МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент
Начальник отдела
суперкомпьютерного моделирования
«НИУ ВШЭ», к.ф.-м.н., доцент

_____ П.С. Костенецкий

«___»_____ 2024 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка рекомендательного сервиса для музыкального стримингового сервиса

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.04.04.2024.308-1489.ВКР

Научный руководитель,
доцент кафедры СП, к.ф.-м.н.
_____ Г.И. Радченко

Автор работы,
студент группы КЭ-229
_____ А.Е. Колмаков

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы магистранта
студенту группы КЭ-229
Колмакову Антону Евгеньевичу,
обучающемуся по направлению
09.04.04 «Программная инженерия»
(магистерская программа «Искусственный интеллект и инженерия данных»)

1. Тема работы (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)
Разработка рекомендательного сервиса для музыкального стримингового сервиса.

2. Срок сдачи студентом законченной работы: 20.05.2024 г.

3. Исходные данные к работе

3.1. Schedl M. Deep Learning in Music Recommendation Systems. // Frontiers in Applied Mathematics and Statistics, 2019. – 61–70 pp.

3.2. Zhang S., Yao L., Sun A., Tay Y. Deep Learning Based Recommender System: A Survey and New Perspectives. // ACM Computing Surveys, 2018. – 1–38 pp.

3.3. Zbigniew W.R., Wieczorkowska A., Tsumoto S. Recommender Systems for Medicine and Music. // Springer Nature, 2021. – 236 p.

3.4. Andreas A., Miguel G., Le Z. Microsoft Recommenders: Best Practices for Production-Ready Recommendation Systems. // Companion Proceedings of the Web Conference, 2020. – 50–51 pp.

4. Перечень подлежащих разработке вопросов

4.1. Провести анализ предметной области.

4.2. Проанализировать и подготовить исходные данные.

4.3. Спроектировать систему.

4.4. Выбрать модель и провести ее обучение.
4.5. Реализовать сервис для взаимодействия с обученной моделью.
4.6. Произвести тестирование модели.
5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
доцент кафедры СП, к.ф.-м.н. Г.И. Радченко

Задание принял к исполнению А.Е. Колмаков

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

В современном мире музыкальные стриминговые сервисы стали доминирующим способом потребления музыки. На фоне стремительного роста рынка (за 2023 год аудитория российских стриминговых сервисов выросла на 25% [1]) конкуренция между платформами усиливается, делая персонализацию пользовательского опыта ключевым фактором успеха. Алгоритмы, учитывающие индивидуальные предпочтения пользователей и формирующие релевантную подборку музыкальных произведений, повышают вовлеченность и лояльность аудитории. Как следствие, это приводит к увеличению аудитории и дохода стримингового сервиса за счет более активного взаимодействия пользователей с платформой.

Также на фоне растущего рынка крупные («Spotify», «Apple Music») и локальные игроки («Яндекс.Музыка», «Звук») ведут активную борьбу за пользователей, инвестируя значительные ресурсы в развитие своих рекомендательных систем. Из-за этого у пользователей повышаются ожидания к качеству предлагаемой сервисами музыки.

В условиях растущей конкуренции, усложнения запросов и требований к персонализации, разработка эффективного и конкурентоспособного рекомендательного сервиса является актуальной задачей для музыкальных стриминговых сервисов.

Постановка задачи

Целью выпускной квалификационной работы является разработка рекомендательного сервиса для музыкального стримингового сервиса. Для достижения поставленной цели необходимо решить следующие задачи:

1) провести обзор существующих решений и методов;
2) описать требования и смоделировать систему;
3) подготовить и проанализировать исходные данные;
4) реализовать модель и провести ее обучение;
5) реализовать сервис для взаимодействия с обученной моделью;

6) произвести тестирование модели.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения, списка литературы и приложения. Объем работы составляет 55 страниц, объем списка литературы – 28 источников.

В первой главе описывается предстоящая задача, проводится обзор исследований, научной литературы и программных аналогов.

Вторая глава посвящена анализу требований, исходных данных и моделированию создаваемой системы.

В третьей главе описывается предобработка данных, выбор и обучение рекомендательных моделей, реализация рекомендательной системы.

В четвертой главе представлены результаты тестирования созданной системы.

В приложении А содержится исходный код обученных моделей.

В приложении Б содержится исходный код рекомендательного сервиса и пример его работы.

В приложении В пример работы рекомендательного сервиса.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Описание предметной области

В настоящее время музыкальные стриминговые сервисы становятся все более популярными среди пользователей по всему миру. Они предлагают широкий спектр музыкальных треков и альбомов, доступных для прослушивания в любое время и в любом месте при наличии интернет-соединения. Однако с ростом объема музыкального контента становится все сложнее для пользователей найти именно те треки, которые соответствуют их вкусам и предпочтениям [2].

Для решения этой проблемы многие стриминговые платформы внедряют рекомендательные системы. Рекомендательные системы – это программные инструменты, которые анализируют предпочтения пользователя и предлагают контент, который, как предполагают рекомендательные системы, будет интересен пользователю. Эти системы основаны на алгоритмах машинного обучения, которые анализируют историю прослушивания пользователя, его предпочтения, оценки и даже действия других пользователей для предсказания того, что ему понравится.

Разработка эффективной рекомендательной системы для музыкального стримингового сервиса требует глубокого понимания как технических аспектов, так и музыкальных предпочтений пользователей. Важно учитывать разнообразные факторы, влияющие на предпочтения пользователей, такие как жанр музыки, артисты, предпочтения в зависимости от настроения или времени суток, а также новые тренды и рекомендации от друзей или экспертов.

Степень разработанности темы

Область рекомендательных систем музыки с применением искусственного интеллекта представляет собой динамично развивающуюся сферу исследований. На сегодняшний день рассматриваемая тема привлекает внимание как академического сообщества, так и коммерческих организаций, что подчеркивается рядом ключевых характеристик.

Научные исследования: в последние десятилетия наблюдается рост академических исследований, посвященных рекомендательным системам в области музыки. Публикации в журналах и конференциях, таких как «ACM Transactions on Interactive Intelligent Systems» [3] и «Conference on Recommender Systems» [4], предоставляют обзоры современных методов, алгоритмов и подходов к созданию рекомендательных систем музыки.

Технологические разработки: коммерческие платформы, такие как «Spotify», «Apple Music», и другие стриминговые сервисы [5], активно интегрируют рекомендательные системы в свою архитектуру. Эти платформы активно применяют искусственный интеллект для анализа музыкальных предпочтений и предоставления персонализированных рекомендаций.

Открытые исследовательские проекты: существует ряд открытых исследовательских проектов, направленных на разработку и совершенствование рекомендательных систем музыки. Проекты с открытым исходным кодом, такие как «Surprise» [6] и «LensKit» [7], обеспечивают доступ к инструментам и ресурсам для тестирования и разработки новых методов рекомендаций.

Специализированные конференции и семинары: в сфере рекомендательных систем проводятся специализированные конференции и семинары, такие как «ACM RecSys» [4], посвященные обмену знаний и опытом в области разработки и исследования рекомендательных систем.

Обширный объем литературы и практические реализации подчеркивают высокую степень проработанности темы рекомендательных систем музыки с применением искусственного интеллекта. Однако, несмотря на достигнутые успехи, существуют вызовы, такие как улучшение точности рекомендаций, учет динамичных музыкальных предпочтений и решение проблемы холодного старта, которые продолжают привлекать внимание исследователей и разработчиков.

История исследований в области рекомендательных систем музыки с использованием искусственного интеллекта насчитывает несколько деся-

тилетий и отличается активным развитием и постоянными инновациями. Для понимания современного состояния области необходимо взглянуть на ключевые этапы развития, общие понятия и классические подходы, современные достижения и вызовы.

История вопроса

Первые попытки создания рекомендательных систем музыки уходят в прошлое и связаны с разработкой методов коллаборативной фильтрации. Однако настоящий бум в исследованиях рекомендательных систем начался в 2000-х годах, когда с распространением интернета и цифровых медиа музыкальные сервисы начали активно применять рекомендации для улучшения пользовательского опыта. Один из прорывных моментов в области был конкурс «Netflix Prize» (2006) [8], который способствовал развитию методов коллаборативной фильтрации.

С течением времени, с развитием искусственного интеллекта и машинного обучения, рекомендательные системы музыки стали более сложными и персонализированными. Современные системы учитывают не только историю прослушивания пользователя, но и контекст, в котором музыка слушается (например, время суток, настроение, активность пользователя и др.) [9].

Общие понятия и классические подходы [10]

Коллаборативная фильтрация: коллаборативная фильтрация основана на анализе пользовательских действий и предоставляет рекомендации на основе сходства между пользователями или объектами. Этот классический подход включает в себя методы соседей, матричные разложения и другие.

Контентные методы: контентные методы анализируют свойства и характеристики музыкальных треков и предоставляют рекомендации на основе сходства между контентом. Методы могут включать в себя анализ текстов песен, жанров, артистов и других характеристик.

Гибридные подходы: гибридные подходы комбинируют коллаборативную фильтрацию и контентные методы для улучшения качества рекомендаций. Данные подходы позволяют учесть как историю пользователя, так и характеристики контента.

Классические работы и программные продукты

Среди классических работ, посвященных рекомендательным системам музыки, следует выделить работы таких авторов, как «Item-based Collaborative Filtering Recommendation Algorithms» [11] и «Music Recommendation and Discovery in the Long Tail» [12].

Коммерческие платформы, такие как «Spotify», «Apple Music», «Amazon Music» и «Deezer», успешно интегрировали рекомендательные системы в свои продукты, предоставляя миллионам пользователей персонализированные музыкальные рекомендации [5].

Современное состояние исследований и разработок в области рекомендательных систем характеризуется интеграцией современных методов машинного обучения, обработки естественного языка и анализа больших данных, что позволяет значительно улучшить точность и персонализацию рекомендаций.

Следующие разделы данного исследования более подробно рассмотрят современные подходы, результаты проведенных исследований и анализ существующих программных продуктов в области рекомендательных систем музыки.

Современные подходы

Современные рекомендательные системы для музыкальных стриминговых сервисов, основанные на искусственном интеллекте, стремятся предоставлять пользователю персонализированные, контекстно-ориентированные рекомендации с высокой точностью. Развитие технологий машинного и глубокого обучения значительно способствует эволюции подходов к созданию эффективных систем рекомендаций.

Глубокое обучение: использование глубоких нейронных сетей позволяет обрабатывать сложные зависимости в музыкальных данных. Примером может служить применение рекуррентных нейронных сетей (RNN) для моделирования последовательностей музыкальных предпочтений, как это описано в статье «What to play next? A RNN-based music recommendation system» [13], где RNN использовались для предсказания следующего трека в плейлисте пользователя.

Контекстуальные рекомендации: учет контекста взаимодействия пользователя с музыкой (время суток, активность, эмоциональное состояние) позволяет адаптировать рекомендации к различным сценариям использования. Например, методы контекстуальной адаптации, изменяющие рекомендации в зависимости от местоположения, времени суток и погоды, были исследованы в работе «A sequence-based and context modelling framework for recommendation» [14].

Совмещение методов: использование гибридных подходов, объединяющих коллаборативную фильтрацию и контентные методы, способствует учету как характеристик пользователя, так и особенностей контента. Примером работы в этом направлении может служить исследование в области рекомендательных систем для фильмов, где используются матричные разложения в сочетании с коллаборативной фильтрацией для учета предпочтений пользователей, как это было показано в работе «The BellKor Solution to the Netflix Grand Prize» [15].

Интерпретируемость моделей: создание моделей, которые могут объяснить свои решения, важно для повышения доверия пользователей и обеспечения прозрачности в процессе принятия рекомендаций. Использование методов, таких как локальные признаки влияния (LIME) [16], для объяснения, почему конкретный трек был рекомендован.

Современные подходы к рекомендательным системам музыки активно интегрируют передовые методы машинного обучения и уделяют внимание аспектам контекста и пользовательского взаимодействия, что

способствует созданию более точных, персонализированных и удовлетворительных музыкальных рекомендаций. Современные музыкальные рекомендательные системы не только адаптируются под текущие условия и предпочтения пользователя, но также учитывают динамическую природу музыкального вкуса и взаимосвязи между различными музыкальными жанрами и контекстами прослушивания.

Вывод по первой главе

На основании анализа предметной области и существующих решений, можно сделать несколько ключевых выводов. Рекомендательные системы для музыкальных стриминговых сервисов представляют значительный интерес как в академических, так и в коммерческих кругах. В частности, актуальность разработки эффективного рекомендательного сервиса для музыкального стриминга обусловлена быстрым ростом рынка и повышением требований к персонализации пользовательского опыта.

В процессе анализа были рассмотрены различные подходы и методы, такие как коллаборативная фильтрация, контент-базированная фильтрация и гибридные системы. Каждый из этих подходов имеет свои преимущества и недостатки, и выбор конкретной модели или метода должен базироваться на специфических требованиях к системе и имеющихся данных.

2. ПРОЕКТИРОВАНИЕ

Данная глава посвящена проектированию рекомендательного сервиса (в дальнейшем – РС). Цель разрабатываемого РС предоставить стриминговому сервису инструмент, формирующий персонализированную пользовательскую музыкальную подборку.

Проект РС разрабатывается с учетом требований конкретного стримингового сервиса – «Zaycev.net» [17]. Однако предложенное решение остается универсальным и может быть адаптировано для использования с различными музыкальными стриминговыми платформами независимо от их специфической инфраструктуры.

