МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М.В.Ломоносова

ЭКОНОМИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА МАТЕМАТИЧЕСКИХ МЕТОДОВ АНАЛИЗА ЭКОНОМИКИ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

«Применение методов машинного обучения для построения рекомендательной системы для онлайн-ритейла»

Выполнил студент

Группы э403

Кузнецов Никита Петрович

Научный руководитель:

Андрейцев Антон Игоревич

Москва

2022

Аннотация

При поиске в онлайн-магазинах у людей часто возникает проблема с нахождением продукта, который бы в полной мере удовлетворял потребности их изначального запроса. Это может приводить к недопотреблению товаров, которые по какой-либо причине не оказались в начале списка, выданного по запросу. То есть клиенты не в полной мере удовлетворяют свои потребности, а производители получают меньше прибыли, чем могли бы. Данная работа направлена на построение рекомендательной системы для магазина одежды, которая бы лучше всего прогнозировала потребности клиентов и сортировала бы на основании этого товары в наличии. Для реализации системы использовались внутренние данные магазина Spin4Spin по поведению пользователей на сайте при выборе одежды. Были построены модели коллаборативной фильтрации контента на основании ближайших соседей с разными метриками расстояния (косинусная, евклидова) и градиентного бустинга над решающими деревьями. Полученные результаты показывают, что модель градиентного бустинга примерно в 5 раз превосходит модель коллаборативой фильтрации по оцениваемым метрикам сортировки

JEL: C 51, C 53, C 60, C 61

Ключевые слова: рекомендательные системы, онлайн-ритейл

Оглавление

Введение и постановка задачи

У пользователей онлайн сервисов и магазинов при увеличении объемов контента или товаров появляется проблема с поиском того, что сможет удовлетворить их потребности. Задача магазина в данном случае – предоставить пользователю услугу по сортировке товаров в наличии, которые бы наилучшим образом подходили пользователю. В оффлайн-магазинах за это отвечают продавцы-консультанты. В онлайн-сфере эта задача решается с помощью рекомендательных систем.

Задачей данной работы является написание рекомендательной системы для онлайн-магазина одежды Spin4Spin[1] и выявление наиболее удачной модели рекомендаций для онлайн-ритейла одежды на основе конкретных метрик. Магазин специализируется на перепродаже одежды и аксессуаров премиум-брендов и известных дизайнеров (Haider Ackermann, Carol Christian Poell, Saint Laurent Paris, Vetements и др.). Попадая на сайт магазина, пользователь без изначального конкретного запроса под свои нужды сразу теряется во множестве разнообразных предметов одежды по бренду, категории, цветам, ценовой категории и другим параметрам.

Всего на сайте присутствует в среднем 3000 вещей различных наименований, от 433 брендов, по 22 большим группам категорий, а ценовой разброс составляет от 500 рублей до 1 млн.

Задача состоит в том, чтобы для каждого пользователя отсортировать весь этот запас вещей так, чтобы он наилучшим образом отражал его предпочтения, что предполагало бы в перспективе увеличение количества заказов и увеличение количества дополнительных вещей в заказе, о которых пользователь мог вообще не знать, из-за неоптимальной сортировки, но при этом они могли бы ему понравиться. Рекомендации на сайте

[1] spin4spin.com

показываются в виде карусели из вещей под карточкой товара, как показано на рисунке 1.

Рисунок 1 Текущий вид рекомендаций на сайте

До построения специальной системы рекомендации на сайте представляли из себя только товары той же категории, что и просматриваемый, а хочется, чтобы рекомендации были персонифицированными.

Специфичным аспектом в данной задаче по сравнению с рекомендациями музыки или фильмов, например, является уникальность каждого товара, то есть если происходит покупка товара, то информация о том, кто и как его оценивал по сути пропадает, так как новые пользователи не смогут взаимодействовать с этим товаром, следовательно не смогут становиться похожими, так как в матрице взаимодействий у них всегда на месте купленных товаров будут нули. Ну и, естественно, эти товары нельзя рекомендовать, так что наличие взаимодействий с ними у старых пользователей так же не будет иметь никакого смысла. Вторым специфическим аспектом является отсутствие оценок для товаров, что, опять же, обосновано уникальностью каждого товара и невозможностью применения этих оценок вне выстраиваемой рекомендательной системы. Третьей проблемой является низкий процент активных зарегистрированных пользователей от всех активных. Это вынуждает использовать в качестве пользователей уникальные ClientID Яндекс.Метрики[2], подключенной к нашему сайту. Более подробное описание пользователей приведено в разделе с описанием данных.

Первая проблема вынуждает сделать объектом рекомендации что-то более агрегированное, чем просто уникальный товар, то есть то, что не пропадет после удачной рекомендации. В качестве такого объекта я выбираю сочетания бренд-группа категорий для каждого товара, что подразумевает достаточную похожесть товаров среди групп и возможность сбора более

[2] metrika.yandex.ru/

объемного количества взаимодействий по каждому объекту для рекомендации. Более подробно на том, как подбиралась агрегированная группа я также остановлюсь в разделе описания данных.

Вторая проблема решается за счет использования информации только о неявных взаимодействиях пользователей и объектов. То есть используются данные о просмотрах карточек товара, добавления в корзину или в список избранного.

Конкретным результатом решения данной задачи является выдача списка штрихкодов каждому пользователю на основании его взаимодействий с товарами и оценка качества этой выдачи. Оценка качества моделей и их сравнительной эффективности проводилась с помощью метрик precision@10 и recall@10, о чем подробнее описано в разделе методологии решения.

Исследовательским вопросом данной работы является выявление того, какой алгоритм лучше всего подходит для вычисления персонифицированных рекомендаций в онлайн-магазине одежды.

В результате построения системы я выявляю, что алгоритм градиентного бустинга над решающими деревьями в задаче рекомендаций показывает результат в 5.6 раз лучше по точности первых 10 рекомендаций и в 5.5 раза лучше по полноте тех же 10 рекомендаций, чем лучшая из моделей коллаборативной фильтрации на основе поиска ближайших соседей.

Данная работа устроена следующим образом. В обзоре литературы я описываю основные подходы к построению рекомендательных систем и то, как они изменялись со временем. Затем описаны методы сбора данных и как они используются в построении рекомендательной системы. В методологии решения описывается подбор оптимальных моделей для вычисления рекомендаций. В конце приводится сравнение моделей на основании метрик качества сортировки, описание возможных причин для полученных результатов и дальнейшие шаги развития системы рекомендаций.

Весь код, использованный для написания данной работы, находится в открытом доступе[3]. Данные доступны по запросу к автору.

[3] github.com/salamoslam/recommendations

Обзор литературы

В данном разделе я остановлюсь на том, как со временем менялись подходы к прогнозированию предпочтений людей с помощью рекомендательных систем.

Изначально рекомендательные системы понадобились в Интернет-среде с ростом количества информации, которое нужно просмотреть пользователю для получения запроса, удовлетворяющего его предпочтения. По сути, любая система рекомендаций представляет из себя алгоритм, который сортирует все доступные для просмотра пользователем объекты в определенной сфере (будь то фильмы, музыка, статьи и др.) на основании трех групп параметров: атрибуты пользователей, атрибуты объектов и взаимодействия пользователей и объектов. Рекомендации помогают пользователю справиться с перегрузом информации, экономя тем самым время пользователя и позволяя провайдеру контента реализовывать его эффективнее. Таким образом, по сути, алгоритм заменяет продавца-консультанта из магазина. В случае, когда объектами являются какие-либо товары (предметы одежды, например), у продавцов могут появляться стимулы для рекомендаций плохо продающихся или залежавшихся товаров, однако, в данной работе я концентрируюсь именно на предсказании предпочтений.

