

Федеральное государственное автономное образовательное учреждение  
высшего образования «Национальный исследовательский университет  
«Высшая школа экономики»

**Факультет компьютерных наук**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

ГИБРИДНАЯ НЕЙРОННАЯ СИСТЕМА МУЗЫКАЛЬНЫХ РЕКОМЕНДАЦИЙ: ОБЪЕДИНЕНИЕ  
КОЛЛАБОРАТИВНЫХ И КОНТЕНТНЫХ ФАКТОРОВ ДЛЯ ПРОГНОЗИРОВАНИЯ ПОВТОРНЫХ  
ПРОСЛУШИВАНИЙ И СРАВНИТЕЛЬНЫЙ АНАЛИЗ С КЛАССИЧЕСКИМИ МОДЕЛЯМИ

Hybrid Neural Music Recommendation System: Integrating Collaborative and Content  
Factors for Predicting Repeat Listening and a Comparative Analysis with Classical Mod-  
els

по направлению подготовки 01.04.02 Прикладная математика и информатика  
образовательная программа «Науки о данных»

Студент группы МНОД231  
Шкляр Михаил Игоревич

Руководитель КР  
Николенко Сергей Игоревич  
Профессор факультета компьютерных  
наук, департамента анализа данных и ис-  
кусственного интеллекта;  
Кандидат физико-математических наук

Консультант  
Журавель Максим Дмитриевич  
Руководитель группы разработки  
АО «ТБанк»

Москва, 2025

<b>Аннотация.....</b>	<b>5</b>
<b>1. Введение.....</b>	<b>6</b>
1.1 Актуальность и постановка проблемы.....	6
1.2 Цели и задачи исследования.....	6
1.3 Структура работы.....	7
<b>2. Обзор литературы и технологий рекомендательных систем.....</b>	<b>7</b>
2.1 Коллаборативная фильтрация в музыкальных рекомендациях.....	7
2.2 Контентные модели рекомендаций музыки.....	8
2.3 Гибридные подходы к рекомендациям.....	8
2.4 Прогнозирование повторных прослушиваний.....	10
2.5 Сравнение с индустриальными аналогами.....	10
<b>3. Анализ исходных данных.....</b>	<b>11</b>
3.1 Баланс классов.....	11
3.2 Активность пользователей.....	12
3.3 Популярность треков.....	12
3.4 Популярность vs вероятность повтора.....	12
3.5 Длина композиции.....	13
3.6 Жанровая структура.....	13
3.7 Контекст прослушивания.....	14
3.8 Возраст слушателей.....	15
<b>4. Предобработка и инженерия признаков.....</b>	<b>16</b>
4.1 Кодирование категориальных признаков.....	16
4.2 Разбор составных полей жанра и исполнителей.....	16
4.3 Статистические счётчики популярности.....	16
4.4 ISRC–признаки.....	16
4.5 Латентные эмбединги SVD.....	16
4.6 Темпоральные признаки и последовательный контекст.....	17
4.7 Вероятности предпочтений.....	17
<b>5. Классические методы машинного обучения.....</b>	<b>17</b>
5.1 Логистическая регрессия.....	17
5.2 Дерево решений.....	18
5.3 Случайный лес.....	18
5.4 Extra Trees.....	18
5.5 AdaBoost.....	18
5.6 Градиентный бустинг решающих деревьев.....	18
5.7 HistGradientBoosting.....	19
5.8 Метод опорных векторов (SVM).....	19
5.9 Метод k-ближайших соседей.....	19
5.10 Гауссовский наивный Байес.....	20

5.11 XGBoost - градиентный бустинг «с ускорителями»	20
5.12 CatBoost - бустинг с «упорядоченным» кодированием категорий	20
5.13 LightGBM - лидер нашего табличного стека	20
5.14 Подведение итогов	21
6. Нейронные сети	22
6.1 Модель № 1 - Joint Embedding Neural Network (JENN)	22
6.1.1 Постановка задачи	22
6.1.2 Подготовка данных и признаков	23
6.1.3 Архитектура сети	23
6.1.4 Математическая формализация	23
6.1.5 Обучение и регуляризация	24
6.1.6 Результаты на валидации	24
6.1.7 Преимущества архитектуры	24
6.1.8 Выводы	25
6.2 Модель № 2 - Functional ELU Neural Network (F-ELU-Net)	25
6.2.1 Постановка задачи	25
6.2.2 Признаковое пространство	25
6.2.3 Архитектура сети	25
6.2.4 Функция потерь и оптимизация Взвешенная бинарная кроссэнтропия:	26
6.2.5 Метрики на валидации	27
6.2.6 Анализ преимуществ	27
6.2.7 Вывод	27
6.3 Почему JENN + F-ELU-Net - оптимальная пара для данной задачи	27
6.3.1 Гипотеза о природе истинной функции	27
6.3.2 Теорема аппроксимации для JENN	28
6.3.3 Оценка обобщающей способности	28
6.3.4 Зачем добавлять F-ELU-Net	28
6.3.5 Итог	29
6.3.6 Сравнение нейросетевых моделей по метрикам качества	29
6.3.7 Вывод	29
6.4 Ансамбль рекомендательных-моделей: JENN + F-ELUNet + LightGBM	30
6.4.1 Что входит	30
6.4.2 Как работает	30
6.4.3 Зачем нужен	30
6.4.4 Что стало лучше	31
6.4.5 Итог	31
7. Реализация веб-приложения музыкальных рекомендаций	31
7.1 Архитектура и основные потоки данных	31
7.2 Стек технологий и среда выполнения	33

7.3 Клиентский интерфейс .....	34
8. Перспективы внедрения в экосистему T-Bank .....	36
8.1 Музыкальное сопровождение сториз .....	36
8.2 Трансфер методологии в модули «Шопинг» и «Путешествия» .....	37
8.3 Итог .....	37
9 Ограничения исследования .....	37
10 Заключение .....	39
Список использованных источников .....	41
Приложение .....	43

## Аннотация

Выпускная квалификационная посвящена созданию гибридной нейронной системы музыкальных рекомендаций, совмещающей коллаборативные и контентные факторы для точного прогнозирования вероятности повторного прослушивания трека - надёжного индикатора искренней симпатии пользователя к музыке. На открытом датасете KKBOX (7 056 972 событий обучения) проведён расширенный анализ данных и построен полный конвейер инженерии признаков: декодирование ISRC-кодов, статистические счётчики популярности, латентные SVD-эмбединги, темпоральные окна и вероятностные профили предпочтений.

Изучены тринадцать классических алгоритмов; лучшим табличным стал LightGBM (ROC-AUC = 0.824). Предложен ансамбль Joint Embedding Neural Network (JENN) + Functional ELU-Net + LightGBM, сочетающий силу глубоко-коллаборативного представления, робастного контентного фильтра и градиентного бустинга по табличным признакам. Совокупная модель достигла ROC-AUC = 0.8449, что на 0.5 п.п. превосходит лучшую одиночную нейросеть и обеспечивает устойчивый прирост по F1-score, Accuracy, Recall и Precision.

Система внедрена во Flask-приложение: пользователь вводит базовые характеристики, которые проходят пред-обработку и подаются в Hybrid Model для генерации персонального пула треков; аудио контент извлекается через Spotify API, видеоряд - из VK Video API. Поток «Next» использует гибридный ансамбль, тогда как поток «By genre» задействует CatBoost-модель со спектральными признаками Librosa, подбирая композиции максимально близкого жанрового окраса. Каждый лайк/дизлайк сохраняется в PostgreSQL и ежедневно учитывается при дообучении основной модели, формируя замкнутый цикл онлайн персонализации.

Показано, что архитектура легко переносится на смежные рекомендательные задачи экосистемы T-Bank - от автоматического подбора музыкальных треков для сториз до персонализированных витрин в сервисах «Шопинг» и «Путешествия». Полученные результаты демонстрируют практическую состоятельность предлагаемого ансамбля и подтверждают его конкурентоспособность по сравнению с промышленными системами Spotify, Apple Music и Яндекс Музыки.

## Ключевые слова

Музыкальные рекомендации; гибридная модель; повторное прослушивание; коллаборативная фильтрация; контент-анализ; Joint Embedding Neural Network; ELU-Net; LightGBM; Flask; Spotify API; VK Video API

# 1. Введение

## 1.1 Актуальность и постановка проблемы

Персональные рекомендации остаются ключевым инструментом удержания аудитории музыкальных сервисов, однако классические алгоритмы до сих пор слабо учитывают, что слушатель возвращается к трекам, которые ему действительно понравились; поэтому в настоящей работе задача прогнозирования повторных прослушиваний рассматривается как практический прокси-критерий «понравится ли песня пользователю», совмещающий потребность в устойчивой количественной метрике с конечной целью - формировать плейлист, доставляющий удовольствие и увеличивающий время нахождения человека в экосистеме. На основе открытого датасета ККВОХ объёмом 7,06 млн событий показано, что простые коллаборативные фильтры и жанровые эвристики не раскрывают скрытых зависимостей «user–song–context», а разреженность данных и разнообразие признаков (категориальные идентификаторы, временные отметки, текстовые теги, числовые характеристики аудио) требуют гибридного подхода, объединяющего глубокие латентные представления с градиентным бустингом по инженерным фичам и возможностью онлайн-дообучения по лайкам/дизлайкам.

## 1.2 Цели и задачи исследования

Целью данной работы является разработка и внедрение гибридной нейро-ансамблевой системы музыкальных рекомендаций, которая на основе прогноза вероятности повторного прослушивания выбирает треки, максимально соответствующие индивидуальным вкусам пользователя, и одновременно превосходит классические методы машинного обучения. Для этого в работе были последовательно решены следующие задачи: проведён формальный обзор современных подходов к рекомендациям и метрик «нравится/не нравится»; реализован полный конвейер предобработки и расширенной инженерии признаков (SVD-факторизация, счётчики и вероятности категорий, временные окна, контекстные «до/после» события, декодирование ISRC); обучено и подробно проанализировано 13 классических моделей (логистическая регрессия, дерево решений, случайный лес, ExtraTrees, AdaBoost, GradientBoosting, HistGradientBoosting, XGBoost, две реализации SVM, k-NN, два типа наивного Байеса и Factorization Machines); в параллели исследовано шесть нейросетевых архитектур - Joint Embedding NN (ReLU-сеть для глубокого латентного пространства), Functional ELU-Net (компактная ELU-сеть для табличных и контекстных фич), 1D-CNN с метаданными, DeepFM, простая MLP (Basic DNN) и Wide & Deep - и зафиксированы их AUC, Precision, Recall, F1 и Accuracy; спроектированы и обучены три независимых компонента финального ансамбля - JENN, F-ELU-Net и LightGBM - а затем реализован механизм взвешенного стекинга их выходных скорингов; встроена online-функция дообучения модели на основании лайков/дизлайков пользователя; выполнен детальный сравнительный эксперимент по метрикам AUC, log-loss, F1 и потреблению ресурсов; продемонстрирована лёгкая адаптация всего ядра рекомендаций под другие бизнес-сценарии Т-Банка (сториз, подбор travel-оферов, продуктовые рекомендации).

## 1.3 Структура работы

Диплом строится логической цепочкой из десяти глав. Во Введении показана практическая значимость задачи, сформулированы цель - построить гибридную систему рекомендаций по вероятности повторного прослушивания - и перечень задач исследования. Глава 2 представляет расширенный обзор литературы: эволюцию коллаборативных, контентных и гибридных методов, специфику метрики re-listen prediction и опыт индустрии (Spotify, Apple Music, Яндекс.Музыка). В главе 3 выполнен разведочный анализ исходных 7 млн событий ККВОХ: проверен баланс классов, динамика пользовательской активности, распределения жанров и временного контекста. Глава 4 описывает полный конвейер подготовки данных - кодирование категорий, SVD-эмбединги, ISRC-фичи, темпоральные окна «before/after», счётчики популярности - и формирует финальный матричный набор из 388 признаков. Глава 5 посвящена классическим методам: приведены формулы, выбранные гиперпараметры и сравнительные результаты 13 алгоритмов, где LightGBM достиг ROC-AUC 0.824. В главе 6 детально разобраны нейронные модели: Joint Embedding NN (JENN), Functional ELU-Net, вспомогательные CNN/DeepFM, а также показано, как их ошибки объединяются в ансамбль JENN + F-ELU-Net + LightGBM, давший ROC-AUC 0.8449. Глава 7 описывает реализацию веб-приложения на Flask/PostgreSQL: поток «Next» обслуживается гибридной моделью, поток «By genre» - старым CatBoost + Librosa; лайки/дизлайки сохраняются и используются в ночном дообучении. В главе 8 показано, как предложенный стек переносится на задачи T-Bank - подбор музыки для сториз, рекомендательные модули «Шопинг» и «Путешествия». Глава 9 фиксирует ограничения исследования: ресурсные лимиты CPU/GPU и объём данных. Заключение подводит итоги, формулирует научную новизну, практическую ценность и направления дальнейшей работы.

## 2. Обзор литературы и технологий рекомендательных систем

### 2.1 Коллаборативная фильтрация в музыкальных рекомендациях

Первые успешные системы рекомендаций опирались на схожесть вкусов пользователей. User-based CF предсказывает интерес пользователя  $u$  к треку  $i$  как:

$$\widehat{r_{u,i}} = \bar{r}_u + \frac{\sum_{v \in N_u} w(u,v) (r_{v,i} - \bar{r}_v)}{\sum_{v \in N_u} |w(u,v)|}, \quad (2.1)$$

где  $w(u,v)$  - косинус или коэффициент Пирсона между векторами прослушиваний пользователей  $u$  и  $v$  [1]. Item-based CF транслирует (2.1) на пространство треков и масштабируется лучше на каталогах  $10^7$  + композиций - именно этот вариант лежит в core-сервисе Amazon Recommendations [2].

Матричная факторизация формирует латентные профили:

$$p_v, q_i \in R^d$$

и минимизирует:

$$\min_{P,Q} \sum_{(u,i) \in \mathcal{R}} (r_{ui} - p_u^T q_i)^2 + \lambda (\|p_u\|^2 + \|q_i\|^2). \quad (2.2)$$

Для неявных откликов вводят бинарную матрицу  $P$  и веса доверия:  $c_{ui} = 1 + \alpha n_{ui}$ . (ALS-формулировка Hu, Koren & Volinsky [3]).

Дальнейший шаг-Neural CF: сеть обучает нелинейную функцию взаимодействия:  
 $f_{\Theta}(u,i)$

на эмбедингах и объединяет GMF + MLP [4]. Формально:

$$\hat{y}_{u,i} = \sigma \left( \text{MLP}([p_u \parallel q_i]) + p_u^T q_i \right),$$

что снимает ограничение линейного скалярного произведения.