2.1. Требования к системе

Раздел описывает ключевые требования к РС, которые определяют основу для последующего проектирования и разработки. Требования к РС подразделяются на две основные категории: функциональные и нефункциональные требования.

Нефункциональные требования

Нефункциональные требования, описанные ниже, основываются на входных данных, состоящих из 1,7 миллионов строк.

1. Время отклика: время на обработку запроса и выдачу 60 рекомендаций не должно быть более 50 миллисекунд.
2. Время обучения модели: процесс обучения модели должен занимать не более 24 часов на графическом ускорителе уровня NVIDIA RTX 3070.
3. Использование графических ускорителей: для ускорения обучения модели необходимо использовать графические ускорители.
4. Микросервисная архитектура и общение по REST API: РС должен быть реализован в виде микросервиса, обеспечивающего обработку запросов через REST API. Общение между сервисами (РС и стриминговый

сервис) должно быть организовано посредством HTTP запросов с использованием JSON или XML для передачи данных.

5. Применение контейнеризации: РС должна быть реализована в виде контейнера с использованием средства автоматизации Docker.

Функциональные требования

Функциональные требования к РС оформлены в виде диаграммы вариантов использования UML (рисунок 1). Диаграмма позволяет визуализировать основные функции РС и взаимодействия с музыкальным сервисом.

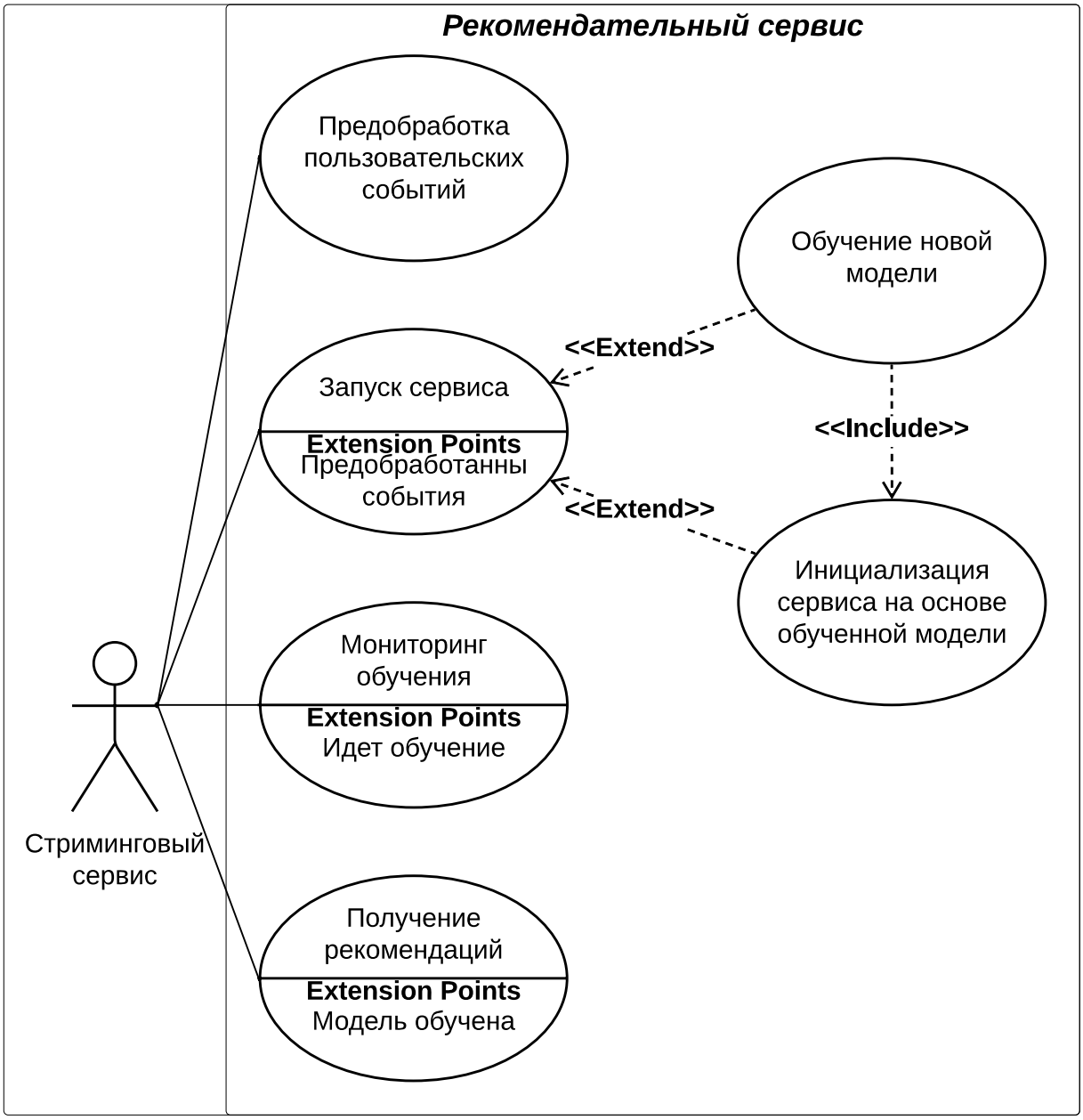

Рисунок 1 – UML диаграмма вариантов использования РС

Приведем описание вариантов использования РС.

1. Предобработка событийных данных: конвертация исходных данных в формат, подходящий для анализа методами машинного обучения. Этапы предобработки включают очистку, нормализацию, агрегацию данных и сохранение данных в требуемом формате. В зависимости от особенностей архитектуры каждого стримингового сервиса механизм предобработки будет иметь свои уникальные аспекты реализации.

2. Запуск сервиса: включает в себя запуск процесса обучения новой модели или использование предварительно обученной модели в зависимости от того, была ли предоставлена модель на момент запуска.

3. Мониторинг состояния обучения модели: возможность наблюдения за ходом обучения модели в реальном времени.

4. Предоставление персонализированных рекомендаций: система должна предоставлять рекомендации для конкретного пользователя, позволяя выбирать объем предложений и предоставлять возможность исключать уже известные пользователю треки.

2.2. Анализ исходных данных

Музыкальный стриминговый сервис [17] предоставил данные, на основании которых необходимо предоставлять рекомендации. Данные представлены в виде CSV файлов, описанных в таблице 1.

Таблица 1 – Описание CSV файлов

Файл	Описание	Количество записей
artists.csv	Таблица с артистами	75 839
authors.csv	Таблица с авторами	57 125
events.csv	Таблица с событиями	147 499 237
genres.csv	Таблица с жанрами	388
tracks.csv	Таблица с треками	417 494

Таблица с артистами

Описание признаков таблицы artists.csv представлено в таблице 2.

Таблица 2 – Признаки таблицы artists.csv

Признак	Описание
artist_id	ID артиста
artist_name	Имя артиста

Таблица с авторами

Описание признаков таблицы authors.csv представлено в таблице 3.

Таблица 3 – Признаки таблицы authors.csv

Признак	Описание
id	ID автора
name	Имя автора
union_id	Перечисление ID всех авторов (в случае нескольких авторов)

Таблица с событиями

Описание признаков таблицы events.csv представлено в таблице 4.

Таблица 4 – Признаки таблицы events.csv

Признак	Описание
trackId	Уникальный идентификатор трека
type	Тип события
personId	Уникальный идентификатор авторизированного пользователя
actionTypeContext	Причина возникновения события
initiatorContext	Место возникновения события
deviceTimestampUtc	Время на устройстве пользователя
artistId	Уникальный идентификатор пользователя
trackTimeMilliseconds	Время трека, на котором возникло событие (в миллисекундах)
durationMilliseconds	Длительность прослушивания трека (в миллисекундах)
countryIsoCode	Код страны в формате ISO 3166-1
regionIsoCode	Код региона в формате ISO 3166-2

Признаки type, actionTypeContext, initiatorContext, countryIsoCode и regionIsoCode являются категориальными и ограничены предопределенным набором уникальных значений.

Описание видов событий (type) представлено в таблице 5.

Таблица 5 – Возможные виды событий (type) в таблице events.csv

Событие	Описание
CACHE	Кеширование трека
DOWNLOAD	Загрузка трека
FAVORITE_ADD	Добавление трека в «избранное»
FAVORITE_REMOVE	Удаление трека из «избранного»
NOT_SUGGEST	Добавление трека в «не предлагать»
UN_NOT_SUGGEST	Удаление трека из «не предлагать»
PLAYLIST_ADD	Добавление трека в плейлист
PLAYLIST_REMOVE	Удаление трека из плейлиста
REWIND_TRACK_TIME	Перемотка трека
STREAMING_DESTROY	Завершение стриминга
STREAMING_END	Окончание стриминга трека
STREAMING_FULL_WITHOUT_REWIND	Окончание стриминга трека без перемоток
STREAMING_PAUSE	Пауза стриминга
STREAMING_RESUME	Продолжение воспроизведения стриминга
STREAMING_START	Начало стриминга

Описание причин событий (actionTypeContext) представлено в таблице 6.

Таблица 6 – Возможные причины событий (actionTypeContext) в таблице events.csv

Причина	Описание
UNKNOWN	Причина не определена
AUTO	Автоматическое действие (например, переключение трека по окончанию воспроизведения)
USER	Пользовательское действие

Описание мест возникновений событий (initiatorContext) представлено в таблице 7.

Таблица 7 – Возможные места возникновений событий (initiatorContext) в таблице events.csv

Место возникновения события	Описание
Headset	Действие с гарнитуры
AppWidget	Действие с виджета на рабочем столе
Bluetooth	Действие от Bluetooth устройства
Notification	Действие из нотификации в шторке уведомлений
Application	Действие из приложения

Таблица с жанрами

Описание признаков таблицы genres.csv представлено в таблице 8.

Таблица 8 – Признаки таблицы genres.csv

Признак	Описание
id	ID жанра
name	Название жанра

Таблица с треками

Описание признаков таблицы tracks.csv представлено в таблице 9.

Таблица 9 – Признаки таблицы tracks.csv

Признак	Описание
TRACK_ID	ID трека
TRACK	Название трека
DURATION	Продолжительность трека (в секундах)
ARTIST	ID артистов
AUTHORS	ID авторов
GENRES	ID жанров

Анализируя исходные данные следует заключить, что информация о пользовательской активности предоставлена детально. Однако сведения о контенте оказываются неполными и не подходят для применения методов контентной фильтрации. В связи с этим детализированные данные о взаимодействиях пользователей с конкретными треками, представленные в файле events.csv, могут стать основой для разработки системы на основе коллаборативной фильтрации.

2.3. Обработка исходных данных

Анализ данных показал, что разрабатываемый РС будет основан на методах коллаборативной фильтрации. Методы коллаборативной фильтрации используют так называемые матрицы взаимодействий, которые отражают историю взаимодействий пользователей с различными объектами (например, фильмами, книгами или товарами). Эти матрицы помогают выявить скрытые предпочтения и антипатии пользователей, основываясь на предыдущих действиях и оценках пользователей. Затем

алгоритмы коллаборативной фильтрации анализируют данные из матрицы вазимодействий для выявления схожих пользователей или объектов и предлагают рекомендации, которые могут быть интересны пользователю на основе предпочтений его «соседей» по данным [18].

Исходный набор данных позволяет оценить пользовательские предпочтения для формирования матрицы взаимодействий. Оценка будет строиться на основе различных событий, каждое из которых будет иметь определенный вес, указанный в таблице 10. Сумма весов событий определит итоговую оценку трека для каждого пользователя.

Таблица 10 – Веса событий

Событие	**Вес**
STREAMING_END	От -0,3 до +0,7
STREAMING_FULL_WITHOUT_REWIND	+1
FAVORITE_ADD	+5
NOT_SUGGEST	-5
FAVORITE_REMOVE	-5
UN_NOT_SUGGEST	+5

Все события, кроме STREAMING_END, имеют константные значения веса. Вес события STREAMING_END рассчитывается по формуле (1).

$$\text{weight}_{\text{STREAMING\_END}} = \frac{\text{eventDuration}}{\text{trackDuration}} \text{-0,3}, \quad (1)$$

где $\text{weight}_{\text{STREAMING\_END}}$ – вес события STREAMING_END;

eventDuration – продолжительность события прослушивания трека (признак durationMilliseconds из таблицы events.csv);

trackDuration – продолжительность трека (признак DURATION из таблицы track.csv).

Такой подход к формированию оценки учитывает не только факт прослушивания трека, но и его длительность относительно общей продолжительности трека. Например, события, когда пользователь прослушал больше половины трека, оцениваются выше, чем события, когда трек был прослушан менее чем на треть. Данная оценка трактует

короткое прослушивание трека как неприязнь к треку, уменьшая итоговый пользовательский рейтинг трека.

Явные пользовательские события (FAVORITE_ADD, NOT_SUGGEST, FAVORITE_REMOVE, UN_NOT_SUGGEST), имеют значительно больший вес, чем неявные (STREAMING_END, STREAMING_FULL_WITHOUT_REWIND). Это означает, что когда пользователь совершает конкретные действия, то это будет иметь большее значение при формировании итоговой оценки предпочтений пользователя. Это важно, потому что явные действия часто более информативны и точно отражают предпочтения пользователя, чем неявные действия. Таким образом, учитывая явные действия с большим весом, система рекомендаций может лучше понять и учесть интересы пользователя, что приведет к более точным рекомендациям.

