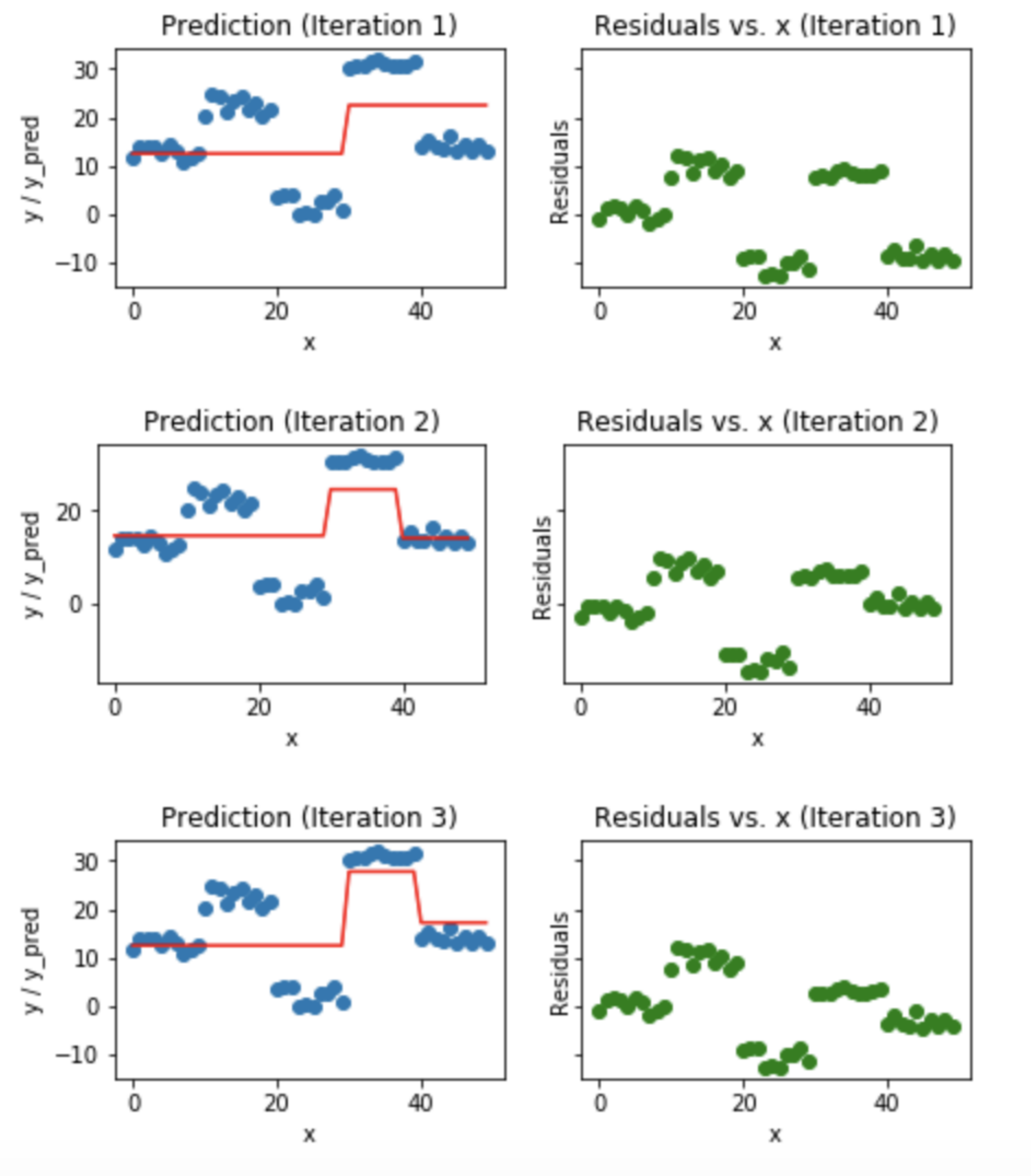
- На каждой новой итерации предсказываем не исходные ответы, а остатки
- Чтобы прибавить их к предсказанию композиции и получить более близкие к реальным ответы