Первым рассматриваемым методом для генерации рекомендаций является коллаборативная фильтрация. Этот метод впервые был представлен в 1992 году в фильтрационной системе e-mail’ов и в 1994 году в системе рекомендаций новостей GroupLens [1] (Hua-Ming Wang, 2015). Он представляет из себя реализацию идеи, что людям с похожими предпочтениями нужно рекомендовать похожие товары. Для того, чтобы понять, как определяется похожесть людей в такой постановке, нужно ввести понятие матрицы предпочтений. Она представляет собой таблицу, где индексами строк являются пользователи, а индексами столбцов – товары, сама таблица заполнена оценками каждого пользователя каждого товара. Как правило, так как множество товаров достаточно объемное, чтобы рекомендательная система вообще имела смысл, оценок товаров у каждого пользователя не так много, что приводит к большой разреженности матрицы (много нулей). Целью алгоритма коллаборативной фильтрации является заполнение нулевых значений в этой матрице, которую в дальнейшем я также буду называть матрица пользователь-объект или матрица предпочтений, ее схематичный вид представлен на рисунке 2.

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	0	3	0	3	4
User 2	4	0	0	2	0
User 3	0	0	3	0	0
User 4	3	0	4	0	3
User 5	4	3	0	4	4

Рисунок 2 Матрица пользователь-товар. Источник: medium.com[4]

Итак, первым пунктом реализации алгоритма является формирование матрицы предпочтений. Таким образом, каждому пользователю будет соответствовать вещественный вектор. Далее, для каждого вектора можно найти наиболее похожие на него, используя разные метрики расстояния. В основном, в качестве такой метрики используется евклидово расстояние, но также может использоваться и косинусное расстояние, хотя оно и не является полноценной метрикой. Из похожих векторов (т.н. соседей) отбирается k ближайших к вектору пользователя, для кого ищется рекомендация. Пользователю рекомендуются те товары, для которых получены оценки, взвешенно усредненные по похожести от ближайших соседей, с которыми пользователь ранее не взаимодействовал. Данный алгоритм часто использовался в магазинах электронной коммерции, таких как Amazon или TaoBao [1], однако, с увеличением объемов обрабатываемых данных он сталкивается с определенными проблемами.

В первую очередь, с увеличением числа пользователей увеличивается время поиска соседей, а, соответственно, и выдачи персонализированной рекомендации. С увеличением числа товаров увеличивается размерность, провоцируя, закономерно, проклятие размерности и, следовательно, проблемы с поиском релевантных соседей. Из-за этого потенциально может ухудшаться качество рекомендации, на которую пользователи уже будут хуже откликаться и, возможно, уходить с платформы с неудовлетворенными

[4] medium.com/recommendation-systems/

потребностями. Аналогичным способом можно подбирать похожие друг на друга товары и рекомендовать пользователям в рамках неперсонифицированной рекомендации.

Вторым основным методом вычисления рекомендаций являются техники матричной факторизации. Они основаны на вычислении неявных характеристик пользователей и объектов. Эти характеристики могут быть как интерпретируемыми, так и совершенно нет, а также может варьироваться их количество. Характеристики фильмов, например, могут показывать серьезность фильма или ориентацию на определенный пол зрителя [2] (Yehuda Koren, 2009). Таким образом, каждому пользователю и объекту сопоставляется вектор определенной размерности, то есть эмбеддинг. При вычислении у похожих товаров будут похожие эмбеддинги, что будет явно отражаться на выдаче рекомендаций для пользователя. Прогнозирование оценки того, насколько пользователю будет интересен конкретный товар, осуществляется путем перемножения векторов пользователя и товара.

Чтобы вычислить эмбеддинги на множестве имеющихся оценок пользователями товаров под каждую существующую оценку подгоняется скалярное произведение эмбеддингов с помощью квадратичной ошибки, регуляризованной на сумму норм этих векторов. Регуляризирующий множитель подбирается на кросс-валидации.

$$\min_{q^*,p^*} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2) \quad (1)$$

Где K – множество пар пользователь-товар, для которых есть оценка r_{ui}

r_{ui} – оценка товара i пользователем u

q_i – эмбеддинг товара i

p_u – эмбеддинг пользователя u

λ – регуялизирующий множитель

Существуют два способа минимизации данного выражения: стохастический градиентный спуск и метод чередующихся наименьших квадратов (alternating least squares, далее ALS)

При использовании градиентного спуска сначала вычисляются ошибки в предсказании рейтингов

$$e_{ui} \stackrel{\text{def}}{=} r_{ui} - q_i^T p_u \quad (2)$$

А потом обновляются параметры:

$$q_i \leftarrow q_i + \gamma(e_{ui}p_u - \lambda q_i) \quad (3)$$

$$p_u \leftarrow p_u + \gamma(e_{ui}q_i - \lambda p_u) \quad (4)$$

Однако в некоторых случаях бывает эффективнее использовать ALS. Идея метода заключается в том, что, так как минимизируемое выражение не является выпуклым, нужно по очереди фиксировать изначально неизвестные p_u и q_i и решать задачу МНК. То есть сначала, например, фиксируется p_u, пересчитываются q_i путем решения МНК, а потом наоборот. Это обеспечивает уменьшение минимизируемого выражения на каждом шаге до сходимости.

Зачастую градиентный спуск работает быстрее, чем ALS. Однако, бывают случаи, когда бывает предпочтительнее использовать последний. Например, при вычислении ALS все q_i вычисляются независимо от всех остальных признаков объекта, и, аналогично для пользователей, p_u вычисляются независимо, что позволяет распараллелить работу алгоритма [3] (al., 2008).

Вторым местом, где ALS может пригодиться, являются рекомендательные системы, построенные на большом количестве неявных данных (как фильмы, например). При этом тренировочные данные неразреженные и проходиться циклом по всем оценкам получается не так эффективно [4] (Hu, Koren, & Volinsky, 2008).

Для генерации рекомендаций также можно использовать и более традиционные для машинного обучения алгоритмы. Например, в работе [5] (Park, Kang, & Byun, 2021) для генерации рекомендаций применяются алгоритмы Word2Vec для определения товаров комлементов по последовательностям кликов пользователей внутри сессии на сайте и градиентный бустинг над решающими деревьями с таргет-вектором, указывающим на покупку какого-то товара, для выявления ключевых признаков, приведших к покупке. Для использования этого метода в статье генерируются эмбеддинги для товаров с помощью алгоритма Word2Vec, оптимальные последовательности которых (то есть те, которые привели к покупке) и являются объектами для обучения. Однако, ничего не мешает использовать в качестве признаков не только векторные представления для товаров, но и какие-то признаки пользователей (например, пол, возраст, источник трафика) и объектов (цена, количество просмотров, время просмотров и др.), а также признаки из взаимодействия (просмотр, оценка, добавление в корзину/избранное). На базе этого можно

классифицировать объекты по принципу купит или не купит их пользователь и рекомендовать первые.

Помимо традиционных методов машинного обучения при построении систем рекомендаций в последнее время начинают использоваться и нейронные сети. Так, традиционный фреймворк коллаборативной фильтрации может быть улучшен с помощью навешивания на него сверху многослойного перцептрона [6] (He, Liao, Zhang, Nie, & Xia Hu, 2017).