В предобработанном датасете следует исключить оценки, близкие к нулю, так как они могут не нести смысловой нагрузки или быть результатом случайных действий пользователей. Исключение таких оценок поможет улучшить качество модели и предоставляемых рекомендаций, так как модель будет оперировать лишь существенными и информативными данными. В данном случае следует удалить все оценки, входящие в диапазон от -2 до 4. Исключив этот диапазон, в данных будут оставлены оценки, на формирование которых повлияли явные пользовательские события, и при этом отфильтрованы случайные прослушивания треков.

Кроме того, необходимо проанализировать выбросы и аномалии и провести фильтрацию данных. В данном случае можно воспользоваться фильтрацией по квантилям за счет чего будут удалены слишком высокие и слишком низкие оценки.

2.4. Моделирование системы

Данный раздел посвящен описанию основных компонентов системы, их взаимосвязей и методов, применяемых в ходе разработки для соответствия всем установленным требованиям.

Сервис предобработки данных

Отвечает за сбор данных из разнообразных источников, их обработку и создание итогового CSV-файла для последующего использования рекомендательным сервисом. Разработка сервиса требует индивидуального подхода для каждого музыкального стримингового сервиса из-за различий в форматах и способах хранения данных.

Сервис рекомендаций

Отвечает за генерацию рекомендаций и тренировку модели машинного обучения. Должен иметь возможность инициализации на основе ранее обученной модели для обеспечения быстрого восстановления после сбоев. Разработчики, интегрирующие сервис, должны иметь возможность наблюдения за процессом обучения через консольный вывод. Обучение должно поддерживать ускорение с помощью графических ускорителей, совместимых с CUDA. Сервис должен быть упакован в Docker-контейнер для облегчения развертывания и обеспечения согласованности среды выполнения.

Для выбора рекомендательной модели необходимо провести исследования наиболее результативных рекомендательных моделей и выбрать наилучшую. Исследование можно осуществить на базе исследовательской работы Microsoft «Microsoft Recommenders: Best Practices for Production-Ready Recommendation Systems» [19], в которой представлены результаты обучения различных рекомендательных моделей.

Пользовательские рекомендации должны предоставляться через REST API, которое реализует GET-метод со следующими параметрами:

– user_id (int) – идентификатор пользователя;

– track_count (int, опционально) – количество треков для рекомендации, по умолчанию 60;

– remove_known (bool, опционально) – исключать ли уже известные пользователю треки, по умолчанию false.

В ответ должен возвращаться список ID рекомендуемых треков в формате JSON.

Взаимодействие сервисов

Для демонстрации взаимодействия сервисов воспользуемся диаграммой последовательности (рисунок 2), которая демонстрирует различные варианты инициализации РС.

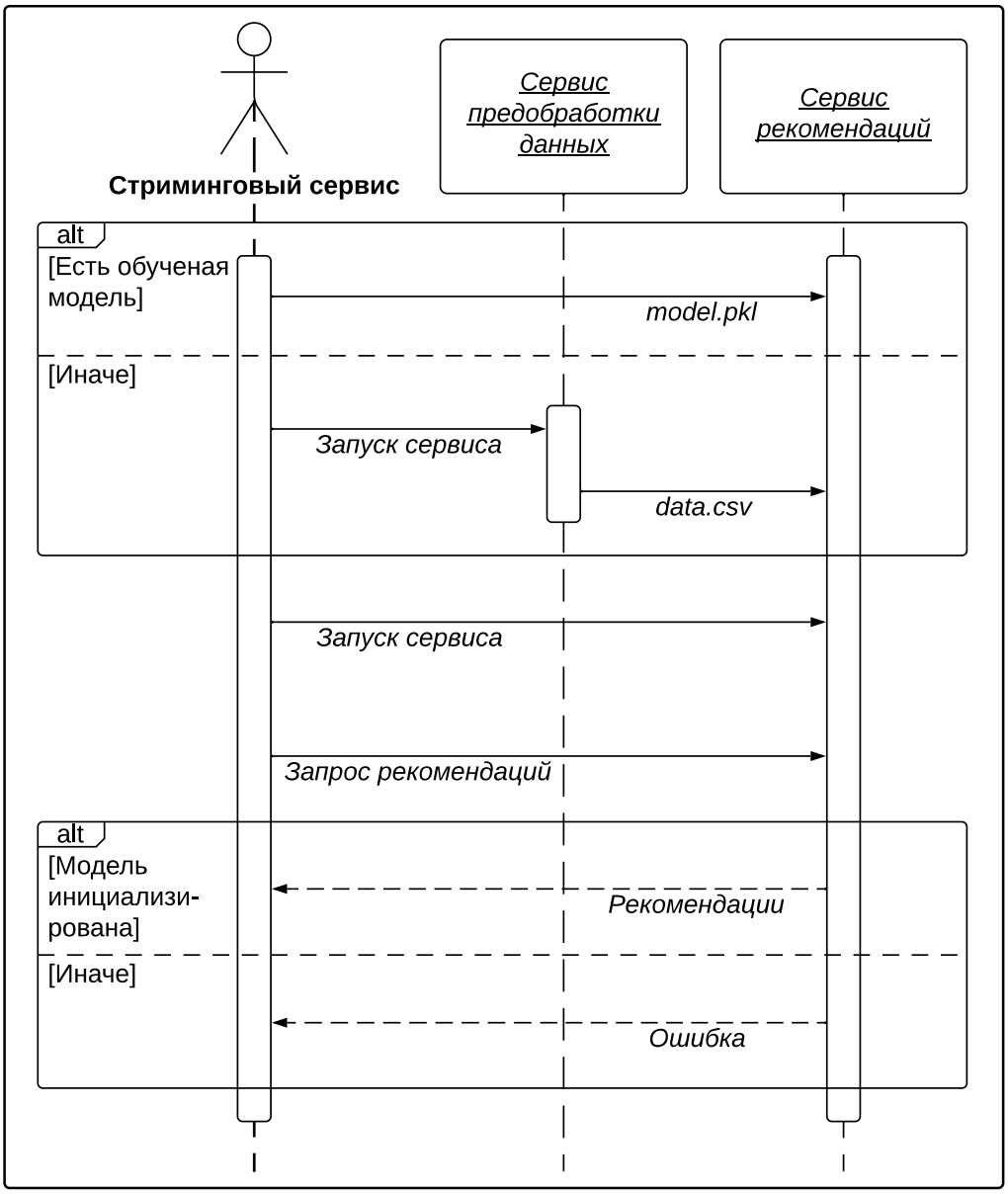

Рисунок 2 – UML диаграмма последовательности РС

Вывод по второй главе

Во второй главе были рассмотрены требования к разработке РС для музыкального стримингового сервиса, анализ исходных данных, их предобработка, а также моделирование системы.

Были определены функциональные и нефункциональные требования, необходимые для разработки эффективного РС. Важными аспектами стали время отклика, время обучения модели и использование микросервисной архитектуры.

Проведен детальный анализ предоставленных данных, включающий таблицы с информацией об артистах, авторах, событиях, жанрах и треках. Описаны структуры данных и выявлены наиболее значимые для рекомендаций признаки.

Разработан подход к предобработке данных, включающий фильтрацию и нормализацию данных, а также формирование матрицы взаимодействий. Для оценки треков использовались различные события, каждое из которых получало определенный вес в зависимости от его влияния на пользовательские предпочтения.

Описаны основные компоненты системы, такие как сервис предобработки данных и сервис рекомендаций. Подробно рассмотрены методы и инструменты, используемые для разработки этих компонентов.

В результате выполнения всех описанных шагов была создана основа для разработки и дальнейшей реализации РС, который отвечает всем поставленным требованиям.

3. РЕАЛИЗАЦИЯ

3.1. Предобработка данных

Конечная задача этапа предобработки данных – сформировать матрицу взаимодействий в формате CSV на основании файлов events.csv и tracks.csv. Для решения данной задачи подходит язык программирования Python и библиотека pandas. Чтение файлов events.csv и tracks.csv представлено в листинге 1.

Листинг 1 – Чтение файлов events.csv и tracks.csv

```
events_dtype = {
    'trackId': 'int32',
    'type': 'category',
    'personId': 'int64',
    'actionTypeContext': 'category',
    'initiatorContext': 'category',
    'deviceTimestampUtc': 'int64',
    'artistId': 'int32',
    'trackTimeMilliseconds': 'int64',
    'durationMilliseconds': 'int32',
    'countryIsoCode': 'category',
    'regionIsoCode': 'category'
}
events_chunks_df = pd.read_csv('/content/raw_data/events.csv',
dtype=events_dtype, chunksize=10_000_000)
chunks = []

for chunk in tqdm(events_chunks_df):
    chunks.append(chunk)

events_df = pd.concat(chunks, ignore_index=True)
chunks = None

for col, dtype in events_dtype.items():
    events_df[col] = events_df[col].astype(dtype)

tracks_df = pd.read_csv('/content/raw_data/tracks.csv')
```

Чтение файла events.csv происходит по частям размером 10 миллионов строк, поскольку исходный файл весит 14 гигабайт и не помещается целиком в оперативную память. Также для оптимизации размера датафрейма были явно указаны типы данных.

После чтения исходных данных требуется произвести очистку, оставив только необходимые для формирования матрицы взаимодействий данные (листинг 2).

Листинг 2 – Фильтрация данных

```
valid_type = ['STREAMING_END', 'STREAMING_FULL_WITHOUT_REWIND', 'STREAM-
ING_FULL', 'NOT_SUGGEST', 'FAVORITE_REMOVE', 'UN_NOT_SUGGEST', 'FAVOR-
ITE_ADD']
valid_initiatorContext = ['Application', 'Notification', 'Headset', 'Blue-
tooth', 'AppWidget']
def event_filter(df):
    return df[(df['type'].isin(valid_type)) &
(df['initiatorContext'].isin(valid_initiatorContext))]
duration_df = tracks_df[['TRACK_ID', 'DURATION']].copy()
duration_df['DURATION'] *= 1000  # Конвертация из секунд в миллисекунды
duration_df = duration_df[(duration_df['DURATION'] >= 30 * 1000) & (dura-
tion_df['DURATION'] <= 15 * 60 * 1000)]
duration_df.rename(columns={'TRACK_ID': 'trackId', 'DURATION': 'trackDura-
tion'}, inplace=True)
filtered_events = event_filter(events_df)
filtered_events = filtered_events.merge(duration_df, on='trackId',
how='right')
```

Обработка отфильтрованных данных согласно пункту 2.3 главы «Проектирование» представлена в листинге 3.

Листинг 3 – Расчет пользовательских рейтингов

```
FULL_STREAMING_TYPES = {'STREAMING_FULL_WITHOUT_REWIND', 'STREAMING_FULL'}
NEGATIVE_EVENTS = {'NOT_SUGGEST', 'FAVORITE_REMOVE'}
POSITIVE_EVENTS = {'UN_NOT_SUGGEST', 'FAVORITE_ADD'}
# Векторизованный расчет оценки
filtered_events['score'] = 0.0  # Инициализация столбца оценок
# Условия для STREAMING_END
condition = (filtered_events['type'] == 'STREAMING_END') & (fil-
tered_events['durationMilliseconds'] > 0) & (fil-
tered_events['trackDuration'] > 0)
filtered_events.loc[condition, 'score'] = (fil-
tered_events['durationMilliseconds'] / filtered_events['trackDuration'] -
0.3).clip(upper=0.7)
# Условия для других типов событий
filtered_events.loc[filtered_events['type'].isin(FULL_STREAMING_TYPES),
'score'] = 1
filtered_events.loc[filtered_events['type'].isin(NEGATIVE_EVENTS), 'score']
= -5
filtered_events.loc[filtered_events['type'].isin(POSITIVE_EVENTS), 'score']
= 5
# Агрегация результатов
rating_df = filtered_events.groupby(['personId',
'trackId'])['score'].sum().reset_index()
rating_df['personId'] = rating_df['personId'].astype('int')
# Переименуем столбцы
rating_df.rename(columns={
    'personId': 'userID',
    'trackId': 'itemID',
    'score': 'rating'
}, inplace=True)
```

График плотности вероятности значения оценок до удаления данных представлен на рисунке 3.

