ИЗВЕСТИЯ РАН. ТЕОРИЯ И СИСТЕМЫ УПРАВЛЕНИЯ, 2024, № 6, с. 91–110

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

УДК 004.89, 004.048, 004.67

ПРИМЕНЕНИЕ БУСТИНГА В РЕКОМЕНДАТЕЛЬНЫХ СИСТЕМАХ

© 2024 г. М. А. Жарова[a], *, В. И. Цурков[b], **

[a]МФТИ, Долгопрудный, Россия

[b]ФИЦ ИУ РАН, Москва, Россия

**e-mail: zharova.ma@phystech.edu*

***e-mail: v.tsurkov@frccsc.ru*

Поступила в редакцию 07.08.2024 г.

После доработки 01.09.2024 г.

Принята к публикации 16.10.2024 г.

В современной цифровой эпохе рекомендательные системы прочно укрепились, став важным инструментом для эффективного управления информационными потоками. Их востребованность в значительной степени обусловлена динамикой нынешнего общества, а именно информационным переизбытком и необходимостью персонализации данных. С расширением сфер применения рекомендательных алгоритмов появляется и немало нестандартных случаев, для которых использование классических подходов не столь эффективно. Рассматривается один из таких: малое число объектов при относительно большом числе пользователей в условиях высокой корреляции между некоторыми объектами. Для моделирования предлагается градиентный бустинг — алгоритм машинного обучения на основе ансамбля решающих деревьев.

Ключевые слова: рекомендательные системы, бустинг, пользователи и объекты, корреляция, калибровка

DOI: 10.31857/S0002338824060083, **EDN:** SUDEVR

APPLICATION OF BOOSTING IN RECOMMENDER SYSTEMS

M. A. Zharova[a], *, V. I. Tsurkov[b], **

[a]Moscow Institute of Physics and Technology (MIPT), Dolgoprudny, Moscow oblast, Russia

[b]Federal Research Center "Computer Science and Control", Russian Academy of Sciences, Moscow, Russia

**e-mail: zharova.ma@phystech.edu*

***e-mail: v.tsurkov@frccsc.ru*

In today's digital era, recommender systems have gained a strong foothold, becoming an important tool for effectively managing information flows. Their demand is largely due to the dynamics of current society, namely information overload and the need to personalize data. With the expansion of the scope of application of recommendation algorithms, many non-standard cases appear, for which the use of classical approaches is not so effective. This paper examines one of these: a small number of objects with a relatively large number of users in conditions of high correlation between some objects. For modeling, it is proposed to use gradient boosting, a machine learning algorithm based on an ensemble of decision trees.

Keywords: recommender systems, boosting, users and items, correlation, calibration

Введение. В настоящий момент самым частым подходом к построению рекомендательных систем является гибридный, который, как правило, представляет собой комбинацию модели матричной факторизации и какого-либо контентного метода [1]. На первом этапе происходит начальный отбор кандидатов, здесь коллаборативные модели в отдельности способны давать более точные и быстрые прогнозы [2]. При помощи контентных данных решается проблема

холодного старта, свойственная всем алгоритмам коллаборативной фильтрации, а также происходит дополнительное уточнение рекомендаций на основе метаданных.

Таким образом, для получения качественного решения необходимы коллаборативные данные – о взаимодействии пользователей и объектов. Обычно они представляются в виде матрицы и в большинстве классических случаев являются разреженными из-за того, что пользователи взаимодействуют далеко не со всеми объектами. Эта проблема, как известно, решается на этапе построения модели матричной факторизации: через разложение матриц и поиск скрытых факторов [3].

Но что если число объектов в данных много меньше числа пользователей – например, на 1 млн уникальных пользователей приходится лишь 10 уникальных объектов? Во-первых, в такой ситуации матрицы уже нельзя считать достаточно разреженными, поэтому подобную задачу эффективнее решать методами формирования последовательностей, используя теорию информации [4]. Более того, в приведенном порядке соотношений 1:100 000 векторы для каждого пользователя будут иметь настолько низкую размерность, что среди них появится много дубликатов. Например, в простейшем случае, когда не учитывается вес взаимодействия пользователя с объектом, все числа в матрице будут равняться либо 0, либо 1. Тогда для 10 объектов суммарно можно получить лишь $2^{10} = 1024$ уникальных строк.

Можно было бы это исправить, взяв лишь часть из имеющихся пользователей, которые уникальны по поведению, далее рассчитать для них первичные рекомендации, а затем сделать переранжирование с учетом метаданных. Но в условиях поставленной задачи добавляется проблема корреляции некоторых объектов друг с другом. Так как число объектов крайне ограничено, нужно удалить часть из них, что означает еще большее уменьшение числа признаков или, другими словами, понижение размерности “коллаборативных” векторов для пользователей, из которых состоит матрица.

В качестве альтернативы можно также рассмотреть чистый контентный подход [5]. Он довольно тривиальный и в классическом виде учитывает только метаданные пользователей или объектов в отдельности, поэтому проблема сильной разницы в их числе пропадает. Если рассматривать вышеупомянутый метод на основе характеристик объектов, высокая корреляция между некоторыми из них сохраняется, из-за чего алгоритм поиска похожих просто деградирует до выдачи одинаковых предсказаний для определенных пар объектов. В случае применения контентного алгоритма на основе характеристик пользователей главной проблемой может оказаться ограниченность данных: технически взять их можно из анкетирования, проведения опросов, поиска в сторонних источниках и т.п. Обычно подобные мероприятия не носят обязательного характера, соответственно информацию и достоверную картину про всех пользователей получить сложно. Более того, немаловажную роль здесь играют различные законы о персональных данных.

В ходе анализа получается, что использование классических подходов неуместно для имеющихся данных главным образом из-за малого числа объектов и корреляции между некоторыми из них. Чтобы решить последнюю проблему, можно независимо обучить несколько отдельных моделей под каждый объект. Эта идея была бы трудно реализуемой при большом количестве разнообразных объектов, как это чаще всего бывает в задачах рекомендаций; но в рассматриваемых условиях их количество как раз невелико.

Таким образом, каждая из независимых моделей будет представлять собой обыкновенный классификационный алгоритм. Для его обучения можно использовать метаданные без каких-либо дополнительных преобразований, а коллаборативные перевести в одномерные признаки путем агрегации признаков действий на разных временных периодах: например, посчитать количество просмотров, среднюю сумму покупок, число звонков и т.п. за последние неделю/месяц/квартал. Это позволит учитывать в модели наиболее ценные данные о взаимодействиях пользователей и объектов, чтобы идея не сводилась к обычному контентному методу на основе только метаинформации.

Итак, реализовав описанный подход, можно решить сразу ряд проблем: малого числа объектов, высокой корреляции между некоторыми из них, а также учесть в одноуровневом каскаде моделей и коллаборативные, и метаданные. Затем полученные предсказания обязательно необходимо откалибровать, чтобы получить сравнимые вероятности заинтересованности пользователей во всех объектах.

1. Постановка задачи. Формализуем данный подход для математической модели. Пусть имеется множество пользователей U и их характеристики; множество объектов P и их характеристики; информация о взаимодействиях $R = U \times P$ за некоторый временной промежуток.

Составим $n = |P|$ отдельных наборов данных по пользователям U, каждый из них будет выглядеть следующим образом:

1) характеристики пользователей U_i, где $U_i \in U$ – подмножество пользователей, у которых нет соответствующего i-го объекта $i = (1, n)$;

2) рассчитанные агрегированные признаки по выбранному объекту для каждого пользователя (количество взаимодействий различного типа: потраченная сумма, средние показатели и статистики);

3) характеристики выбранного объекта p.

В данной задаче, как и в любой рекомендательной системе, важно обратить внимание на временные особенности – за какой промежуток использовать признаковое пространство и на каком собирать целевую переменную. Это поможет сконструировать правдоподобный процесс работы системы на реальных пользователях, определить примерное время переобучения модели в совокупности с возможностями доступных вычислительных ресурсов и корректно оценить качество, исключив проблему утечки данных [6].

Модели матричной факторизации основываются на заполнении пропусков в матрице user-item, что позволяет спрогнозировать заинтересованность пользователей в объектах, с которыми они еще не взаимодействовали [7]. При формировании списка итоговых рекомендаций те объекты, с которыми целевое взаимодействие уже было, не попадают в него. Это аналогичным образом необходимо учитывать в предложенной архитектуре из независимых классификаторов, а именно на этапе формирования множества пользователей для обучения и тестирования: выборки для каждой модели должны быть отфильтрованы так, чтобы в них попадали только пользователи, у которых нет взаимодействий с соответствующим объектом за определенный промежуток времени T_1.

При работе с моделями на коллаборативных данных одним из немаловажных аспектов является также проблема холодного старта. В рассматриваемой задаче на каждый месяц приходится не более 5% новых пользователей, для которых имеется ряд контентных характеристик. Не будем исключать их из выборки, а признаки, рассчитанные на основе коллаборативных, просто заполним нулями.

Еще одна важная особенность, связанная со временем – период созревания целевой переменной после предоставления новых рекомендаций пользователям. Для надежной оценки качества между начальной и конечной датами тестирования должно пройти не менее T_2 времени, чтобы пользователи успели отреагировать. Также при выборе T_2 можно отталкиваться и от цели использования системы: на каком горизонте нужно спрогнозировать поведение пользователей – коротком или более долгосрочном.

С учетом вышеперечисленных требований можно определить схему формирования тренировочной и тестовой выборок (рис. 1): T_1 – промежуток времени для взятия признакового пространства, T_2 – период, за который необходимо рассматривать целевое событие для достоверной оценки работы модели. Другими словами, будем считать, что взаимодействие произошло, если пользователь в течение промежутка T_2 хотя бы раз провзаимодействовал с соответствующим объектом.