Авторы этой работы предлагают модель, в которой во входной слой заходят векторы характеристик пользователей и объектов. Далее эти векторы преобразуются в более плотные с точки зрения заполненности векторов эмбеддинги пользователей и товаров на первом полносвязном слое. Далее эти векторы скармливаются так называемым «нейронным слоям коллаборативной фильтрации». Переходы между слоями осуществляются, как и в обычных нейронных сетях, с помощью взвешивания параметров с предыдущего слоя и применения функции нелинейности (tanh/sigmoid/reLU и др.). На последнем слое в качестве функции активации применяется сигмоида для того, чтобы предсказанные значения лежали в промежутке [0,1], так как таргет-вектором является вектор, показывающий релевантен ли пользователю конкретный объект (то есть бинарная классификация). В целом, такая архитектура позволяет авторам добиться более высоких значений целевых метрик по сравнению с фильтрацией по k ближайшим соседям или матричной факторизации. Прежде всего, оценивалась метрика hit ratio at k, которая показывает, попал ли последний объект, с которым взаимодействовал пользователь в список k сгенерированных по всем предыдущим взаимодействиям рекомендаций.

Сделав обзор основных методов, применявшихся для построения рекомендательных систем, важно отметить ситуацию, где все они оказываются достаточно бесполезны. Это люди, которые первый раз зашли в магазин или приложение и нет никакой информации об их предпочтениях, ни явной, ни неявной. В таком случае невозможно определить, на кого по своим предпочтениям он может быть похож, поэтому невозможно вычислить персонифицированную рекомендацию. Поэтому в ситуациях «холодного старта» приходится рекомендовать пользователю либо как-то эвристически (самые просматриваемые, с самым высоким рейтингом или новинки), либо на каждый просматриваемый товар выдавать потоварные рекомендации на основе вычисленных эмбеддингов товаров или наиболее похожих товаров по оценкам из матрицы предпочтений.

Данные

Для начала опишу полную схему сбора и подготовки сырых данных к таблицам, на основании которых и будут работать наши модели. В качестве сырых данных с сайта используются логи сайта, в которых записываются переходы каждого человека по каждой ссылке на сайте. Нас интересуют только взаимодействия с товарами, то есть просмотр карточки товара. Также используются данные по добавлению товаров в корзины и вишлисты.

Сырые данные

Хиты сайта

Основная часть информации получена из переходов людей по страницам сайта, где отмечается, главным образом, какой пользователь смотрел какой товар. Всего с начала октября 2021 года по настоящий момент собрано 570 тыс. переходов людей по ссылкам, из них 170 тыс. просмотры карточек товара, то есть минимальная информация, которая нам подходит для замеров взаимодействий пользователей и товаров.

Перед дальнейшим описанием важно отметить, кого я считаю отдельным пользователем. На нашем сайте зарегистрировано 3197 человек и их активность с октября 2021 года представлена на графике 1.

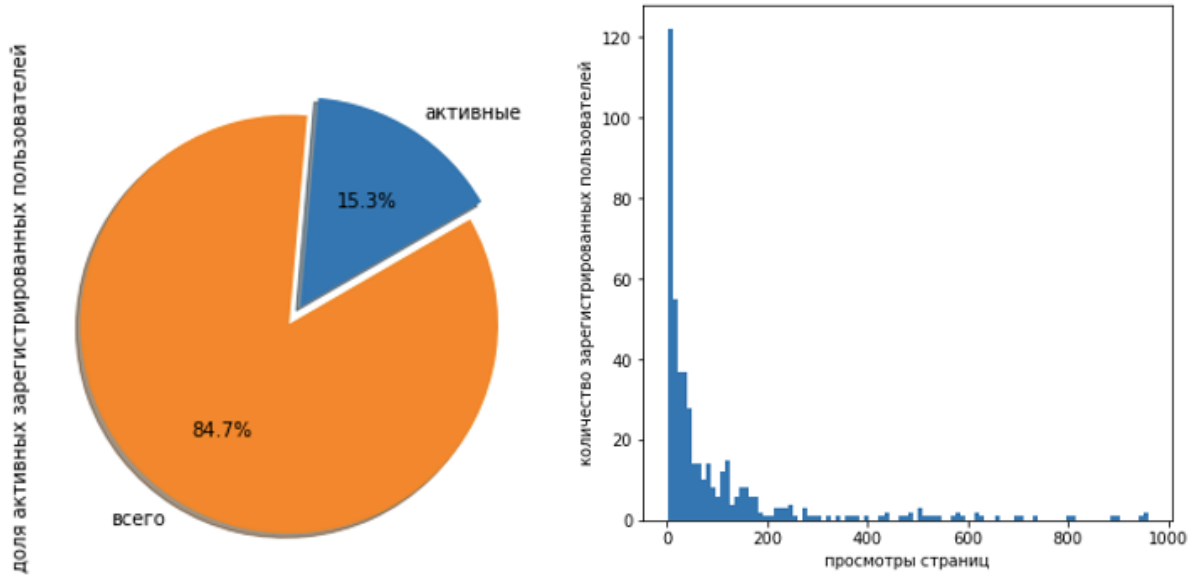

График 1 Активность зарегистрированных пользователей. Источник: составлено автором

Столь низкая активность зачастую объясняется тем, что даже, если пользователь зарегистрирован, то при заходе на сайт он авторизуется либо если он собирается оформлять заказ и хочет применить программу лояльности, либо он комитент, то есть человек, который сдал свои вещи на комиссию и хочет посмотреть, продались они или нет. Таким образом, не имеет смысла ориентироваться на зарегистрированных пользователей, сосредоточимся на другом подходе. Яндекс.Метрика, подключенная к нашему сайту, позволяет отслеживать передвижения по сайтам человека до попадания к нам и после, наделяя его уникальным *ym_client_id* при первом посещении сайта. Ограничения по использованию этих идентификаторов следующие: каждый новый идентификатор генерируется для пользователя при заходе из другого браузера или с другого устройства, а также идентификатор не позволяет следить за трафиком людей, находящихся в режиме «инкогнито» или в режиме «частного доступа». Тем не менее, не будем акцентироваться на них, так как они сами пожелали оставаться незаметными в Интернете и точно не желают получать персонифицированные рекомендации.

Всего людей, зашедших на карточку хотя бы одного товара: 45967 чел.