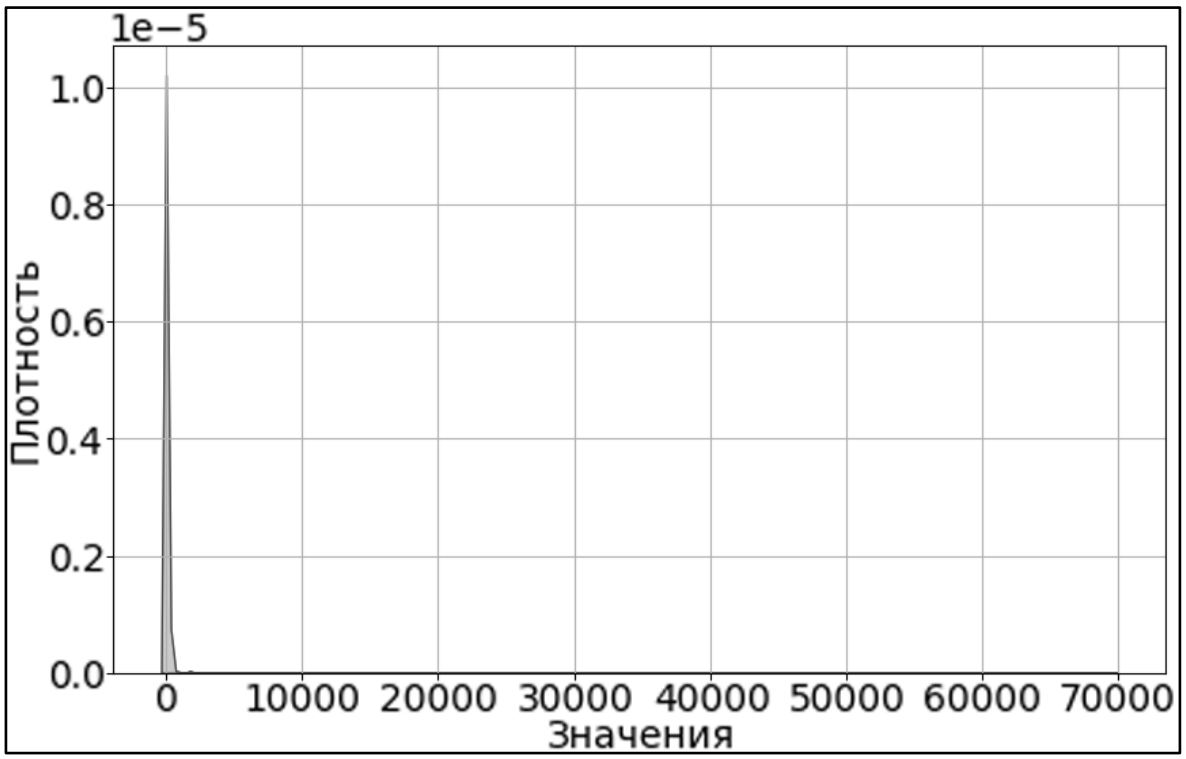

Рисунок 3 – График плотности вероятности оценок (rating) до удаления данных

Характеристики датафрейма с рейтингами (rating_df) до удаления данных представлены в таблице 11.

Таблица 11 – Характеристики rating_df до удаления данных

Параметр	**Столбец**		
	userID	**itemID**	**rating**
count	1,565386e+07	1,565386e+07	1,565386e+07
mean	4,971136e+07	2,142880e+07	2,032512e+00
std	9,605081e+06	6,488626e+06	2,328187e+01
min	1,000000e+00	3,200000e+01	-3,651977e+02
25%	5,149818e+07	2,308157e+07	-2,743103e-01
50%	5,191984e+07	2,477394e+07	-3,636306e-02
75%	5,224248e+07	2,488664e+07	1,700000e+00
max	5,265662e+07	1,003069e+08	6,999969e+04

Удаление значений близких к нулю представлено в листинге 4.

Листинг 4 – Удаление околонулевых значений

```
rating_df=rating_df[(rating_df['rating'] < -2) | (rating_df['rating'] > 4)]
```

Удаление аномальных значений с использованием квантилей (0,005 и 0,995) представлено в листинге 5.

Листинг 5 – Удаление выбросов

```
Q1 = rating_df['rating'].quantile(0.005)
Q3 = rating_df['rating'].quantile(0.995)
rating_df = rating_df[(rating_df['rating']>=Q1)&(rating_df['rating']<=Q3)]
rating_df.sort_values(by='rating')
```

Характеристики датафрейма с рейтингами (rating_df) после удаления выбросов и ненужных данных представлены в таблице 12.

Таблица 12 – Характеристики rating_df после предобработки данных

Параметр	**Столбец**		
	userID	**itemID**	**rating**
count	1,709440e+06	1,709440e+06	1,709440e+06
mean	5,074227e+07	2,036225e+07	1,383649e+01
std	6,782702e+06	7,422983e+06	1,821466e+01
min	2,350000e+03	4,200000e+01	-6,016488e+00
25%	5,137688e+07	1,705916e+07	5,100000e+00
50%	5,186558e+07	2,469109e+07	7,781010e+00
75%	5,221624e+07	2,488183e+07	1,502485e+01
max	5,265643e+07	1,003069e+08	1,669990e+02

График плотности вероятности значения оценок после предобработки данных представлен на рисунке 4.

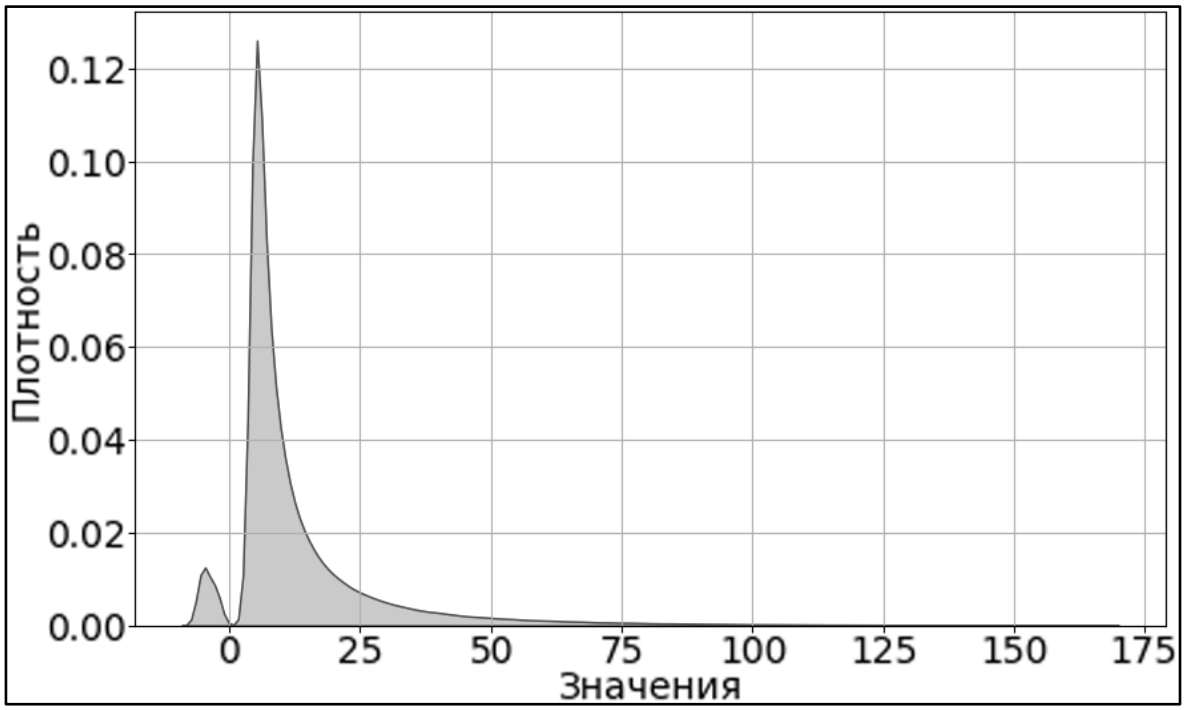

Рисунок 4 – График плотности вероятности оценок (rating) после предобработки данных

Итоговый датасет содержит 1 709 440 строки. Сохранение датаста представлено в листинге 6.

Листинг 6 – Сохранение датасета в формате CSV

```
rating_df.to_csv('rating_filtered_df.csv', index=False)
```

3.2. Выбор и обучение моделей

В данном пункте были реализованы и адаптированы шесть моделей коллаборативной фильтрации, представленные в исследовании «Microsoft Recommenders: Best Practices for Production-Ready Recommendation Systems» (2020) [19]. Для каждой из моделей была проведена настройка параметров с целью оптимизации эффективности на датасете, полученном в результате предобработки данных.

Сравнение качества рекомендаций, представляемых моделями, будет осуществлено при помощи метрик MAP [20], nDCG@k [21], Precision@k [22] и Recall@k [22].

Все рассматриваемые модели были обучены на выборке, составляющей 3,5% от исходного датасета (182 301 записей). Данные были разделены в соотношении 75% на 25%, что соответствует 136 725 записям в обучающем наборе и 45 576 записям в тестовом наборе. Разделение выборки проводилось с использованием фиксированного значения параметра seed, обеспечивающего воспроизводимость условий эксперимента для всех моделей.

Ниже представлены ключевые параметры и результаты, полученные для каждой из обученных моделей.

Alternating Least Squares (ALS) [23]

Alternating Least Squares (ALS) – метод, используемый для факторизации матриц в контексте рекомендательных систем. ALS решает задачу минимизации потерь, поочередно фиксируя один набор параметров (например, пользовательские векточры) и оптимизируя другой (например, векторы товаров).

Реализация модели представлена в листинге 1 приложения А. Параметры обучения модели представлены в таблице 13.

Таблица 13 – Параметры обучения ALS

Параметр	Значение	Описание
Число факторов (rank)	10	Определяет размерность скрытого пространства, в котором представлены пользователи и элементы
Максимальное количество итераций (maxIter)	15	Максимальное число итераций алгоритма, прежде чем он остановится, если не сошелся ранее
Неявные предпочтения (implicitPrefs)	False	При значении True модель предполагает, что данные о предпочтениях неявные
Коэффициент регуляризации (regParam)	0,05	Используется для уменьшения переобучения, добавляя штраф к величине параметров модели
Стратегия при холодном старте (coldStartStrategy)	'drop'	Метод обработки новых пользователей или элементов во время прогнозирования
Ненегативные значения (nonnegative)	False	При значении True все факторы должны быть неотрицательными

Bilateral Variational Autoencoder (BiVAE) [24]

Bilateral Variational Autoencoder (BiVAE) – модификация вариационного автоэнкодера, предназначенная для использования в коллаборативных рекомендательных системах. Основное отличие BiVAE заключается в том, что BiVAE обучает две связанные модели: одну для пользователей и одну для предметов рекомендаций, что позволяет более точно моделировать интересы пользователей и свойства предметов. Это повышает качество и персонализацию рекомендаций.

Реализация модели представлена в листинге 2 приложения А. Параметры обучения модели представлены в таблице 14.

Таблица 14 – Параметры обучения BiVAE

Параметр	Значение	Описание
Число скрытых факторов (k)	50	Размерность скрытого пространства в автоэнкодере
Структура энкодера (encoder_structure)	[100]	Архитектура энкодера, указывающая количество и размерность скрытых слоев
Функция активации (act_fn)	'tanh'	Функция активации, используемая в слоях нейронной сети

Окончание таблицы 14

Параметр	Значение	Описание
Вероятностная модель (likelihood)	'pois'	Тип используемого распределения для моделирования данных – пуассоновская вероятностная модель
Число эпох (n_epochs)	500	Количество полных проходов по набору данных в процессе обучения
Размер батча (batch_size)	128	Количество образцов данных, обрабатываемых за одну итерацию
Скорость обучения (learning_rate)	0,001	Скорость, с которой модель обновляет параметры

Bayesian Personalized Ranking (BPR) [25]

Bayesian Personalized Ranking (BPR) – метод машинного обучения, основанный на байесовском подходе к персонализированной ранжировке. BPR оптимизирует порядок рекомендаций на основе предпочтений пользователей, предпочитая положительные взаимодействия перед отрицательными.

Реализация модели представлена в листинге 3 приложения А. Параметры обучения модели представлены в таблице 15.

Таблица 15 – Параметры обучения BPR

Параметр	Значение	Описание
Число скрытых факторов (k)	200	Размерность векторов скрытых факторов для пользователей и элементов
Максимальное количество итераций (max_iter)	100	Максимальное число итераций для оптимизации модели
Скорость обучения (learning_rate)	0,01	Шаг обновления параметров в процессе оптимизации
Регуляризация (lambda_reg)	0,001	Коэффициент регуляризации для контроля переобучения

FastAI Embedding Dot Bias (FastAI) [26]

FastAI Embedding Dot Bias (FastAI) – метод, реализованный в библиотеке FastAI, использующий векторные представления (эмбеддинги) для пользователей и товаров с добавлением члена смещения для каждого пользователя и товара, чтобы улучшить качество рекомендаций.

Реализация модели представлена в листинге 4 приложения А. Параметры обучения модели представлены в таблице 16.

Таблица 16 – Параметры обучения FastAI

Параметр	Значение	Описание
Количество факторов (n_factors)	40	Количество латентных факторов, используемых в модели
Диапазон предсказываемых значений (y_range)	[-6, 100]	Диапазон, в котором модель ограничивает свои прогнозы
Коэффициент весового распада (wd)	0,1	Параметр регуляризации для снижения переобучения модели
Максимальная скорость обучения (lr_max)	0,005	Максимальная скорость обучения

Neural Collaborative Filtering (NCF) [27]

Neural Collaborative Filtering (NCF) – это подход к рекомендательным системам, который интегрирует многослойные перцептроны (MLP) и метод обобщенной матричной факторизации (GMF) для моделирования скрытых линейных и нелинейных структур взаимодействий между пользователями и товарами. Это позволяет изучать более сложные зависимости, улучшая точность рекомендаций.

Реализация модели представлена в листинге 5 приложения А. Параметры обучения модели представлены в таблице 17.

Таблица 17 – Параметры обучения NCF

Параметр	Значение	Описание
Количество эпох (n_epochs)	100	Общее количество проходов по обучающему набору данных
Размер батча (batch_size)	256	Количество примеров, обрабатываемых в одном обучающем проходе
Скорость обучения (learning_rate)	0,001	Шаг оптимизации параметров модели
Количество факторов (n_factors)	4	Размерность факторов, используемых в модели
Размеры слоев (layer_sizes)	[16, 8, 4]	Размеры скрытых слоев модели
Коэффициент смешивания GMF и MLP (alpha)	0,5	Степень влияния обобщенной матричной факторизации и многослойного перцептрона на итоговую модель

Simple Algorithm for Recommendation (SAR) [28]

Simple Algorithm for Recommendation (SAR) – базовый алгоритм для создания рекомендаций, который использует совместную фильтрацию, основанную на подсчете и анализе совместных взаимодействий между пользователями и товарами. SAR учитывает популярность и схожесть элементов для формирования рекомендаций.