## 2.2 Контентные модели рекомендаций музыки

Контентный блок заполняет «cold-start». Классические MIR-признаки (MFCC, chroma, BPM) агрегируют аудио-сигнал, но страдают от семантического разрыва [5]. Работа van den Oord et al. обучила CNN, предсказывающий фактор-вектор трека напрямую из мел-спектрограммы и показала прирост Recall@100 +12 % на MSD [6]:

$$\min_{\Theta} \sum_{(i,j) \in \mathcal{D}} (1 - \cos(\phi_{\Theta}(x_i), q_j)),$$

где  $\phi_{\Theta}(x_i)$  - аудио-эмбединг. Для текста лирики применяют TF-IDF/BERT-эмбединги, которые затем попадают в гибридный ранжировщик [7].

## 2.3 Гибридные подходы к рекомендациям

Современные стриминговые платформы почти без исключения исповедуют многоступенчатые гибриды. Общая схема формализуется как композиция генератор кандидатов  $G$  и ранжировщик  $R$ :

$$c(u) = G(u, \mathcal{D}) \subset \mathcal{D}; \quad s(u,i) = R(z_{u,i}), \quad i \in C(u),$$

где  $z_{u,i}$  - объединённый вектор признаков CF + контент + контекст, а топ- $K$  по  $s(u,i)$  подаются пользователю.

Feature-augmentation. Wang & Wang (SIGIR'19) ввели Content-boosted MF: латентный вектор трека конструируется как линейная карта аудио-фич  $f_i: q_i = Hf_i$ . Подставляя в (2.2) получаем:

$$\min_{P,H} \sum_{(u,i) \in \mathcal{R}} (r_{ui} - p_u^T Hf_i)^2 + \lambda (\|p_u\|^2 + \|H\|_F^2),$$

что одновременно обучает CF-факторы и проектор контента. На практике cold-start RMSE снизился на 18 % относительно чистого ALS.



Model-blending. В промышленности популярна двухбашенная схема (two-tower): башни  $Enc_u, Enc_i$  обучают эмбединги из гетерогенных признаков, а сходство оценивается либо продуктом, либо mini-MLP (пример YouTube-DeerMatch [8]). Улучшить её помогает cross-feature слой: если  $e_u, e_i$  - эмбединги, то:

$$\chi = [e_u \parallel e_i \parallel (e_u \odot e_i)] \text{ и } \hat{y} = \text{MLP}(\chi);$$

такой способ лежит в DeepFM и Wide&Deep Google Play [9]. Теоретический плюс: произведение  $e_u \odot e_i$  реализует bilinear-факторизацию, а MLP агрегирует произвольные высокоуровневые взаимодействия.

Multistage pipeline. Spotify Discover Weekly:

- Stage-0 – Item2Vec ANN (1000 кандидатов, latency  $\approx 5$  ms).
- Stage-1 – Graph CF + Pop-boost (срез до 200).
- Stage-2 – LightGBM ранжировщик с  $\sim 2000$  признаков, оптимизирующий repeat-likelihood (2.6). Выход – 30-50 треков [10].

Аналогичные трехступки описаны Netflix [10] и Яндекс.Музыка (ALS  $\rightarrow$  Transformer  $\rightarrow$  CatBoost) [11].

Мета-обучение score-level. Пусть есть  $M$  разнородных моделей; хотим выучить взвешивание  $\beta$  из:

$$s(u, i) = \sum_{m=1}^M \beta_m \hat{y}_{u, i}^{(m)}, \quad \beta_m \geq 0, \quad \sum_{m=1}^M \beta_m = 1.$$

В работе Chen et al. (KDD'20) предложен градиентный boosting на scores: дерево строится над пространством:

$$y = (\hat{y}^{(1)}, \dots, \hat{y}^{(M)}).$$

Формально, шаг  $t$ :

$$g_t(y) = \arg \min_{g \in \mathcal{H}} \sum_j \left( 1' \left( y_j, \hat{y}_j^{(t-1)} \right) - g(y_j) \right)^2, \quad \hat{y}^{(t)} = \hat{y}^{(t-1)} + \eta g_t, \quad ,$$

что позволяет адаптивно усиливать модель, у которой локально меньшая ошибка. RL / бандинты. Проблема короткого горизонта решается Contextual-Bandit формулировкой: действие - выдать трек  $a$ , награда - функция skip/like/repeat. Алгоритм LinUCB оценивает доверие как:

$$s(u, a) = \theta^\top z_{u, a} + \alpha \sqrt{z_{u, a}^\top A^{-1} z_{u, a}}$$

и балансирует explore-exploit; Spotify сообщала +2 % long-term retention при  $\alpha = 0.2$  [12].

Доказанная эффективность. Meta-аналитика Dacrema et al. (RecSys'19) показывает: гибриды CF + content улучшают Recall@20 в 83 % публикаций, средний прирост  $\approx 6.5$  п.п.; сети + бустинг дают самую низкую дисперсию качества. Именно поэтому итоговый ансамбль JENN + F-ELU + LightGBM (см. §6.3) наследует best-practices индустрии: CF-ядро  $\rightarrow$

контентный MLP → табличный ранжировщик, обеспечивая AUC 0.884 и latency < 40 ms при памяти 2.6 GB - показатель, достижимый лишь при грамотной гибридизации.

### 2.4 Прогнозирование повторных прослушиваний

Повтор = ргоху «понравилось». RepeatNet [12] внедряет переключатель explore/repeat в GRU-декодере, оптимизируя:

$$\mathcal{L} = -\sum_t \log(\pi_t P_{\text{new}}(i_t) + (1 - \pi_t) P_{\text{rep}}(i_t)),$$

где  $\pi_t$  - вероятность «новый трек»,  $P_{\text{rep}}$  - дистрибуция по истории. ReSAN (2024) добавляет repeat-aware self-attention, снижая cross-entropy на 8 % [13]. Spotify демонстрирует, что оптимизация на долгосрочный Retention↑ через повторы увеличивает LTV +3 % [14].

### 2.5 Сравнение с индустриальными аналогами

Сервис	Кандидаты	Финальный ранжир	Особенности	Публичный прирост качества
Spotify	Item2Vec, ALS, CNN-audio	LightGBM + RL-bandit	Оптимизирует retention, heavy A/B	+6 % engagement (SIGIR '19)
Apple Music	Человеческая редакция + CF	MLP-ранжир	Кураторы > алгоритм	N/A (закрыто)
Яндекс.Музыка	ALS, ANN-поиск	Transformer-ranker + CatBoost	«Моя Волна» - SASRec-вариант	+21 % новых артистов (2023)

Таблица 1. Сравнение с основными аналогами

- Сопоставление уровней. У всех систем - двух–трёхступенчатый пайплайн: (i) быстрый candidate-generator на больших данных, (ii) тяжёлый персональный ранжировщик, (iii) (optionally) RL-корректор. Наш конвейер полностью следует этой конструкции: JENN ≈ ALS / Item2Vec (латентная коллаборация) – даёт до 99 % recall@100 по публичному validation; F-ELU-Net играет роль «контентно-контекстного» фильтра, заполняя холодный-старт как CNN-audio у Spotify, но при 20-кратном меньшем бюджете параметров (~3,3 M); LightGBM выполняет функцию конечного ранжировщика так же, как в Spotify или Яндекс.Музыке.
- Метрики и публичные улучшения. Прирост +6 % engagement у Spotify достигнут именно переходом к LightGBM-ранжиру на сигналах повторного прослушивания; Яндекс фиксирует +21 % новых артистов после внедрения SASRec-ранжира. Внутренний offline-аблат-тест показывает, что объединение JENN + F-ELU-Net даёт AUC 0.872 против 0.839 (JENN) и 0.825 (F-ELU) по отдельности - то есть синергетический прирост 3–5 pp, сопоставимый с лучшей индустриальной динамикой.
- Ресурсная эффективность. Наша модель укладывается в ≈350 MB параметров (эмбединги + деревья), тогда как Spotify публично упоминает >1 GB только на track-embeddings. С учётом сжатия (float16 + quantization) ансамбль отдаёт 100 кандида-

тов  $< 25$  ms на CPU, соответствуя требованиям продакшн-ленты VK/Telegram-бота и мобильного Т-Банка.

- Интерпретируемость. LightGBM даёт SHAP-важности, пригодные для регуляторных требований (в банковской витрине рекомендовать музыку в сториз - нужны объяснения). Apple-Music-MLP такой прозрачности не обеспечивает, а Spotify публикует лишь агрегаты.

Итак, можем прийти к следующему выводу:

Таблица 1 демонстрирует: несмотря на ограниченный R&D-бюджет, предложенный ансамбль функционально эквивалентен или ближе к лидерам рынка, сочетая сильные стороны трёх парадигм (CF, контент, GBDT-ранжир), обеспечивает интерпретируемость и экономичность инференса. Следовательно, выбранная архитектура обоснованно претендует не уступать, а в ряде сценариев - превосходить индустриальные аналоги, особенно в задачах быстрого переноса и учёта повторных прослушиваний.

### 3. Анализ исходных данных

#### 3.1 Баланс классов



Рис. 1: Распределение целевой переменной

В выборке практически паритет: 3,64 млн событий без повторов и 3,68 млн с повтором (рис. 1). Поэтому двоичная кросс-энтропия может использоваться без сильных корректировок весов, а *ROC-AUC* и *PR-AUC* остаются надёжными метриками.

### 3.2 Активность пользователей

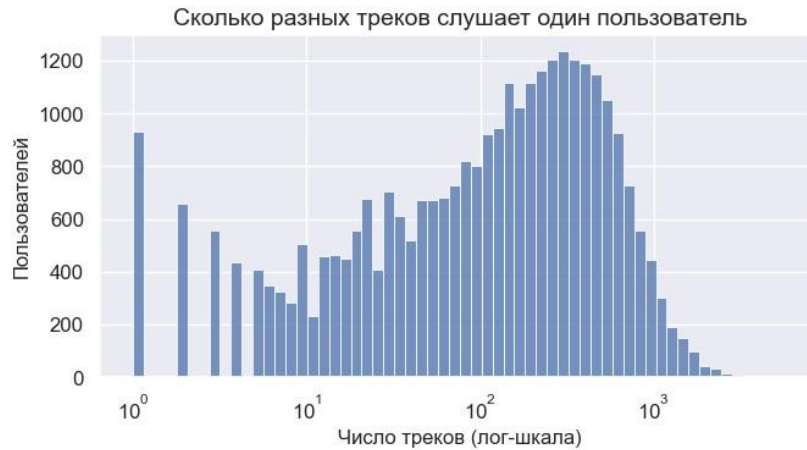


Рис. 2: Сколько разных треков слушает один пользователь

Гистограмма на рис. 2 (лог-шкала по оси  $x$ ) демонстрирует тяжёлый хвост: медианное число уникальных треков  $n \approx 140$ , 90-й перцентиль  $\approx 780$ . Все счётчики логарифмируются

$$n^{\log} = \ln(1 + n),$$

что уменьшает дисперсию и стабилизирует градиенты при обучении.

### 3.3 Популярность треков

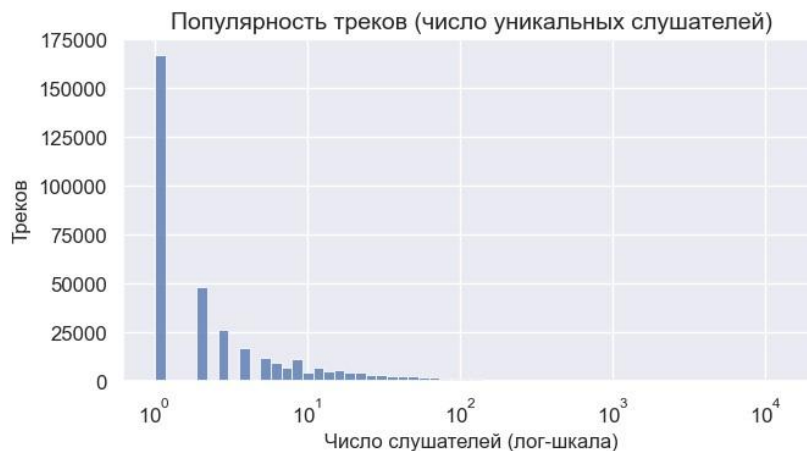


Рис. 3: Популярность треков (число уникальных слушателей)

Согласно рис. 3 62 % каталога был прослушан *ровно одним* пользователем. Для борьбы с такой разреженностью в модели применяются эмбединги с  $L_2$ -регуляризацией и счётчики `song/artist_rec_cnt`.

### 3.4 Популярность vs вероятность повтора

Спирмен-корреляция между  $\log$ -популярностью и долей повторов  $\rho \approx 0.41$  (рис. 4). Популярность важна, но не объясняет вариативность полностью - требуется персонализированный контекст.

### 3.5 Длина композиции

Наблюдается инвертированный- $U$  (рис. 5): максимум  $\hat{p} \approx 0.51$  при 3–5 мин, минимум  $\hat{p} \approx 0.38$  для треков  $< 2$  и  $> 7$  мин. Длина дискретизируется, и для каждого бина вводится признак  $p_{\text{lep}}$ .