Возвращаясь к архитектуре системы, можно формализовать понятие целевой переменной. Она представляет собой бинарный признак, который равен 1, если пользователь взаимодействовал с соответствующим объектом, и 0 – если нет, взаимодействия рассматриваются на временном промежутке T_2. Далее необходимо собрать все предсказания воедино. Здесь важно учесть, что значения, полученные от каждой модели, не являются чистыми вероятностями склонности к тому или иному объекту. Это лишь степень уверенности модели в том, что конкретная запись принадлежит положительному классу. Поэтому чтобы можно было сравнивать

Рис. 1. Формирование обучающих и тестовых наборов данных во времени.

предсказания отдельных моделей друг с другом (это понадобится для составления итогового ранжированного списка рекомендаций), необходимо дополнительно провести калибровку для всех классификаторов. Главные условия, которые должны при этом соблюдаться, – отсутствие переранжирования между преобразованными вероятностями и исходными значениями, а также соответствие среднего значения целевой переменной итоговым вероятностям [8].

2. Методы обучения: бустинговые алгоритмы. После того, как схема рекомендательной системы зафиксирована, необходимо выбрать наиболее подходящие алгоритмы для каждой ее отдельной части. В первую очередь происходит обучение классификаторов, и при выборе модели важно учитывать тот факт, что размеры датасетов невелики (порядка нескольких сотен тысяч записей). Интерпретировать это можно следующим образом: чем популярнее объект, тем больше им пользуются и тем меньшему количеству пользователей его можно рекомендовать. Соответственно в обучающую выборку модели для такого объекта попадет мало пользователей. С одной стороны, это можно было бы решить, взяв больший промежуток T_1 для обучения. Но это может повлечь ряд проблем, первая из которых – не совсем актуальные рекомендации, так как модель обучится на старых данных. Также может уменьшиться число уникальных пользователей, так как по определению необходимо обучать классификаторы только на тех, кто не взаимодействовал с соответствующим объектом на T_1; соответственно увеличится количество записей для одних и тех же пользователей, что повышает риск переобучения. И желательно, чтобы все классификаторы, даже те, которые не для популярных объектов, были обучены на примерно одинаковых по размеру выборках, чтобы их результаты можно было сравнить друг с другом.

Все алгоритмы искусственного интеллекта верхнеуровнево можно разделить на две группы: классические (машинного обучения) и с применением нейронных сетей (глубокого обучения). В условиях малых датасетов последние демонстрируют менее эффективные результаты, это обусловлено требованием большого объема данных, необходимых для обучения нейросетей, поскольку их сложная архитектура и значительное количество параметров увеличивают риск переобучения на небольших выборках. Классические методы показывают более стабильные и точные результаты в таких условиях [9, 10]. Более того, для работы с табличными структурированными данными такие алгоритмы в принципе более эффективны [11, 12].

Далее при выборе конкретной модели необходимо учитывать, что доступных признаков порядка 500 единиц, среди которых встречаются категориальные и со сложной структурой. При работе с табличными данными, содержащими большое количество сложных признаков, эффективнее использовать нелинейные алгоритмы. Лучше всего с этой задачей справляется градиентный бустинг. Более того, существует ряд современных библиотек, которые позволяют автоматически решать многие проблемы – заполнение пропусков, обработка категориальных признаков, экстраполяции предсказаний на диапазонах данных вне обучающей выборки [13, 14].

Классический градиентный бустинг представляет собой ансамблевую технику, которая объединяет слабые модели, а именно деревья решений для последовательного улучшения предсказаний путем минимизации функции потерь. Модель дерева решений представляет собой бинарное дерево: вершине на каждом уровне соответствует один из признаков данных (без повторений по уровням), разбиение на двух потомков происходит по некоторому значению порога, которое подбирается путем перебора. Последовательность признаков также не случайна, они располагаются согласно критериям информативности, например по дисперсии выборки для задачи регрессии или по формуле кроссэнтропии для классификации. Основная идея бустинга состоит в пошаговом обучении деревьев таким образом, чтобы каждое следующее компенсировало ошибки предыдущей. Так, итоговая модель обновляется на каждом шаге с учетом градиента функции потерь, что позволяет постепенно улучшать предсказания.

Основные шаги метода следующие:

1. Инициализация. На этом этапе подбирается некая функция $a_0(x)$: $X \to \mathbb{R}$, где X – признаковое пространство, состоящее из действительных чисел, размерность которого равна количеству признаков (столбцов) в данных. Обычно эта функция представляет собой среднее значение целевой переменной для задачи регрессии или логарифм отношения шансов для задачи классификации. На каждом следующем шаге также будет подбираться некоторая функция уже путем решения оптимизационной задачи, а итоговая модель будет являться их композицией. В последующем описании каждая отдельная подбираемая функция именуется “базовый алгоритм”, а композиция – просто “алгоритм” с указанием номера шага. В этих терминах на этапе инициализации создается базовый алгоритм $a_0(x)$. Соответственно алгоритмом на m-м шаге будет $a_m(x)$, где x – произвольный вектор из признакового пространства X; в обучающей выборке содержится n таких векторов $x_1, \ldots, x_n$.

2. Вычисление вектора сдвигов. Каждый следующий базовый алгоритм строится таким образом, чтобы уменьшить суммарную ошибку уже имеющейся системы. Для этого на каждом шаге необходимо найти оптимальный вектор значений, на которые нужно сдвинуть предсказания. Другими словами, на некотором m-м шаге решается следующая задача:

$$\sum_{i=1}^{n} L\left(y_i, a_{m-1}\left(x_i\right) + \gamma_i\right) \to \min_{\gamma}, \frac{dy}{dx},$$

где L – функция потерь, y_i – истинные значения целевой переменной, n – количество наблюдений в обучающей выборке, а γ_i, $i = \overline{1,n}$, – компоненты вектора сдвига γ, его можно вычислить следующим образом:

$$\gamma = -\nabla F_m = (s_1, \ldots, s_n)^{\mathrm{T}},\ s_i = -\frac{\partial L\left(y_i, z_i\right)}{\partial z_i},\ z_i = a_{m-1}\left(x_i\right),$$

$$F_m\left(\gamma\right) = \sum_{i=1}^{n} L\left(y_i, a_{m-1}\left(x_i\right) + \gamma_i\right).$$

3. Построение нового дерева. Здесь m-й базовый алгоритм фактически предсказывает производные функции потерь с отрицательным знаком, вычисленные в точках, соответствующих ответам композиции на обучающей выборке на $(m-1)$-м шаге построения алгоритма:

$$f_m\left(x\right) = \operatorname*{argmin}_{f} \frac{1}{n} \sum_{i=1}^{n} \left(f\left(x_i\right) - \gamma_i\right)^2.$$

Информация об исходной функции потерь $L(y, z)$ уже содержится в выражении для вектора оптимального сдвига γ. В ходе оптимизации подбирается такой базовый алгоритм f решающего дерева, который минимизирует суммарное значение ошибки по всем объектам обучающей выборки $x_1, \ldots, x_n$.

4. Обновление модели. Построенное на предыдущем шаге дерево добавляется в итоговую модель с некоторым коэффициентом η:

$$a_m(x) = a_{m-1}(x) + \eta f_m(x),$$

где η – коэффициент обучения, контролирующий вклад каждого нового дерева в общую модель.

5. Повторение шагов 2–4. Процесс повторяется до тех пор, пока не будет достигнуто заданное число итераций или пока улучшения в модели на каждом шаге не станут незначительными.

На текущий момент существует множество модификаций градиентного бустинга, самые известные представлены в библиотеках Light Gradient Boosting Machine (LightGBM) [15], Extreme Gradient Boosting (XGBoost) [16] и Categorical Boosting (CatBoost) [17]. Рассмотрим каждый из них подробнее.

2.1. L i g h t G B M – это эффективная реализация градиентного бустинга, разработанная для повышения скорости обучения и уменьшения потребления памяти при работе с большими объемами данных [18]. В его основе лежат несколько ключевых идей оптимизации.

1. LightGBM разбивает непрерывные признаки по бинам, образуя гистограммы. По ним алгоритм ищет оптимальные точки разбиения, что позволяет прежде всего снизить вычислительную сложность, так как тогда необходимо обработать ограниченный набор дискретных значений бинов. Данное упрощение также помогает задействовать меньше памяти. Формально это можно описать следующим образом: сначала непрерывный признак l разбивается на k бинов:

$$l \to \left\{b_1, b_2, \ldots, b_k\right\}.$$

Затем для каждого бина b_j вычисляется гистограмма:

$$B_j = \sum_{i \in b_j} g_i,\ g_i = \frac{\partial L\left(y_i, z_i\right)}{\partial z_i},$$

здесь g_i – градиенты для i-го объекта обучающей выборки, в B_j суммируются только попавшие в бин b_j. Далее оптимальное разбиение находится путем поиска максимального прироста информации:

$$Gain\left(s\right) = \frac{1}{2}\left(\frac{G_L^2\left(s\right)}{H_L\left(s\right) + \lambda} + \frac{G_R^2\left(s\right)}{H_R\left(s\right) + \lambda} - \frac{\left(G_L\left(s\right) + G_R\left(s\right)\right)^2}{H_L\left(s\right) + H_R\left(s\right) + \lambda}\right) - \alpha,$$

$$G_L = \sum_{i \in D_L} g_i,\ G_R = \sum_{i \in D_R} g_i,\ g_i = \frac{\partial L(y_i, z_i)}{\partial z_i},$$

$$H_L = \sum_{i \in D_L} h_i,\ H_R = \sum_{i \in D_R} h_i,\quad h_i = \frac{\partial L^2(y_i, z_i)}{\partial z_i^2},$$

где D_L и D_R – подмножества данных, попадающих в левое и правое поддеревья после разбиения. Таким образом, G_L и G_R можно трактовать как соответствующие суммы элементов вектора градиента функции потерь, а H_L и H_R – суммы диагональных элементов матрицы Гессе функции потерь в левых и правых поддеревьях, λ – коэффициент $L2$-регуляризации, α – параметр регуляризации, контролирующий минимальный прирост информации необходимый для осуществления разбиения.