Реализация модели представлена в листинге 6 приложения А. Параметры обучения модели представлены в таблице 18.

Таблица 18 – Параметры обучения SAR

Параметр	Значение	Описание
Тип метрики сходства (similarity_type)	'jaccard'	Выбор алгоритма для расчета сходства между элементами – индекс Жаккара
Нормализация матрицы сходства (normalize)	True	Применение нормализации к значениям в матрице сходства для улучшения производительности рекомендаций

Оценка моделей

В таблице 19 представлены конечные метрики и временные параметры обученных моделей.

Таблица 19 – Метрики и временные параметры моделей

Модель	MAP	nDCG@k	Precision @k	Recall@k	Время обучения, [с]	Время предсказания, [с]
BiVAE	0,10554	0,17291	0,10902	0,19227	368,32	36,12
SAR	0,08201	0,14547	0,08227	0,15915	3,13	1,34
BPR	0,05159	0,09739	0,06684	0,11617	1,18	38,72
FastAI	0,04638	0,08345	0,05432	0,10162	18,52	33,45
NCF	0,04118	0,08520	0,05221	0,10073	7968,79	177,27
ALS	0,00013	0,00028	0,00016	0,00018	6,38	55,69

Для визуализации данных представим метрики в виде столбчатой диаграммы (рисунок 5).

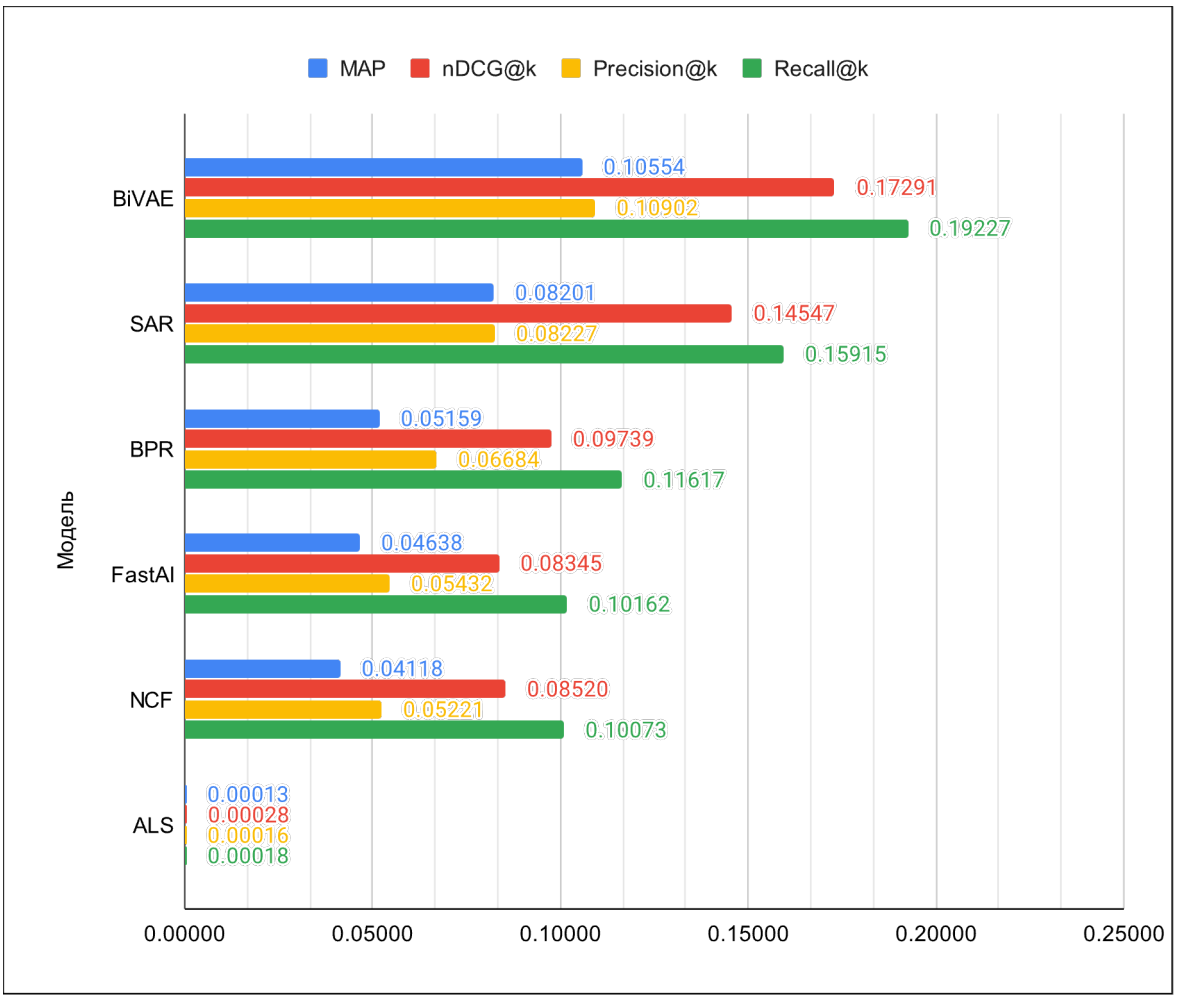

Рисунок 5 – Диаграмма метрик обученных моделей

Охарактеризуем каждую модель на основе таблицы 19 и рисунка 5.

BiVAE показывает наилучшие результаты по всем метрикам качества среди представленных моделей, но обучение и предсказание занимают значительно больше времени по сравнению с другими алгоритмами.

SAR занимает мало времени на обучение и предсказание. Показатели качества ниже, чем у BiVAE, но превосходят остальные алгоритмы, что делает SAR хорошим выбором для систем с ограничениями по времени.

BPR требует минимальное время на обучение, однако время предсказания довольно высокое. Метрики качества средние.

NCF занимает очень много времени на обучение и предсказание, при этом показывая одни из наименьших результатов по метрикам качества.

ALS показывает крайне низкие результаты по всем метрикам, несмотря на быстрое время обучения.

Общий вывод: если скорость обучения и предсказания не является критическим фактором, BiVAE представляется лучшим выбором из-за высоких показателей качества. SAR может быть хорошим компромиссом, предлагая приемлемое качество с низким временем на обучение и предсказание.

3.3. Реализация рекомендательного сервиса

Для реализации рекомендательной системы была выбрана модель BiVAE, представленная в листинге 8 приложения Б. Этот выбор обусловлен тем, что модель BiVAE соответствует всем критериям, предъявляемым к РС, и показывает наилучшие результаты по сравнению с другими моделями, прошедшими тестирование в рамках данного проекта.

Для создания REST API используется веб-фреймворк FastAPI, упомянутый в листинге 7 приложения Б. Серверная часть приложения развертывается на ASGI сервере Uvicorn, что позволяет обеспечить высокую производительность и асинхронную обработку запросов.

Обеспечение изоляции и управление рабочей средой происходит посредством Docker. Создается специализированный Docker-образ, который настраивается для работы в CUDA-окружении, как указано в листинге 9 приложения Б. В образ включаются все необходимые зависимости из фала requirements.txt (листинг 10 приложения Б).

Для сборки и запуска РС необходимо выполнить команды, описанные в листинге 7.

Листинг 7 – Сборка и запуск docker контейнера

```
docker build -t recomend-system --build-arg IMAGE_NAME=nvidia/cuda .
docker run --gpus=all -p 0.0.0.0:80:80 recomend-system:latest
```

После запуска контейнера начинается автоматическая инициализация модели BiVAE. Процесс обучения модели можно отслеживать в ре-

альном времени с помощью логов, выводимых в консоль Docker (рисунок 6).

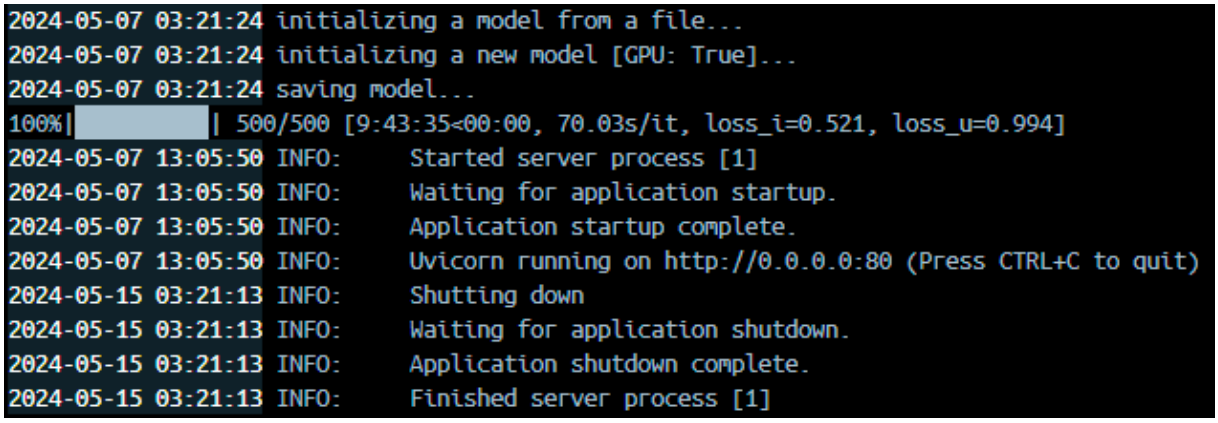

Рисунок 6 – Процесс обучения модели

После того как обучение модели завершается, начинает функционировать REST API, который играет ключевую роль в обеспечении коммуникации между PC и стриминговым сервисом. Этот API позволяет стриминговому сервису направлять запросы непосредственно к модели, чтобы в режиме реального времени получать актуальные рекомендации, основанные на последних данных и пользовательских предпочтениях.

Используемый веб-фреймворк FastAPI выделяется не только высокой производительностью и асинхронностью, но и предоставлением широких возможностей для документирования API. В частности, FastAPI интегрируется с Swagger UI, что позволяет разработчикам легко взаимодействовать с API через интерактивный интерфейс. Эта документация, отображаемая в Swagger UI (рисунок 1 листинга В), оказывается невероятно полезной для тестирования API и его интеграции в различные приложения.

Вывод по третьей главе

В третьей главе была рассмотрена реализация PC для музыкального стримингового сервиса. Была проведена предобработка данных, которая включала очистку, нормализацию и агрегацию данных. Эти шаги были необходимы для создания матрицы взаимодействий, на основе которой далее строились рекомендации. После подготовки данных были выбраны и

обучены несколько моделей коллаборативной фильтрации, среди которых Alternating Least Squares (ALS), Bilateral Variational Autoencoder (BiVAE), Bayesian Personalized Ranking (BPR), FastAI Embedding Dot Bias (FastAI), Neural Collaborative Filtering (NCF) и Simple Algorithm for Recommendation (SAR). Каждая из моделей была тщательно настроена и протестирована для определения наилучшей эффективности на подготовленном наборе данных.

Сравнительный анализ показал, что модель BiVAE продемонстрировала наилучшие результаты по всем основным метрикам качества, таким как Mean Average Precision (MAP), Normalized Discounted Cumulative Gain (nDCG@k), Precision@k и Recall@k. Несмотря на то, что BiVAE требует больше времени на обучение и предсказание по сравнению с другими моделями, её высокие показатели точности делают её предпочтительным выбором для реализации РС. В качестве альтернативы для систем с ограниченными временными ресурсами может быть использована модель SAR, которая обеспечивает приемлемое качество рекомендаций при минимальном времени на обучение и предсказание.

Реализация РС включала создание микросервисной архитектуры, использующей REST API для взаимодействия со стриминговым сервисом. Сервис был развернут в Docker-контейнере, обеспечивающем изоляцию и консистентность рабочей среды. Для тестирования и интеграции API использовалась документация в формате Swagger UI, что должно упростить взаимодействие с системой и ее тестирование.

Таким образом, в третьей главе была успешно реализована и протестирована рекомендательная система, отвечающая всем установленным требованиям и готовая к внедрению в стриминговый сервис.

4. ТЕСТИРОВАНИЕ

Глава посвящена тестированию рекомендательного сервиса с целью проверки соответствия готового решения предъявляемым функциональным (таблица 20) и нефункциональным требованиям (таблица 21).

Таблица 20 – Функциональное тестирование

№	Дейвствие	Ожидаемое поведение	Пройден
1	Предобработка данных	Получение предобработанного датасета в формате CSV	Да
2	Сборка Docker образа	Docker образ собирается	Да
3	Запуск Docker контейнера с предобработанными данными	Docker контейнер запускается. Начинается процесс обучения модели. По завершению обучения становится доступен GET метод REST API	Да
4	Запуск Docker контейнера с обученной моделью	Docker контейнер запускается. Модель инициализируется обученной моделью. По завершению инициализации модели становится доступен GET метод REST API	Да
5	Мониторинг процесса обучения	В консоль Docker контейнера выводится прогресс обучения	Да
6	Получение рекомендаций для существующего пользователя	Ответ с кодом 200 со списком рекомендованных треков размером 60	Да
7	Получение рекомендаций для отсутствующего пользователя	Ответ с кодом 500 с описанием ошибки	Да
8	Получение ограниченного числа рекомендаций (track_count = 10)	Ответ с кодом 200 со списком рекомендованных треков размером 10	Да
9	Получение рекомендаций без известных пользователю треков (remove_known = True)	Ответ с кодом 200 со списком рекомендованных треков. Список может содержать ранее прослушанные треки	Да

Таблица 21 – Нефункциональное тестирование

№	Параметр	Ожидаемый результат	Итоговый результат
1	Время отклика (получения рекомендаций)	50 миллисекунд	30 миллисекунд
2	Время обучения модели	Менее 24 часов	9 часов 45 минут
3	Использование графических ускорителей для обучения модели	Используется	Используется
4	Микросервисная архитектура с использованием REST API	Используется	Используется
5	Применение Docker контейнеризации	Используется	Используется

Вывод по четвертой главе

Тестирование подтвердило успешное выполнение всех требований.