Распределение их числа по количеству просмотров, а также самые просматриваемые сочетания брендов и категорий и, что будет важнее в дальнейшем анализе, распределение людей по просмотрам брендов-категорий, приведено на графиках 2(1) и 3 (все цифры приведены за период с 6 октября 2021 по 18 апреля 2022):

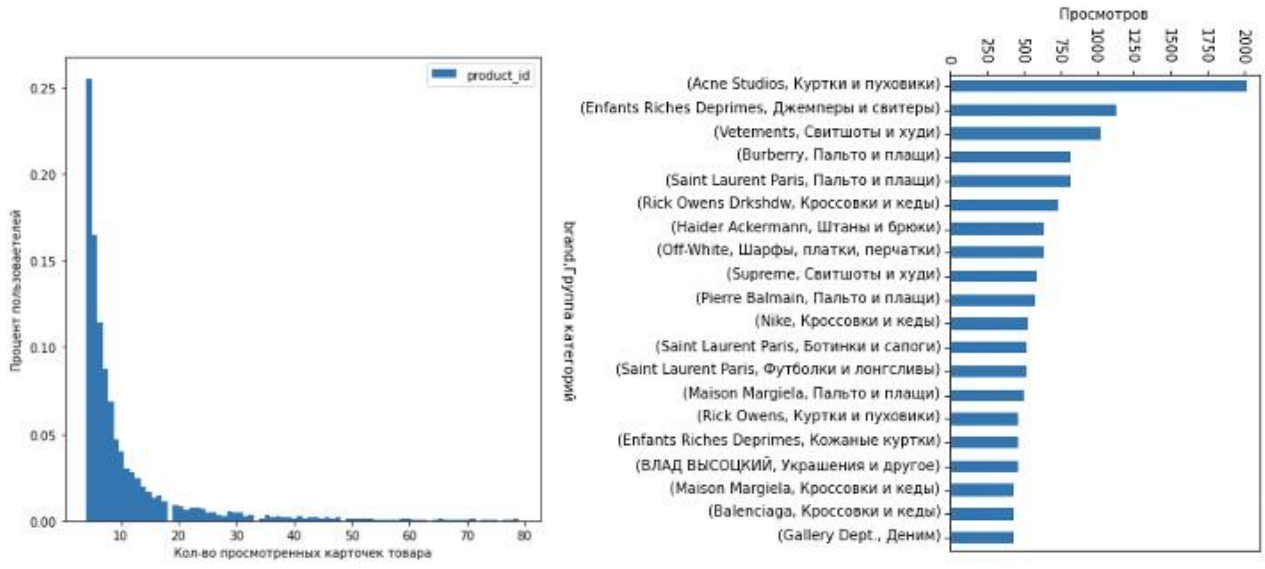

График 2 Распределение пользователей по числу просмотров (1) и самые популярные бренды-категории (2). Источник: составлено автором

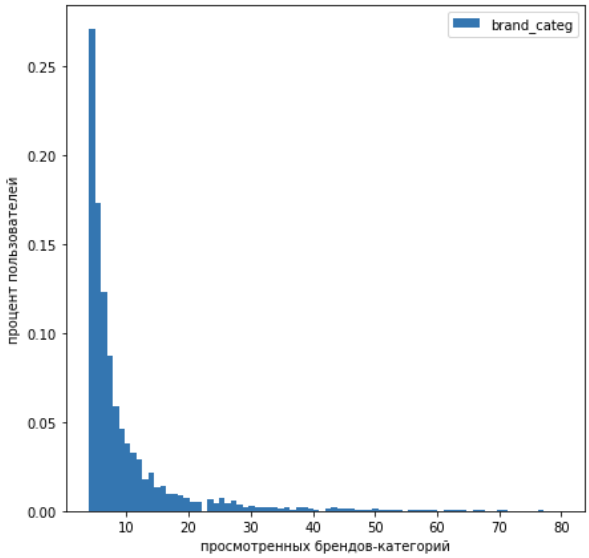

График 3 Распределения пользователей по числу просмотров брендов-категорий. Источник: составлено автором

Как видно по графикам, бóльшая часть пользователей взаимодействует с лишь с малым количеством товаров. Если группировать товары в ранее оговоренные группы бренд-категория, то ситуация остается примерно такой же. С этим не так приятно работать, как с более толстохвостым распределением, что будет подробнее описано в разделе с подбором модели коллаборативной фильтрации.

Корзины и избранное

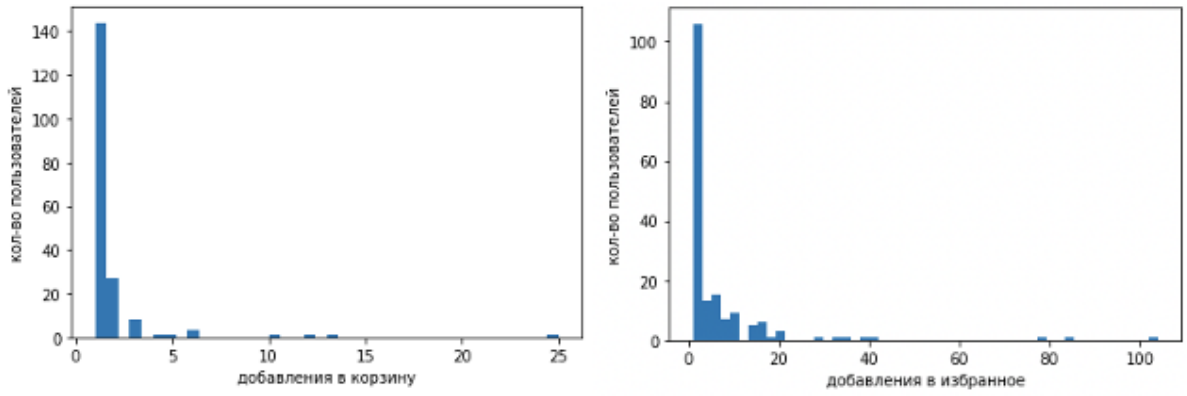

График 4 Распределения пользователей по числу добавлений в корзину или избранное. Источник: составлено автором

В корзинах и избранном активность пользователей схожа с просмотрами карточек товара, то есть бóльшая часть людей добавляла в эти списки 1–2 товара. К сожалению, часть

добавлений в корзину оказалось без указания *ym_client_id,* что не дает возможности отследить, кто именно что туда добавлял.

Предобработанные данные

Датасет для коллаборативной фильтрации

Как и описывалось в обзоре литературы, основным набором данных для применения моделей коллаборативной фильтрации является матрица пользователь-товар. Она заполняется данными по неявным оценкам товаров: просмотрами, добавлениями в избранное, корзины. Для генерации «оценки» я наделяю каждое взаимодействие определенным количеством очков. Просмотру каждого товара я ставлю одно очко, причем они суммируются для одного товара (5 кликов на один товар – 5 очков). Добавлениям в избранное ставятся 5 очков, а добавлениям в корзину – 10. Очки за добавление в избранное взяты из статистического соотношения количества просмотров товаров к добавлению в избранное среди пользователей, добавлявших что-либо в избранное (просмотров примерно в 5 раз больше). С добавлениями в корзину ситуация сложнее, так как по выборке людей, добавлявших в избранное, отношение корзин к избранному 1 к 18, а среди пользователей, добавлявших в корзину примерно 1 к 1. Всего добавлений в избранное в 4 раза больше, чем в корзину, но на результатах модели это сильно не отразится. В общем, очки нам нужны, чтобы расставить приоритеты и привести к одной шкале разные типы взаимодействий.

Взаимодействия с конкретными товарами далее группируются по бренду и категории, что далее будет называться объектами. Всего сочетаний брендов-категорий было 1275, но так как нам нужно рекомендовать объект, который не кончится при удачной рекомендации и по которому должна быть собрана статистика, то бренды и категории обрезаются по критерию более двух вещей в наличии на данный момент. В итоге получается матрица 30255 пользователей на 531 объект. Остальные активные пользователи (15 712 чел.) только на товары брендов-категорий, у которых в наличии на момент сбора данных меньше двух вещей.

Так как не все из этих пользователей взаимодействовали с достаточным количеством объектов, я оставляю лишь тех, для кого есть отклик на 3 и более бренда-категории, чтобы при попадании векторов таких людей в соседи кому-то хотя бы по двум координатам была возможность рекомендовать хоть один товар от этого человека. Таким образом остается 3065 пользователей, из которых активность за последние две недели проявил 121

пользователь. Заполненность итоговой матрицы взаимодействиями – 1,8%. Схему того, какие пользователи доходят до этапа валидации модели, можно видеть на рисунке 3.

