

Ранжирование и рекомендательные системы

Ранжирование

Пример

Яндекс

машинное обучение

Х Найти

ПОИСК КАРТИНКИ ВИДЕО КАРТЫ МАРКЕТ НОВОСТИ ПЕРЕВОДЧИК ЕЩЁ

w [Машинное обучение — Википедия](#)
ru.wikipedia.org > Машинное обучение ▾
Машинное обучение (англ. Machine Learning) — класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи...

l [Что такое машинное обучение и почему оно может...](#)
lifehacker.ru > Лайфхакер > ...-mashinnoe-obuchenie ▾
Машинное обучение избавляет программиста от необходимости подробно объяснять компьютеру, как именно решать проблему.

o [Курс «Машинное обучение» 2014 - YouTube](#)
youtube.com > playlist?list=..._b9zqEQiiBtC ▾
Курс "Машинное обучение" является одним из основных курсов Школы, поэтому он является обязательным для всех студентов ШАД.

P [Машинист электропоезда - обучение | Про профессии.ру](#)
propof.ru > Машинист электропоезда ▾
Машинист электропоезда - обучение. И метрополитен, и РЖД приглашают на обучение в собственные учебно-производственные центры.

• [Обучение - машина - Большая Энциклопедия Нефти...](#)
ngpedia.ru > id201843p1.html ▾
После обучения машины или в ходе его, смотря по алгоритму, проводится прогнозирование новых катализаторов...

Пример

Яндекс

машинае обучение

X

Найти

ПОИСК КАРТИНКИ ВИДЕО КАРТЫ МАРКЕТ НОВОСТИ ПЕРЕВОДЧИК ЕЩЁ

w **Машинное обучение — Википедия**

[ru.wikipedia.org](#) > Машинное обучение ▾

Машинное обучение (англ. Machine Learning) — класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи...

l **Что такое машинное обучение и почему оно может...**

[lifehacker.ru](#) > Лайфхакер > ...-mashinnoe-obuchenie ▾

Машинное обучение избавляет программиста от необходимости подробно объяснять компьютеру, как именно решать проблему.

■ **Курс «Машинное обучение» 2014 - YouTube**

[youtube.com](#) > playlist?list=..._b9zqEQiiBtC ▾

Курс "Машинное обучение" является одним из основных курсов Школы, поэтому он является обязательным для всех студентов ШАД.

P **Машинист электропоезда - обучение | Про профессии.ru**

[proprof.ru](#) > Машинист электропоезда ▾

Машинист электропоезда - обучение. И метрополитен, и РЖД приглашают на обучение в собственные учебно-производственные центры.

■ **Обучение - машина - Большая Энциклопедия Нефти...**

[ngpedia.ru](#) > id201843p1.html ▾

После обучения машины или в ходе его, смотря по алгоритму, проводится прогнозирование новых катализаторов...

Ранжирование

- Дан набор запросов $\{q_1, \dots, q_m\}$
- Дан набор документов $\{d_1, \dots, d_n\}$
- Нужно для каждого запроса правильно упорядочить документы
- Что такое «правильно»?

Ранжирование

- Дан набор запросов $\{q_1, \dots, q_m\}$
- Дан набор документов $\{d_1, \dots, d_n\}$
- Рассматриваем пары «запрос-документ» (q, d)
- Для некоторых троек (q, d_1, d_2) известно, что для запроса q документ d_1 должен стоять раньше, чем d_2
- Обозначение: R — множество троек (q, d_1, d_2) , для которых известен такой порядок

Ранжирование

- Раньше: строим модель $a(x)$, которая приближает ответы
- Сейчас: строим модель $a(q, d)$, которая правильно упорядочивает документы для запросов

$$(q, d_1, d_2) \in R \Rightarrow a(q, d_1) > a(q, d_2)$$

Пример

- Для запроса q известны пары $(d_3, d_1), (d_3, d_2), (d_1, d_4)$
- Какие наборы прогнозов модели лучше?
 - $(3, 2, 4, 1)$
 - $(2, 3, 4, 1)$
 - $(3, 4, 2, 1)$
 - $(13, 10, 20, 7)$

Пример

- Для запроса q известны пары $(d_3, d_1), (d_3, d_2), (d_1, d_4)$
- Какие наборы прогнозов модели лучше?
- **(3, 2, 4, 1)**
- **(2, 3, 4, 1)**
- **(3, 4, 2, 1)**
- **(13, 10, 20, 7)**
- Важен порядок, а не абсолютные значения!