Играем в гольф

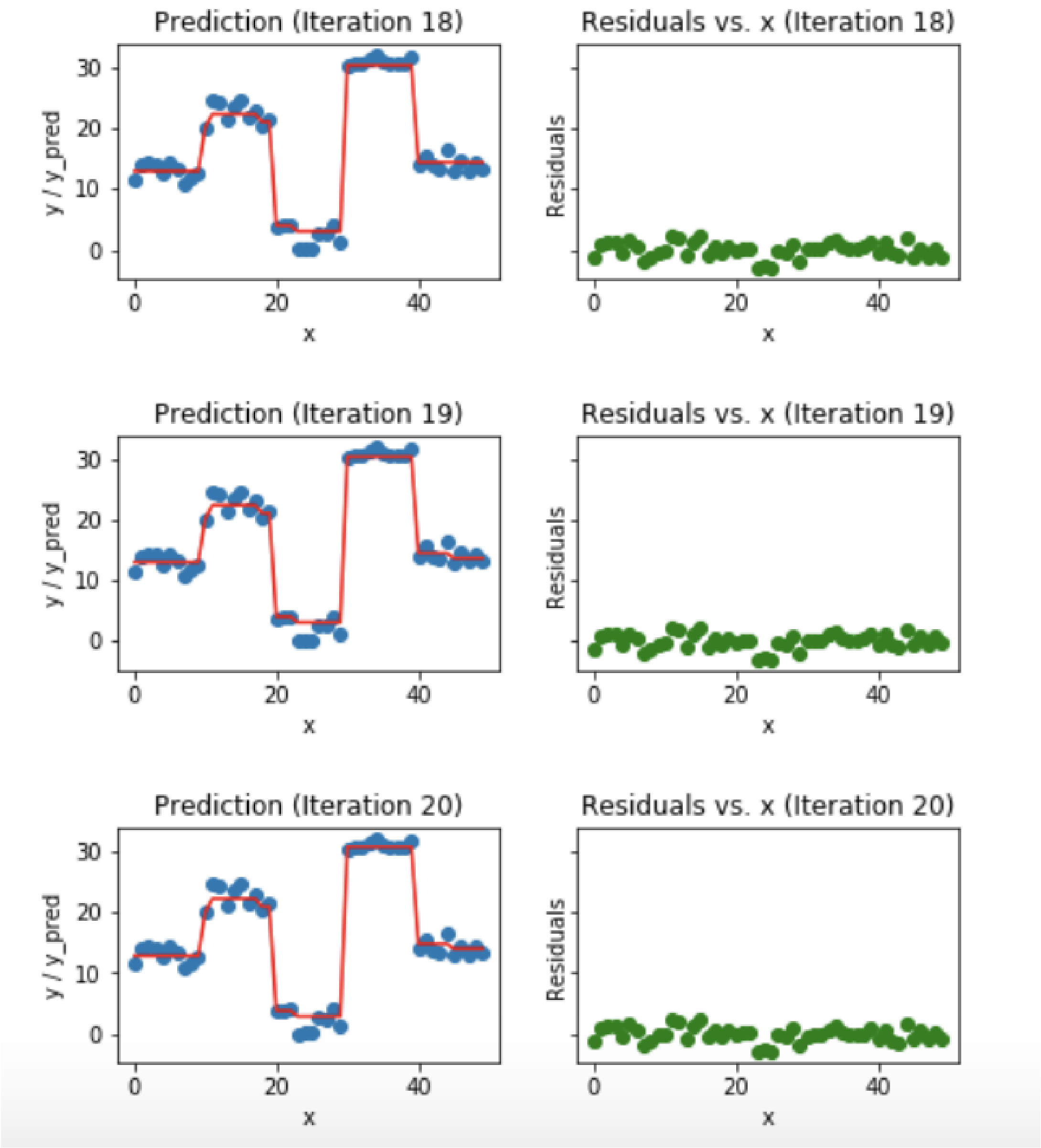




https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d



https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d



https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d

Бустинг для MSE

- Переобучается по мере роста числа базовых моделей (в отличие от случайного леса)
- Композиция деревьев с помощью бустинга **понижает** смещение и **повышает** разброс
- Значит, базовые модели — неглубокие деревья (где-то от 1 до 6 уровней)

- Для сравнения: беггинг **не меняет** смещение и **понижает** разброс
- Поэтому базовые модели — глубокие деревья

Градиентный бустинг в общем случае

- Задача обучения в общем виде:

$$\frac{1}{\ell}\sum_{i=1}^{\ell} L\big(y_i, a(x_i)\big) \to \min_a$$

Градиентный бустинг в общем случае

- Допустим, мы уже обучили (N-1)-у базовую модель:

$$a_{N-1}(x) = \sum_{n=1}^{N-1} b_n(x)$$

- Задача обучения N-й модели:

$$\frac{1}{\ell} \sum_{i=1}^{\ell} L\big(y_i, a_{N-1}(x_i) + b_N(x_i)\big) \to \min_{b_N}$$

Градиентный бустинг в общем случае

- Для MSE есть важное свойство:

$$L(y, a+b) = \big((a+b) - y\big)^2 = \big(b - (y-a)\big)^2 = L(y-a, b)$$

- Поэтому задача обучения очередной базовой модели сводится к замене целевой переменной:

$$\frac{1}{\ell}\sum_{i=1}^{\ell} L\big(y_i, a_{N-1}(x_i) + b_N(x_i)\big) \to \min_{b_N}$$

$$\frac{1}{\ell}\sum_{i=1}^{\ell} L\big(y_i - a_{N-1}(x_i), b_N(x_i)\big) \to \min_{b_N}$$

Градиентный бустинг в общем случае

- Но далеко не всегда это свойство выполнено!
