Бустинги

Гирдюк Дмитрий Викторович, Никольская Анастасия Николаевна

24 апреля 2025

СПбГУ, ПМ–ПУ

Сравнение подходов к ансамблированию [1]

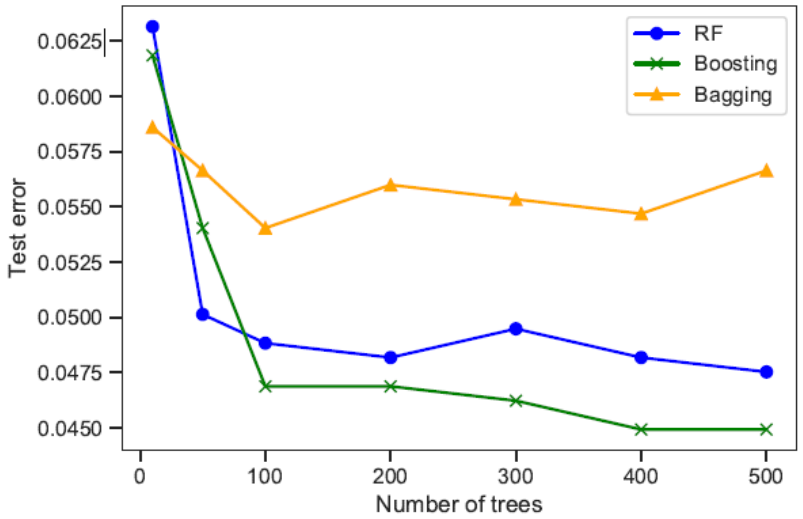


Figure 18.5: Preditive accuracy vs size of tree ensemble for bagging, random forests and gradient boosting with log loss. Adapted from Figure 15.1 of [HTF09]. Generated by spam_tree_ensemble_compare.ipynb.