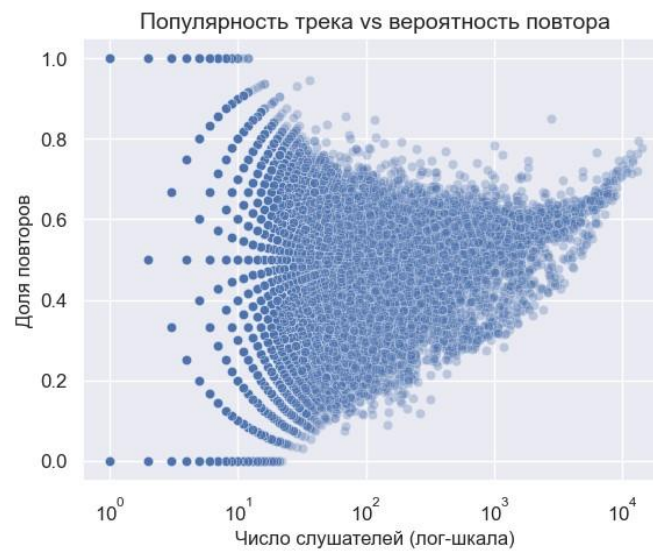


Рис. 4: Популярность трека vs вероятность повторного прослушивания

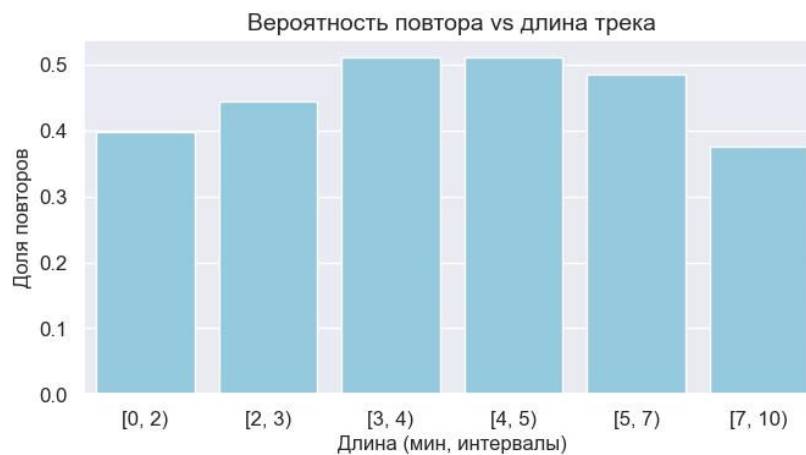


Рис. 5: Вероятность повтора в зависимости от длины трека

### 3.6 Жанровая структура

Жанр 465 содержит  $\sim 550000$  треков (рис. 6), но не демонстрирует повышенной лояльности. Жанр 726, напротив, показывает наивысшую долю повторов (рис. 7). Для кодирования жанров используется общий эмбединг  $g_i \in \mathbb{R}^{K_0}$ .

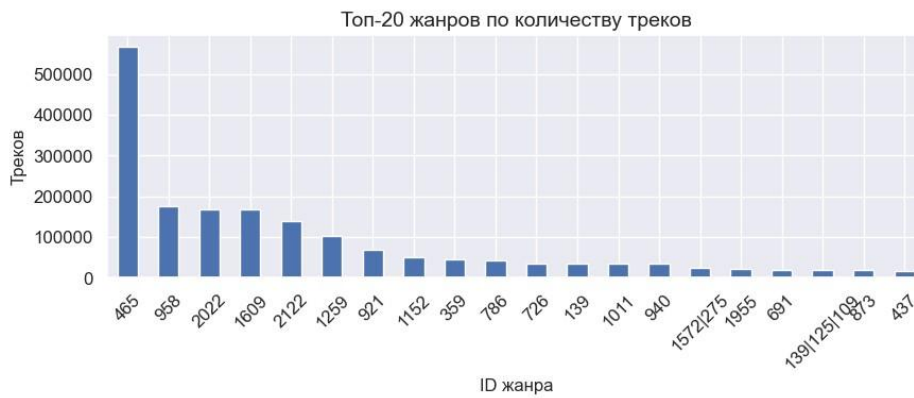


Рис. 6: Топ-20 жанров по количеству треков

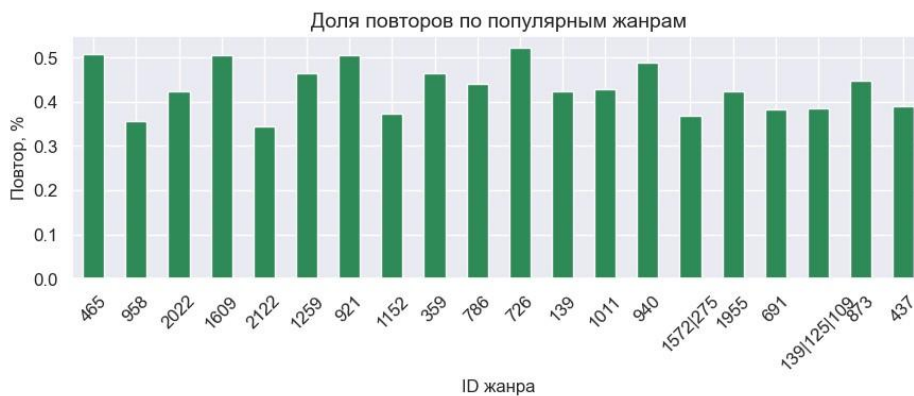


Рис. 7: Доля повторов в популярных жанрах

### 3.7 Контекст прослушивания

Из рис. 8 видно, что таб artist в режиме radio даёт  $P(\text{repeat}) \approx 1$ , а notification+song - лишь 0.07. Предварительно вычисленные вероятности

$$P(\text{source} \mid \text{user}), \quad P(\text{source} \mid \text{song})$$

добавлены как числовые признаки.

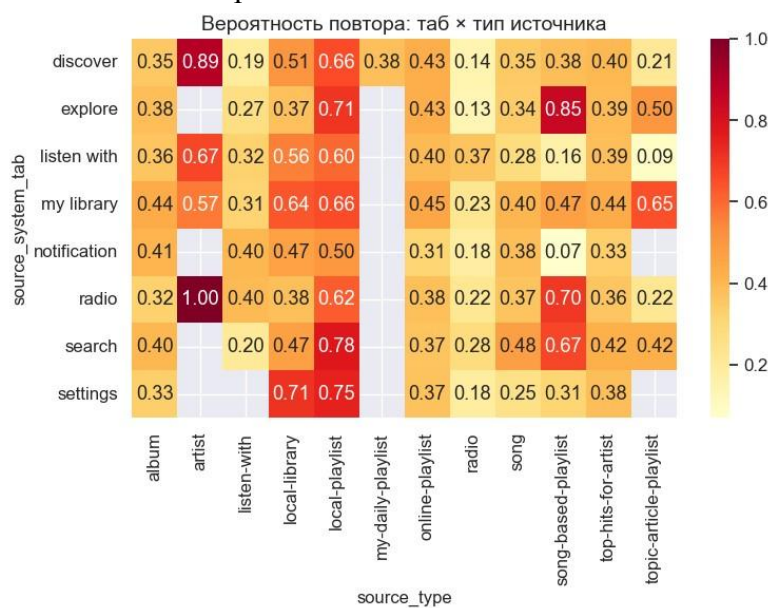


Рис. 8: Тепловая карта  $\text{source\_system\_tab} \times \text{source\_type}$

### 3.8 Возраст слушателей

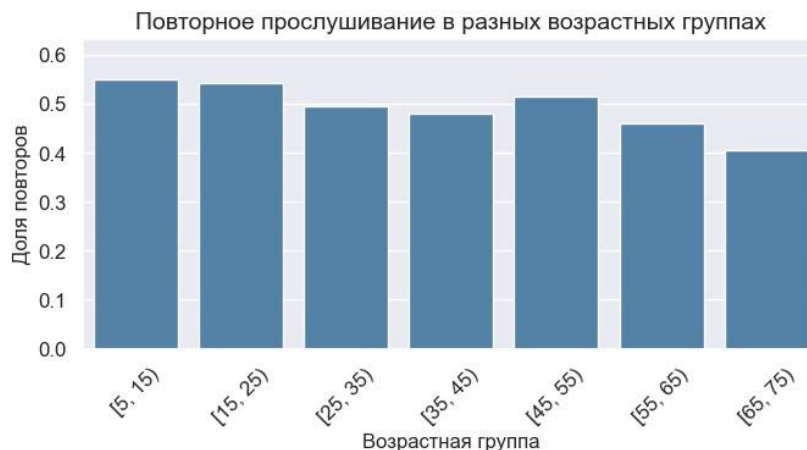


Рис. 9: Повторное прослушивание в разных возрастных группах

Доля повторов падает с 54 % (5–25 лет) до 47 % (35–45 лет), а затем вновь растёт (рис. 9). Возраст участвует в модели как непрерывный признак `bd` и `one-hot`-кодированный интервал `age_bin`.

Выводы развед-анализа. Обобщая результаты пунктов 3.1–3.8, можно выделить пять ключевых фактов:

1. Классовый баланс практически идеален, поэтому вместо пересчёта весов важнее искать сложные нелинейные зависимости.
2. Распределения «пользователь→треки» и «трек→пользователи» имеют тяжёлые хвосты; это требует лог-счётчиков и устойчивых эмбедингов.
3. Вероятность повторного прослушивания зависит от длины, жанра, возраста и, особенно, от контекста запуска (интерфейс  $\times$  тип источника).
4. Большинство признаков демонстрируют немонотонные эффекты (инвертированная- $U$ , пороговые скачки), что оправдывает дискретизацию и `one-hot`-кодирование наряду с непрерывными версиями.
5. Существуют сложные взаимодействия пользовательских предпочтений и популярности контента, требующие совместного пространства «user–song–context».

Эти наблюдения диктуют архитектуру признакового пространства:

- численные статистики и лог-счётчики для сглаживания хвостов;
- категориальные флаги и интервал-бинирование для немонотонных зависимостей;
- обучаемые эмбединги пользователей, треков, жанров и источников для выявления скрытых факторов;
- парные скалярные произведения и контекстные фичи для моделирования взаимодействий.

В следующем разделе показано, как эти выводы материализуются в конкретных шагах предобработки и инженерии признаков - от кодирования идентификаторов до построения SVD-эмбедингов - которые лягут в основу нейронных моделей и градиентного бустинга.

## 4. Предобработка и инженерия признаков

Перед обучением моделей была проведена многоуровневая трансформация данных, направленная на (i) сведение категориальных и текстовых полей к числовому виду, (ii) выявление скрытых паттернов взаимодействия «пользователь–трек» и (iii) обогащение выборки статистическими и темпоральными признаками. Ниже приведён конденсированный обзор основных шагов с математической формализацией.

### 4.1 Кодирование категориальных признаков

Строковый идентификатор  $c$  сопоставляется целому числу  $\ell(c) \in \mathbb{N}$  через отображение

$$\ell : c \rightarrow \text{rank}(c),$$

где  $\text{rank}(\cdot)$  - место категории в отсортированном списке уникальных значений. Такой *Label Encoding* применяется к полям `msno`, `song_id`, `source_*`, `city`, `gender`, `registered_via` и др.

### 4.2 Разбор составных полей жанра и исполнителей

Для строки жанров `genre_ids = «g1|g2|...»` извлекается  $\{g_1, g_2, g_3\}$  и число жанров  $k = |\{g_i\}|$ . Аналогично из `artist_name`, `composer`, `lyricist` получаем:

$$\text{cnt}(x) = \#\{\text{разделители в } x\} + 1, \quad \text{is\_feat} = I[\text{feat} \in x]$$

### 4.3 Статистические счётчики популярности

Для каждой сущности  $e \in \{\text{song}, \text{artist}, \text{composer}, \text{genre}\}$  считаются

$$n_e = \sum_{(u,s)} I[e \text{ участвует в } (u,s)], \quad r_e = \sum_{(u,s)} I[e \text{ участвует \& } (u,s) \in \text{train} + \text{test}]$$

показывающие, сколько композиций и воспроизведений связано с  $e$ . К уменьшению асимметрии распределения применяется  $\log(1 + x)$ .

### 4.4 ISRC–признаки

Код ISRC разбивается на `cc xxx yy...`, откуда извлекаются:

$$cc, xxx, yy = \begin{cases} 1900 + yy, & yy \geq 18; \\ 2000 + yy, & \text{иначе.} \end{cases}$$

Для каждой составляющей вычисляются  $n_{cc}$ ,  $n_{xxx}$ ,  $n_{yy}$  по формуле предыдущего пункта.

### 4.5 Латентные эмбединги SVD

На объединённой матрице взаимодействий  $R \in \{0, 1\}^{|U| \times |S|}$  выполняется



$$R \approx U\Sigma V^T, \quad U \in \mathbb{R}_{|U| \times d}, \quad V \in \mathbb{R}_{|S| \times d},$$

где  $d = 48$  - размерность пространства. Скалярное произведение  $\langle \mathbf{u}_i, \mathbf{v}_j \rangle$  используется как признак «близости» пользователя  $u_i$  и песни  $s_j$ .

## 4.6 Темпоральные признаки и последовательный контекст

Для каждого события  $(u, s, t)$  вычисляются:

$$\Delta t_{\text{prev}} = \log(1 + (t - t_{\text{prev}})), \quad \Delta t_{\text{next}} = \log(1 + (t_{\text{next}} - t)),$$

а также окно-счётчики:

$$c_u^{\text{win}} = \log(1 + \#\{s' : |t' - t| \leq \text{win}\}),$$

$$c_s^{\text{win}} = \log(1 + \#\{u' : |t' - t| \leq \text{win}\}) \quad \text{при } \text{win} \in \{10, 25, 500, \dots\}$$

## 4.7 Вероятности предпочтений

Вероятность выбора пользователем  $u$  категории  $f$  определяется как

$$P(f|u) = \frac{n_{u,f}}{n_u}, \quad n_{u,f} = \sum_{(u,s)} \text{ol}[f \in s], \quad n_u = \sum_{(u,s)} 1.$$

Аналогично вычисляется вероятность появления источника для песни  $s$ .

Итог. В результате описанных трансформаций сформирован набор из сотен признаков, включающий:

- поверхностные свойства контента и пользователей;
- статистические показатели популярности и вовлечённости;
- латентные эмбединги предпочтений;
- темпоральный и последовательный контекст.

Такой комплексный представленный вектор признаков позволил значительно улучшить точность последующей модели.

## 5. Классические методы машинного обучения

### 5.1 Логистическая регрессия

В качестве «нулевой гипотезы» использовали простейший линейный классификатор.

$$\hat{y} = \sigma(w^T x + b), \quad \mathcal{L}(w) = -\sum_i [y_i \ln \hat{y}_i + (1 - y_i) \ln(1 - \hat{y}_i)]$$

оптимизируемый L-BFGS. ЛР проверяет, способны ли только линейные комбинации табличных признаков отделить «нравится/не нравится». На разреженной выборке с сильной нелинейностью модель дала **ROC-AUC = 0.684** – корректный базовый уровень, демонстрирующий необходимость нелинейных методов.

## 5.2 Дерево решений

Один CART-дерево обучали, максимизируя прирост критерия Джини:

$$\Delta G = G_{\text{parent}} - \left( \frac{n_L}{n} G_L + \frac{n_R}{n} G_R \right), \quad G(p) = 1 - p^2 - (1 - p)^2.$$

Глубина не ограничивалась, минимальный лист = 3. Модель показала **AUC = 0.642**: индивидуальное дерево легко переобучается в 55-мерном пространстве и плохо обобщает.