- Например, для логистической функции потерь оно не работает

$$L(y, a) = \log(1 + \exp(-ya))$$

Градиентный бустинг в общем случае

Можно показать, что задача

$$\sum_{i=1}^{\ell} L\big(y_i, a_{N-1}(x_i) + b_N(x_i)\big) \to \min_{b_N}$$

примерно совпадает с задачей

$$\sum_{i=1}^{\ell} (b_N(x_i) - s_i)^2 \to \min_{b_N}$$

Где

$$s_i = -\frac{\partial L}{\partial z}\bigg|_{z=a_{N-1}(x_i)}$$

Градиентный бустинг в общем случае

- Задачу построения следующей модели в композиции можно свести к задаче регрессии с новой целевой переменной
- Новая целевая переменная — вектор антиградиента функции потерь в точке текущего прогноза (остатки)
- Мы как бы строим новую модель, чтобы она как можно сильнее снизила ошибку композиции

Обучение градиентного бустинга

- Основные гиперпараметры:
  - Число деревьев
  - Размер шага
  - Глубина дерева
- В имплементациях могут быть и другие важные настройки
  - Регуляризация
  - Семплирование объектов
  - и т.д.

Learning rate (размер шага)

- Базовые модели - неглубокие деревья с низким качеством
- Базовая модель должна приближать вектор антиградиента
- Может сделать это плохо и вместо шага в нужную сторону мы получим случайное блуждание
- Не будем сильно доверять каждому отдельному базовому алгоритму, уменьшим его вклад в итоговую модель домножив результат на коэффициент $0 < \gamma <= 1$

$$a_N(x) = a_{N-1}(x) + \gamma b_N(x)$$

- Аналог размера шага в градиентном спуске
- Замедление обучения, большее число базовых алгоритмов в модели

Learning rate