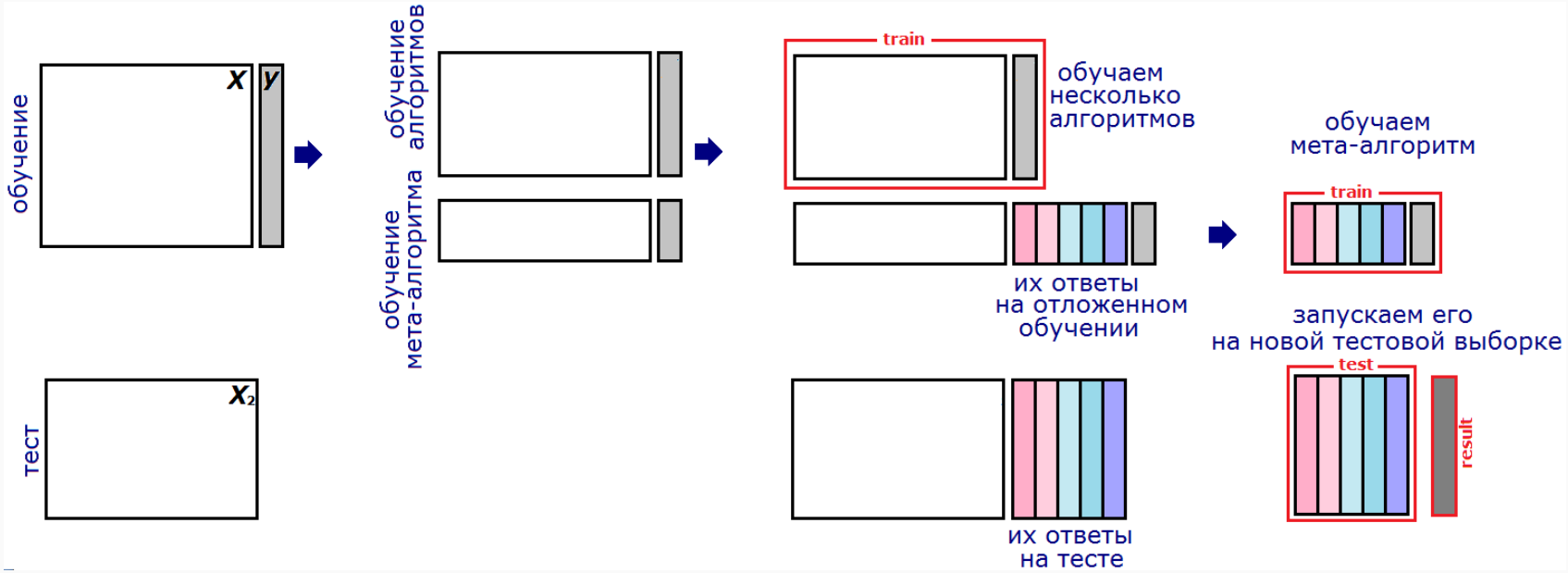
Blending и Stacking

- Попробуем строить модель поверх других моделей.
- Способов того, как это сделать, тоже хватает. Разберем 2 основных: блэндинг и стэкинг.

Blending

- Делим обучающую (!) выборку на 2 части.
- На первой части обучаем базовые модели. На отложенной части делаем предсказания обученными моделями и получаем набор из так называемых метапризнаков.
- Используем известные таргеты для обучения метамодели на метапризнаках.

Схема блендинга [2]