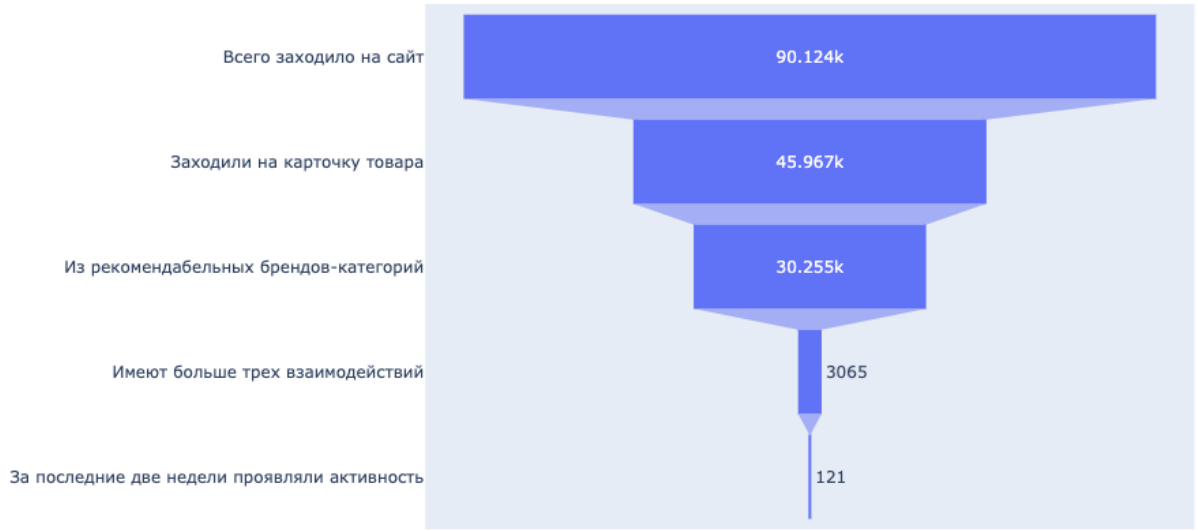

Рисунок 3 Воронка фильтрации пользователей по количеству взаимодействий. Источник: составлено автором

Датасет для модели с использованием градиентного бустинга над решающими деревьями

Для более традиционного для машинного обучения метода я собираю датасет, состоящий из трех типов признаков: признаки пользователя (их распределения описаны в предыдущем разделе), признаки объекта, признаки взаимодействий пользователей и объектов. Конкретные описания переменных приведены в Приложении 1.

Чтобы создать таргет-вектор, я складываю векторы добавлений в корзину и избранное и бинаризую его. То есть, наблюдение относится к первому классу, если товар из группы бренд-категория был добавлен в корзину или избранное, остальные наблюдения класса 0. Взаимодействия с корзинами и избранным складываются, чтобы увеличить численность предсказываемого класса, чтобы избежать чрезмерного дисбаланса классов. Численность положительного класса в тренировочной выборке составляет 334 наблюдения. Отличие от коллаборативной фильтрации в представлении: итоговый датасет состоит из строк взаимодействия каждого пользователя с каждым объектом объединенных с признаками пользователей и объектов, суммарно 16 065 405 строк. В отличие от датасета для коллаборативной фильтрации, я использую данные по всем пользователям, которые взаимодействовали хотя бы с одной группой бренд-категория, то есть 30 255 пользователей по 531 объекту.

Методология решения

В данном разделе описываются два подхода к вычислению рекомендаций: коллаборативная фильтрация и градиентный бустинг над решающими деревьями.

Коллаборативная фильтрация

Я строю модель поиска соседей для пользователей по двум схемам и на основании двух способов измерения расстояния между векторами. Первая схема основана на поиске соседей по векторам пользователей целиком (используя все объекты из 531), то есть используя все ячейки, включая и большинство нулей. Поиск соседей в этом способе осуществляется с помощью библиотеки faiss[5]. Второй способ поиска соседей основан на обрезании для целевого пользователя (для которого вычисляются рекомендации) всех отсутствующих взаимодействий, и поиск осуществляется только по заполненным ячейкам (то есть по коротким векторам, см. рисунок 4). Его реализация потребовала ручного написания.

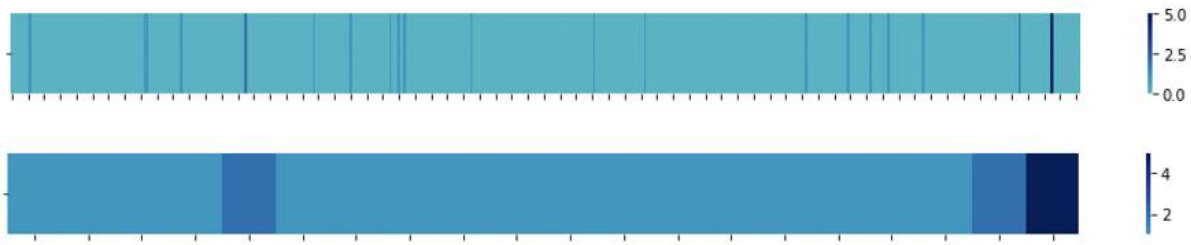

Рисунок 4 Сравнение плотностей векторов для поиска примерных и точных соседей. Источник: составлено автором

Для поиска соседей используются две популярные метрики: евклидова и косинусная (формулы 5 и 6 соответственно)

$$\|x - y\|_2 = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{5}$$

$$\cos(\theta) = 1 - \frac{xy}{\|x\|\,\|y\|} \tag{6}$$

Проблема, упомянутая в разделе описания данных (про тонкохвостые распределения взаимодействий пользователей) выражается в следующем. При поиске ближайших соседей их качество для пользователей из группы с маленьким количеством взаимодействий по умолчанию низкое, так как вектора из матрицы взаимодействия для них наиболее

[5] github.com/facebookresearch/faiss

разрежены и мала вероятность точных совпадений по взаимодействиям с конкретными объектами. Как видно по графику 5, качество соседей убывает по количеству взаимодействий пользователей (чем выше, тем более ориентированы в одну сторону соответствующие векторы).

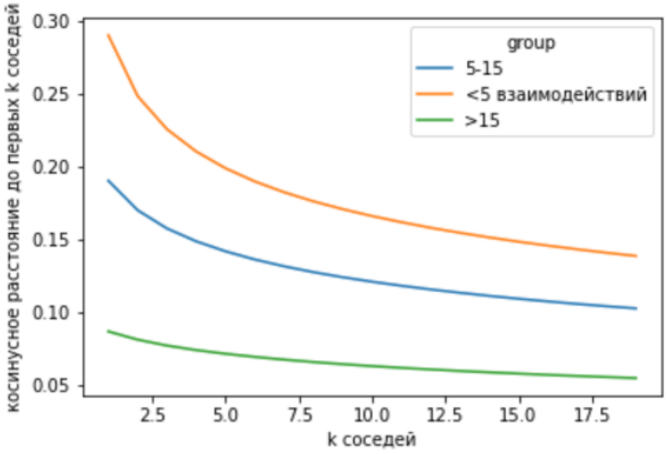

График 5 Среднее расстояние до соседей по группам пользователей. Источник: составлено автором

Это происходит по двум причинам: для более активных пользователей, подбирается мало соседей, по причине того, что их группа сама по себе малочисленна (видно по меньшей вариации среднего расстояния до k первых соседей) и пользователи-соседи с большим количеством взаимодействий имеют больше непустых координат, что автоматически увеличивает расстояние между ними.