2. В обычном градиентном бустинге используется подход level-wise, при котором дерево растет по уровням, т.е. все листья на текущем уровне добавляются одновременно. LightGBM же применяет метод leaf-wise, при котором дерево растет по листьям: на каждом шаге построения дерева выбирается лист с наибольшей ошибкой для разбиения. Такой подход более эффективен для поиска сложных зависимостей, но есть риск, что дерево будет расти в глубину только по одной ветке, что повлечет переобучение. Такая проблема распространена при обучении на малых объемах данных [19].

3. LightGBM поддерживает параллельное обучение, может эффективно использовать многопроцессорные системы и распределенные вычислительные кластеры, что ускоряет его работу, особенно на больших объемах данных.

2.2. XGBoost – это усовершенствованный алгоритм градиентного бустинга, в котором применяется оптимизация второго порядка и два типа регуляризации [20].

1. XGBoost использует вторые производные функции потерь для более точного подбора оптимального решения. Формально нужно подобрать алгоритм $f(x)$ – скалярную функцию, решающую следующую задачу:

$$\sum_{i=1}^{n} L\left(y_i, a_{m-1}(x_i) + f(x_i)\right) \to \min_f.$$

Для этого функция L раскладывается в ряд Тейлора до второго члена с центром в ответе композиции a_{m-1}:

$$L\left(y_i, a_{m-1}(x_i) + f(x_i)\right) \approx L\left(y_i, a_{m-1}(x_i)\right) + s_i f(x_i) + \frac{1}{2} h_i f^2(x_i),$$

$$s_i = \frac{\partial L(y_i, z_i)}{\partial z_i},\ h_i = \frac{\partial L^2(y_i, z_i)}{\partial z_i^2},\ z_i = a_{m-1}(x_i),\qquad i = \overline{1,n}.$$

Первое слагаемое не зависит от нового базового алгоритма, поэтому можно его не учитывать.

2. XGBoost включает регуляризационные члены, которые предотвращают численные проблемы, связанные с малым значением второй производной, а также добавляют штрафы за количество листьев и норму коэффициентов, помогая избежать переобучения:

$$\Omega = \beta T + \frac{1}{2}\delta \sum_{j=1}^{T} w_j^2.$$

Здесь T – количество листьев в дереве, w_j – вес узла, β и δ – гиперпараметры регуляризации, задаются вручную или путем экспериментального подбора в зависимости от конкретной задачи. Полная функция потерь с регуляризацией и отбрасыванием первого слагаемого имеет вид:

$$\sum_{i=1}^{n} \left[s_i f(x_i) + \frac{1}{2} h_i f^2(x_i) \right] + \beta T + \frac{1}{2}\delta \sum_{j=1}^{T} w_j^2.$$

3. При построении дерева XGBoost использует, как и LightGBM, критерий информативности, зависящий от оптимального вектора сдвига:

$$Gain(s) = \frac{1}{2}\left(\frac{G^2(s)}{H(s) + \lambda} - \frac{\left(G_L^2(s) + G_R^2(s)\right)}{H_L(s) + H_R(s) + \lambda} \right) - \alpha.$$

4. Критерий останова в XGBoost также зависит от оптимального сдвига: дерево перестает расти, когда дальнейшее разбиение не приводит к значительному улучшению критерия информативности.

2.3. C a t B o o s t – библиотека градиентного бустинга, которая специально оптимизирована для работы с категориальными признаками [21].

1. Обычный бустинг требует предварительной обработки категориальных признаков, например с помощью one-hot encoding. Этот метод не создает ложного ранжирования между категориями признака, но может значительно увеличить размерность данных. Существует альтернативный алгоритм кодирования target encoding, который отражает влияние отдельных категорий на целевую переменную [22]. Этот достаточно эффективный метод автоматически применяется в CatBoost:

$$Encoded\_Feature(x_i) = \sum_{j \neq i} I(x_j = x_i) y_j \Big/ \sum_{j \neq i} I(x_j = x_i),$$

где y_j – целевая переменная, $I(x_j = x_i)$ – индикатор функции, равный 1, если $x_j = x_i$. Также можно использовать схему порядковых кодов: она основывается на порядке появления данных. Подобный подход уменьшает риск утечки информации и снижает риск переобучения. Формула порядкового кодирования выглядит следующим образом:

$$Ordered\_Target\_Mean(x_i) = \sum_{j<i} I(x_j = x_i) y_j \Big/ \sum_{j<i} I(x_j = x_i).$$

2. CatBoost применяет новую метрику совокупного убывания (ordered boosting), которая снижает риск переобучения и позволяет создать более устойчивую модель. Идея заключается в применении градиентного бустинга на случайных подвыборках данных в каждой итерации:

$$L(y, \hat{y}) = \sum_{i=1}^{n} l\left(y_i, \hat{y}_{(i)}\right),$$

где $\hat{y}_{(i)}$ – предсказание для i-го примера, основанное на модели, обученной на примерах до i.

3. CatBoost поддерживает также многомерный градиентный бустинг, что позволяет одновременно решать несколько задач регрессии или классификации.

Таким образом, LightGBM рекомендуется использовать в задачах, где важна скорость обучения и низкое потребление памяти. XGBoost наиболее эффективен там, где нужна высокая точность и стабильность модели, особенно на больших объемах данных. CatBoost идеально подходит для датасетов с категориальными признаками благодаря встроенным механизмам кодирования, а также в условиях ограниченного количества информации (табл. 1, рис. 2).

Таблица 1. Сравнение моделей LightGBM, XGBoost, CatBoost

Характеристика	LightGBM	XGBoost	CatBoost
Разработчик	Microsoft	DMLC	Yandex
Год выпуска	2016	2014	2017
Базовые деревья	Несимметричные, рост по листьям	Несимметричные, рост по уровням	Симметричные
Метод разбиения	GOSS	Гистограммный на отсортированных данных	Жадный
Поддержка категориальных признаков	Может интерпретировать порядковые признаки	Необходима предобработка вручную	Может обрабатывать самостоятельно
Текстовые неструктурированные признаки	Не поддерживаются	Не поддерживаются	Несколько встроенных методов, включая Наивного Байеса и мешок слов

Окончание таблицы 1

Борьба с дисбалансом классов	Встроенные методы Weighted Loss и Focal Loss	Встроенные методы Weighted Loss и Focal Loss	Автоматическая балансировка классов и Balanced Cross-Entropy
Скорость обучения	Высокая благодаря росту по листьям и большому числу оптимизаций на разных шагах алгоритма	Медленнее, чем LGBM, но может использовать оптимизации column block и sparsity-aware split finding	Высокая скорость благодаря поддержке категориальных данных и упрощенному перебору гиперпараметров

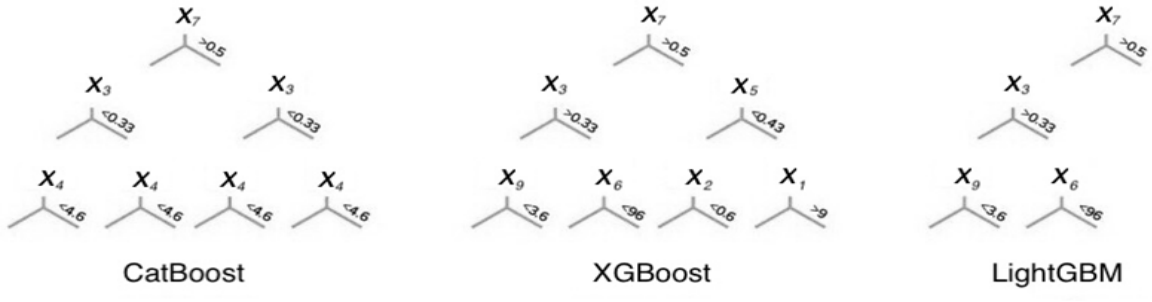

Рис. 2. Сравнение базовых деревьев LightGBM, XGBoost, CatBoost.

Возвращаясь к особенностям решаемой задачи с учетом сравнения алгоритмов бустинга, наиболее подходящим для моделирования в данной задаче будет CatBoost, так как наборы данных для каждого классификатора небольшие, скорость обучения моделей не столь важна, качество стоит на первом месте, в датасете достаточно много категориальных признаков. Таким образом, для обучения каскада моделей классификации в дальнейшем будет использоваться CatBoost.

3. Методы калибровки. Калибровка моделей машинного обучения важна для получения точных вероятностных предсказаний. Числа, которые выдают большинство моделей классификации, CatBoost в частности, отождествляются со степенью уверенности алгоритма в принадлежности того или иного объекта какому-либо классу. Чтобы привести их к чистым интерпретируемым вероятностям, нужна калибровка. В данной задаче это особенно необходимо, так как для формирования списка рекомендаций для каждого пользователя необходимо будет сравнивать вероятности от каждой модели друг с другом.

Цель процедуры калибровки – преобразовать выходные числа модели так, чтобы они соответствовали среднему значению целевой переменной. Для этого необходимо построить калибровочную кривую: разбить отрезок [0, 1] на части (бины) и в каждой рассчитать среднее значение таргета. Полученная кривая в координатах “уверенности модели – калиброванные вероятности” аппроксимируется некоторой функцией (рис. 3), которая затем применяется к значениям уверенности модели; ее результат – искомые вероятности. Остается только выбрать метод аппроксимации и количество частей, на которые следует разбить диапазон [0, 1]. Более того, важно учитывать следующие требования [23, 24]:

1) функция должна быть монотонной, чтобы не было переранжирования вероятностей после калибровки;

2) откалиброванные вероятности должны соответствовать среднему значению целевой переменной внутри бинов;

3) желательно, чтобы после калибровки распределение вероятностей не приобретало разрывный вид;

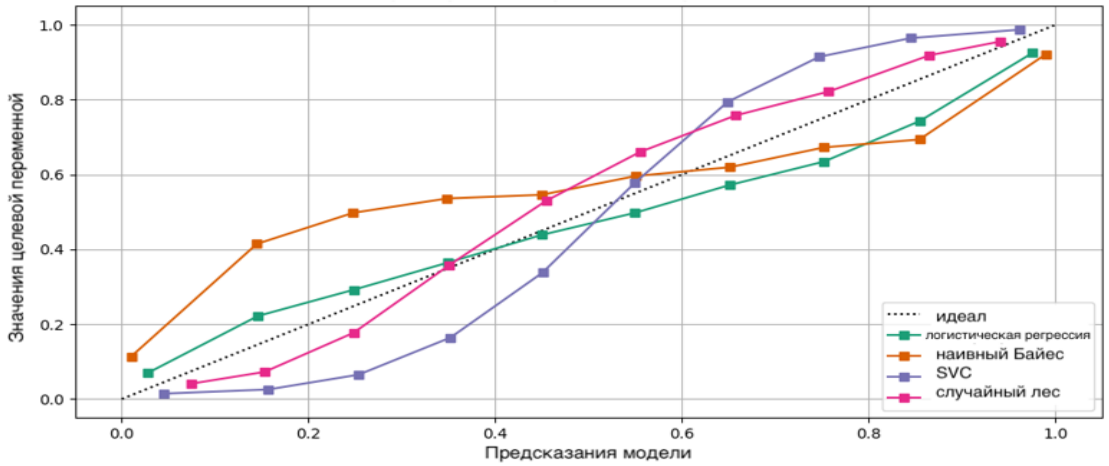

Рис. 3. Примеры калибровочных функций.