ЗАКЛЮЧЕНИЕ

В рамках данной работы был разработан рекомендательный сервис для музыкального стримингового сервиса. В ходе выполнения поставленной задачи были решены следующие задачи.

1. Проведен анализ предметной области.
2. Проанализированы требования и исходные данные, на основании которых было произведено проектирование архитектуры рекомендательного сервиса.
3. Проанализированы и подготовлены данные для обучения рекомендательных моделей.
4. Исследованы и обучены рекомендательные модели. Для каждой из моделей проведена настройка параметров с целью оптимизации их эффективности на подготовленных данных. На базе наиболее результативной модели была реализована рекомендательная система.
5. Проведено функциональное и нефункциональное тестирование, которое подтвердило соответствие системы установленным требованиям

На момент написания выпускной квалификационной работы разработанный сервис предложен к внедрению в рамках стримингового сервиса «Zaycev.net» [17].

ЛИТЕРАТУРА

1. Коммерсантъ. Стриминги наращивают темп. [Электронный ресурс] URL: https://www.kommersant.ru/doc/6310128 (дата обращения: 23.02.2024 г.).

2. Демин В.А. Digital-трансформация маркетинга в музыкальной индустрии: актуальные тенденции. // Всероссийская научно-практическая конференция, 2021. – № 15. – С. 248–253.

3. Wanling C., Yucheng J., Li C. Critiquing for Music Exploration in Conversational Recommender Systems. // International Conference on Intelligent User Interfaces, 2021. – №26. – 480–490 pp. DOI: 10.1145/3397481.3450657.

4. List of conferences. [Электронный ресурс] URL: https://recsys.acm.org (дата обращения: 10.02.2024 г.).

5. Аблякимова А.Н., Ибраимов А.Г. Обзор популярных музыкальных стриминг-сервисов. // Информационно-компьютерные технологии в экономике, образовании и социальной сфере, 2018. – № 4. – С. 147–155.

6. A Python scikit for recommender systems. [Электронный ресурс] URL: https://surpriselib.com (дата обращения: 10.02.2024 г.).

7. Python Tools for Recommender Experiments. [Электронный ресурс] URL: https://lenskit.org (дата обращения: 10.02.2024 г.).

8. The Netflix Prize. [Электронный ресурс] URL: https://www.netflixprize.com (дата обращения: 10.02.2024 г.).

9. Как это работает? Рекомендации в Яндекс.Музыке. [Электронный ресурс] URL: https://yandex.ru/blog/company/92883 (дата обращения: 10.02.2024 г.).

10. Янкина Е.А. Подходы к построению рекомендательных систем, виды гибридизации систем на основе коллаборативной и контентной фильтрации. // Сборник трудов Всероссийской научной конференции молодых ученых, аспирантов и студентов, 2018. – № 16. – С. 267–270.

11. Sarwar B., Karypis G., Konstan G., Riedl J. Item-based collaborative filtering recommendation algorithmus. // Proceedings of the international conference on World Wide Web, 2001. – № 10. – 285–295 pp.

12. Celma O. Music Recommendation and Discovery: The Long Tail, Long Fail, and Long Play in the Digital Music Space. // Springer, 2010. – C. 210. DOI: 10.1007/978-3-642-13287-2.

13. Miao J., Ziyi Y., Chen Z. What to play next? A RNN-based music recommendation system. // Asilomar Conference on Signals, Systems, and Computers, 2017. – № 51. – 356–358 pp. DOI: 10.1109/ACSSC.2017.8335200.

14. Gunjan K. A sequence-based and context modelling framework for recommendation. / K. Gunjan, J. Houssem, P. Michael // Dublin: University College Dublin, 2019. – 167 p. DOI: 10.1016/j.eswa.2021.114665.

15. Andreas T., Michael J. The BigChaos Solution to the Netflix Grand Prize. // Netflix Prize, 2009. – C. 52.

16. Marco T., Sameer S., Carlos G. «Why Should I Trust You?»: Explaining the Predictions of Any Classifier [Электронный ресурс] // arXiv.org. 2016. Дата обновления: 16.08.2016. URL: https://arxiv.org/abs/1602.04938v3 (дата обращения: 10.04.2024 г.).

17. Музыкальный портал «Zaycev.net». [Электронный ресурс] URL: https://zaycev.net (дата обращения: 10.04.2024 г.).

18. What is collaborative filtering? [Электронный ресурс] URL: https://cotera.co/blog/what-is-collaborative-filtering (дата обращения: 10.04.2024 г.).

19. Andreas A., Miguel G., Le Z. Microsoft Recommenders: Best Practices for Production-Ready Recommendation Systems. // Companion Proceedings of the Web Conference, 2020. – 50–51 pp. DOI: 10.1145/3366424.3382692.

20. Mean Average Precision (MAP) in ranking and recommendations. [Электронный ресурс] URL: https://www.evidentlyai.com/ranking-metrics/mean-average-precision-map (дата обращения: 10.04.2024 г.).

21. Normalized Discounted Cumulative Gain (NDCG) explained. [Электронный ресурс] URL: https://www.evidentlyai.com/ranking-metrics/ndcg-metric (дата обращения: 10.04.2024 г.).

22. Precision and recall at K in ranking and recommendations. [Электронный ресурс] URL: https://www.evidentlyai.com/ranking-metrics/precision-recall-at-k (дата обращения: 10.04.2024 г.).

23. Source code for pyspark.ml.recommendation. [Электронный ресурс] URL: https://spark.apache.org/docs/latest/api/python/_modules/pyspark/ml/recommendation.html#ALS (дата обращения: 10.04.2024 г.).

24. Quoc-Tuan T., Aghiles S., Hady W. Bilateral Variational Autoencoder for Collaborative Filtering. // Proceedings of the 14th ACM International Conference on Web Search and Data Mining, 2021. – 292–300 pp. DOI: 10.1145/3437963.3441759.

25. Steffen R., Christoph F., Zeno G., Lars S. BPR: Bayesian Personalized Ranking from Implicit Feedback [Электронный ресурс] // arXiv.org. 2012. Дата обновления: 09.05.2012. URL: https://arxiv.org/abs/1205.2618 (дата обращения: 10.04.2024 г.).

26. FastAI documentation. [Электронный ресурс] URL: https://docs.fast.ai/collab.html (дата обращения: 10.04.2024 г.).

27. Xiangnan H., Lizi L., Hanwang Z., Liqiang N., Xia H., Tat-Seng C. Neural Collaborative Filtering. // Proceedings of the 26th International Conference on World Wide Web, 2017. – 173–182 pp. DOI: 10.1145/3038912.3052569.

28. Smart Adaptive Recommendations documentation. [Электронный ресурс] URL: https://microsoft.github.io/SynapseML/docs/Explore%20Algorithms/Other%20Algorithms/Smart%20Adaptive%20Recommendations/ (дата обращения: 10.04.2024 г.).

ПРИЛОЖЕНИЯ

Приложение А. Код обучения моделей

Листинг 1 – Обучение ALS

```
import sys
import pandas as pd

import pyspark
from pyspark.ml.recommendation import ALS

from recommenders.utils.timer import Timer
from recommenders.datasets.python_splitters import python_random_split
from recommenders.evaluation.spark_evaluation import SparkRankingEvaluation
from recommenders.utils.spark_utils import start_or_get_spark

TOP_K = 10
RAND_SEED = 42
SAMPLE_FRAC = 0.035
USER_COL = 'userID'
ITEM_COL = 'itemID'
RATING_COL = 'rating'
PREDICTION_COL = 'prediction'

spark = start_or_get_spark("ALS PySpark", memory="16g")
spark.conf.set("spark.sql.analyzer.failAmbiguousSelfJoin", "false")

data = pd.read_csv('rating_filtered_df.csv')
unique_users = data[USER_COL].unique()
sampled_users = pd.Series(unique_users).sample(frac=SAMPLE_FRAC, ran-
dom_state=RAND_SEED)
data = data[data[USER_COL].isin(sampled_users)]
data.head()

train, test = python_random_split(data, 0.75, seed=RAND_SEED)

print ("N train", train.size)
print ("N test", test.size)

train = spark.createDataFrame(train)
test = spark.createDataFrame(test)

header = {
    "userCol": USER_COL,
    "itemCol": ITEM_COL,
}

als = ALS(
    rank=10,
    maxIter=15,
    implicitPrefs=False,
    regParam=0.05,
    coldStartStrategy='drop',
    nonnegative=False,
    seed=RAND_SEED,
    **header)
with Timer() as t:
    model = als.fit(train)
print(f"Time for training is {t} seconds")

with Timer() as t:
```

```
    users = train.select(USER_COL).distinct()
    items = train.select(ITEM_COL).distinct()
    user_item = users.crossJoin(items)
    dfs_pred = model.transform(user_item)

    dfs_pred_exclude_train = dfs_pred.alias("pred").join(
        train.alias("train"),
        (dfs_pred[USER_COL] == train[USER_COL]) & (dfs_pred[ITEM_COL] ==
train[ITEM_COL]),
        how='outer'
    )

    top_all =
dfs_pred_exclude_train.filter(dfs_pred_exclude_train[f"train.{RATING_COL}"]
.isNull()) \
        .select('pred.' + USER_COL, 'pred.' + ITEM_COL, 'pred.' + PREDIC-
TION_COL)

    top_all.cache().count()
print(f"Time for predictions is {t} seconds")

rank_eval = SparkRankingEvaluation(test, top_all, k = TOP_K,
col_user=USER_COL, col_item=ITEM_COL,
                                   col_rating=RATING_COL,
col_prediction=PREDICTION_COL,
                                   relevancy_method="top_k")

eval_map = rank_eval.map_at_k()
eval_ndcg = rank_eval.ndcg_at_k()
eval_precision = rank_eval.precision_at_k()
eval_recall = rank_eval.recall_at_k()

print(f'''MAP:\t\t{eval_map}
nDCG@k:\t\t{eval_ndcg}
Precision@k:\t{eval_precision}
Recall@k:\t{eval_recall}''')
spark.stop()
```

Листинг 2 – Обучение BiVAE

```
import os
import sys
import torch
import cornac
import pandas as pd

from recommenders.datasets.python_splitters import python_random_split
from recommenders.models.cornac.cornac_utils import predict_ranking
from recommenders.utils.timer import Timer
from recommenders.evaluation.python_evaluation import map, ndcg_at_k, pre-
cision_at_k, recall_at_k

TOP_K = 10
RAND_SEED = 42
SAMPLE_FRAC = 0.035
USER_COL = 'userID'
ITEM_COL = 'itemID'
RATING_COL = 'rating'
PREDICTION_COL = 'prediction'
```

```
data = pd.read_csv('rating_filtered_df.csv')
unique_users = data[USER_COL].unique()
sampled_users = pd.Series(unique_users).sample(frac=SAMPLE_FRAC, ran-
dom_state=RAND_SEED)
data = data[data[USER_COL].isin(sampled_users)]
data.head()

train, test = python_random_split(data, 0.75, seed=RAND_SEED)

print ("N train", train.size)
print ("N test", test.size)

bivae = cornac.models.BiVAECF(
    k=50,
    encoder_structure=[100],
    act_fn="tanh",
    likelihood="pois",
    n_epochs=500,
    batch_size=128,
    learning_rate=0.001,
    seed=RAND_SEED,
    use_gpu=torch.cuda.is_available(),
    verbose=True
)

with Timer() as t:
    bivae.fit(train_set)
print(f"Time for training is {t} seconds")

with Timer() as t:
    all_predictions = predict_ranking(bivae, train, usercol=USER_COL, item-
col=ITEM_COL, remove_seen=True)
print(f"Time for predictions is {t} seconds")

eval_map = map(test, all_predictions, col_prediction=PREDICTION_COL,
k=TOP_K)
eval_ndcg = ndcg_at_k(test, all_predictions, col_prediction=PREDICTION_COL,
k=TOP_K)
eval_precision = precision_at_k(test, all_predictions,
col_prediction=PREDICTION_COL, k=TOP_K)
eval_recall = recall_at_k(test, all_predictions,
col_prediction=PREDICTION_COL, k=TOP_K)
print(f'''MAP:\t\t{eval_map}
nDCG@k:\t\t{eval_ndcg}
Precision@k:\t{eval_precision}
Recall@k:\t{eval_recall}''')
```