Таким образом рекомендации могут подбираться из «кусков нескольких пользователей», что вызывает некие нарушения в предположениях о том, как стоит рекомендовать товары (у похожих пользователей по неявному отклику на товары похожие предпочтения). В данной же ситуации похожий усредненный пользователь оказывается почти синтетическим, что может вызывать низкие показатели оцениваемых далее метрик.

Гиперпараметром в моделях коллаборативной фильтрации является число соседей. Как описывалось в обзоре литературы, рекомендации в этой модели вычисляются как средневзвешенное оценок тех брендов-категорий, которые имеются у ближайших соседей пользователя (пользователь при этом может захотеть провзаимодействовать с товарами того же бренда-категории, что будет считаться за релевантную рекомендацию). Так как в данном методе не используется традиционная минимизация лосс-функции, подбор оптимального числа соседей выполняется на этапе валидации.

Валидация происходит с помощью оценки качества спрогнозированных рекомендаций с помощью метрик precision@k и recall@k. Они являются наиболее подходящими для оценки результатов рекомендационной выдачи [7] (Shani & Gunawardana, 2011). Для их вычисления берется отсортированный по релевантности столбец рекомендаций, вычисленный на основе данных до момента времени в прошлом (например, за две недели), и столбец реальных взаимодействий, прошедших за некоторое время после времени тренировочного датасета (за последние две недели как раз). Precision@k будет равняться тому, какой процент релевантных объектов из всех k рекомендованных. Recall@k равняется тому, какой процент из релевантных, попадает в k рекомендованных. То есть, если смотреть на рисунок 5, то метрика precision@20 будет равна 1/20. Числитель при расчете recall@20 в данном случае 1, знаменателем будет количество релевантных товаров (порог релевантности считаем больше нуля).

	recommends	preferred
('Saint Michael', 'Свитшоты и худи')	5.40	0
('Haider Ackermann', 'Штаны и брюки')	4.85	0
('Vetements', 'Свитшоты и худи')	4.80	0
('Enfants Riches Deprimes', 'Футболки и лонгсливы')	4.15	0
('Saint Laurent Paris', 'Рубашки')	4.00	0
('Balenciaga', 'Свитшоты и худи')	3.95	0
('Saint Laurent Paris', 'Пальто и плащи')	3.95	0
('Saint Laurent Paris', 'Пиджаки и жилеты')	3.95	0
('Nike', 'Кроссовки и кеды')	3.60	0
('Vetements', 'Футболки и лонгсливы')	3.30	0
('Maison Margiela', 'Кроссовки и кеды')	3.25	0
('Haider Ackermann', 'Пальто и плащи')	3.20	10
('Off-White', 'Футболки и лонгсливы')	3.05	0
('Saint Laurent Paris', 'Футболки и лонгсливы')	3.00	0
('Greg Lauren', 'Свитшоты и худи')	2.75	0
('Balenciaga', 'Кроссовки и кеды')	2.65	0
('Saint Laurent Paris', 'Деним')	2.45	0
('Burberry', 'Пальто и плащи')	2.45	0
('Maison Margiela', 'Штаны и брюки')	2.45	0
('Saint Laurent Paris', 'Ботинки и сапоги')	2.40	0

Рисунок 5 Векторы рекомендаций и фактических взаимодействий отсортированных по наилучшим рекомендациям

Для валидации я создаю такую же матрицу предпочтений, но составленную на период двумя неделями ранее той, на которой буду тестировать рекомендации. В этой матрице находятся только пользователи, у которых было взаимодействий больше, чем с тремя объектами на тот момент. Всего в «старой» матрице 2818 пользователей. Далее из них я

выделяю тех, у кого в аналогичной матрице на текущий момент изменились вектора (то есть произошло хоть одно новое взаимодействие). Всего выходит выборка из 121 пользователя, как показано на рисунке 6.

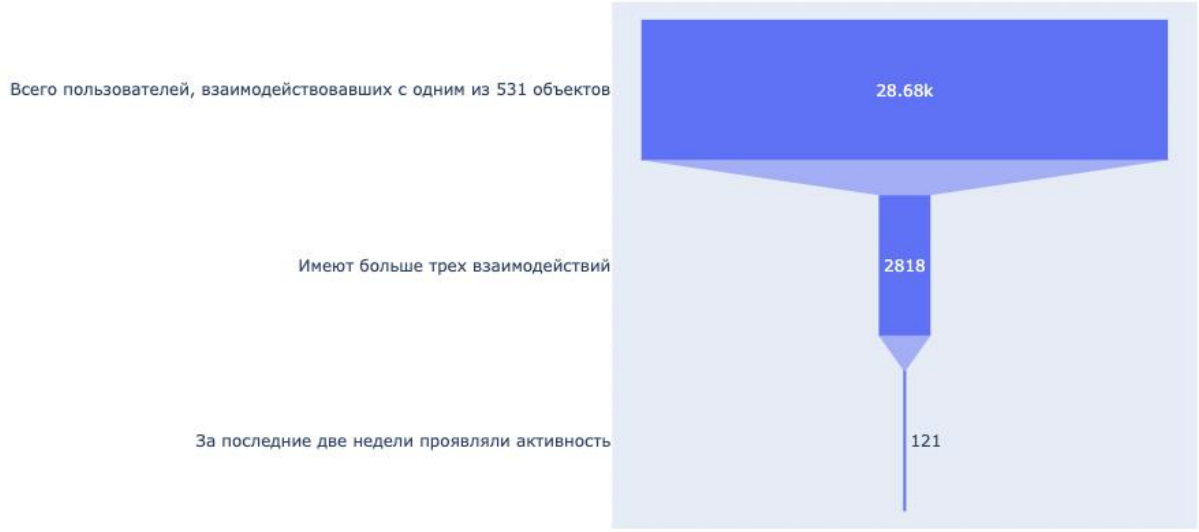

Рисунок 6 Отбор пользователей в валидационную выборку. Источник: составлено автором

Для этих пользователей ищется варьирующееся количество соседей по данным двухнедельной давности, вычисляются рекомендации и оцениваются на основании precision@10 и recall@10 с новыми взаимодействиями. Выбирается то, количество соседей, при котором значения метрик максимальное (либо выбираем по наибольшей из отдельных метрик, либо по F-мере).

Градиентный бустинг

Для обучения этой модели берется датасет со всеми доступными признаками пользователей, признаками объектов и просмотрами товаров. На основании этого решается задача бинарной классификации, какой пользователь добавит в корзину или избранное какой объект с наибольшей вероятностью. Порог классификации в данной задаче нас не интересует, так как нам важно проранжировать для каждого пользователя все возможные объекты на основании предсказанных вероятностей принадлежностей к классу «добавил в корзину или избранное».

Так как в полном датасете 16 миллионов строк, из которых принадлежат к целевому классу только 334 наблюдения, приходится составлять тренировочные датасеты следующим

образом. Во-первых, чтобы результаты были сравнимы с аналогичными для коллаборативной фильтрации, тренировочная выборка собирается на момент двухнедельной давности (она состоит из 15 057 000 строк). Во-вторых, для устранения сильного дисбаланса классов тренировочный датасет составляется из всех наблюдений класса добавлений в корзину и избранного (класс 1), и случайной выборки из элементов противоположного класса (используются выборки размером от 10 000 до 1 млн. наблюдений). Далее на основании этих данных обучается модель градиентного бустинга над решающими деревьями. Используется модель из библиотеки lightGBM[6] с гиперпараметрами: learning_rate – 0.03, n_estimators – 5000, категориальные признаки – *ym_client_id, brand_categ*.