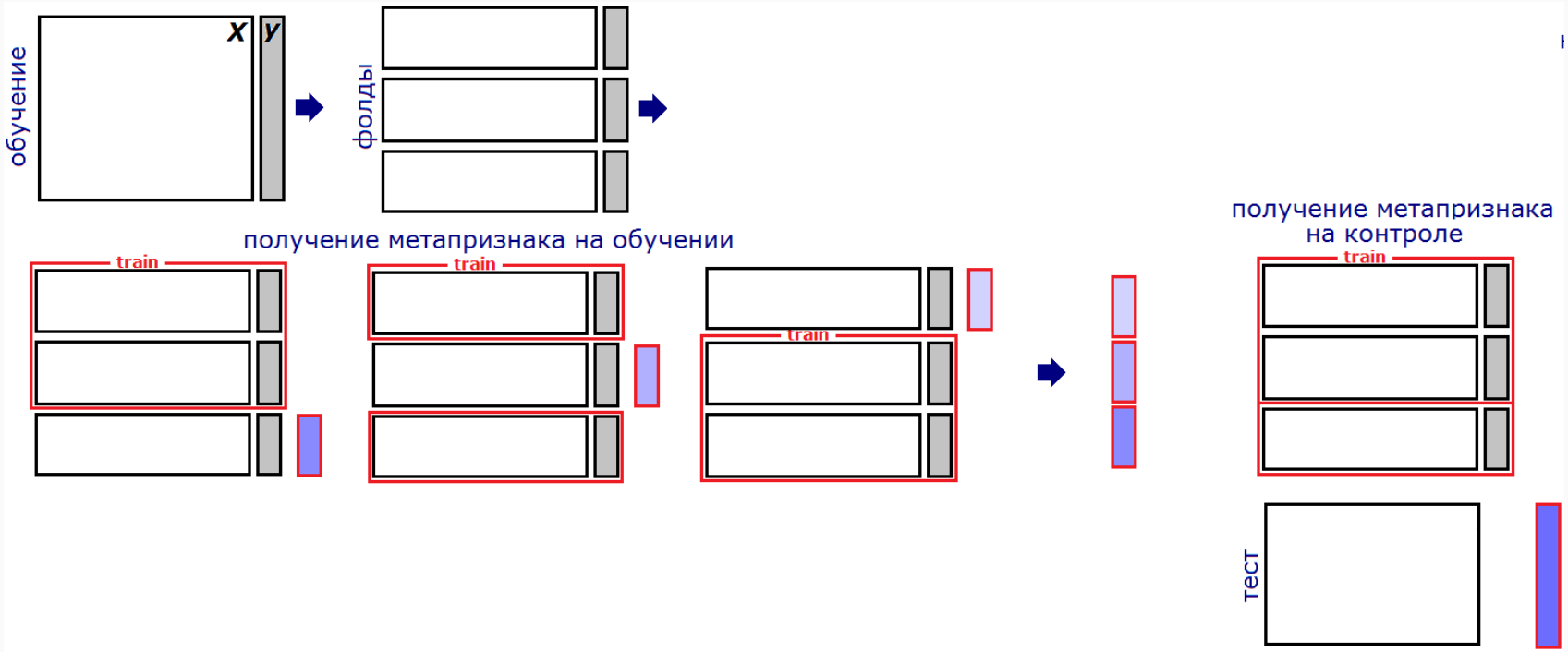
Stacking i

- Проблема блендинга очевидна: приличная часть данных уходит на обучение метамодели. Частично побороть это можно путем проведения обучения блендинга несколько раз с разными разбиениями данных.
- Но есть и другая идея (скорее даже набор идей), которая называется стэкингом.

Stacking ii

- Как в кросс-валидации делим обучающую выборку на K частей/фолдов. Последовательно обучаем все базовые модели на подмножествах из $K-1$ фолдов, предсказываем на оставшемся.
- Из предсказаний на них собираем для каждой базовой модели свой метапризнак (можно также несколько раз разбить на фолды и усреднить предсказания на объектах). На полученных метапризнаках обучаем метамодель.
- После чего все базовые алгоритмы обучаются на всей тренировочной выборке. Ими производится предсказание на тестовой выборке для получения метапризанков, которые подаются в обученную метамодель.

Схема стекинга [2]



Блендинг и стекинг: обсуждение i

- В случае с блендингом и стекингом можно вместе использовать разные типы моделей. Почему бы не объединить kNN, логистические регрессии с различными параметрами регуляризации и случайный лес, а в качестве метамодели еще раз использовать случайный лес? Обычно все правда попроще, так как разные методы требуют разных данных.
- Строгой теории, связывающей эти подходы с дилеммой смещения-дисперсии нет, блендинг и стекинг могут немного улучшить качество в сравнении с наилучшей моделью в ансамбле. Они же часто (раньше) использовались в разных соревнованиях, так как помогали эксплуатировать проблемы в датасетах.

Блендинг и стекинг: обсуждение ii

- Махинация в стекинге с переобучением базовых моделей на всем тренировочном датасете очевидна. Признаки базовых моделей на обучении и тесте разные. Есть разные способы побороться с ней, например, регуляризацией метамодели или зашумление метапризнаков.
- Стекинг и блендинг можно делать многоуровневым! Муторно, в реальных проектах почти не встречается, но на соревнованиях таким не брезгуют заниматься, если время и вычислительные ресурсы позволяют.
- Добавлять исходные признаки к метапризнакам можно, но очень осторожно.
- В scikit-learn есть реализация стэкинга.

Boosting

- Бустинг – альтернативный подход к ансамблированию, основная задача которого состоит в уменьшении смещения (хотя дисперсия обычно также снижается).
- Достигается это за счет построения ансамбля из достаточно "слабых" моделей, причем не параллельно, как в бэггинге, а последовательно, где каждая новая базовая модель обучается таким образом, чтобу уменьшить суммарную ошибку текущего ансамбля.
- Бустинг – общая идея, на основе которой в свое время было построено достаточно много разнообразных алгоритмов.
- Наиболее общая версия бустинга, обобщающая известные ранее, "градиентный бустинг", была разработана в конце прошлого века. На текущий момент все основные библиотеки, реализующие бустинг (xgboost [3], catboost [4], lightgbm [5]), реализуют именно ее.

Последовательное конструирование ансамбля i

- Пусть каждый следующий алгоритм уточняет предсказание суммы предыдущих:

$$a_t(\boldsymbol{x}) = a_{t-1}(\boldsymbol{x}) + \lambda_t h(\boldsymbol{x}; \boldsymbol{\theta}_t)$$

- Тогда последовательное обучение сводится к решению последовательности задач

$$\arg\min_{\lambda, \boldsymbol{\theta}} \sum_{i=1}^{N} L\left(y^{(i)}, a_{t-1}(\boldsymbol{x}^{(i)}) + \lambda h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})\right), \; t = 1, \ldots, T$$

- При решении новой задачи параметры предыдущих моеделей в ансамбле не обновляются, также как не обновляются и их весовые коэффициенты
- Чаще всего весовые коэффициенты имеют одно фиксированное значение - 1.