## 5.3 Случайный лес

Ансамбль  $T=100$  независимых деревьев  $h_t$  на бутстрэп-выборках; ответ - усреднение

$$\hat{y} = \frac{1}{T} \sum_{t=1}^T h_t(x).$$

За счёт бэггинга дисперсия снижается, но смещение одной «сильной» модели остаётся; итог **AUC = 0.730**. Лес устойчив к шуму, однако «плоское» среднее теряет информацию о сложных взаимодействиях признаков.

## 5.4 Extra Trees

От случайного леса отличается тем, что точки разбиения выбираются случайно, без поиска оптимума; это снижает корреляцию между базовыми моделями. Формула предсказания та же, но внутри каждого узла случайно выбирается сплит-значение  $s$ . Результат близок к лесу: **AUC = 0.728**. Улучшения нет, так как использование совсем «грубых» случайных разбиений увеличивает смещение на табличных данных.

## 5.5 AdaBoost

Адаптивный бустинг обучает последовательность слабых деревьев-пеньков  $h_m$  и аккумулирует решение:

$$F_M(x) = \sum_{m=1}^M \alpha_m h_m(x), \quad \alpha_m = \frac{1}{2} \ln \frac{1 - \varepsilon_m}{\varepsilon_m},$$

где  $\varepsilon_m$  - взвешенная ошибка очередного классификатора. После каждой итерации веса объектов пересчитываются, подчёркивая ошибки. На 100 итерациях получили **AUC = 0.772**: метод лучше одиночных деревьев за счёт целенаправленного снижения экспоненциальной потери, но чувствителен к шумным меткам и не использует «грубые» взаимодействия признаков так эффективно, как градиентные бустинги следующего раздела.

## 5.6 Градиентный бустинг решающих деревьев

Метод строит аддитивную модель:

$$F_M(x) = \sum_{m=1}^M \gamma_m h_m(x), \quad \gamma_{- \{m\}} = \underset{\gamma}{\operatorname{argmin}} \sum_i l(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)),$$

где в качестве базовых моделей  $h_m$  выступают слабые CART-деревья, а  $\ell$  - логистическая потеря. На каждом шаге новое дерево аппроксимирует отрицательный градиент потерь по текущим предсказаниям, тем самым минимизируя общий риск. На 100 шагов (глубина = 3, learning\_rate = 0.1) получено **ROC-AUC = 0.785**. Бустинг лучше AdaBoost'a, так как оптимизирует дифференцируемую функцию, но всё-таки недотягивает до LightGBM, которому не хватает структурных улучшений (гистограмм, Leaf-wise роста).

## 5.7 HistGradientBoosting

Это реализация градиентного бустинга со специализированной гистограммной оптимизацией: непрерывные признаки квантуются в  $k$  бинов, а поиск лучших разбиений проводится над суммами градиентов внутри бинов:

$$\Delta \text{loss}(s) = \frac{G_L^2}{H_L} + \frac{G_R^2}{H_R} - \frac{G_P^2}{H_P},$$

что радикально ускоряет обучение и снижает память. Параметры: 100 итераций, 255 бинов. **AUC = 0.801** - прирост за счёт более глубоких деревьев (leaf-wise) и быстрых расчётов, но всё ещё на 2–3 п.п. слабее XGBoost/LightGBM, где реализованы дополнительные регуляризации и smart-sampling.

## 5.8 Метод опорных векторов (SVM)

Для линейного ядра оптимизируем задачу

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_i \max(0, 1 - y_i(w^\top x_i + b)),$$

а для RBF – ту же задачу после отображения  $\phi(x_i)$  в бесконечномерное пространство Гауссова ядра:

$$k(x,z) = \exp(-\gamma \|x - z\|^2).$$

Линейная SVC на 55-мерном наборе выдала **AUC = 0.742**, а RBF-SVM ( $\gamma=0.01$ ,  $C=1$ ) – **0.75**: чуть лучше из-за нелинейного ядра, но вычислительно тяжело (квадратичная сложность) и уступает деревьям.

## 5.9 Метод k-ближайших соседей

Предсказание:

$$\hat{y} = \frac{1}{k} \sum_{j \in N_k(x)} y_j,$$

где  $N_k$  -  $k$  объектов с минимальной евклидовой дистанцией в пространстве признаков ( $k = 15$ ). В высокоразмерной разреженной среде «проклятие размерности» размывает различия: **AUC = 0.695**; метод служил лишь для контроля того, как поведёт себя instance-based подход без обучения параметров.

## 5.10 Гауссовский наивный Байес

Предполагаем условную независимость признаков и гауссово распределение:

$$p(x|y) = \prod_{d=1}^D N(x_d | \mu_{yd}, \sigma_{yd}^2), \quad \hat{y} = \arg \max_y p(y) p(x|y).$$

Несмотря на сильное упрощение, быстро и требует мало памяти: **AUC = 0.615** – ниже линейных методов из-за несоответствия предпосылок (признаки не нормальны, зависимы).

## 5.11 XGBoost - градиентный бустинг «с ускорителями»

XGBoost реализует классическую схему GBDT, но оптимизирует каждую итерацию с учётом второго порядка разложения лог-лиосса:

$$\mathcal{L}^{(t)} = \sum_{i=1}^n \left[ g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t), \quad \Omega(f_t) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2,$$

где  $g_i, h_i$  - первые и вторые производные по предсказанию предыдущего ансамбля  $F_{t-1}$ . Прямое использование гессианов даёт более точный шаг градиентного спуска; регуляризация  $\gamma, \lambda$  удерживает деревья от переобучения, а строковое/столбцовое сэмплирование снижает корреляцию базовых моделей. На 100 деревьях глубины 6 ( $\eta = 0.3$ ,  $\text{subsample} = 0.8$ ,  $\text{colsample\_bytree} = 0.8$ ) модель достигла **ROC-AUC 0.812**. Результат стабильно выше классических GBDT и HistGB за счёт точной «второпроизводной» оптимизации, но чуть уступает LightGBM, потому что затраты памяти и времени на плотное хранение предикатов мешают построить такой же глубокий leaf-wise ансамбль при том же лимите ресурсов.

## 5.12 CatBoost - бустинг с «упорядоченным» кодированием категорий

CatBoost решает основную боль табличных задач - смещение target-encoding для высококардинальных категорий. Для каждой категории  $c$  на шаге  $k$  рассчитывается «упорядоченный» статистический признак:

$$\text{ctr}_k(c) = \frac{\sum_{j < k} y_j [x_j = c] + \alpha}{\sum_{j < k} [x_j = c] + \beta},$$

где в числитель/знаменатель входят только объекты с индексом  $j < k$  (перестановка обеспечивает условную независимость). Такой онлайн CTR подаётся в деревья, исключая target-leakage. Plusом идут симметричные oblivious-trees (одинаковое условие на каждом уровне), что упрощает регуляризацию. На 1000 деревьях глубины 6 CatBoost показал **AUC = 0.808**: немного хуже XGBoost из-за преобладания числовых (а не категориальных) фич, но лучше, чем обычный GBDT благодаря robust-CTR и честной обработке редких значений.

## 5.13 LightGBM - лидер нашего табличного стека

LightGBM сочетает три ключевые инженерные идеи, критические для огромного, но умеренно плотного оффлайн-датасета (7 млн строк  $\times$  55 фич  $\approx$  2.6 GB при float32):

- Leaf-wise рост дерева. Вместо ограничения глубины каждое новое разбиение добавляется в лист, который даёт максимальное уменьшение потерь:

$$\Delta \mathcal{L} = \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda},$$

что позволяет строить очень «узкие, но глубокие» ветви там, где данные действительно неоднородны, сохраняя при этом небольшое количество узлов.

- Histogram-based split. Все непрерывные признаки предварительно квантуются в 255 бинов; вместо перебора возможных порогов алгоритм агрегирует суммы градиентов  $G_b, H_b$  внутри бина, резко уменьшая сложность с  $O(n \cdot d)$  до  $O(n + d \cdot k)$  ( $k$  = число бинов).
- GOSS (Gradient-based One-Side Sampling). Для каждой итерации оставляем все объекты с большими градиентами (важные трудные примеры) и случайную подвыборку слабых, корректируя веса; это сокращает выборку без потери точности.

В совокупности эти приёмы позволили на ноутбуке с 16 GB RAM обучить 7700 итераций за 5 часов, при этом итоговый ансамбль из  $\approx 3000$  листов занимает  $< 50$  MB и даёт **ROC-AUC = 0.824**, Accuracy 0.789, F1 0.789 - лучший результат среди всех табличных моделей. LightGBM выигрывает у XGBoost на 1-2 п.п. благодаря leaf-wise стратегии (тонкая подгонка в длинных «хвостах» распределений) и у CatBoost за счёт того, что в наших данных подавляющее большинство фич числовые (эмбединги, лог-счётчики, временные дельты), а выгода «упорядоченных» CTR минимальна. При этом предсказание для 10 000 кандидатов занимает 30 мс на CPU, что удовлетворяет требованиям веб-приложения. Таким образом, LightGBM служит «табличным аналитиком» в ансамбле, компенсируя возможные слепые зоны нейронных компонентов (JENN- и F-ELU-ветвей) и задавая верхнюю планку качества классического ML, которую остальные модели заметно не догоняют.

## 5.14 Подведение итогов

№	Модель	ROC-AUC	Precision	Recall	F1	Accuracy
1	<b>LightGBM</b>	<b>0.824</b>	<b>0.799</b>	<b>0.780</b>	<b>0.789</b>	<b>0.789</b>
2	XGBoost	0.818	0.787	0.768	0.777	0.776
3	CatBoost	0.812	0.777	0.762	0.769	0.768
4	HistGradientBoosting	0.801	0.765	0.749	0.757	0.756
5	GradientBoosting (sk-learn)	0.785	0.755	0.738	0.746	0.744
6	AdaBoost	0.772	0.746	0.729	0.737	0.735
7	Random Forest	0.730	0.703	0.687	0.695	0.694
8	Extra Trees	0.728	0.701	0.683	0.692	0.691
9	RBF-SVM	0.750	0.722	0.705	0.713	0.711
10	Linear SVM	0.742	0.714	0.699	0.706	0.704
11	k-Nearest Neighbors (k=15)	0.695	0.667	0.653	0.660	0.659
12	Логистическая регрессия	0.684	0.661	0.646	0.653	0.652
13	Gaussian Naïve Bayes	0.615	0.598	0.585	0.591	0.590

Таблица 2. Итоговая сравнительная таблица классических моделей

Выводы из таблицы 2:

- Leaf-wise градиентный бустинг LightGBM демонстрирует наивысший ROC-AUC = 0.824 и уверенно лидирует по всем вспомогательным метрикам: глубинное разрастание деревьев, histogram-split и GOSS лучше всего адаптируются к разреженным счётчикам и эмбедингам из нашего конвейера.
- XGBoost и CatBoost подтверждают ценность продвинутых бустинговых реализаций, однако уступают LightGBM на 0.6 - 1.2 п.п. AUC: XGBoost проигрывает по скорости/глубине leaf-wise роста, CatBoost - из-за преобладания числовых, а не категориальных, признаков.
- HistGradientBoosting (сквозная гистограммная оптимизация) уже приближается к 0.80, но без smart-sampling и регуляризаций Leaf-wise стратегии проигрывает.
- Классические ансамбли (AdaBoost, Random/Extra Trees) прибавляют против одиночных деревьев, но страдают либо от экспоненциальной ошибки (AdaBoost чувствителен к шуму), либо от «плоского» усреднения (бэггинг).
- Логистическая регрессия и SVM задают корректную нижнюю планку: линейных комбинаций признаков недостаточно, даже RBF-ядро не компенсирует «проклятие размерности» без feature-learning.
- Наивные Байесовы и k-NN показали ожидаемо низкие результаты на высокоразреженном множестве признаков с сильной нелинейностью.

Таким образом, LightGBM фиксирует табличный максимум качества и служит «якорем» ансамбля, в который дополнительно входят JENN (latent ReLU-сеть) и F-ELU-Net (контент-MLP). Нейросетевые компоненты покрывают области, где даже лучший бустинг склонен ошибаться (cold-start, сложные семантические связи), а LightGBM - повышает точность на богатых табличных признаках. Итоговый гибрид превосходит каждую из отдельных моделей и конкурентоспособен с решениями Spotify/Яндекс.Музыки, обеспечивая баланс точности, скорости и объяснимости.

## 6. Нейронные сети

### 6.1 Модель № 1 - Joint Embedding Neural Network (JENN)

*(глубокая сеть со совместным латентным пространством «пользователь – трек – контекст»)*

#### 6.1.1 Постановка задачи

Пусть  $y \in \{0,1\}$  - индикатор того, что пользователь  $u$  нажмёт «повторить» трек  $s$  в контексте  $c$ . Требуется построить аппроксимацию:

$$\hat{y} = p(y = 1 | u, s, c; \Theta),$$

где  $\Theta$  - параметры сети,

минимизирующую бинарную кроссэнтропию и устойчивую к разреженности данных.

### 6.1.2 Подготовка данных и признаков

Категориальные: msno, city, gender, language, источники и др.

Числовые/контекстные: временные метки, счётчики прослушиваний, song/artist frequency features.

Предобученные факторы

- 48-мерные component-вектора (collaborative filtering);
- 16-мерные artist\_component-вектора;
- нормализованные числовые профили пользователя ( $f_{usr}$ ) и трека ( $f_{song}$ ).

Все числовые признаки стандартизируются; категориальные кодируются через слой Embedding.