4) внутри каждого бина также важно учитывать индивидуальный рост среднего значения целевой переменной.

Если разбить отрезок [0, 1] на несколько частей b_i, $i = \overline{1,n}$ (обычно берется $n = 10$), в каждом посчитать усредненную частоту положительного класса $target\_mean_i$ и среднее значение уверенности модели $conf\_mean_i$, то можно вычислить ошибку калибровки ECE (expected calibration error):

$$target_{mean_i} = \frac{1}{|b_i|}\sum_{j\in b_i} y_i,$$

$$conf\_mean_i = \frac{1}{|b_i|}\sum_{j\in b_i} p_i,$$

$$ECE = \sum_{i=1}^{n}\frac{|b_i|}{N}|target\_mean_i - conf\_mean_i|,$$

где y_i – истинные значения целевой переменной, p_i – предсказания модели, N – размерность данных. Рассмотрим основные алгоритмы калибровки.

3.1. Изотоническая регрессия. Это непараметрический метод, который особенно эффективен, когда предсказания модели показывают сильные отклонения и не следуют линейной зависимости. Калибровочная кривая подбирается как кусочно-постоянная неубывающая функция (рис. 4). На вход алгоритму подаются исходные предсказания, в качестве целевой переменной выступает исходная, а в ходе обучения подбираются интервалы постоянства и значения функции на них путем минимизации среднеквадратичной ошибки [25, 26].

Формализуем подход: пусть имеется набор данных (x_i, y_i), где $i = \overline{1,n}$, x_i – предсказанные значения, а y_i – истинные метки классов. Изотоническая регрессия ищет монотонно неубывающую функцию f, такую, что

$$f(x_1) \le f(x_2) \le \ldots \le f(x_n).$$

Она минимизирует следующую функцию потерь:

$$\sum_{i=1}^{n}(y_i - f(x_i))^2 \to \min.$$

Наиболее часто используемый алгоритм для решения задачи изотонической регрессии – PAVA (pool adjacent violators algorithm). Его принцип работы заключается в следующем:

1) инициализация: сначала предполагается, что $f(x_i) = y_i \quad i = \overline{1,n}$;

2) проверяются пары смежных значений y_i и y_{i+1}: если обнаружено, что $y_i > y_{i+1}$, эти значения объединяются в один отрезок и заменяются средним:

$$y_{new} = \frac{y_i n_i + y_{i+1} n_{i+1}}{n_i + n_{i+1}},$$

где n_i и n_{i+1} – количество элементов в бинах до объединения;

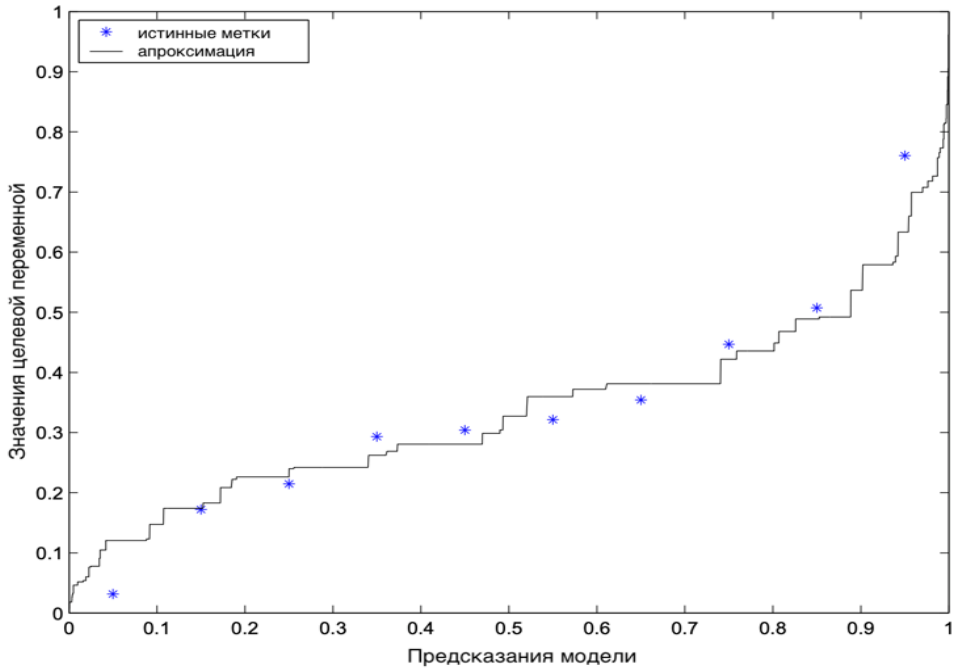

Рис. 4. Пример калибровки изотонической регрессией.

3) процесс повторяется до тех пор, пока все значения не будут следовать монотонно неубывающей последовательности.

Главные преимущества изотонической регрессии в том, что не нужно никаких предположений о распределении данных, алгоритм хорошо работает со сложными нелинейными зависимостями. Основные недостатки – склонность к переобучению, чувствительность к выбросам, нарушение ранжирования внутри бинов и необходимость большого объема данных для стабильной оценки.

3.2. К а л и б р о в к а П л э т т а. Данный метод был предложен Джоном Плэттом для калибровки SVM (support vector machine) [27]. Идея заключается в применении логистической регрессии к выходам модели:

$$\mathrm{P}\left(y = 1x\right) = \frac{1}{1 + \exp\left(A f\left(x\right) + B\right)},$$

где A и B – параметры, которые необходимо подобрать через максимизацию логарифмического правдоподобия:

$$\min_{A,B} \sum_{i=1}^{n} \left[y_i \log\left(\mathrm{P}\left(y_i = 1\#f\left(x_i\right)\right)\right) + \left(1 - y_i\right)\log\left(1 - \mathrm{P}\left(y_i = 1\#f\left(x_i\right)\right)\right)\right],$$

$$\mathrm{P}\left(y_i = 1f\left(x_i\right)\right) = \frac{1}{1 + \exp\left(A f\left(x_i\right) + B\right)}.$$

Далее можно подставить в формулу найденные коэффициенты и использовать ее для преобразования оценок [28]:

$$\mathrm{P}\left(y = 1x\right) = \frac{1}{1 + \exp\left(A f\left(x\right) + B\right)}.$$

Данный метод популярен из-за его простоты и эффективности для калибровочных функций, которые хорошо описываются сигмоидой (рис. 5). Это одновременно и недостаток, так как для сложных кривых он плохо применим. Также стоит отметить, что для получения качественных результатов необходимо значительное количество данных.

3.3. Т е м п е р а т у р н о е ш к а л и р о в а н и е. Этот метод основывается на попытке аппроксимировать калибровочную кривую сигмоидальной функцией, только логиты – действительные числа или вектора, которые формирует модель, – предварительно делятся на

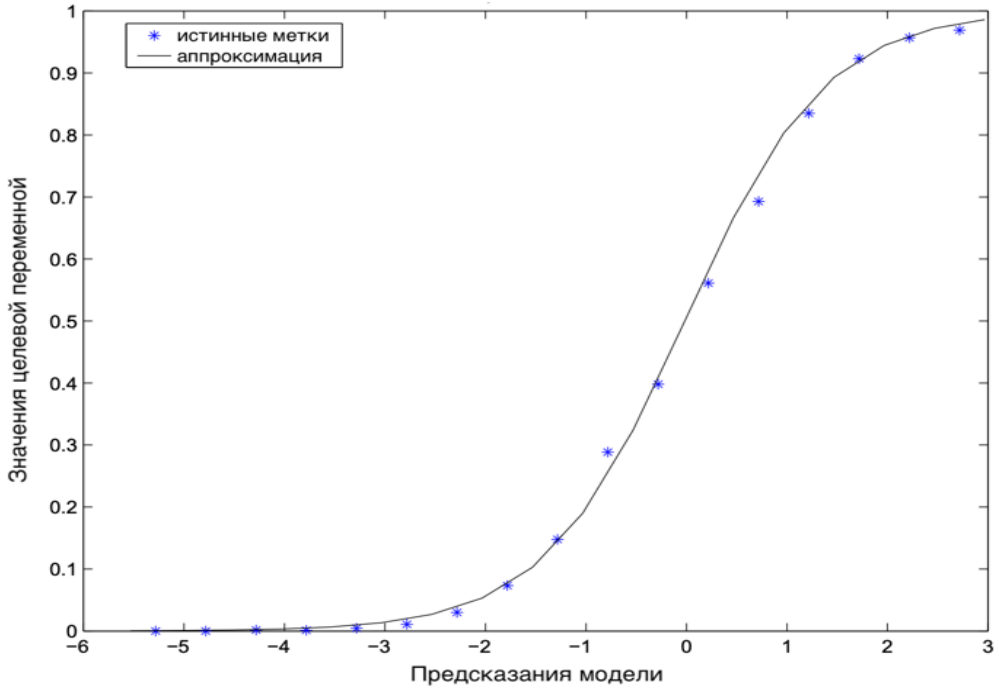

Рис. 5. Пример калибровки Плэтта.