Последовательное конструирование ансамбля ii

- h и $\boldsymbol{\theta}$ задаются контретным базовым алгоритмом. Так, для дерева $\boldsymbol{\theta}$ - это набор его разбиений.
- Например, для квадратичной функции потерь в задаче регрессии

$$L\left(y^{(i)}, a_{t-1}(\boldsymbol{x}^{(i)}) + \lambda h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})\right) =$$
$$= \frac{1}{2}\left(y^{(i)} - a_{t-1}(\boldsymbol{x}^{(i)}) - \lambda h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})\right)^2 = \left(r_t^{(i)} - \lambda h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})\right)^2$$

- Если положить $\lambda = 1$, то мы по сути подгоняем модель $h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$ предсказывать невязку между таргетом и предсказанием текущего ансамбля. В данном случае невязка - антиградиент функции потерь:

$$y^{(i)} - a(\boldsymbol{x}^{(i)}) = -\frac{\partial L(y^{(i)}, a(\boldsymbol{x}))}{\partial a(\boldsymbol{x})}\Big|_{a(\boldsymbol{x})=a_{t-1}(\boldsymbol{x}^{(i)})}$$

Градиентный бустинг i

- Рассмотрим ту же постановку в применении к общей задаче с потерями L. Разложим лосс-функцию в ряд Тейлора до первого члена в окрестности точки $(y^{(i)}, a(\boldsymbol{x}))$

$$L\left(y^{(i)}, a_{t-1}(\boldsymbol{x}^{(i)}) + \lambda h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})\right) \approx$$

$$\approx L\left(y^{(i)}, a_{t-1}(\boldsymbol{x}^{(i)})\right) + \lambda h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) \frac{\partial L(y^{(i)}, a(\boldsymbol{x}))}{\partial a(\boldsymbol{x})}\Big|_{a(\boldsymbol{x})=a_{t-1}(\boldsymbol{x}^{(i)})}$$

- Первое слагаемое постоянное, потому при оптимизации параметров базовой модели итоговая лосс-функция может быть записана следующим образом

$$\arg\min_{\lambda, \boldsymbol{\theta}} \sum_{i=1}^{N} \lambda h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) \left(g_{t-1}^{(i)}\right), t = 1, \ldots, T$$

Градиентный бустинг ii

- Это скалярное произведение, и чтобы его минимизировать, можно обучать базовую модель $h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$ предсказывать антиградиент $-g_{t-1}^{(i)}$ (и для задачи регрессии, и для классификации!)

$$\boldsymbol{\theta}_t = \arg\min \sum_{i=1}^{N} \left(h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) g_{t-1}^{(i)} \right)^2$$

- Итак, независимо от исходной функции, задача сводится к
- λ же можно находить бинарным/линейным поиском для исходной оптимизационной задачи уже после того, как модель $h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$ будет обучена. По крайней мере так было в исходной статье по градиентному бустингу.

Градиентный бустинг: алгоритм

1. Инициализируем первую модель в ансамбле константой:

$$a_0(\boldsymbol{x}) = \arg\min_{\gamma} \sum_{i=1}^{N} L(y^{(i)}, \gamma)$$

2. Для $t = 1, \ldots, T$ повторяем шаги 3-6.
3. Вычисляем псевдоостатки/антиградиенты

$$g_{t-1}^{(i)} = -\frac{\partial L(y^{(i)}, a(\boldsymbol{x}))}{\partial a(\boldsymbol{x})}\Big|_{a(\boldsymbol{x})=a_{t-1}(\boldsymbol{x}^{(i)})}, \; i = 1, \ldots, N$$

4. Подгоняем по ним базовую модель $h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$ на выборке $\{(\boldsymbol{x}^{(i)}, g_{t-1}^{(i)})\}_{i=1}^{N}$.
5. Линейным поиском находим оптимальное значение λ_t.
6. Обновляем ансамбль $a_t(\boldsymbol{x}) = a_{t-1}(\boldsymbol{x}) + \lambda_t h(\boldsymbol{x}; \boldsymbol{\theta}_t)$

Пример бустинга [1]