Тестовая выборка состоит из строк взаимодействия того же 121 пользователя с 531 объектом на настоящий момент, всего 64 251 строка. Чтобы, опять же, можно было сравнивать результаты с моделями коллаборативной фильтрации прогнозируемые вероятности к первому классу принадлежности (топ-10 из которых и являются рекомендациями) сравниваются со всеми новыми взаимодействиями пользователей из тестовой выборки с товарами, а не только с добавлениями в корзину или избранное (то есть, еще и с просмотрами). При этом, важно отметить, что просмотры товаров из брендов-категорий не используются в таргет-векторе при обучении, то есть утечки таргета не происходит.

Так как не совсем ясно, какой оптимальный объем для тренировочной выборки вытягивать из всего датасета, я варьирую это значение от 10 000 до 1 000 000 наблюдений (это размер выборки нулевого класса, первый класс фиксирован в объеме 334 наблюдения), прогнозирую вероятности принадлежностей к первому классу и оцениваю метрики precision@10 и recall@10. Теперь можно перейти к результатам и их обсуждению.

[6] lightgbm.readthedocs.io/en/latest/

Результаты

Сравнения метрик

Сначала рассмотрим значения метрик, полученные для четырех разных моделей коллаборативной фильтрации (разделены по метрике: cosine/euclid и по способу нахождения соседей: faiss/hardcode, первый считается как способ по умолчанию). Тут нужно отметить, что эти метрики рассчитаны по каждому из 121 пользователя тестовой выборки (на валидации моделей) отдельно, а затем усреднены. Следовательно, результаты моделей сравнимы.

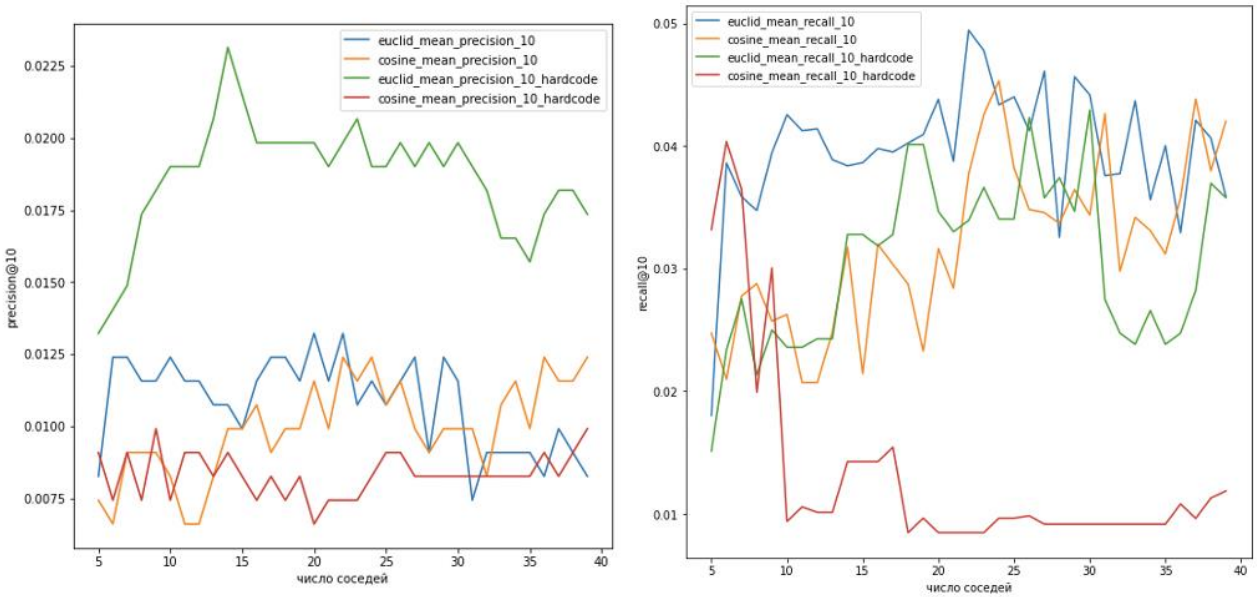

График 6 Значения метрик для разного числа соседей для моделей коллаборативной фильтрации. Источник: составлено автором

Лучшей моделью на основании F-score@10 (является средним геометрическим из метрик precision@10 и recall@10) выходит модель с нахождением соседей только по заполненным ячейкам с евклидовой метрикой и 14 соседями (от 14 до 30 соседей F-score@10 находится на плато, график 7(1), но мы возьмем модель с лучшим precision@10).

Как видно по графику 7(2), результаты модели градиентного бустинга в зависимости от размера выборки показывают значения сильно лучше, чем модели поиска ближайших соседей. Лучшей моделью является обученная на выборке размером 100 000 наблюдений.

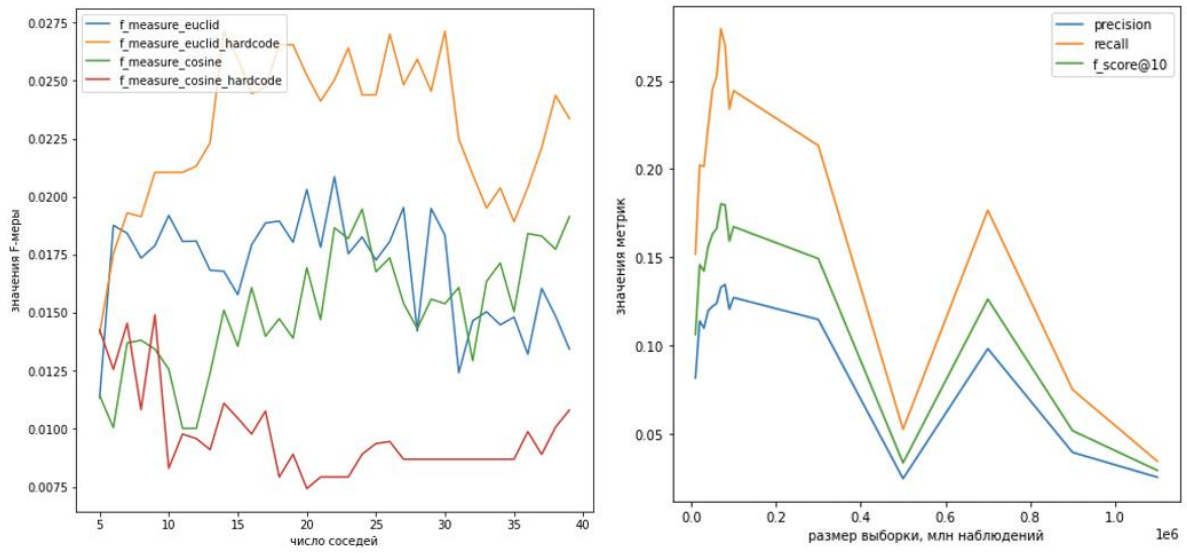

График 7 Значения F-score@10 для коллаборативной фильтрации (1) и значения аналогичных метрик для градиентного бустинга (2). Источник: составлено автором

Значения precision@10 и recall@10 для моделей указаны в следующей таблице:

	nearest neighbors	lightgbm
precision@10	0.023	0.127
recall@10	0.043	0.244

Таблица 1 Сравнение метрик для построенных моделей

Таким образом, модель градиентного бустинга вычисляет рекомендованные бренды-категории с точностью в 5.6 раза лучше и полнотой в 5.5 раз лучше, чем модель коллаборативной фильтрации с поиском ближайших соседей. Значения полученных метрик можно проинтерпретировать следующим образом. Доля релевантных для пользователей брендов-категорий, которые попадают в топ-10 рекомендаций модели градиентного бустинга, составляет 24.4%, а доля рекомендуемых пользователям брендов-категорий, оказывающихся релевантными равна 12.7%.