общий коэффициент температуры T [29]. Он подбирается так, чтобы минимизировать логарифмическую функцию потерь на валидационном наборе данных:

$$\min_T -\frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}_i \log\left(p_i\right) + (1-\hat{y}_i)\log\left(1-p_i\right)\right),$$

где $\hat{y}_i$ – истинные значения целевой переменной, $\hat{p}_i$ – предсказанные моделью значения.

Далее вычисляются корректированные логиты z' путем деления исходных логитов z на значение температуры T, и с их помощью рассчитываются откалиброванные вероятности:

$$z' = \frac{z}{T},\ \hat{p}_i = \exp\left(\frac{z_i}{T}\right)\Big/\sum_{j=1}^{n}\exp\left(\frac{z_j}{T}\right).$$

Алгоритм температурного шкалирования часто применяется для современных глубоких нейронных сетей, он прост в использовании и мало переобучается, так как зависит только от одного параметра – температуры. Но опять же, он качественно аппроксимирует только калибровочные кривые, которые близки по виду к сигмоиде или softmax (рис. 6).

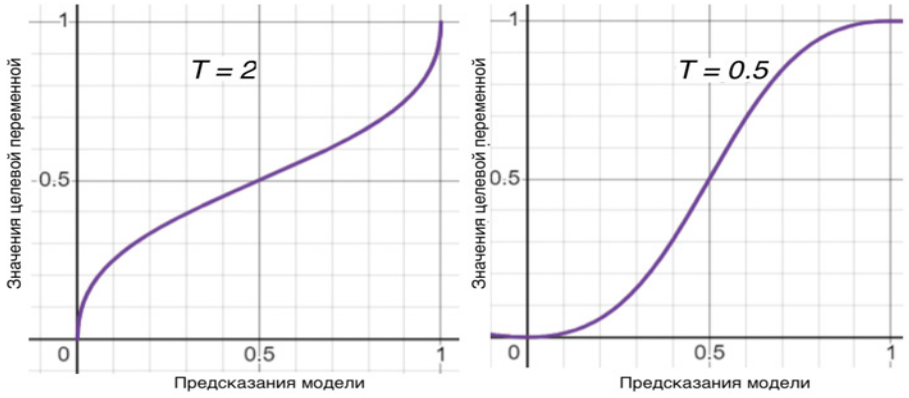

Рис. 6. Примеры калибровки температурным шкалированием.

3.4. Гистограммная калибровка. Данный метод отчасти похож на изотоническую регрессию: отрезок [0, 1] разбивается на части, внутри каждой из которых происходит аппроксимация калибровочной кривой. Отличие в том, что количество и длина этих подотрезков задаются вручную. Можно применять различные модификации для подбора функции на каждом из них, решая таким образом важные проблемы, которые свойственны классическим подходам.

Первым делом необходимо разбить отрезок [0, 1] на бины $b_1,\ldots,b_n$: можно сделать это так, чтобы они были одинаковой длины или же одинаковые по количеству наблюдений (равномощные) [30]. Простейшая постановка задачи – это подобрать для каждого бина некое число θ_i, которое и будет искомой вероятностью. Подбираются они так, чтобы θ_i, $i = 1,\ldots,n$, как можно лучше приближали средние метки классов на соответствующих подотрезках:

$$\sum_{i=1}^{n}\left|\sum_{j=1}^{N} I\left\{q\left(x_j\right) \in b_i\right\} y_i \,/\, |b_i| - \theta_i\right| \to \min_{(\theta_1,\ldots,\theta_n)},$$

где N – размерность данных.

Недостаток такого подхода в том, что в каждом бине предсказания аппроксимируются одним числом, из-за чего теряется их внутреннее ранжирование. Это можно исправить, подобрав не просто числа θ_i, а также коэффициенты наклона k_i. Таким образом, внутри каждого подотрезка приближение будет не константной функцией, а линейной (рис. 7):

$$\frac{1}{|b_i|}\sum y_n = k_i \frac{1}{|b_i|}\sum_{n\in b_i} q\left(x_n\right) \Rightarrow k_i = \frac{\text{mean}_{n\in b_i}\left(y_n\right)}{\text{mean}_{n\in b_i}\left(q\left(x_n\right)\right)}.$$

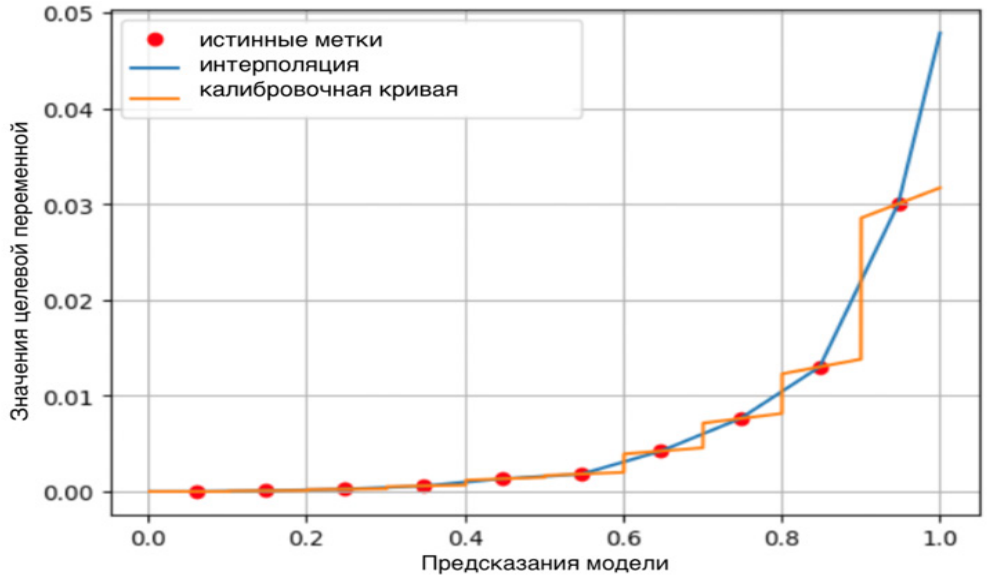

Рис. 7. Пример гистограммной калибровки с одним коэффициентом.

Но тогда может появиться еще одна весомая проблема: нарушения общей монотонности функции, особенно на участках медленного роста калибровочной кривой (рис. 8). Это происходит из-за того, что найденные для каждого бина линейные зависимости обязательно проходят через точку (0, 0). Устранить данный недостаток можно простым добавлением коэффициента смещения b в уравнение (рис. 9). Тогда при поиске оптимальной прямой необходимо руководствоваться также условием, что общая функция калибровки должна быть неубывающей. Упрощенно можно было бы решать эту задачу как построение линейной регрессии на каждом из участков [31].

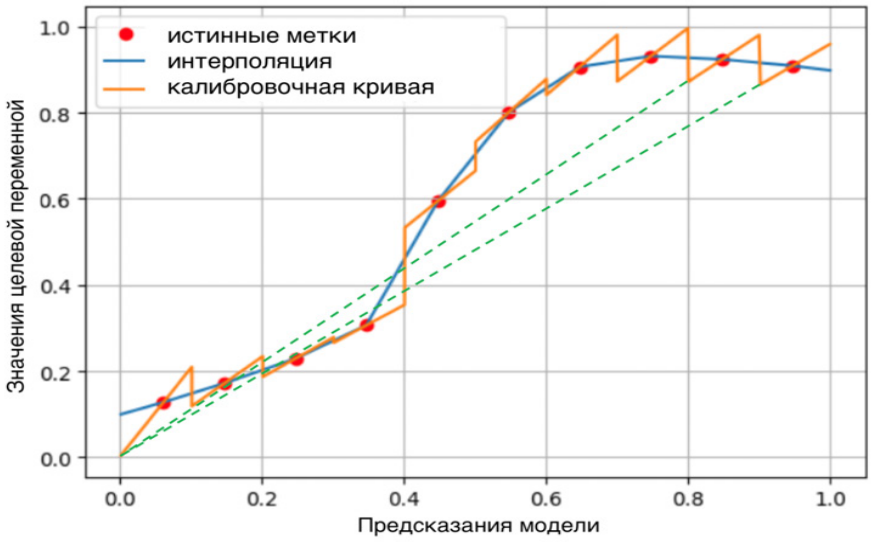

Рис. 8. Пример нарушения ранжирования при гистограммной калибровке с одним коэффициентом.

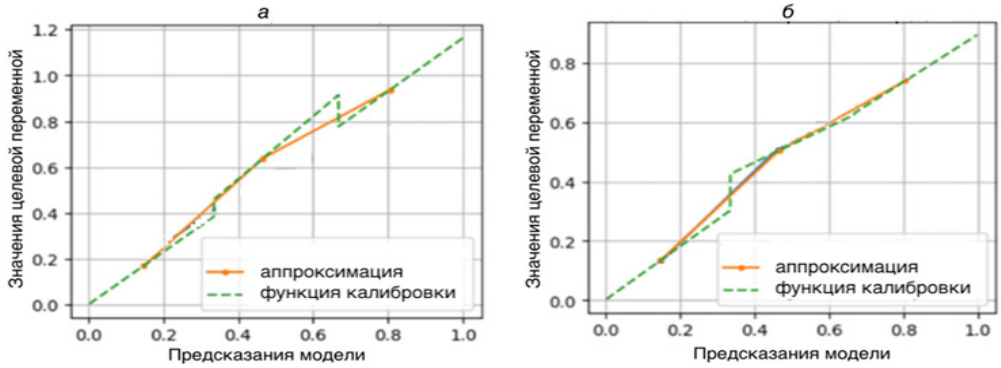

Рис. 9. Сравнение гистограммной калибровки с одним (а) и двумя (б) коэффициентами в критическом случае.