Листинг 3 – Обучение BPR

```
import os
import sys
import cornac
import pandas as pd

from recommenders.datasets.python_splitters import python_random_split
from recommenders.evaluation.python_evaluation import map, ndcg_at_k, pre-
cision_at_k, recall_at_k
from recommenders.models.cornac.cornac_utils import predict_ranking
from recommenders.utils.timer import Timer
```

```
TOP_K = 10
RAND_SEED = 42
SAMPLE_FRAC = 0.035
USER_COL = 'userID'
ITEM_COL = 'itemID'
RATING_COL = 'rating'
PREDICTION_COL = 'prediction'

data = pd.read_csv('rating_filtered_df.csv')
unique_users = data[USER_COL].unique()
sampled_users = pd.Series(unique_users).sample(frac=SAMPLE_FRAC, ran-
dom_state=RAND_SEED)
data = data[data[USER_COL].isin(sampled_users)]

train, test = python_random_split(data, 0.75, seed=RAND_SEED)

print ("N train", train.size)
print ("N test", test.size)

train_set = cornac.data.Dataset.from_uir(train.itertuples(index=False),
seed=RAND_SEED)

bpr = cornac.models.BPR(
    k=200,
    max_iter=100,
    learning_rate=0.01,
    lambda_reg=0.001,
    verbose=True,
    seed=RAND_SEED
)

with Timer() as t:
    bpr.fit(train_set)
print(f"Time for training is {t} seconds")

with Timer() as t:
    all_predictions = predict_ranking(bpr, train, usercol=USER_COL, item-
col=ITEM_COL, remove_seen=True)
print(f"Time for predictions is {t} seconds")

eval_map = map(test, all_predictions, col_prediction=PREDICTION_COL,
k=TOP_K)
eval_ndcg = ndcg_at_k(test, all_predictions, col_prediction=PREDICTION_COL,
k=TOP_K)
eval_precision = precision_at_k(test, all_predictions,
col_prediction=PREDICTION_COL, k=TOP_K)
eval_recall = recall_at_k(test, all_predictions,
col_prediction=PREDICTION_COL, k=TOP_K)

print(f'''MAP:\t\t{eval_map}
nDCG@k:\t\t{eval_ndcg}
Precision@k:\t{eval_precision}
Recall@k:\t{eval_recall}''')
```

Листинг 4 – Обучение FastAI

```
import os
import sys
import numpy as np
import pandas as pd
```

```
import torch
import fastai
from tempfile import TemporaryDirectory

from fastai.collab import collab_learner, CollabDataLoaders, load_learner

from recommenders.utils.timer import Timer
from recommenders.datasets.python_splitters import python_random_split
from recommenders.models.fastai.fastai_utils import cartesian_product,
score
from recommenders.evaluation.python_evaluation import map, ndcg_at_k, pre-
cision_at_k, recall_at_k

TOP_K = 10
RAND_SEED = 42
SAMPLE_FRAC = 0.035
USER_COL = 'userID'
ITEM_COL = 'itemID'
RATING_COL = 'rating'
PREDICTION_COL = 'prediction'

data = pd.read_csv('rating_filtered_df.csv')
unique_users = data[USER_COL].unique()
sampled_users = pd.Series(unique_users).sample(frac=SAMPLE_FRAC, ran-
dom_state=RAND_SEED)
data = data[data[USER_COL].isin(sampled_users)]

data[USER_COL] = data[USER_COL].astype('str')
data[ITEM_COL] = data[ITEM_COL].astype('str')
data.head()
train, test = python_random_split(data, 0.75, seed=RAND_SEED)

print ("N train", train.size)
print ("N test", test.size)

np.random.seed(RAND_SEED)
torch.manual_seed(RAND_SEED)
torch.cuda.manual_seed_all(RAND_SEED)

with Timer() as preprocess_time:
    data = CollabDataLoaders.from_df(train,
                                     user_name=USER_COL,
                                     item_name=ITEM_COL,
                                     rating_name=RATING_COL,
                                     valid_pct=0)

data.show_batch()

learn = collab_learner(data, n_factors=40, y_range=[-6,100], wd=1e-1)
learn.model
with Timer() as t:
    learn.fit_one_cycle(5, lr_max=5e-3)

print(f"Time for training is {t} seconds")
tmp = TemporaryDirectory()
model_path = os.path.join(tmp.name, "fastai_model.pkl")

learn.export(model_path)

learner = load_learner(model_path)
```

```
total_users, total_items = learner.dls.classes.values()
total_items = total_items[1:]
total_users = total_users[1:]

test_users = test[USER_COL].unique()
test_users = np.intersect1d(test_users, total_users)

users_items = cartesian_product(np.array(test_users),np.array(total_items))
users_items = pd.DataFrame(users_items, columns=[USER_COL,ITEM_COL])

training_removed = pd.merge(users_items, train.astype(str), on=[USER_COL,
ITEM_COL], how='left')
training_removed = train-
ing_removed[training_removed[RATING_COL].isna()][[USER_COL, ITEM_COL]]

with Timer() as t:
    top_k_scores = score(learner,
                         test_df=training_removed,
                         user_col=USER_COL,
                         item_col=ITEM_COL,
                         prediction_col=PREDICTION_COL)

print(f"Time for predictions is {t} seconds")

eval_map = map(test, top_k_scores, col_user=USER_COL, col_item=ITEM_COL,
               col_rating=RATING_COL, col_prediction=PREDICTION_COL,
               relevancy_method="top_k", k=TOP_K)
eval_ndcg = ndcg_at_k(test, top_k_scores, col_user=USER_COL,
col_item=ITEM_COL,
                      col_rating=RATING_COL, col_prediction=PREDICTION_COL,
                      relevancy_method="top_k", k=TOP_K)
eval_precision = precision_at_k(test, top_k_scores, col_user=USER_COL,
col_item=ITEM_COL,
                                col_rating=RATING_COL,
col_prediction=PREDICTION_COL,
                                relevancy_method="top_k", k=TOP_K)
eval_recall = recall_at_k(test, top_k_scores, col_user=USER_COL,
col_item=ITEM_COL,
                          col_rating=RATING_COL,
col_prediction=PREDICTION_COL,
                          relevancy_method="top_k", k=TOP_K)

print(f'''MAP:\t\t{eval_map}
nDCG@k:\t\t{eval_ndcg}
Precision@k:\t{eval_precision}
Recall@k:\t{eval_recall}''')

tmp.cleanup()
```

Листинг 5 – Обучение NCF

```
import os
import sys
import shutil
import numpy as np
import pandas as pd
import tensorflow as tf

from recommenders.utils.timer import Timer
from recommenders.models.ncf.ncf_singlenode import NCF
```

```
from recommenders.models.ncf.dataset import Dataset as NCFDataset
from recommenders.datasets.python_splitters import python_random_split
from recommenders.evaluation.python_evaluation import map, ndcg_at_k, pre-
cision_at_k, recall_at_k

TOP_K = 10
RAND_SEED = 42
SAMPLE_FRAC = 0.035
USER_COL = 'userID'
ITEM_COL = 'itemID'
RATING_COL = 'rating'
PREDICTION_COL = 'prediction'

# Model parameters
EPOCHS = 100
BATCH_SIZE = 256

data = pd.read_csv('rating_filtered_df.csv')
unique_users = data[USER_COL].unique()
sampled_users = pd.Series(unique_users).sample(frac=SAMPLE_FRAC, ran-
dom_state=RAND_SEED)
data = data[data[USER_COL].isin(sampled_users)]
data.head()

train, test = python_random_split(data, 0.75, seed=RAND_SEED)

print ("N train", train.size)
print ("N test", test.size)


test = test[test[USER_COL].isin(train[USER_COL].unique())]
test = test[test[ITEM_COL].isin(train[ITEM_COL].unique())]
leave_one_out_test = test.groupby(USER_COL).last().reset_index()
train_file = "./train.csv"
test_file = "./test.csv"
leave_one_out_test_file = "./leave_one_out_test.csv"
train.sort_values(USER_COL).to_csv(train_file, index=False)
test.to_csv(test_file, index=False)
leave_one_out_test.to_csv(leave_one_out_test_file, index=False)

data = NCFDataset(train_file=train_file, test_file=leave_one_out_test_file,
seed=RAND_SEED, overwrite_test_file_full=True)
model = NCF(
    n_users=data.n_users,
    n_items=data.n_items,
    model_type="NeuMF",
    n_factors=4,
    layer_sizes=[16,8,4],
    n_epochs=EPOCHS,
    batch_size=BATCH_SIZE,
    learning_rate=1e-3,
    verbose=10,
    seed=RAND_SEED
)
with Timer() as t:
    model.fit(data)
print(f"Time for training is {t} seconds")

predictions = [[row.userID, row.itemID, model.predict(row.userID,
row.itemID)]
               for (_, row) in test.iterrows()]
```

```
predictions = pd.DataFrame(predictions, columns=['userID', 'itemID', 'pre-
diction'])
predictions.head()

with Timer() as t:

    users, items, preds = [], [], []
    item = list(train.itemID.unique())
    for user in train.userID.unique():
        user = [user] * len(item)
        users.extend(user)
        items.extend(item)
        preds.extend(list(model.predict(user, item, is_list=True)))

    all_predictions = pd.DataFrame(data={USER_COL:users, ITEM_COL:items,
PREDICTION_COL:preds})

    merged = pd.merge(train, all_predictions, on=[USER_COL, ITEM_COL],
how="outer")
    all_predictions = merged[merged.rating.isnull()].drop('rating', axis=1)
print(f"Time for predictions is {t} seconds")

eval_map = map(test, all_predictions, col_prediction='prediction', k=TOP_K)
eval_ndcg = ndcg_at_k(test, all_predictions, col_prediction='prediction',
k=TOP_K)
eval_precision = precision_at_k(test, all_predictions,
col_prediction='prediction', k=TOP_K)
eval_recall = recall_at_k(test, all_predictions,
col_prediction='prediction', k=TOP_K)

print(f'''MAP:\t\t{eval_map}
nDCG@k:\t\t{eval_ndcg}
Precision@k:\t{eval_precision}
Recall@k:\t{eval_recall}''')

model = NCF(
    n_users=data.n_users,
    n_items=data.n_items,
    model_type="GMF",
    n_factors=4,
    layer_sizes=[16,8,4],
    n_epochs=EPOCHS,
    batch_size=BATCH_SIZE,
    learning_rate=1e-3,
    verbose=10,
    seed=RAND_SEED
)

with Timer() as t:
    model.fit(data)
print(f"Time for training is {t} seconds")

model.save(dir_name=".pretrain/GMF")

model = NCF(
    n_users=data.n_users,
    n_items=data.n_items,
    model_type="MLP",
    n_factors=4,
    layer_sizes=[16,8,4],
    n_epochs=EPOCHS,
```

```
    batch_size=BATCH_SIZE,
    learning_rate=1e-3,
    verbose=10,
    seed=RAND_SEED
)

with Timer() as t:
    model.fit(data)
print(f"Time for training is {t} seconds")

model.save(dir_name=".pretrain/MLP")

model = NCF(
    n_users=data.n_users,
    n_items=data.n_items,
    model_type="NeuMF",
    n_factors=4,
    layer_sizes=[16,8,4],
    n_epochs=EPOCHS,
    batch_size=BATCH_SIZE,
    learning_rate=1e-3,
    verbose=10,
    seed=RAND_SEED)
model.load(gmf_dir=".pretrain/GMF", mlp_dir=".pretrain/MLP", alpha=0.5)

with Timer() as t:
    model.fit(data)
print(f"Time for training is {t} seconds")

with Timer() as t:

    users, items, preds = [], [], []
    item = list(train.itemID.unique())
    for user in train.userID.unique():
        user = [user] * len(item)
        users.extend(user)
        items.extend(item)
        preds.extend(list(model.predict(user, item, is_list=True)))

    all_predictions = pd.DataFrame(data={USER_COL:users, ITEM_COL:items,
PREDICTION_COL:preds})

    merged = pd.merge(train, all_predictions, on=[USER_COL, ITEM_COL],
how="outer")
    all_predictions = merged[merged.rating.isnull()].drop('rating', axis=1)

print(f"Time for predictions is {t} seconds")

eval_map2 = map(test, all_predictions, col_prediction='prediction',
k=TOP_K)
eval_ndcg2 = ndcg_at_k(test, all_predictions, col_prediction='prediction',
k=TOP_K)
eval_precision2 = precision_at_k(test, all_predictions,
col_prediction='prediction', k=TOP_K)
eval_recall2 = recall_at_k(test, all_predictions,
col_prediction='prediction', k=TOP_K)

print(f'''MAP:\t\t{eval_map2}
nDCG@k:\t\t{eval_ndcg2}
Precision@k:\t{eval_precision2}
Recall@k:\t{eval_recall2}''')
```