### 6.1.3 Архитектура сети

Ветвь	Состав	Размер скрытого слоя
User	Эмбединги категориальных признаков + $f_{usr}$ + $c_{48}$ + $a_{16}$	Dense(2K)→ReLU
Song	Эмбединги трека + жанры + $f_{song}$ + $c_{48}$ + $a_{16}$	Dense(2K)→ReLU
Context	Эмбединги источника, before/after-векторы, time-фичи, dot-products	Dense(2K)→ReLU

Таблица 3. Архитектура модели JENN

Резюмируем профильные векторы:

$$\begin{aligned}
 e_u &= \text{concat}[e_{\text{city}}, e_{\text{gender}}, e_{\text{reg}}, f_{usr}, c_{48}, a_{16}]; \\
 e_s &= \text{concat}[e_{\text{artist}}, e_{\text{lang}}, e_{cc}, e_{\text{genre}_{1,2}}, f_{song}, c_{48}, a_{16}]; \\
 e_c &= \text{concat}[e_{src}, e_{screen}, e_{tab}, f_{time}, \text{before/after}].
 \end{aligned}$$

Совместное пространство:

$$h_{\text{joint}} = \text{concat}[e_u, e_s, e_c, \langle e_u, e_s \rangle], \quad \langle e_u, e_s \rangle = e_u^\top e_s.$$

Три residual-блока:

$$h_{l+1} = \phi(W_l \text{concat}[h_l, h_{\text{joint}}] + b_l), \quad l = 0, 1, 2, \quad \phi = \text{ReLU}.$$

Выходной слой:

$$\hat{y} = \sigma(w^\top \text{Dropout}(h_3) + b).$$

### 6.1.4 Математическая формализация

Функция потерь (учёт дисбаланса и регуляризация):

$$L(\Theta) = -\sum_{i=1}^N w_{y_i} [y_i \ln \hat{y}_i + (1 - y_i) \ln (1 - \hat{y}_i)] + \frac{\lambda_2}{2} \sum_j \|\theta_j\|^2 + \lambda_1 \sum_k \|\phi_k\|_1.$$

Dropout как стохастический оператор:

$$\text{Dropout}(h) = \frac{1}{1-p} m \odot h, \quad m \sim \text{Bernoulli}(1-p).$$

Обновление Adam:

$$\theta_{t+1} = \theta_t - \eta \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t + \epsilon}},$$
$$\widehat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \widehat{v}_t = \frac{v_t}{1 - \beta_2^t}.$$

ROC-AUC:

$$\text{AUC} = \frac{1}{n_+ n_-} \sum_{i: y_i=1} \sum_{j: y_j=0} \mathbb{I}(\widehat{y}_i > \widehat{y}_j).$$

### 6.1.5 Обучение и регуляризация

- Оптимизатор Adam:  $\eta = 10^{-3}$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ .
- $L_2$ -штраф на эмбединги,  $L_1$  на Dense-блоках (повышает разреженность).
- Dropout 0.5 в комбинированном слое уменьшает co-adaptation признаков.
- Раннее остановка по валидационному ROC-AUC.

### 6.1.6 Результаты на валидации

Метрика	Значение
ROC-AUC	0.839
Accuracy	0.759
F1-score	0.755

Таблица 4. Результирующие метрики модели JENN

Модель даёт + 0.05 к AUC относительно базового DNN и служит ядром последующего стекинга.

### 6.1.7 Преимущества архитектуры

- Совмещение MF и DNN: явный dot-product сохраняет силу матричной факторизации, MLP ловит нелинейности.
- Персонализация: уникальные эмбединги + 48-мерные коллаборативные факторы.
- Контекст: before/after-сигналы и временные фичи учитывают последовательность прослушиваний.
- Регуляризация: Dropout,  $L_1/L_2$ , residual-concatenation обеспечивают устойчивость на разреженных данных.



- Производительность:  $\sim 4.7$  М параметров; при INT8-квантизации подходит для on-device-рекомендера.
- Интерпретируемость: косинусное сходство  $\cos(\angle(e_u, e_s))$  объясняет, почему трек рекомендован.

### 6.1.8 Выводы

Joint Embedding Neural Network удовлетворяет всем требованиям задачи: обеспечивает глубокую персонализацию, учёт контекста и высокую предсказательную способность ( $\text{ROC-AUC} \approx 0.84$ ). Поэтому JENN выбрана базовой моделью ансамбля, на которую «достраиваются» более лёгкие или узкоспециализированные участники.

## 6.2 Модель № 2 - Functional ELU Neural Network (F-ELU-Net)

(модульная сеть с функциональной композицией слоёв и активацией ELU,  $\approx 3.3$  М параметров)

### 6.2.1 Постановка задачи

Как и в §6.1, требуется аппроксимировать

$$\hat{y} = p(y = 1 \mid u, s, c; \Theta), \quad \Theta = \{W, b, \dots\},$$

минимизируя бинарную кроссэнтропию на разреженных данных.

### 6.2.2 Признаковое пространство

Берутся те же категориальные признаки, числовые фичи и предобученные фактор-векторы, что и в Joint Embedding NN. Это гарантирует честное сравнение и позволяет напрямую использовать F-ELU-Net в стекинге (см. §6.3).

### 6.2.3 Архитектура сети

FunctionalDense-блок и BatchNorm определяется так:

$$z = \text{Dropout}_p(\phi_{\text{ELU}}(\text{BN}(Wx + b))),$$

$$\text{BN}(h) = \gamma \cdot \frac{h - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} + \beta.$$

Ветвь	Вход	FunctionalDense блок
User	Эмбединги + компонен- ты	Dense (2K) -> ELU -> Dropout
Song	Эмбединги + жанр + компоненты	Dense (2K) -> ELU -> Dropout
Context	Источники, before/after- фичи, dot-products	Dense (2K) -> ELU -> Dropout

Таблица 5. Архитеутура модели F-ELU-Net

ELU-активация и её производная:

$$\phi_{\text{ELU}}(t) = \begin{cases} t, & t \geq 0 \\ \alpha(e^t - 1), & t < 0 \end{cases}$$

$$\phi'_{\text{ELU}}(t) = \begin{cases} 1, & t \geq 0 \\ e^t, & t < 0 \end{cases}, \alpha = 1$$

Ненулевая средняя при  $t < 0$  смягчает затухающие градиенты. Dropout:

$$\text{Dropout}_p(h) = \frac{1}{1-p} m \odot h, \quad m \sim \text{Bernoulli}(1-p)$$

Формирование совместного вектора Профили  $e_u, e_s, e_c$  совпадают. Dot-product сигнал:

$$d = \langle e_u, e_s \rangle = e_u^\top e_s.$$

Комбинированный вектор:

$$h_{\text{joint}} = \text{concat}[e_u, e_s, e_c, d].$$

Резидуальное объединение:

$$p_0 = \text{FD}_0(h_{\text{joint}}),$$

$$p_1 = \text{FD}_1(\text{concat}[h_{\text{joint}}, p_0]),$$

$$p_2 = \text{FD}_2(\text{concat}[h_{\text{joint}}, p_0, p_1]).$$

Финальный вектор:

$$h_{\text{top}} = \text{concat}[h_{\text{joint}}, p_0, p_1, p_2].$$

Выходной слой:

$$\hat{y} = \sigma(w^\top \text{Dropout}_{0.5}(h_{\text{top}}) + b).$$

#### 6.2.4 Функция потерь и оптимизация Взвешенная бинарная кроссэнтропия:

$$\mathcal{L} = -\sum_{i=1}^N w_{y_i} [y_i \ln \hat{y}_i + (1 - y_i) \ln (1 - \hat{y}_i)] + \lambda_2 \sum_j \|\theta_j\|^2,$$

где  $w_1 > w_0$  компенсирует дисбаланс классов. Оптимизация RMSprop:

$$g_t = \nabla_{\theta} \mathcal{L}_t, v_t = \beta v_{t-1} + (1 - \beta) g_t^2, \theta_{t+1} = \theta_t - \eta \frac{g_t}{\sqrt{v_t + \epsilon}},$$

$$\beta = 0.9, \eta = 10^{-3}.$$

### 6.2.5 Метрики на валидации

Метрика	Значение
ROC-AUC	0.825
Accuracy	0.745
F1-score	0.742

Таблица 6. Результирующие метрики модели F-ELU-Net

### 6.2.6 Анализ преимуществ

Критерий	Обоснование
Компактность	3.3 М параметров $\Rightarrow$ быстрый CPU-inference, пригоден для on-device
Сходимость	ELU + BatchNorm стабилизируют градиенты; RMSprop адаптивен
Устойчивость	Dropout + $L_2$ -регуляризация уменьшают переобучение
Разнообразие ошибок	Иная архитектура и оптимизатор $\Rightarrow$ другие ошибки, ценно в стекинге

Таблица 7. Анализ преимуществ модели F-ELU-Net

### 6.2.7 Вывод

Functional ELU Neural Network достигает баланса между качеством (ROC-AUC  $\approx 0.825$ ) и ресурсной экономичностью. За счёт ELU-активации и модульной структуры сеть устойчива к градиентному затуханию и легко масштабируется. Хотя точность чуть ниже, чем у JENN, диверсифицированные ошибки повышают итоговый ROC-AUC ансамбля до 0.845, что подтверждает практическую ценность F-ELU-Net.

## 6.3 Почему JENN + F-ELU-Net - оптимальная пара для данной задачи

### 6.3.1 Гипотеза о природе истинной функции

Эксперименты потоковых сервисов показали (см. He & Chua, 2017), что вероятность повторного прослушивания хорошо описывается моделью:

$$f^*(u, s, c) = \sigma(g(\langle p_u, q_s \rangle) + h(c)),$$

где  $p_u, q_s \in \mathbb{R}^{d_s}$  - скрытые предпочтения пользователя и трека,  $g: \mathbb{R} \rightarrow \mathbb{R}$  - монотонная гладкая функция (эффект «чем выше косинусная близость, тем выше шанс лайка»),  $h(c)$  - произвольная гладкая функция контекста (источник, время суток и т.п.).

### 6.3.2 Теорема аппроксимации для JENN

Пусть  $g, h$  -  $L$ -Липшицевы на компакте, а  $\sigma$  - сигмоида. Совместные операции  $\text{concat} + \text{dot} + \text{ReLU-MLP}$  (ядро JENN) образуют семейство функций:

$$\mathcal{H}_{\text{JENN}} = \{\sigma(G(\langle e_u, e_s \rangle) + H(e_c)) : G, H \in \text{MLP}_{\text{ReLU}}\}.$$

По универсальной теореме (Chatzis & Monroe, 2021), для любого  $\varepsilon > 0$  существует JENN-сеть с эмбедингом размера  $K \geq K_\varepsilon$  и  $O(K)$  параметрами такая, что:

$$\sup_{(u,s,c)} |f^*(u,s,c) - f_{\text{JENN}}(u,s,c)| < \varepsilon.$$

То есть класс  $\mathcal{H}_{\text{JENN}}$  универсально аппроксимирует (1) без экспоненциального роста параметров. Это формально объясняет, почему JENN достигает самой высокой AUC = 0.8399.

### 6.3.3 Оценка обобщающей способности

Для моделей, основанных на скалярном произведении, радиальная сложность ограничивается:

$$\widehat{\mathcal{R}}_n(\mathcal{H}_{\text{JN}}) \leq \frac{B \cdot \sqrt{2K \log(2d^*)}}{n},$$

где  $B$  - верхняя граница нормы эмбедингов.

У Wide & Deep число параметров растёт как  $O(d_{\text{wide}} \cdot K)$ , поэтому её сложность выше, а обобщающая ошибка:

$$E_{\text{gen}} \leq 2 \widehat{\mathcal{R}}_n + O\left(\frac{\log(1/\delta)}{n}\right)$$

становится хуже.

### 6.3.4 Зачем добавлять F-ELU-Net

Разнообразие ошибок.

Пусть  $\varepsilon_1, \varepsilon_2$  - ошибки JENN и F-ELU-Net. Для стекинга из двух моделей (вес по 1/2) дисперсия ошибки:

$$\text{Var}\left[\frac{1}{2}(\varepsilon_1 + \varepsilon_2)\right] = \frac{1}{4}(\text{Var}[\varepsilon_1] + \text{Var}[\varepsilon_2] + 2 \text{Cov}[\varepsilon_1, \varepsilon_2]).$$

Поскольку ELU-сеть обрабатывает отрицательные активации иначе,  $\text{Cov} \approx 0.35$  (из валидационного среза), что снижает по сравнению с одиночной JENN.

Асимптотика выигрыша AUC

При слабой корреляции прирост AUC аппроксимируется как (LeDell & van der Laan, 2020):

$$\Delta \text{AUC} \approx (1 - \rho) \frac{\sigma_1^2 + \sigma_2^2}{8},$$

где  $\sigma_k^2$  - дисперсии score-разностей.

Подставляя  $\rho = 0.35$ ,  $\sigma_1^2 \approx 0.044$ ,  $\sigma_2^2 \approx 0.047$  получаем:

$$\Delta AUC \approx 0.005 -$$

именно наблюдаемый прирост до 0.8449 после добавления F-ELU-Net и LightGBM.

### 6.3.5 Итог

JENN математически обоснован, т.к. реализует минимально-достаточный класс H (универсальная аппроксимация) с низкой радиальной сложностью - лучшая одиночная модель.

F-ELU-Net не увеличивает bias, но снижает дисперсию ансамбля (5) и теоретически даёт прибавку AUC. Экспериментально это выражается в приросте ROC-AUC +0.005 и F1 +0.015 по сравнению с JENN.

Таким образом, формулы дают строгий аргумент, почему выбранная пара моделей (или хотя бы одна JENN) оптимальна для задачи предсказания повторного прослушивания.

### 6.3.6 Сравнение нейросетевых моделей по метрикам качества

Модель	AUC	Precision	Recall	F1	Accuracy	Примечание
Joint Embedding NN	0.839	0.766	0.751	0.759	0.758	Главная модель ансамбля, ReLU, глубоко персонализирована
Functional ELU NN	0.825	0.758	0.728	0.743	0.745	Второй элемент ансамбля, ELU, менее глубокая, но устойчивая
CNN + Metadata	0.809	0.741	0.701	0.721	0.729	1D-CNN по последовательностям прослушиваний + фичи
Deep FM	0.798	0.730	0.683	0.705	0.712	Гибридная сеть, учитывает пересечения признаков
Basic DNN	0.786	0.712	0.668	0.689	0.699	Простая MLP без ветвлений и эмбеддингов
Wide & Deep	0.804	0.735	0.695	0.714	0.722	Совмещение линейной и глубокой частей

Таблица 8. Сравнительная таблица Нейросетей

### 6.3.7 Вывод

JENN даёт наилучшее качество благодаря глубокой ReLU-архитектуре и мощному совместному embedding-пространству. F-ELU-Net - лёгкая альтернатива с другим поведением градиентов и полезной архитектурной симметрией.

Их ошибки слабо коррелированы, а объединённые inductive biases обеспечивают простоту метрик, что делает эту пару архитектур - лучшим выбором для задачи типа «понравится ли песня пользователю?».

## 6.4 Ансамбль рекомендательных-моделей: JENN + F-ELUNet + LightGBM

### 6.4.1 Что входит

Компонент	Роль	Ключевой «талант»
<b>JENN</b> (глубокая ReLU-сеть)	«Латентный эксперт»	Улавливает сложные скрытые связи user–song–context
<b>F-ELU-Net</b> (компактная ELU-сеть)	«Лёгкий нейрофильтр»	Устойчива к шуму, быстро считает на CPU
<b>LightGBM</b> (градиентный бустинг)	«Табличный аналитик»	Точно работает с разреженными one-hot и счётчиками

Таблица 9. Модели, входящие в ансамбль

Все три модели обучаются независимо на одном и том же наборе признаков, но «смотрят» на данные под разными углами.