Рассмотрим пример поиска такой калибровочной кривой для случая деления [0, 1] на три части. Формализуем каждое из условий для нахождения решения.

1. Средняя преобразованная вероятность должна быть равна среднему значению целевой переменной в каждом бине:

$$\begin{cases} k_1 s_1 + b_1 = t_1, \\ k_2 s_2 + b_2 = t_2, \\ k_3 s_3 + b_3 = t_3. \end{cases}$$

2. Общая функция калибровки должна быть неубывающей и кусочно непрерывной на [0, 1]:

$$\begin{cases} 0 \le k_1 l_1 + b_1, \\ k_1 r_1 + b_1 = k_2 r_2 + b_2, \\ k_2 r_2 + b_2 = k_3 r_3 + b_3, \\ k_3 r_3 + b_3 \le 1, \\ k_1, k_2, k_3 > 0. \end{cases}$$

В уравнениях выше s_i – средние значения исходных предсказаний модели в i-м подотрезке, t_i – средние значения целевой переменной, l_i и r_i – левая и правая границы исходных предсказаний.

Таким образом, количество неизвестных в первой системе равно шести, а уравнений всего три. В совокупности с требованиями второй системы возможных решений будет либо бесконечно много, либо вообще не будет (в случае, если средние значения целевой переменной убывают по бинам “слева направо”). Если посмотреть на задачу геометрически, становится понятно, что чем меньше коэффициент k, тем более сжатыми будут распределения вероятностей в каждом бине. Это нежелательно, так как тогда преобразованные предсказания разделятся на дискретные части; лучше, если они будут образовывать равномерное распределение.

Поэтому дополнительно потребуем, чтобы ширина преобразованных вероятностей была максимальна: $\Delta y = k\Delta x \to \max$; технически это означает максимизацию коэффициента k, а также чтобы расстояния между соответствующими крайними точками в соседних бинах были минимальными [32].

Самый тривиальный способ решения такой задачи – одновременно минимизировать два выражения: разницу между $k_i s_i + b_i$ и t_i (например, по формуле среднеквадратичной ошибки) и средний квадрат расстояний между бинами; оставшиеся неравенства из систем выше будут являться ограничениями. На практике такой способ подбирает коэффициенты с достаточно малыми расстояниями между бинами, но ошибка калибровки получается больше, так как минимизируется разница между средним таргетом и средней вероятностью, их равенства не требуется.

Альтернативный вариант – использовать следующий алгоритм.

1. Выразим коэффициенты b_i из уравнений первой системы, тогда искомые калибровочные функции для каждого бина преобразуются в $y_i(x) = k_i(x - s_i) + t_i$.

2. Для каждого бина находим потенциальных кандидатов на роль коэффициента $k_i : k_l = 0.5(t_{i-1} + t_i)$ и $k_r = 0.5(t_i + t_{i+1})$. Геометрически эти условия означают поиск середины между средними значениями целевых переменных в соседних бинах.

3. Далее выбираем из кандидатов один итоговый коэффициент как $k_i = \min(k_l, k_r)$. Это условие гарантирует, что преобразованные вероятности в соседних бинах не будут пересекаться.

4. На данном этапе между заполненными диапазонами могут быть значительные расстояния. Чтобы это исправить, необходимо увеличивать k_i, пока крайние значения вероятностей не начнут пересекаться с ближайшим соседом. Единственное, что могут быть ограничения на расширение бинов из-за соседних, а также из-за того, что нельзя выходить за пределы [0, 1].

Преимущество этого подхода в том, что ошибка калибровки в нем отсутствует. Но могут быть расстояния между крайними значениями вероятностей в соседних бинах.

Из перечисленных алгоритмов калибровка Плэтта и температурное шкалирование не будут эффективными в рассматриваемой задаче, так как не для всех классификаторов калибровочная кривая хорошо описывается сигмоидой. Ввиду широких возможностей для тонкой настройки алгоритма остановимся на методе гистограммной калибровки, а именно последнем описанном подходе: с подбором двух коэффициентов прямой для каждого диапазона и максимизацией k наряду с минимизацией расстояния между соседними бинами. По сравнению с простейшей реализацией этого алгоритма и изотонической регрессией, модифицированный подход не так сильно переобучается и решает проблему потери ранжирования между предсказаниями внутри отдельных бинов, оставляя при этом распределение преобразованных вероятностей максимально равномерным (табл. 2).

Таблица 2. Сравнение различных алгоритмов калибровки

Алгоритм	Преимущества	Недостатки
Изотоническая регрессия	Достаточно точно приближает калибровочную кривую любой формы	Нарушается ранжирование внутри участков постоянства, склонность к переобучению
Калибровка Плэтта	Подбирается только два коэффициента, почти не переобучается	Эффективна только в случаях, когда калибровочная кривая хорошо описывается сигмоидой
Температурное шкалирование	Подбирается только два коэффициента, почти не переобучается, не требуется дополнительных моделей	Качественно приближает лишь ограниченный класс функций

Окончание таблицы 2

Гистограммная калибровка	Можно достаточно точно приблизить калибровочную кривую любой формы, возможности для модификации решают важные проблемы переранжирования внутри бинов, непрерывности функции, нестабильности целевой переменной	Сильно зависит от выбора разбиения по бинам, также необходимо подбирать количество коэффициентов в зависимости от сложности задачи

Примем во внимание еще некоторые особенности решаемой задачи и добавим к улучшенному алгоритму гистограммной калибровки несколько необходимых модификаций.

1. В дальнейшем необходимо будет сравнивать выходы моделей друг с другом. Но если их предсказания в среднем сильно отличались друг от друга до калибровки, эта же тенденция может сохраниться и после. Например, у одной модели все уверенности могут быть расположены на отрезке от 0 до 0.5, а у другой – от 0.5 до 1. Тогда при формировании итоговых рекомендаций объект, соответствующий второй модели, будет всегда выше первого (рис. 10). Это косвенно может быть связано с так называемой проблемой “популярности”: чем больше пользователей взаимодействуют с некоторым объектом, тем выше для него будет среднее значение целевой переменной и тем выше значения предсказаний как до, так и после калибровки. Исправить ситуацию можно нормализацией выходных значений модели, например привести их к логнормальному распределению:

$$f(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}}\exp\left(-\frac{(\ln x-\mu)^2}{2\sigma^2}\right),\ x>0.$$

Его особенность состоит в том, что логарифм от него дает нормальное распределение: $X \approx \log N\left(\mu,\sigma^2\right) \Rightarrow \ln(X) \approx N\left(\mu,\sigma^2\right)$. После применения такого преобразования распределения предсказаний моделей становятся более схожими, можно регулировать их также при помощи коэффициента сдвига по оси *Ox*.

2. Другой немаловажный момент заключается в том, что для некоторых объектов средние значения целевой переменной ведут себя нестабильно во времени. Поэтому если ориентироваться на одно число внутри каждого бина, повышается риск переобучения и снижается обобщающая

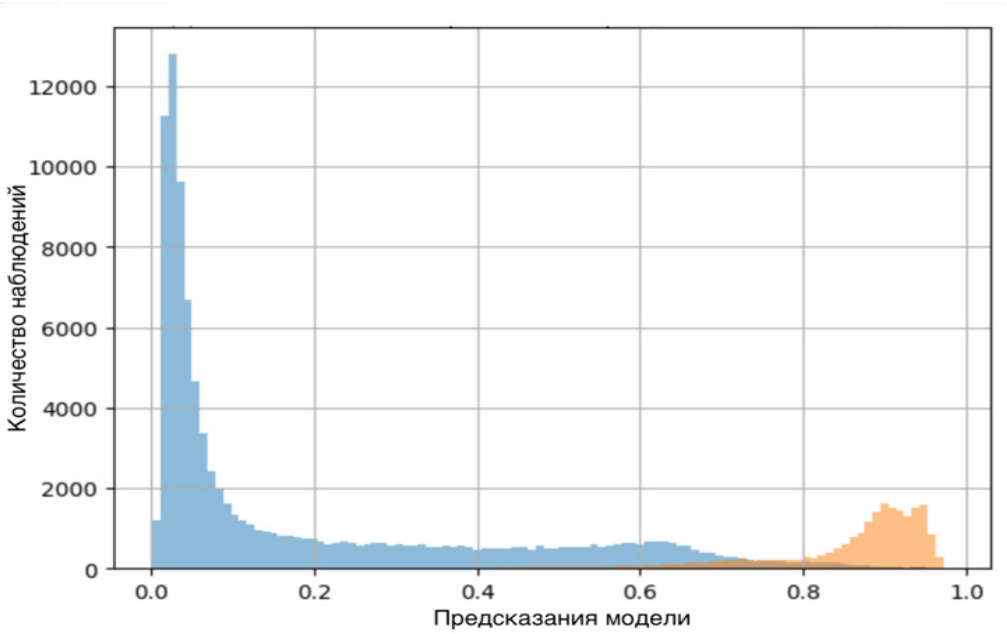

Рис. 10. Пример неравномерно распределенных предсказаний двух моделей.

способность модели. Попробовать исправить эту проблему можно через аппроксимацию кумулятивного среднего значения целевой переменной во времени. Это позволит сгладить выбросы и выявить тренд, что сделает модель более устойчивой. Зависимости целевых переменных во времени достаточно хорошо описываются степенной функцией со сдвигом вида $(kx + b)^n$. Другими словами, необходимо подобрать коэффициенты k, b и показатель степени n так, чтобы итоговая функция наилучшим образом проходила через имеющиеся точки значений таргета (рис. 11). Далее коэффициент наклона для участка калибровочной кривой будет вычисляться как отношение предсказанного тренда и фактического значения целевой переменной (рис. 12):

$$linear\_coef = \frac{expected\_mean\_target}{fact\_mean\_target} = \frac{\min\left(1,(kx+b)^n\right)}{fact\_mean\_target}.$$

Теперь, когда все методы решения и особенности задачи определены, можно приступать к применению их на практике.