Возможные объяснения

Модель коллаборативной фильтрации на основе поиска соседей с помощью библиотеки faiss предположительно проигрывает по точности предсказаний hardcode-исполнению, так

как поиск соседей в ней фокусируется на скорости исполнения (при этом ищутся примерные соседи). В моем же методе ищутся точные соседи и только по заполненным ячейкам взаимодействия, что улучшает точность, но приводит к падению скорости примерно в 3 раза (17 секунд против 54 на одну итерацию на этапе валидации).

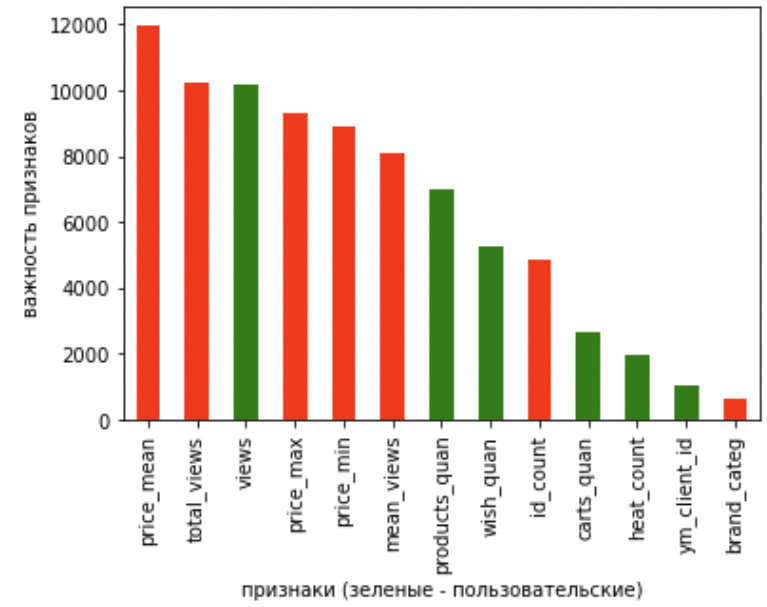

График 8 Важность признаков в модели град. бустинга

Перейдем к обсуждению результатов бустинга. Если посмотреть на важность признаков в этой модели (график 8), то наиболее важную роль в предсказании релевантности объекта для пользователя играют именно признаки объектов. Это может говорить о том, что признаки пользователей по своей специфике (из-за того, какие данные доступны) плохо описывают их предпочтения, даже несмотря на то, что это не пользователи с «холодного старта», так как у них всех изначально было взаимодействий больше, чем с 3 брендами-категориями.

С другой стороны, бустинг может обыгрывать коллаборативную фильтрацию как раз-таки за счет использования дополнительных признаков, помимо взаимодействий. Тот же heat_count достаточно незначим, хотя является основной составляющей матрицы взаимодействия для поиска соседей. Также, бустинг акцентируется на по определению более ценных взаимодействиях (добавление в корзину и избранное), которые позволяют прогнозировать релевантность объекта в виде просмотра карточки товара лучше, чем просмотры похожих пользователей.

Дальнейшие планы

После подбора оптимальной модели коллаборативной фильтрации, она была в тестовом режиме подключена к сайту. Это было реализовано следующим образом. Сначала для пользователей с достаточным числом взаимодействий с объектами (выборка из 3065 пользователей на момент написания) были предпосчитаны рекомендованные объекты (на обновляемой основе, каждые полчаса). Так как алгоритм выдает рекомендации с точностью до бренда-категории, а пользователь видит в каруселькe 5 рекомендуемых товаров с возможностью пролистать дальше, то показывать в первичной выдаче товары из групп брендов-категорий по убыванию релевантности групп не имело смысла (так как в некоторых группах может быть 30 товаров, например). Поэтому я беру по товару из каждой рекомендуемой группы по убыванию их релевантности и показываю в первых двух пятерках в карусели, дальше по второму товару и так далее до конца товаров в наличии из групп. Соответствия список штрихкодов - ClientID после подсчета кладутся в базу данных (используется база данных mongoDB[7]). С помощью фреймворка Flask[8] была написана API, которая подключена к базе данных, и в ответ на get-запрос с ClientID пользователя, заходящего на карточку товара, это приложение вытягивает из базы список рекомендуемых штрихкодов для него.

Это было реализовано для дальнейшего проведения A/B теста, для сравнения эффективности рекомендаций, которые сейчас реализованы на сайте (товары той же категории, что и просматриваемый). То есть каждому второму пользователю должны будут показываться старые рекомендации, а другой группе на основе коллаборативной фильтрации. После этого нужно смотреть на различия в просмотрах, добавлениях в корзину и избранном (ну и в заказах, конечно же) по группам. Аналогично, нужно будет запустить A/B тест по сравнению моделей коллаборативной фильтрации и градиентного бустинга в онлайне для окончательного определения оптимальной модели и дальнейшей отладки гиперпараметров.

[7] mongodb.com/

[8] flask

Список литературы

1. Hua-Ming Wang, G. Y. (2015). Personalized recommendation system K-neighbor algorithm optimization.
2. Yehuda Koren, R. B. (2009). Matrix factorization techniques for recommender systems.
3. al., Y. Z. (2008). Large-Scale Parallel Collaborative Filtering for the Netflix Prize. 337-348.
4. Park, S.-J., Kang, C.-U., & Byun, Y.-C. (2021). Extreme Gradient Boosting for Recommendation System by Transforming Product Classification into Regression Based on Multi-Dimensional Word2Vec. *Symmetry*.
5. Hu, Y., Koren, Y., & Volinsky, C. (2008). Collaborative Filtering for Implicit Feedback Datasets. 263-272.
6. He, X., Liao, L., Zhang, H., Nie, L., & Xia Hu, T.-S. C. (2017). Neural Collaborative Filtering.
7. Shani, G., & Gunawardana, A. (2011). Evaluating Recommendation Systems.

Приложение 1

Описание признаков для модели градиентного бустинга над решающими деревьями.

Используются следующие признаки пользователей:

- *views* – число переходов по страницам
- *products* – число просмотров карточек товара
- *carts_quan* – число добавлений в корзину
- *wish_quan* – число добавлений в избранное

Признаки объекта следующие:

- *mean_price* – средняя цена товаров группы бренд-категория, руб.
- *min_price* – минимальная цена, руб.
- *max_price* – максимальная цена, руб.
- *total_views* – суммарное число просмотров товаров из группы
- *id_count* – всего вещей бренда-категории в наличии
- *mean_views* – среднее число просмотров объекта (всего просмотров объекта / всего вещей объекта)

Признаки взаимодействий состоят из:

- *heat_count* – просмотры товаров
- *carts_counter* – добавлений в корзину
- *wishlist_counter* – добавления в избранное.