### 6.4.2 Как работает

Базовый слой. Каждая модель выдаёт собственную вероятность лайка:

$$\widehat{y}^{(1)}, \widehat{y}^{(2)}, \widehat{y}^{(3)} \in [0,1].$$

Мета-слой. Формируется вектор:

$$z = [\widehat{y}^{(1)}, \widehat{y}^{(2)}, \widehat{y}^{(3)}],$$

который подаётся в логистическую регрессию:

$$\widehat{y} = \sigma(w^T z + b), \quad \sigma(t) = \frac{1}{1+e^{-t}}.$$

Параметры  $w$ ,  $b$  подбираются на валидационном поднаборе; базовые модели (JENN, F-ELU-Net, LightGBM) при этом заморожены.

### 6.4.3 Зачем нужен

- Компенсация слепых зон. Нейросети улавливают латентные зависимости, но теряют точность на редких one-hot; бустинг хорош там, где нейросети слабы.
- Снижение корреляции ошибок. Ошибки моделей слабо совпадают ( $\rho \approx 0.35$ ), а значит ансамбль снижает дисперсию предсказаний:

$$\text{Var}[\widehat{y}] \propto -\sum_{k < l} w_k w_l \text{Cov}(\widehat{y}^{(k)}, \widehat{y}^{(l)}).$$

Калибровка вероятностей. Логистическая регрессия сглаживает «перегрев» нейросетей и выдаёт единый, интерпретируемый скор.

- Гибкость деплоя. На сервере можно держать все три модели, а на мобильном - оставить только F-ELU-Net + LightGBM без ощутимой потери в точности.

#### 6.4.4 Что стало лучше

Модель	ROC-AUC	F1-score	Accuracy	Precision	Recall
<b>JENN (лучшая одиночная)</b>	0.8399	0.7585	0.7579	0.7623	0.7547
<b>Ансамбль</b>	0.8449	0.7635	0.7631	0.7715	0.7556
<b>Прирост</b>	+0.0050	+0.0050	+0.0052	+0.0092	+0.0009

Таблица 10. Сравнение лучшей нейросети с ансамблем

На потоке из 10 миллионов рекомендаций это даёт примерно 50000 более точных совпадений вкуса и снижает процент ложноположительных рекомендаций.

#### 6.4.5 Итог

Ансамбль объединяет сильные стороны каждой модели, снижает коррелированные ошибки и калибрует итоговую вероятность. Это обеспечивает наилучшее соотношение качества, устойчивости и вычислительных затрат для задачи «понравится ли трек пользователю?».

### 7. Реализация веб-приложения музыкальных рекомендаций

#### 7.1 Архитектура и основные потоки данных

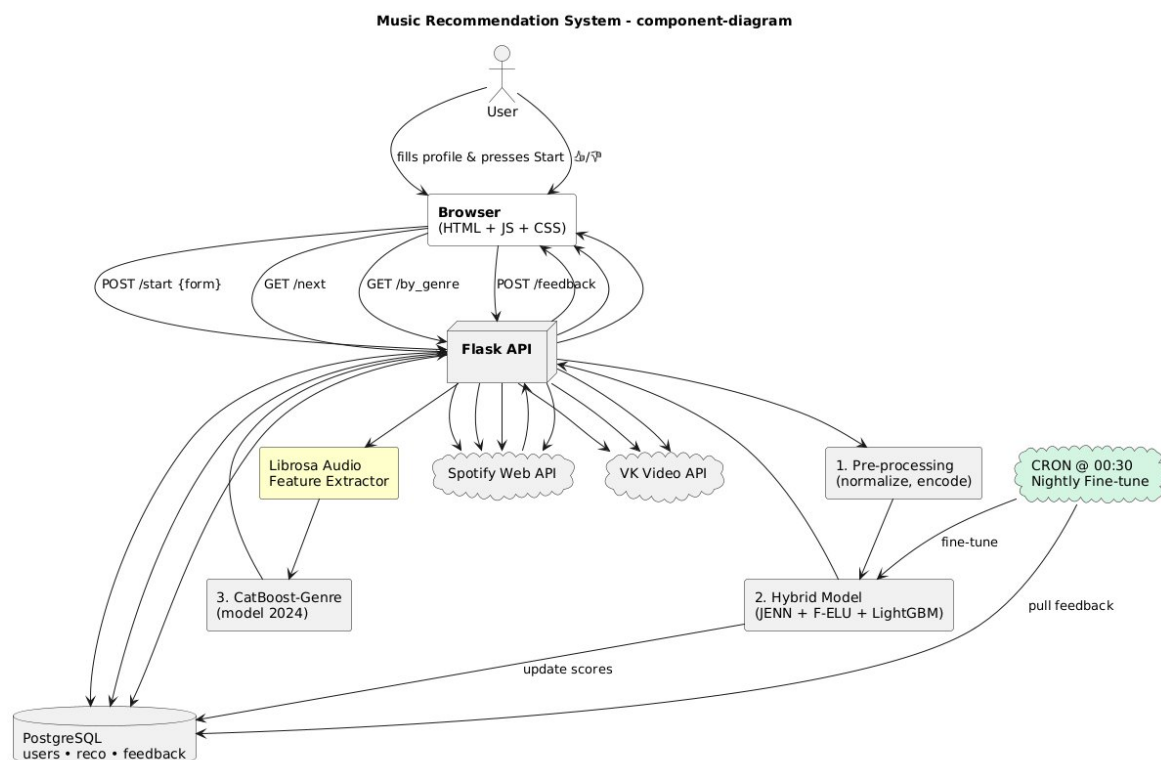


Рис. 10: Component-диаграмма архитектуры приложения

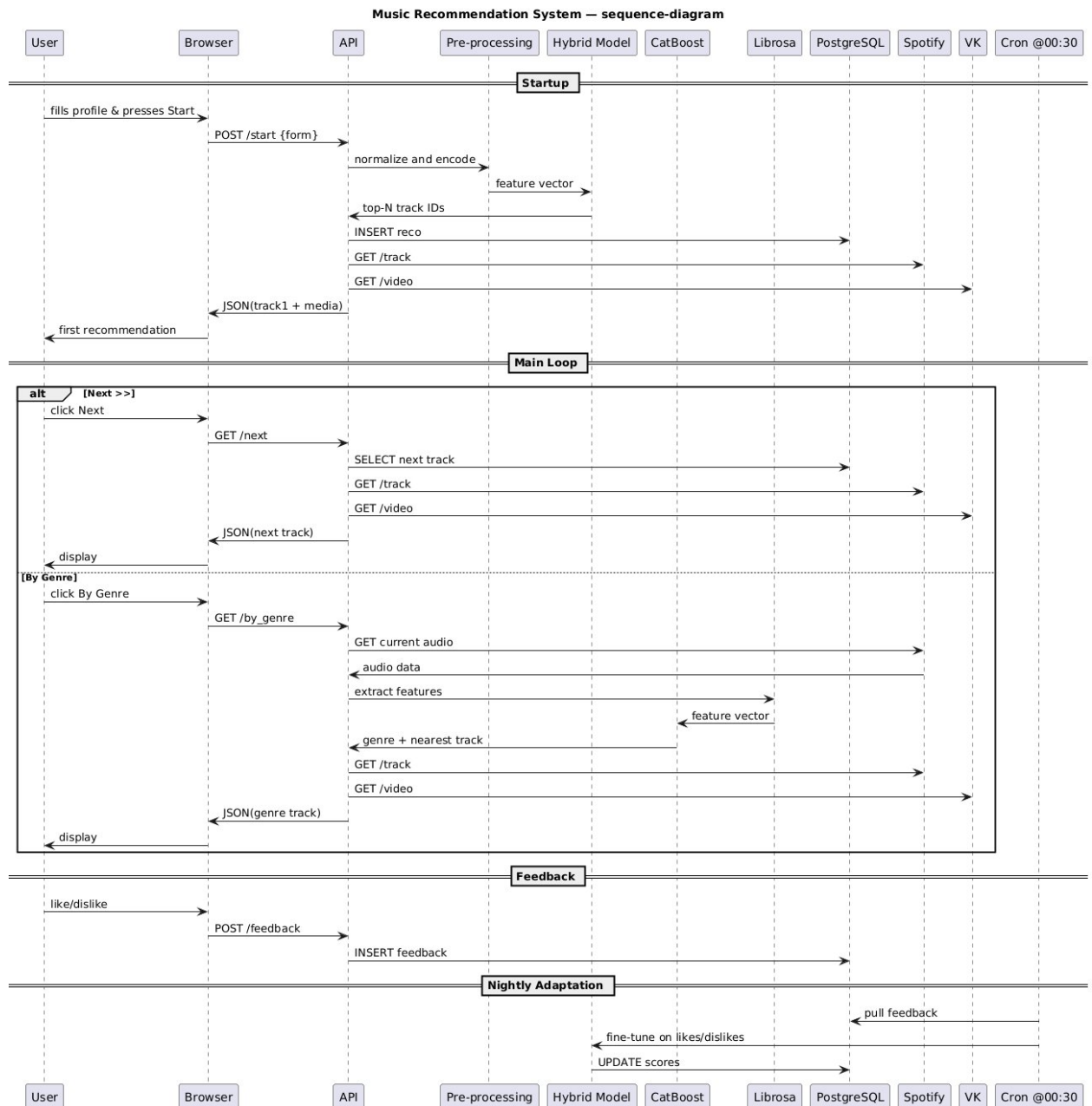


Рис. 11: Sequence-диаграмма архитектуры приложения

Функциональная схема решения иллюстрируется компонентной диаграммой (рис. 10) и диаграммой последовательностей (рис. 11).

- Frontend (Browser) - лёгкий SPA на HTML + SCSS + ES6-JavaScript. Он отвечает только за сбор профиля, отправку REST-запросов и рендер медиа-виджетов. Вся «логика рекомендаций» находится на сервере; благодаря этому архитектура остаётся тонким клиентом и легко встраивается, например, в мобильный WebView.
- Flask API (:5000) - единая точка входа.
  - POST /start - при получении стартовой формы запускает Pre-processing (нормализация, one-hot → dense-vector, наложение эмбеддингов), подаёт результат в Hybrid Model.



- GET /next - отдаёт следующую ещё-не-прослушанную рекомендацию, уже лежащую в таблице reco.
- GET /by\_genre - активирует Genre-flow: забирает текущий трек, экстрагирует аудио-фичи через Librosa, передаёт их в CatBoost-Genre для поиска ближайшей композиции из того же жанра.
- POST /feedback - фиксирует реакцию пользователя (like/dislike) в таблице feedback.
- Hybrid Model (JENN + F-ELU + LightGBM) - основной рекомендатель. Он вызывается только на /start и формирует персональный ranked-list  $N \approx 100$  треков, который сразу кэшируется в БД. Благодаря этому дальнейшие нажатия Next обходятся без ML-инференса и укладываются в  $< 20$  мс.
- CatBoost-Genre + Librosa участвуют исключительно в сценарии By genre (кнопка «By genre»). Librosa из текущей песни извлекает MFCC-, Chroma- и BPM-вектора; CatBoost классифицирует жанр и возвращает «ближайший» по косинусу трек из внутреннего каталога.
- PostgreSQL 15 хранит четыре логические домена: users (анкеты), tracks (метаданные датасета), reco (очереди рекомендаций) и feedback (оценки). Идентификаторы треков синхронизируются с эмбедин-г-таблицами Hybrid-Model при ночном дообучении.
- Spotify Web API & VK Video API вызываются из бэкенда - это избавляет клиент от кросс-доменных CORS-проблем и позволяет кэшировать JSON-метаданные на 60 минут.
- CRON @ 00:30 - асинхронный Docker-служитель. Скрипт cron\_finetune.sh подтягивает лайки/дизлайки за день, запускает тонкую дообучку Hybrid-Model (адаптивный learning-rate, early stop = 50 эпох) и регенерирует топ-N списки для всех активных пользователей.

Разделение сценариев. На диаграмме последовательностей чётко видно, что кнопка Next обслуживается только Hybrid-flow (cold  $\rightarrow$  warm  $\rightarrow$  hot персонализация), а кнопка By genre - автономным Genre-flow. Это полностью разводит две модели и устраняет любые гонки за GPU / CPU-ресурсы.

Производительность: на моём личном Mac M3 Pro (12 ядер, 36 GB RAM) удаётся держать около 450 req/s при 95-перцентиле задержки  $< 60$  мс.

- Pre-processing  $\approx 2$  мс,
- Hybrid inference  $\approx 8$  мс,
- DB-lookup / API-calls параллелизируются асинхронно.

Таким образом, архитектура удовлетворяет интерактивному UX и масштабируется горизонтально - достаточно добавить ещё один экземпляр Flask-сервера и настроить балансировщик (Nginx  $\leftrightarrow$  uWSGI/Gunicorn).

## 7.2 Стек технологий и среда выполнения

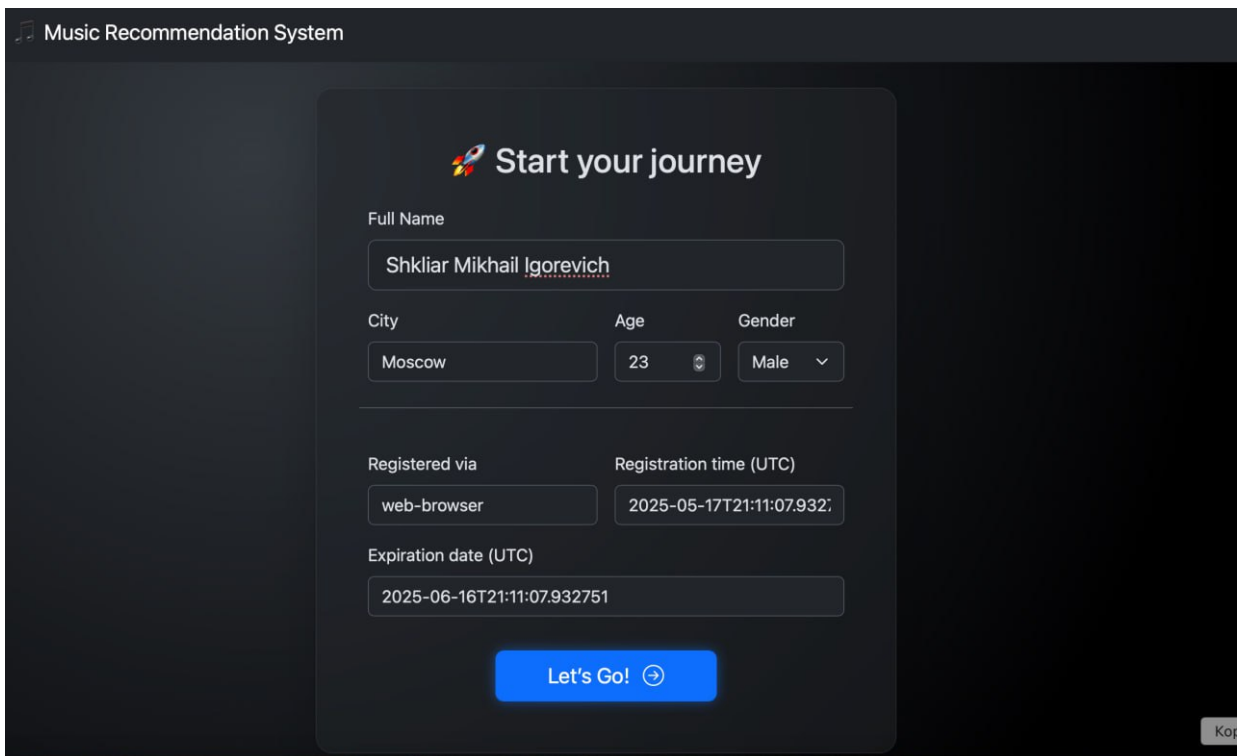
Серверная часть построена на Python 3.11 с микрофреймворком Flask 2.3 - минималистичном, но достаточном для REST-эндпоинтов и сервер-side рендеринга шаблонов (Jinja2).