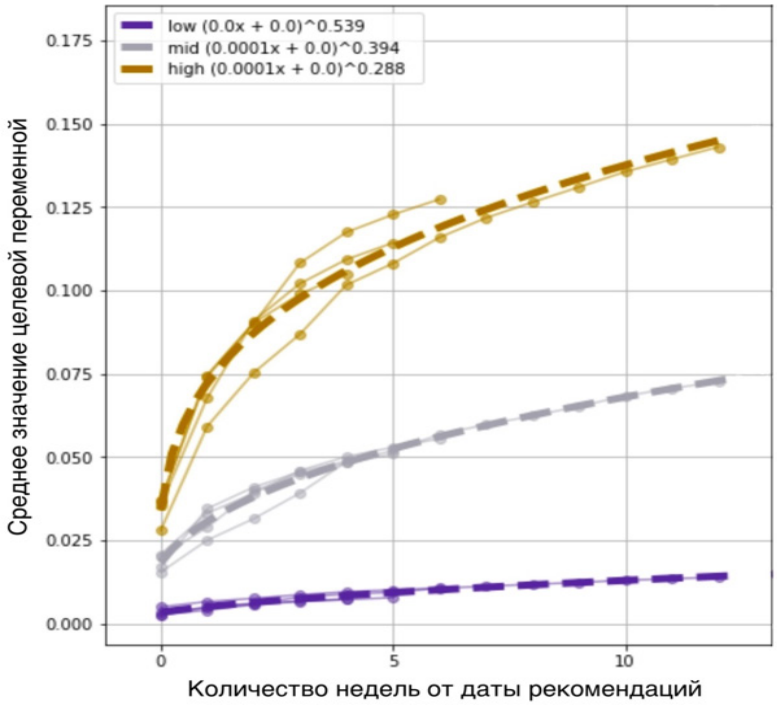

Рис. 11. Аппроксимация кумулятивной доли значений целевой переменной в разрезе групп по величине склонности.

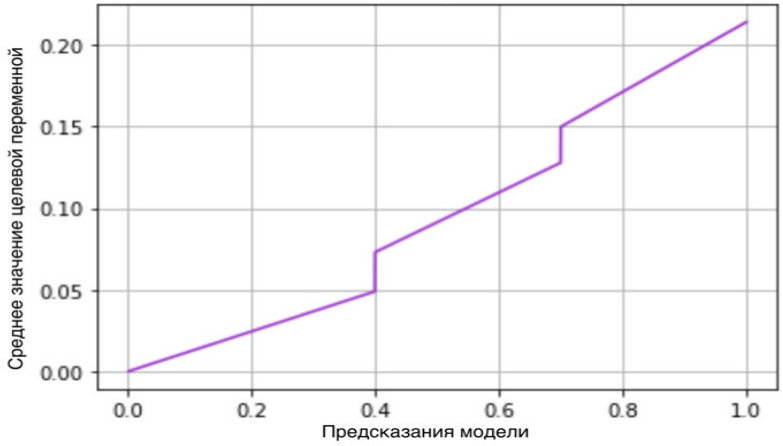

Рис. 12. Калибровочная функция для трех бинов с аппроксимацией таргета.

4. Проведение эксперимента. На основе исследования осталось собрать данные, обучить модели, откалибровать их и собрать в единую систему. Последовательность основных действий и результаты следующие.

1. Сбор данных. Для каждого классификатора необходимо сформировать отдельный набор данных, суммарно их получится 10 штук. В состав пользователей должны входить только те, которые не имеют соответствующего объекта на дату выдачи рекомендаций моделью: мета-информация по пользователям и объектам и агрегированные коллаборативные факторы (для новых пользователей заполняются нулями). В качестве периода формирования обучающего датасета и срока готовности целевой переменной возьмем значения $T_1 = 3$ мес., $T_2 = 1$ мес. (рис. 1); размер выборки для out-of-time тестирования – 1 мес. (рис. 13). Для некоторых объектов наблюдается дисбаланс в целевой переменной (рис. 14), в дальнейшем это можно исправить коэффициентами весов для классов при обучении.

2. Предобработка данных. Стандартно предлагается удалить неинформативные и уникальные признаки с высокой долей пропущенных значений (более 70%), а затем отобрать наиболее значимые по SHAP-значениям (shapley additive explanations). Последнее необходимо для снижения риска переобучения и ускорения процесса обучения [33, 34], в качестве инструмента используется встроенный алгоритм CatBoost Feature Selector [21].

3. Обучение моделей и подбор гиперпараметров. Для всех 10 экспериментальных объектов обучаются отдельные модели классификации CatBoostClassifier [21]. Для подбора гиперпараметров выделяется 30% от обучающего набора данных, для оптимизации процесса используется optuna [35]. Часть гиперпараметров фиксируется вручную (табл. 3), на них в совокупности с наилучшими подобранными значениями обучаются финальные модели (табл. 4).

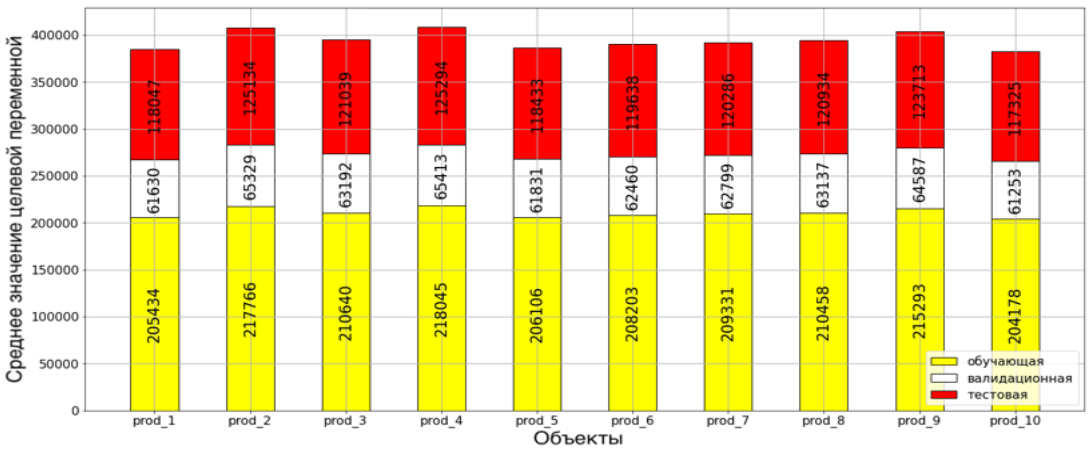

Рис. 13. Размеры выборок для классификаторов.

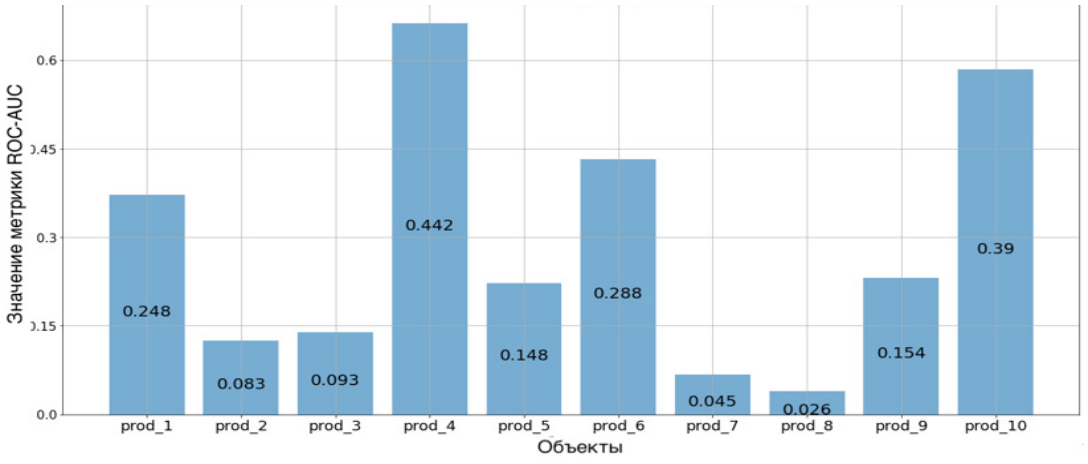

Рис. 14. Доли целевых событий в тренировочных выборках.

Таблица 3. Фиксированные гиперпараметры моделей

Гиперпараметр	Значение
iterations	1000
loss_function	‘Logloss’
one_hot_max_size	5
boosting_type	‘Ordered’
n_estimators	50
scale_pos_weight	w*

* Задается для каждой модели вручную как отношения размера отрицательного класса к положительному.

Таблица 4. Оптимизируемые гиперпараметры моделей

Гиперпараметр	Диапазон для подбора и шаг
learning_rate	[0.001, 0.1], 0.001
depth	[2, 8], 1
l2_leaf_reg	[1, 10], 1
bagging_temperature	[0.01, 0.9], 0.05
border_count	[32, 255], 5
min_data_in_leaf	[5, 100], 5
min_child_samples	[10, 200], 1
subsample	[0.2, 1], 0.1
rsm	[0.1, 1], 0.1
random_strength	[0, 10], 0.1

4. Промежуточная оценка качества. Для полученных классификаторов можно уже рассчитать некоторые метрики: в качестве таковой рассматривается ROC-AUC (receiver operating characteristic area under curve), так как она не зависит от порога и позволяет оценить общую ранжирующую способность обученных алгоритмов (рис. 15).

5. Калибровка предсказаний. Для перехода от степени уверенности модели к интерпретируемым значениям вероятности производится калибровка улучшенным гистограммным методом, описанным ранее.

6. Составление списка рекомендаций. На данном этапе остается соединить все вероятности в единую рекомендательную систему: для каждого пользователя отсортировать все имеющиеся для него значения по убыванию, это и будет итоговый список рекомендаций. Оценка качества полученной системы проводится по метрикам Hit Rate, Precision@3 и Precision@5 (табл. 5).