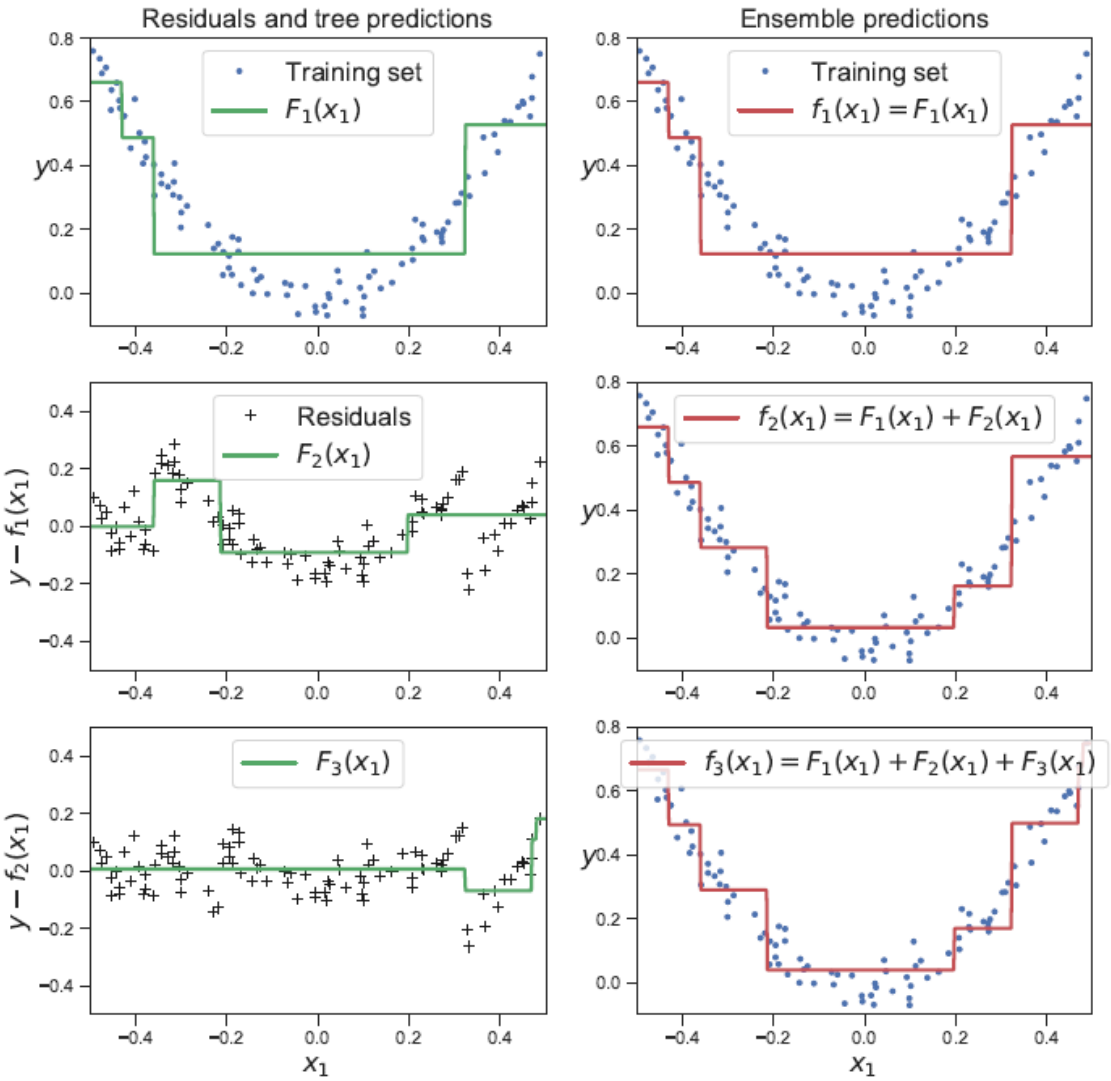


Figure 18.6: Illustration of boosting using a regression tree of depth 2 applied to a 1d dataset. Adapted from Figure 7.9 of [Gér19]. Generated by boosted_regr_trees.ipynb.