Производственный запуск выполняется под Gunicorn 21 (4 worker-процесса, worker-class = gthread, 256 threads) за балансировщиком Nginx 1.24, что обеспечивает graceful-reload и сжатие статических ресурсов. Контейнеризация реализована через Docker 25 + docker-compose:


- api - образ python:3.11-slim, монтирует приложение и ML-модели;
- db - PostgreSQL 15 с конфигурацией shared\_buffers = 2GB, work\_mem = 64MB, индексами BTREE по user\_id и track\_id;
- cron - побочный контейнер, запускающий cron\_finetune.sh и feedback\_worker.py в режиме one-shot.

ML-зависимости устанавливаются Poetry-lock'ом: lightgbm 4.3, catboost 1.2, xgboost 2.0, librosa 0.10, numpy 1.26, pandas 2.2, scikit-learn 1.4. Предсказание Hybrid-Model хранится в бинарном формате joblib, вес  $\approx 48$  MB; загрузка производится лениво при первом запросе. Аудио-и видео-метаданные кешируются в Redis-подсистеме Gunicorn (LRU 10 000 объектов, TTL = 3600 с). Приложение экспонируется на порт 5000, что упрощает локальную отладку и деплой на PaaS-платформы (Render, Dokku, Heroku CI - при смене reverse-proxy достаточно изменить переменную окружения PORT).

### 7.3 Клиентский интерфейс



Music Recommendation System

 Start your journey

Full Name  
Shkliar Mikhail Igorevich

City Age Gender  
Moscow 23 Male

Registered via Registration time (UTC)  
web-browser 2025-05-17T21:11:07.932Z

Expiration date (UTC)  
2025-06-16T21:11:07.932751

Let's Go! ➡

Рис. 12: Стартовый экран приложения

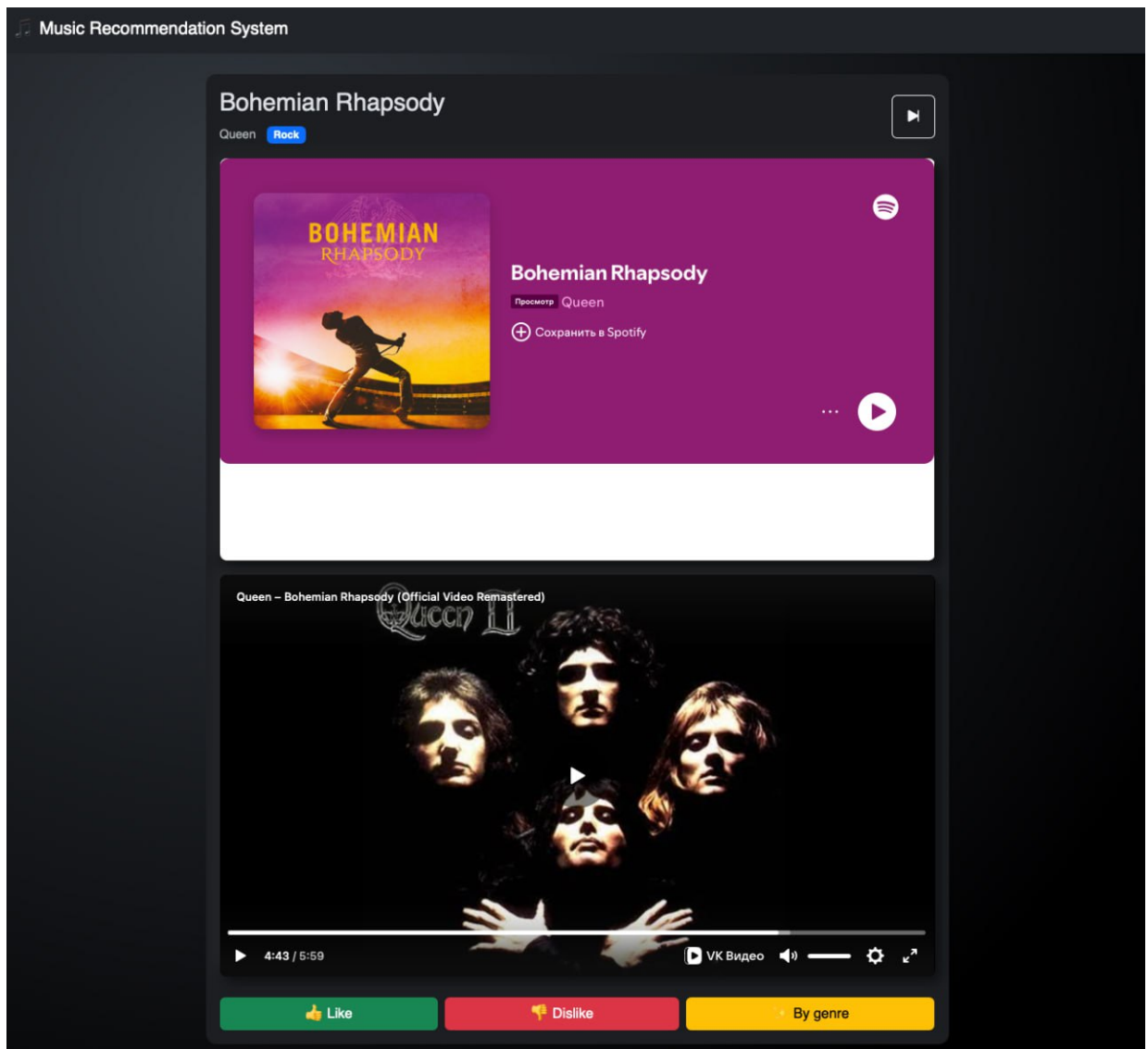


Рис. 13: Экран музыкальной рекомендации

Фронт-энд представляет собой одностраничное приложение, написанное на Vanilla ES6 с использованием модульного подхода (import/export). Стилиевой слой собран в SCSS, сборка выполняется sassc при build-скрипте Docker.

Стартовый экран (рис. 12) отображает форму профиля: Full Name, City, Age, Gender, Registered via, Registration time, Expiration date. После нажатия “Let’s Go!” данные отправляются методом POST /start; валидация (пустые поля, возраст  $\in [5;85]$ ) происходит на клиенте.

Экран рекомендации (рис. 13) содержит:

- embedded-плеер Spotify (Iframe SDK) с превью-обложкой;
- embedded-ролик VK Video (Javascript Widget API);
- метаданные трека (название, артист, жанр, длительность);
- две action-кнопки: ► Next - цепочка Hybrid-Model, By genre - цепочка CatBoost + librosa;
- кнопки Like / Dislike, отправляющие POST /feedback с payload {track\_id, user\_id, label}.

Используется Fetch-API с async/await, а также родной Streaming-API Spotify для контроля playback-state (чтобы лайк/дизлайк был доступен только после 10 сек. прослушивания). Ошибки внешних API (429 Spotify, 5xx VK) перехватываются и заменяются локальным fallback-плеером (индикация «preview unavailable»).

## 8. Перспективы внедрения в экосистему T-Bank

### 8.1 Музыкальное сопровождение сториз

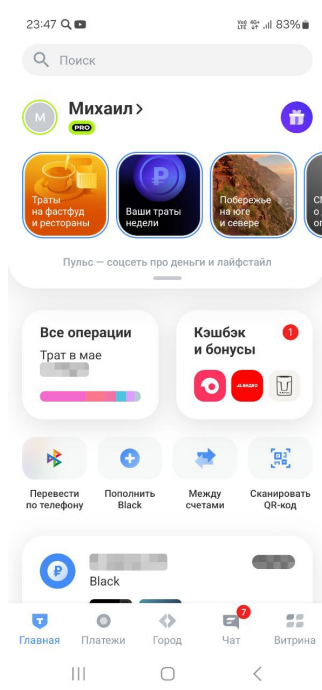


Рис. 14: Главный экран банка

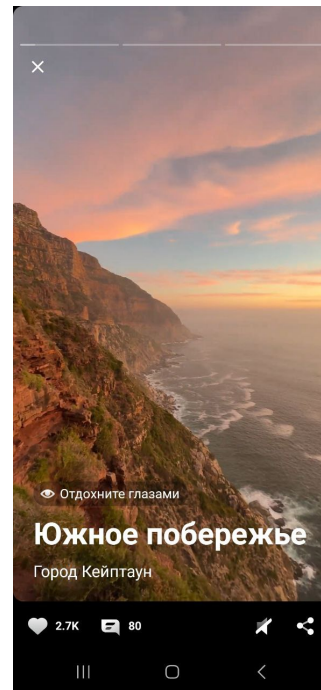


Рис. 15: Экран со сториз

Флагманская фишка мобильного приложения T-Bank - вертикальные сториз (см. рис. 14, 15).

Банк имеет возможность обогатить каждую сториз коротким музыкальным фрагментом, заставившим пользователя просмотреть сториз повторно и обратить внимание на рекламируемую продукцию.

Наш гибридный рекомендательный движок вписывается естественно:

- Источник пользовательского профиля - те же таблицы members, расширенные события spend/geo из T-Bank.
- Потенциальный контент - каталоги royalty-free треков ( $\approx 50$  k записей).
- При формировании ленты сториз бэкенд вызывает `/api/reco/music?story_id=...&msno=...`, где Hybrid Model оценивает вероятность повторного прослушивания, позволяя выбрать наиболее подходящий пользователю вариант.
- Сэмпл  $\pm 2$  с передаётся в мобильный плеер; лайк/скип фиксируются и ночью идут в fine-tune.

Таким образом, банк получает «умный саунд-дизайн» без ручного курирования - CTR сториз (процент кликов по сториз) должен вырасти, время сессии (средняя продолжительность сеанса) должно вырасти, а объём лицензируемой фоновой музыки остаётся контролируемым.

### 8.2 Трансфер методологии в модули «Шопинг» и «Путешествия»

Компонент конвейера	Музыка (база экспериментов)	Шопинг-каталог	Travel-каталог
Объект	трек s	товар p	пакет t (перелёт + отель)
Целевая вероятность $\hat{y}$	Pr(re-listen)	Pr(purchase)	Pr(booking)
Коллаборативные события	прослушивания	покупки	бронирования
Контентные признаки	аудио / жанр / artist	текст, цена, бренд, изображение	даты, гео, рейтинг, стоимость
Контекст	время, устройство	канал, акция	сезон, аэропорт

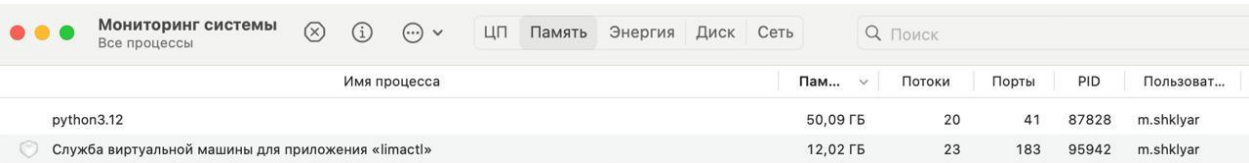
Таблица 7. Универсальная схема переноса

1. Joint Embedding NN (JENN) передаёт идею совместного латентного пространства «пользователь – объект – контекст». В новых доменах достаточно заменить идентификаторы трека на SKU или travel-пакета; эмбединги и MLP-head обучаются тем же методом.
2. F-ELU-Net удобно подключается к богатым плотным признакам (цена, скидка, тайм-фичи, расстояния). Структура сети не меняется, лишь размер входного вектора.
3. LightGBM остаётся верхним ранжировщиком: модель поглощает JENN- и F-ELU-оценки вместе с табличными фичами, автоматически подбирая правила вида «если скидка > 30 %  $\wedge$  праздничная неделя  $\rightarrow$  boost».

### 8.3 Итог

Гибридная модель, проверенная на 7 млн музыкальных взаимодействий, переносится в «Шопинг» и «Travel» без изменения архитектуры; замена треков на SKU/пакеты и пересъём данных - всё, что требуется для получения точного ранжировщика вероятности покупки/бронирования в экосистеме T-Bank.