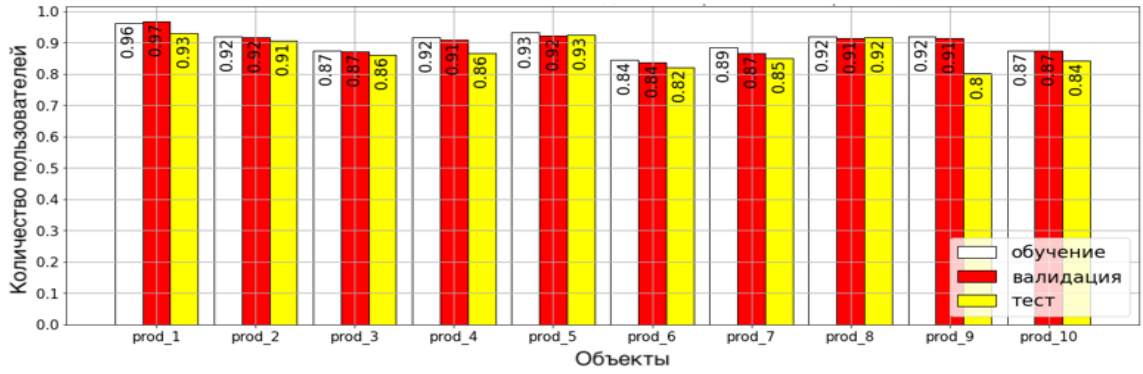

Рис. 15. Значения ROC-AUC всех моделей на разных выборках.

Таблица 5. Значения метрик итоговой рекомендательной системы

Названия метрик	Значение
Hit Rate	0.0859
Precision@3	0.0462
Precision@5	0.0352

Заключение. Построена рекомендательная система, которая может быть использована для нестандартного случая малого числа уникальных объектов в условиях высокой корреляции между некоторыми из них. Для этого предложен способ обучения отдельных классификаторов по каждому объекту. В данных представлены как метапризнаки, так и агрегаты коллаборативных, модели обучены на предварительно подобранных гиперпараметрах. В конце производится калибровка для перехода к вероятностям и формируются списки итоговых рекомендаций по пользователям.

В качестве алгоритма для обучения выбран бустинг, так как наборы данных по каждому объекту небольшие по размеру и хорошо структурированы. Конкретная выбранная реализация – модель CatBoost из-за ее способности гибко обрабатывать категориальные признаки, а также высокого качества на открытых данных. В процессе выбора метода калибровки обнаружен ряд недостатков стандартных методов, которые могут быть решены дополнительными надстройками для легко адаптируемого алгоритма гистограммной калибровки.

Промежуточное качество классификаторов достаточное, отдельные модели показывают значительную ранжирующую способность на соответствующих выборках. Меры качества итоговой рекомендательной системы также высокие, причем наблюдается нелинейное повышение точности на k первых объектах с ростом k, что свидетельствует о том, что финальный алгоритм способен правильно ранжировать подходящие объекты по релевантности.

Результат может быть применен для улучшения качества работы рекомендательных алгоритмов при малом числе уникальных объектов, часть из которых взаимосвязана друг с другом. Описанный подход позволяет независимо получить вероятности заинтересованности пользователей во всех объектах, которые могут быть использованы для любых других целей. Технически метод гибкий в плане добавления новых объектов в общий набор и предоставляет широкие возможности для быстрого переобучения и повышения качества отдельных классификаторов.

СПИСОК ЛИТЕРАТУРЫ

1. *Cano E., Morisio M.* Hybrid Recommender Systems: A Systematic Literature Review // Intelligent Data Analysis. 2017. V. 21. P. 1487–1524.
2. *Al-bashiri H., Abdulhak M., Romli A., Hujainah F.* Collaborative Filtering Recommender System: Overview and Challenges // J. Computational and Theoretical Nanoscience. 2017. V. 23. P. 9045–9049.
3. *Jahrer M., Toscher A.* Collaborative Filtering Ensemble // J. Machine Learning Research. 2012. V. 18. P. 61–74.
4. *Ahn H., Kang H., Lee J.* Selecting a Small Number of Products for Effective User Profiling in Collaborative Filtering // Expert Systems with Applications. 2010. V. 37. P. 3055–3062.
5. *Zharova M., Tsurkov V.* Neural Network Approaches for Recommender Systems // J. Computer and Systems Sciences International. 2024. V. 62. P. 1048–1062.
6. *Castells P., Moffat A.* Offline Recommender System Evaluation: Challenges and New Directions // AI Magazine. 2022. V. 43. P. 225–238.
7. *Bokde D., Girase S., Mukhopadhyay D.* Matrix Factorization Model in Collaborative Filtering Algorithms: A Survey // Procedia Computer Science. 2015. V. 49. P. 136–146.
8. *Filho T., Song H., Perello-Nieto M.* Classifer Calibration: a Survey on How to Assess and Improve Predicted Class Probabilities // Machine Learning. 2023. P. 3211–3260.
9. *Alzubaidi L., Bai J., Al-Sabaawi A.* A Survey on Deep Learning Tools Dealing with Data Scarcity: Definitions, Challenges, Solutions, Tips, and Applications // J. Big Data. 2023. V. 10. № 46.
10. *Grinsztajn L., Oyallon E., Varoquaux G.* Why do Tree-based Models Still Outperform Deep Learning on Tabular Data? // arXiv:2207.08815v1, 2022.
11. *Alzubaidi L., Zhang J., Humaidi A.* Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions // arXiv:2207.08815v1, 2022.
12. *Borisov V., Leemann T., Sebler K.* Deep Neural Networks and Tabular Data: A Survey // arXiv:2110.01889v3, 2022.
13. *Bentejac C., Csorgo A., Martinez-Munoz G.* A Comparative Analysis of Gradient Boosting Algorithms // Artificial Intelligence Review. 2020. V. 54. P. 1937–1967.
14. *Sahour H., Gholami V., Torkaman J.* Random Forest and Extreme Gradient Boosting Algorithms for Streamflow Modeling Using Vessel Features and Tree-rings // Environmental Earth Sciences. 2021. V. 80. № 747.

15. Имплементация модели LightGBM на Python // GitHub. Microsoft LightGBM: webcite https://github.com/microsoft/LightGBM (accessed: 10.07.2024).
16. Имплементация модели XGBoost на Python // GitHub. Distributed (Deep) Machine Learning Community XGBoost: webcite https://github.com/dmlc/xgboost (accessed: 10.07.2024).
17. Имплементация модели CatBoost на Python // GitHub. CatBoost: webcite https://github.com/catboost/catboost (accessed: 10.07.2024).
18. *Ke1 G., Meng Q., Finley T.* LightGBM: A Highly Efficient Gradient Boosting Decision Tree // Advances in Neural Information Processing Systems. 2017. P. 3146–3154.
19. Эксперименты с моделью LightGBM // Kaggle. LightGBM experiments: webcite https://www.kaggle.com/code/prashant111/lightgbm-classifier-in-python (accessed: 10.07.2024).
20. *Chen T., Guestrin C.* XGBoost: A Scalable Tree Boosting System // arXiv:1603.02754v3, 2016.
21. *Dorogush A., Prokhorenkova L., Gusev G.* CatBoost: Unbiased Boosting with Categorical Features // arXiv:1706.09516v5, 2019.
22. *Pargentn F., Pfisterer F., Thomas J., Bischl D.* Regularized Target Encoding Outperforms Traditional Methods in Supervised Machine Learning with High Cardinality Features // Computational Statistics. 2022. V. 37. P. 2671–2692.
23. *Niculescu-Mizil A., Caruana R.* Predicting Good Probabilities with Supervised Learning // Machine Learning, Proc. 22nd Intern. Conf. (ICML). Bonn, Germany, 2005. P. 625–632.
24. *Guo C., Pleiss G., Sun Y., Weinberger K.* On Calibration of Modern Neural Networks // arXiv:1706.04599v2, 2017.
25. *Barlow R., Bartholomew D., Bremner J., Brunk H.* Statistical Inference under Order Restrictions: The Theory and Application of Isotonic Regression // Royal Statistical Society. Series A: General. 1974. V. 137. P. 92–93.
26. *Zadrozny B., Elkan C.* Transforming Classifier Scores into Accurate Multiclass Probability Estimates // The Eighth ACM SIGKDD Intern. Conf. on Knowledge Discovery and Data Mining. Edmonton, 2002.
27. *Platt J.* Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods // Advances in Large Margin Classifiers. Cambridge: MIT Press, 2000. P. 61–74.
28. *Zadrozny B., Elkan C.* Transforming Classifier Scores into Accurate Multiclass Probability Estimates // Proc. 8th ACM SIGKDD Intern. Conf. on Knowledge Discovery and Data Mining. N. Y., 2002. P. 694–699.
29. *Guo C., Pleiss G., Sun Y., Weinberger K.* On Calibration of Modern Neural Networks // Machine Learning, Proc. 34th Intern. Conf. (ICML). Sydney, 2017.
30. *Gupta C., Ramdas A.* Distribution-free Calibration Guarantees for Histogram Binning without Sample Splitting // arXiv:2105.04656v2, 2021.
31. *Naeini M., Cooper G.* Binary Classifier Calibration Using an Ensemble of Piecewise Linear Regression Models // Knowledge and Information Systems. 2018. V. 54. P. 151–170.
32. *Filho T., Song H., Perello-Nieto M.* Classifier Calibration: a Survey on How to Assess and Improve Predicted Class Probabilities // Machine Learning. 2023. V. 112. P. 3211–3260.
33. *Wang H., Liang Q., Hancock J., Khoshgoftaar T.* Feature Selection Strategies: a Comparative Analysis of SHAP-value and Importance-based Methods // J. Big Data. 2024. V. 11. № 44.
34. *Gebreyesus Y., Dalton D., Nixon S., Chiara D., Chinnic M.* Machine Learning for Data Center Optimizations: Feature Selection Using Shapley Additive exPlanation (SHAP) // Future Internet. 2023. V. 15. № 88.
35. Имплементация библиотеки для подбора гиперпараметров Optuna на Python // GitHub. Optuna: webcite https://github.com/optuna/ optuna (accessed: 20.07.2024).