Градиентный бустинг: лосс-функции

Name	Loss	$-\partial \ell(y_i, f(\boldsymbol{x}_i))/\partial f(\boldsymbol{x}_i)$
Squared error	$\frac{1}{2}(y_i - f(\boldsymbol{x}_i))^2$	$y_i - f(\boldsymbol{x}_i)$
Absolute error	$\lvert y_i - f(\boldsymbol{x}_i)\rvert$	$\operatorname{sgn}(y_i - f(\boldsymbol{x}_i))$
Exponential loss	$\exp(-\tilde{y}_i f(\boldsymbol{x}_i))$	$-\tilde{y}_i \exp(-\tilde{y}_i f(\boldsymbol{x}_i))$
Binary Logloss	$\log(1 + e^{-\tilde{y}_i f_i})$	$y_i - \pi_i$
Multiclass logloss	$-\sum_c y_{ic} \log \pi_{ic}$	$y_{ic} - \pi_{ic}$

Table 18.1: Some commonly used loss functions and their gradients. For binary classification problems, we assume $\tilde{y}_i \in \{-1, +1\}$, and $\pi_i = \sigma(2f(\boldsymbol{x}_i))$. For regression problems, we assume $y_i \in \mathbb{R}$. Adapted from [HTF09, p360] and [BH07, p483].

Градиентный бустинг: обсуждение i

- Итого, градиентный бустинг – общая методология ансамблирования, нацеленная на уменьшение смещения.
- Достаточно быстро поняли, что λ_t можно заменить константой и выделить как отдельный гиперпараметр. Этот параметр по сути является скоростью обучения (learning rate), и позволяет контролировать вклад каждой базовой модели в итоговый ансамбль.
- На практике чаще всего градиентный бустинг строят над решающими деревьями. Причем важно использовать достаточно неглубокие деревья (вплоть до глубины 1, известные также как решающие пни). Иначе для глубоких деревьев очень быстро базовые модели начнут переобучаться.

Градиентный бустинг: обсуждение ii

- Контроль глубины деревьев в ансамбле наряду со скоростью обучения – основные способы борьбы с переобучением градиентного бустинга.
- Градиентный бустинг над решающими деревьями зачастую имеет способы оценки важности признаков. Например, можно замерять суммарное изменение информативности в каждой нелистовой вершине дерева со сплитом по конкретному признаку. А затем усреднять эти значения по всему ансамблю. И выводить нормализованные относительные значения (признак с наибольшим суммарным вкладом ставится в соответствии 100%).
- Несмотря на эффективность случайного леса, на практике бустинг зачастую показывает себя эффективнее с точки зрения итоговой точности.

Градиентный бустинг: обсуждение iii

- По большому счету, градиентный бустинг на табличных данных является де-факто алгоритмом номер 1 и используется повсеместно.
- Как было ранее отмечено, уже продолжительное время существует 3 приличные библиотеки для обучения градиентного бустинга (xgboost [3], catboost [4], lightgbm [5]). Своих тонкостей у каждой очень много. Впрочем, основные идеи уже так или иначе позаимствованы друг у друга. Например, изначально поддержка категориальных переменных была лишь у catboost, но с некоторых пор остальные две также умеют работать с ними.
- В scikit-learn также присутствует 2 реализации градиентного бустинга, но их результаты зачастую хуже, чем у 3 указанных библиотек.

Использованные источники i

1. *Murphy K. P.* **Probabilistic Machine Learning: An introduction.** MIT Press, 2022. URL: probml.ai.
2. *Дьяконов А.* **Стекинг (Stacking) и блендинг (Blending).** URL: https://alexanderdyakonov.wordpress.com/2017/03/10/c%D1%82%D0%B5%D0%BA%D0%B8%D0%BD%D0%B3-stacking-%D0%B8-%D0%B1%D0%BB%D0%B5%D0%BD%D0%B4%D0%B8%D0%BD%D0%B3-blending/.
3. **XGBoost.** URL: https://xgboost.readthedocs.io/en/stable/.
4. **CatBoost.** URL: https://xgboost.readthedocs.io/en/stable/.
5. **LightGBM.** URL: https://lightgbm.readthedocs.io/en/stable/.