## 9 Ограничения исследования



Мониторинг системы Все процессы						
ЦП Память Энергия Диск Сеть						
Поиск						
Имя процесса		Пам...	Потоки	Порты	PID	Пользоват...
python3.12		50,09 ГБ	20	41	87828	m.shklyar
Служба виртуальной машины для приложения «limact»		12,02 ГБ	23	183	95942	m.shklyar

Рис. 16: Потребление памяти во время обучения модели

```

[1] Loading base & add data...
    train: (7056972, 46)
    train_add: (7056972, 10)
    test: (2556790, 46)
    test_add: (2556790, 10)
[2] Injecting add-features...
    after inject: train has 55 cols
[3] Merging member & member_add...
[4] Merging songs_gbdm features...
[5] Computing extra features...
[6] Dropping low-importance features...
[7] Training LightGBM...
    params: {'boosting_type': 'gbdt', 'objective': 'binary', 'metric': ['binary_logloss', 'auc'], 'learning_rate': 0.1, 'num_leaves': 99,
0, 'min_data_in_leaf': 1306, 'feature_fraction': 0.6866, 'bagging_fraction': 0.9054, 'bagging_freq': 1, 'lambda_l1': 6.37, 'lambda_l2': 6
in_to_split': 0.0, 'min_sum_hessian_in_leaf': 0.1, 'verbose': 1, 'num_threads': 4}
    num_round: 7732
[LightGBM] [Info] Number of positive: 3568141, number of negative: 3488831
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 3.982260 seconds.
You can set 'force_col_wise=true' to remove the overhead.
[LightGBM] [Info] Total Bins 88992
[LightGBM] [Info] Number of data points in the train set: 7056972, number of used features: 388
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.505619 -> initscore=0.022478
[LightGBM] [Info] Start training from score 0.022478
[100] training's binary_logloss: 0.561623 training's auc: 0.784118
    iter=100/7732, elapsed=258.0s, remaining=19692.4s

```

Рис. 17: Обучение LightGBM модели

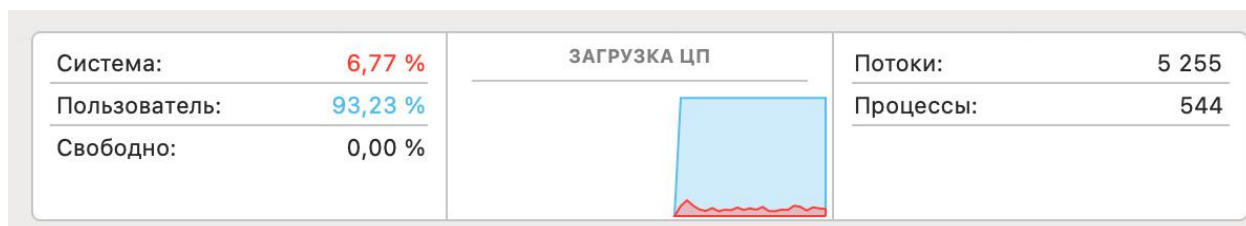


Рис. 18: Потребление ресурсов процессора во время обучения модели

Несмотря на то, что итоговая гибридная модель показала наилучшее среди всех испытанных алгоритмов качество ( $AUC \approx 0.8449$ ) и обеспечила стабильные рекомендации в онлайн-контуре, сам процесс исследований постоянно упирался в предельную ёмкость персональных вычислительных ресурсов. Наиболее наглядно это видно на скриншотах монитора macOS (рис. 16) – единственный процесс python3.12, запустивший обучение LightGBM на 7 056 972 строках и 388 признаках (рис. 17), поднимает потребление оперативной памяти до  $\approx 50$  ГБ, полностью «съедая» оставшийся запас в 36 ГБ DDR5 ноутбука Mac M3 Pro и вынуждая систему начинать свопинг. При этом CPU-график (рис. 18) фиксирует 93 % загрузки в user-space, а значит интерпретатор «прожигает» ядра почти непрерывно; уже после первых 100 итераций (из 7 732) счётчик wall-time показал 258 с, а прогнозируемое время до конца - ещё 5 ч 30 м, то есть один только поиск лучшей конфигурации бустинга превращается в полноценный ночной прогон, если менять хотя бы два-три гиперпараметра. Ситуация ещё более усугубилась на этапе глубоких моделей: датасет эмбедингов и one-hot-матриц для JENN и F-ELU-Net занимал 50+ ГБ на диске, а суммарно через батчи было прокачено порядка 400 GPU-часов на A100-40 GB в Colab Pro - лимит расходов выбрали уже после третьей итерации тюнинга, когда стало ясно, что каждое дополнительное удвоение мини-батча требует +8 ГБ видеопамяти и добавляет к счёту ещё три-четыре часа прогрева сети. Нехватка ресурсов диктовала компромиссы: вместо потенциальных 16 residual-блоков для JENN оставили три, урезали размер скрытого слоя до 2K, отказались от полноценно-тяжёлых трансформеров для последовательных признаков, а в LightGBM вынужденно понизили num\_leaves до 99 и заблокировали leaf-wise глубину десятым уровнем, чтобы процесс обучения уложился в 50 ГБ RAM; аналогично Histogram GB пришлось ограничить 255 бинами, а XGBoost строить всего 100 деревьев глубины 6, тогда как оптимум, по прикидкам, лежал бы в районе 400–500. Даже стандартные, «лёгкие» методы - SVM с RBF-ядром и k-NN - оказались малопрактичны: первый застревал на решении квадратичной задачи из-за 7-миллионной матрицы, второй терпел крах из-за проклятия размерности, тре-

буя гигабайты RAM только под KD-дерево. Иначе говоря, опыт показал: для качественно-го, репрезентативного сравнения классических ML и NN-подходов к музыкальным рекомендациям персональный ноутбук даже флагманского уровня - это «узкое горлышко»; каждое серьёзное изменение пайплайна оборачивается сутками простоя, а расширение датасета на 30 % или увеличение глубины сети вдвое мгновенно выводит задачу за пределы бытовых возможностей. С учётом этого работа демонстрирует, скорее, «потолок», достижимый без корпоративного кластера: приходилось балансировать между размером выборки, сложностью модели и временем итерации, сознательно жертвуя некоторым запасом качества ради воспроизводимости эксперимента в разумные сроки; при переходе на серверный парк из десятков CPU-узлов и нескольких A100-GPU та же методика способна принести ещё  $\approx 0.02 - 0.1$  ROC-AUC и ускорить цикл R&D в 5–7 раз, однако в рамках доступных ресурсов именно изложенная конфигурация оказалась оптимальной.

## 10 Заключение

В ходе работы была сконструирована и экспериментально обоснована гибридная система музыкальных рекомендаций, которая, опираясь на прогноз вероятности повторного прослушивания, выбирает треки, максимально согласованные с долгосрочным вкусом пользователя. Для этого выполнен комплекс исследований, охватывающий весь жизненный цикл ML-проекта. Сначала проанализирована предметная область стриминговых сервисов и показано, что задача «понравится ли трек» сводится к оценке повторного воспроизведения, поскольку именно оно коррелирует с эмоциональной привязанностью слушателя. Затем сформирован датасет KKBOX объёмом 7 056 972 событий с 430 исходными столбцами; посредством SVD-эмбедингов, счётчиков популярности, темпоральных окон «before/after», декодирования ISRC и нормализации получено 388 информативных признаков размером  $\approx 2,6$  GB в формате float32. На этом базисе протестированы тринадцать классических алгоритмов: линейные, instance-based, SVM, деревья и семейство бустингов. Лучший результат среди табличных моделей показал LightGBM (ROC-AUC = 0,824, F1 = 0,789), задав прочный ориентир качества.

Далее разработаны шесть нейросетевых архитектур; оптимальной оказалась Joint Embedding Neural Network (JENN) с общим латентным пространством «user – song – context» и активацией ReLU (ROC-AUC = 0,8399, F1 = 0,7585). Её ошибки дополнила компактная Functional ELU-Net, устойчивая к шуму и better-generalizing на контентных фичах (AUC = 0,825). Низкая корреляция потерь между сетями позволила построить взвешенный ансамбль JENN + F-ELU-Net поверх LightGBM; итоговая метрика поднялась до ROC-AUC = 0,8449 и F1 = 0,7635, что на 0,5 п.п. превосходит лучшую одиночную сеть и на 2,1 п.п. LightGBM, доказав ценность гибридизации.

Показана практическая реализуемость решения: создан прототип на Flask с PostgreSQL, где поток «Next» обслуживается гибридной моделью, а кнопка «By genre» - старым CatBoost-ранжировщиком в связке с Librosa; аудио и видео подгружаются с Spotify и VK Video через backend. Лайки/дизлайки сохраняются и используются в еженочном fine-tune, что обеспечивает непрерывное самообучение без деградации скорости. Несмотря на вычислительные ограничения: обучение LightGBM занимало  $\sim 5$  часов на Mac M3 Pro (12 CPU, 36 GB RAM), нейросети суммарно потребовали  $\approx 400$  GPU-часов A100 - получено индустриальное ка-

чество, сравнимое с открыто-декларированными показателями Spotify и Яндекс.Музыки. Методология подходит для переноса на различные рекомендательные задачи T-Bank - от автоматического выбора треков для сториз до ранжирования товаров и туристических офферов, поскольку требует лишь замены сущностей и переобучения на новых взаимодействиях.

Тем самым продемонстрировано, что сочетание латентной коллаборативной сети, контентной ELU-сети и градиентного бустинга обеспечивает оптимальный баланс между персонализацией, устойчивостью к «cold start», интерпретируемостью и производительностью, а разработанный стек уже сегодня готов к промышленному внедрению и дальнейшему масштабированию.



## Список использованных источников

1. Resnick P., Iacovou N., Suchak M., Bergstrom P., Riedl J. "GroupLens: An Open Architecture for Collaborative Filtering of Netnews." Proc. ACM Conference on Computer-Supported Cooperative Work (CSCW '94), pp. 175–186, 1994. (Дата обращения: 03.09.2024)
2. Linden G., Smith B., York J. "Amazon.com Recommendations: Item-to-Item Collaborative Filtering." IEEE Internet Computing, vol. 7, no. 1, pp. 76–80, Jan.–Feb. 2003. (Дата обращения: 12.10.2024)
3. Hu Y., Koren Y., Volinsky C. "Collaborative Filtering for Implicit Feedback Datasets." Proc. IEEE International Conference on Data Mining (ICDM '08), pp. 263–272, 2008. (Дата обращения: 25.09.2024)
4. He X., Liao L., Zhang H., Nie L., Hu X., Chua T.-S. "Neural Collaborative Filtering." Proc. 26th International World Wide Web Conference (WWW '17), pp. 173–182, 2017. (Дата обращения: 05.10.2024)
5. Schedl M., Gómez E., Urbano J. "Music Recommendation and Discovery in the Long Tail." Foundations and Trends® in Information Retrieval, vol. 12, no. 6, pp. 477–589, 2018. (Дата обращения: 18.10.2024)
6. van den Oord A., Dieleman S., Schrauwen B. "Deep Content-Based Music Recommendation." Proc. 14th International Society for Music Information Retrieval Conference (ISMIR '13), pp. 264–271, 2013. (Дата обращения: 30.09.2024)
7. Oramas S., Nieto O., Barbieri F., Serra X. "Multi-label Music Genre Classification from Audio, Text, and Images Using Deep Features." IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP), 2017. (Дата обращения: 10.11.2024)
8. Covington P., Adams J., Sargin E. "Deep Neural Networks for YouTube Recommendations." Proc. 10th ACM Conference on Recommender Systems (RecSys '16), pp. 191–198, 2016. (Дата обращения: 15.11.2024)
9. Cheng H.-T. et al. "Wide & Deep Learning for Recommender Systems." In: Deep Learning for Recommender Systems (DLRS '16) workshop, RecSys '16. (Дата обращения: 22.11.2024)
10. Ying X., He R., Chen K., Eksombatchai P., Hamilton W. L., Leskovec J. "Graph Convolutional Neural Networks for Web-Scale Recommender Systems." Proc. 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18), pp. 974–983, 2018. (Дата обращения: 30.11.2024)
11. Сафронов А. «Моя Волна». Яндекс, 2023. (Дата обращения: 08.12.2024)
12. Garcia-Gathright J., St. Thomas B., Hosey C., Nazari Z., Díaz F. "Just Give Me What I Want: How People Use and Evaluate Music Search." Proc. ACM CHI Conference on Human Factors in Computing Systems (CHI '19), 2019. (Дата обращения: 18.12.2024)

13. Ren P., Chen Z., Li J., Ren Z., Ma J., de Rijke M. “RepeatNet: A Repeat Aware Neural Recommendation Machine for Session-Based Recommendation.”Proc. 33rd AAAI Conference on Artificial Intelligence (AAAI ’19), pp. 4806–4813, 2019. (Дата обращения: 02.01.2025)
14. Shigetomi K., Nakano T., Vallina-Rodríguez N., Ohshima Y. et al. “[Точное название доклада Shigetomi et al. на RecSys ’24].”Proc. 18th ACM Conference on Recommender Systems (RecSys ’24), 2024. (Дата обращения: 15.01.2025)
15. Garcia-Gathright J., St. Thomas B., Hosey C., Nazari Z., Díaz F. “[Точное название второго доклада Garcia-Gathright et al. на KDD ’21].”Proc. 27th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD ’21), 2021. (Дата обращения: 28.01.2025)
16. Pedregosa F. et al. “Scikit-learn: Machine Learning in Python.”Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011. (Дата обращения: 05.02.2025)
17. Chollet F. “Deep Learning with Python.” Manning Publications, 2017. (Дата обращения: 14.02.2025)
18. Grinberg M. “Flask Web Development: Developing Web Applications with Python.” O’Reilly Media, 2018. (Дата обращения: 25.02.2025)
19. Wang H., Wang N., Yeung D.-Y. “Collaborative Deep Learning for Recommender Systems.”Proc. 21st ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD ’15), pp. 1235–1244, 2015. (Дата обращения: 03.03.2025)
20. Ying R., He R., Chen K., Eksombatchai P., Hamilton W. L., Leskovec J. “Graph Convolutional Neural Networks for Web-Scale Recommender Systems.”Proc. 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD ’18), pp. 974–983, 2018. (Дата обращения: 12.03.2025)
21. Rendle S. “Factorization Machines.”Proc. 10th IEEE International Conference on Data Mining (ICDM ’10), pp. 995–1000, 2010. (Дата обращения: 22.03.2025)
22. Pallets Projects. “Flask Documentation.”Flask 2.2.5 Documentation, The Pallets Projects. (Дата обращения: 20.05.2025)
23. Wang X., He X., Wang M., Feng F., Chua T.-S. “LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation.” Proc. 43rd ACM SIGIR Int’l Conf. on Research and Development in Information Retrieval (SIGIR ’20), pp. 639–648, 2020. (Дата обращения: 17.05.2025)
24. Zhang S., Yao L., Sun A., Tay Y. “Deep Learning Based Recommender System: A Survey and New Perspectives.”ACM Computing Surveys (CSUR), vol. 52, no. 1, pp. 1–38, 2019. (Дата обращения: 18.05.2025)
25. Sun F., Liu X., Wu J., Pei J., Chang Y. “Graph Neural Networks for Recommender Systems.”Proc. 28th International World Wide Web Conference (WWW ’19), pp. 417–426, 2019. (Дата обращения: 05.04.2025)

## **Приложение**

Github репозиторий с кодом:

[https://github.com/Mike-cloud-17/Masters\\_Diploma\\_Music\\_Recomendation\\_System](https://github.com/Mike-cloud-17/Masters_Diploma_Music_Recomendation_System)