Листинг 6 – Обучение SAR

```
import sys
import logging
import numpy as np
import pandas as pd
from recommenders.utils.timer import Timer
from recommenders.datasets.python_splitters import python_stratified_split
from recommenders.models.sar import SAR
from recommenders.evaluation.python_evaluation import map, ndcg_at_k, pre-
cision_at_k, recall_at_k
%load_ext autoreload
%autoreload 2
TOP_K = 10
RAND_SEED = 42
SAMPLE_FRAC = 0.035
USER_COL = 'userID'
ITEM_COL = 'itemID'
RATING_COL = 'rating'
PREDICTION_COL = 'prediction'
data = pd.read_csv('rating_filtered_df.csv')
unique_users = data[USER_COL].unique()
sampled_users = pd.Series(unique_users).sample(frac=SAMPLE_FRAC, ran-
dom_state=RAND_SEED)
data = data[data[USER_COL].isin(sampled_users)]
data[RATING_COL] = data[RATING_COL].astype(np.float32)
data.head()
train, test = python_stratified_split(data, ratio=0.75, col_user=USER_COL,
col_item=ITEM_COL, seed=RAND_SEED)
print ("N train", train.size)
print ("N test", test.size)
logging.basicConfig(level=logging.DEBUG,
                    format='%(asctime)s %(levelname)-8s %(message)s')

model = SAR(
    col_user=USER_COL,
    col_item=ITEM_COL,
    col_rating=RATING_COL,
    similarity_type="jaccard",
    normalize=True
)
with Timer() as t:
    model.fit(train)
print(f"Time for training is {t} seconds")

with Timer() as t:
    top_k = model.recommend_k_items(test, top_k=TOP_K, remove_seen=True)
print(f"Time for predictions is {t} seconds")

eval_map = map(test, top_k, col_user=USER_COL, col_item=ITEM_COL,
col_rating=RATING_COL, k=TOP_K)
eval_ndcg = ndcg_at_k(test, top_k, col_user=USER_COL, col_item=ITEM_COL,
col_rating=RATING_COL, k=TOP_K)
eval_precision = precision_at_k(test, top_k, col_user=USER_COL,
col_item=ITEM_COL, col_rating=RATING_COL, k=TOP_K)
eval_recall = recall_at_k(test, top_k, col_user=USER_COL,
col_item=ITEM_COL, col_rating=RATING_COL, k=TOP_K)
print(f'''MAP:\t\t{eval_map}
nDCG@k:\t\t{eval_ndcg}
Precision@k:\t{eval_precision}
Recall@k:\t{eval_recall}''')
```

Приложение Б. Код рекомендательного сервиса

Листинг 7 – Точка входа в сервис – main.py

```
from fastapi import FastAPI, HTTPException
from typing import List

from models.bivae import BivaeModel

app = FastAPI()
bivae = BivaeModel()


@app.get("/recommendations/")
async def get_recommendations(user_id: int, track_count: int = 60, re-
move_known: bool = False) -> List[int]:
    try:
        return bivae.recommend(user_id, count=track_count, re-
move_seen=remove_known)
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

Листинг 8 – Рекомендательная модель – bivae.py

```
import torch
import cornac
import pandas as pd
import shutil


class BivaeModel:
    _rand_seed = 42
    _model_path = "models/bivae.pkl"
    _trainset_path = "models/bivae.pkl.trainset"
    _data_path = "data/rating_filtered_df.csv"

    _train_set = None
    _model = None

    def __init__(self):
        try:
            print("initializing a model from a file...")
            self._model = cornac.models.BiVAECF.load(self._model_path)
            self._train_set = cornac.data.Dataset.load(self._trainset_path)
        except:
            print(f"initializing a new model [GPU:
{torch.cuda.is_available()}]...")

            data = pd.read_csv(self._data_path)
            if False: # make True for fast run
                unique_users = data['userID'].unique()
                sampled_users = pd.Series(unique_users).sample(frac=0.001,
random_state=self._rand_seed)
                data = data[data['userID'].isin(sampled_users)]
                print(sampled_users)
            self._train_set = cor-
nac.data.Dataset.from_uir(data.itertuples(index=False),
seed=self._rand_seed)
            self._model = None
            tmp_model = cornac.models.BiVAECF(
                k=50,
                encoder_structure=[100],
```

```
                act_fn="tanh",
                likelihood="pois",
                n_epochs=500,
                batch_size=128,
                learning_rate=0.001,
                seed=self._rand_seed,
                use_gpu=torch.cuda.is_available(),
                verbose=True
            )
            print(f"saving model...")
            tmp_model.fit(self._train_set)
            tmp_model_path = tmp_model.save(save_dir="models/tmp",
save_trainset=True)
            tmp_trainset_path = f"{tmp_model_path}.trainset"
            shutil.move(tmp_model_path, self._model_path)
            shutil.move(tmp_trainset_path, self._trainset_path)
            print(f"model save to {self._model_path}")
            self._model = tmp_model
        print(f"model initialization completed")
    def recommend(self, user_id: int, count: int, remove_seen: bool) ->
list:
        if self._model is None:
            raise Exception("Model has not yet been initialized")
        return self._model.recommend(user_id, k=count, re-
move_seen=remove_seen, train_set=self._train_set)
```

Листинг 9 – Файл сборки Docker образа – Dockerfile

```
ARG IMAGE_NAME
FROM ${IMAGE_NAME}:12.4.1-runtime-ubuntu22.04 as base
ENV NV_CUDA_LIB_VERSION "12.4.1-1"
FROM base as base-amd64
ENV NV_CUDA_CUDART_DEV_VERSION 12.4.127-1
ENV NV_NVML_DEV_VERSION 12.4.127-1
ENV NV_LIBCUSPARSE_DEV_VERSION 12.3.1.170-1
ENV NV_LIBNPP_DEV_VERSION 12.2.5.30-1
ENV NV_LIBNPP_DEV_PACKAGE libnpp-dev-12-4=${NV_LIBNPP_DEV_VERSION}
ENV NV_LIBCUBLAS_DEV_VERSION 12.4.5.8-1
ENV NV_LIBCUBLAS_DEV_PACKAGE_NAME libcublas-dev-12-4
ENV NV_LIBCUBLAS_DEV_PACKAGE
${NV_LIBCUBLAS_DEV_PACKAGE_NAME}=${NV_LIBCUBLAS_DEV_VERSION}
ENV NV_CUDA_NSIGHT_COMPUTE_VERSION 12.4.1-1
ENV NV_CUDA_NSIGHT_COMPUTE_DEV_PACKAGE cuda-nsight-compute-12-
4=${NV_CUDA_NSIGHT_COMPUTE_VERSION}
ENV NV_NVPROF_VERSION 12.4.127-1
ENV NV_NVPROF_DEV_PACKAGE cuda-nvprof-12-4=${NV_NVPROF_VERSION}
ENV NV_LIBNCCL_DEV_PACKAGE_NAME libnccl-dev
ENV NV_LIBNCCL_DEV_PACKAGE_VERSION 2.21.5-1
ENV NCCL_VERSION 2.21.5-1
ENV NV_LIBNCCL_DEV_PACKAGE
${NV_LIBNCCL_DEV_PACKAGE_NAME}=${NV_LIBNCCL_DEV_PACKAGE_VERSION}+cuda12.4
FROM base as base-arm64
ENV NV_CUDA_CUDART_DEV_VERSION 12.4.127-1
ENV NV_NVML_DEV_VERSION 12.4.127-1
ENV NV_LIBCUSPARSE_DEV_VERSION 12.3.1.170-1
ENV NV_LIBNPP_DEV_VERSION 12.2.5.30-1
ENV NV_LIBNPP_DEV_PACKAGE libnpp-dev-12-4=${NV_LIBNPP_DEV_VERSION}

ENV NV_LIBCUBLAS_DEV_PACKAGE_NAME libcublas-dev-12-4
ENV NV_LIBCUBLAS_DEV_VERSION 12.4.5.8-1
```

```
ENV NV_LIBCUBLAS_DEV_PACKAGE
${NV_LIBCUBLAS_DEV_PACKAGE_NAME}=${NV_LIBCUBLAS_DEV_VERSION}
ENV NV_CUDA_NSIGHT_COMPUTE_VERSION 12.4.1-1
ENV NV_CUDA_NSIGHT_COMPUTE_DEV_PACKAGE cuda-nsight-compute-12-
4=${NV_CUDA_NSIGHT_COMPUTE_VERSION}
ENV NV_LIBNCCL_DEV_PACKAGE_NAME libnccl-dev
ENV NV_LIBNCCL_DEV_PACKAGE_VERSION 2.21.5-1
ENV NCCL_VERSION 2.21.5-1
ENV NV_LIBNCCL_DEV_PACKAGE
${NV_LIBNCCL_DEV_PACKAGE_NAME}=${NV_LIBNCCL_DEV_PACKAGE_VERSION}+cuda12.4

FROM base-${TARGETARCH}

ARG TARGETARCH

RUN apt-get update && apt-get install -y --no-install-recommends \
    cuda-cudart-dev-12-4=${NV_CUDA_CUDART_DEV_VERSION} \
    cuda-command-line-tools-12-4=${NV_CUDA_LIB_VERSION} \
    cuda-minimal-build-12-4=${NV_CUDA_LIB_VERSION} \
    cuda-libraries-dev-12-4=${NV_CUDA_LIB_VERSION} \
    cuda-nvml-dev-12-4=${NV_NVML_DEV_VERSION} \
    ${NV_NVPROF_DEV_PACKAGE} \
    ${NV_LIBNPP_DEV_PACKAGE} \
    libcusparse-dev-12-4=${NV_LIBCUSPARSE_DEV_VERSION} \
    ${NV_LIBCUBLAS_DEV_PACKAGE} \
    ${NV_LIBNCCL_DEV_PACKAGE} \
    ${NV_CUDA_NSIGHT_COMPUTE_DEV_PACKAGE} \
    && rm -rf /var/lib/apt/lists/*

RUN apt-mark hold ${NV_LIBCUBLAS_DEV_PACKAGE_NAME}
${NV_LIBNCCL_DEV_PACKAGE_NAME}
ENV LIBRARY_PATH /usr/local/cuda/lib64/stubs

RUN apt-get update && apt-get install -y --no-install-recommends \
    python3-pip python3-dev \
    && pip3 install --no-cache-dir fastapi-cli

COPY requirements.txt .
RUN pip3 install --no-cache-dir -r requirements.txt --index-url
https://mirrors.sustech.edu.cn/pypi/simple

COPY ./app /app
COPY ./data /data
COPY ./models /models

CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "80"]
```

Листинг 10 – Список пакетов – requirements.txt

```
fastapi == 0.111.0
torch == 2.3.0
cornac == 1.18.0
pandas == 2.2.2
numpy == 1.26.4
scipy == 1.13.0
Cython == 3.0.10
```

Приложение В. Работа рекомендательного сервиса

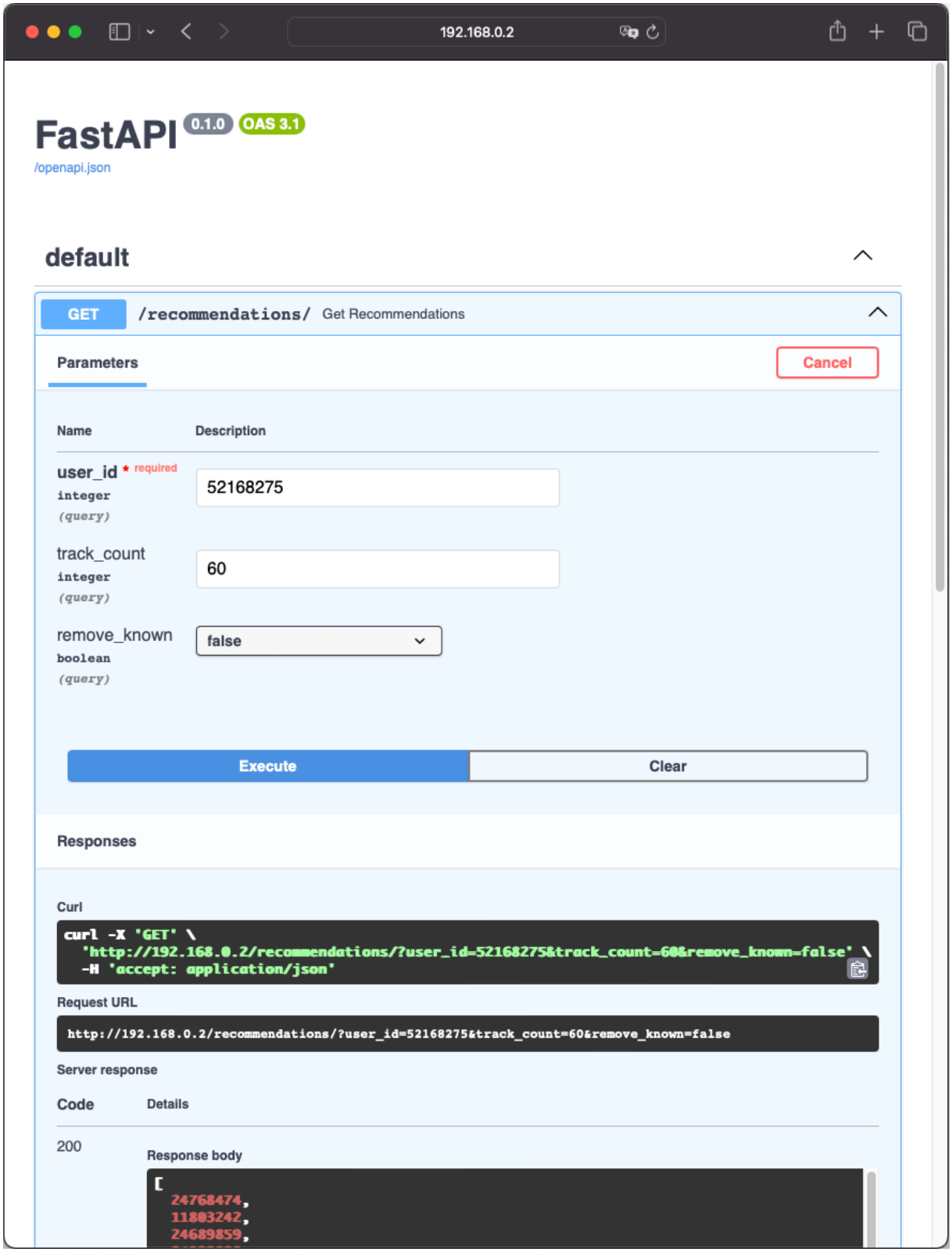

Рисунок 1 – Swagger UI