Методы ранжирования

Целевая переменная

- Объекты — пары «запрос-документ» $x_i = (q, d)$
- Ответы — числа y_i
- Требование — если есть объекты (q, d_1) и (q, d_2) , такие что $y_1 > y_2$, то должно быть $a(q, d_1) > a(q, d_2)$

Целевая переменная, пример

- $(q_1, d_1), 1$
- $(q_1, d_2), 0.7$
- $(q_1, d_3), 0$
- $(q_2, d_1), 0$
- $(q_2, d_2), 1$
- Для q_1 должны получить ранжирование (d_1, d_2, d_3)
- Для q_2 должны получить ранжирование (d_2, d_1)

Поточечный (pointwise) подход

- Обучим модель $a(q, d)$, чтобы она как можно точнее приближала ответы y_i
- Например, линейная регрессия:

$$\sum_{(q,d,y) \in R} (\langle w, x(q, d) \rangle - y_i)^2 \rightarrow \min_w$$

- $x(q, d)$ — признаки для пары «запрос-документ»

Поточечный (pointwise) подход

- Простой в реализации
- Можно использовать любую из известных моделей (линейные, деревья, случайные леса, нейронные сети...)
- Восстанавливает точные значения y_i , хотя нас интересует порядок

Попарный (pairwise) подход

- В ранжировании требуется правильно располагать пары документов — формализуем лосс

$$\sum_{(q, d_i, d_j) \in R} [a(q, d_i) - a(q, d_j) < 0]$$

- Штрафуем, если второй документ из пары оказался раньше

Попарный (pairwise) подход

- Получили разрывный функционал — сложно оптимизировать
- Перейдём к гладкой верхней оценке (как в линейных классификаторах):

$$\sum_{(q, d_i, d_j) \in R} [a(q, x_i) - a(q, x_j) < 0] \leq \sum_{(q, d_i, d_j) \in R} L(a(q, x_i) - a(q, x_j))$$

- Пример: $L(z) = \log(1 + e^{-z})$

Попарный (pairwise) подход

- Сложнее поточечного (больше слагаемых в функционале)
- Обычно даёт качество выше, чем поточечный
- Реализации: SVM^{light}, xgboost (rank:pairwise)

Задачи ранжирования

- Поиск
- Рекомендации
- Автодополнение

Рекомендательные системы

Рекомендательные системы

- С какими рекомендательными системами вы встречались?

Рекомендательные системы

- Фильмы, видео
- Музыка
- Книги
- Приложения
- Товары
- Посты в социальных сетях
- Баннерные системы
- Люди (социальные сети, сервисы знакомств)
- Услуги (рестораны, отели, ...)
- Научные публикации

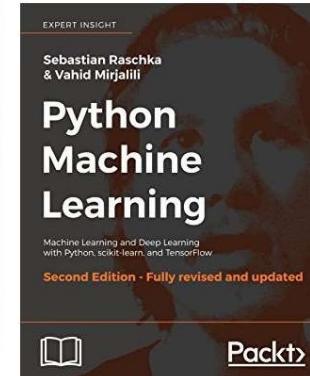
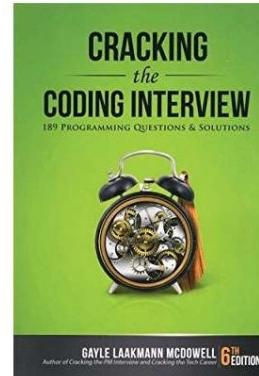
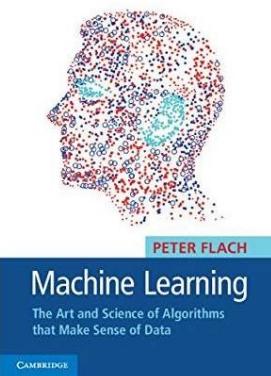
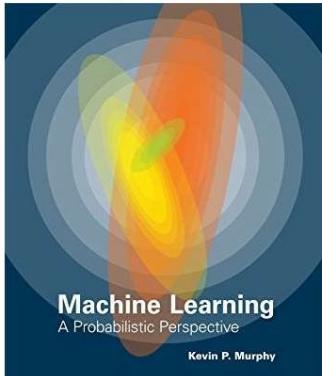
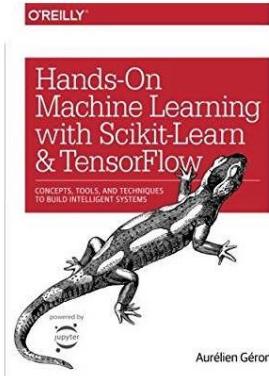


Рекомендательные системы

- Рекомендательные системы сокращают объём информации, необходимый для принятия решения
- Не нужно читать отзывы на 1000 фильмов — модель сама выберет лучший
- Netflix: 80% просмотренных фильмов найдены через рекомендательную систему
- Amazon: 35% продаж через полки рекомендаций
- Youtube: 60% просмотров благодаря рекомендациям

Amazon

Recommendations for you in Books



Amazon

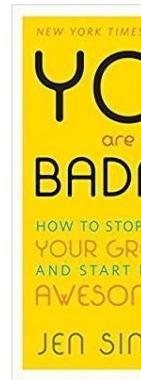
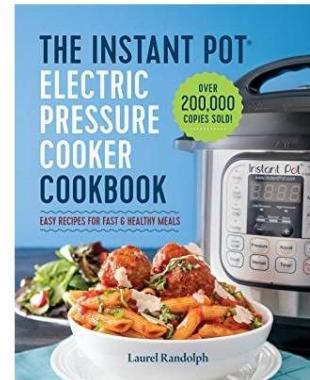
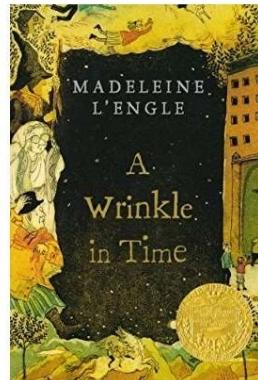
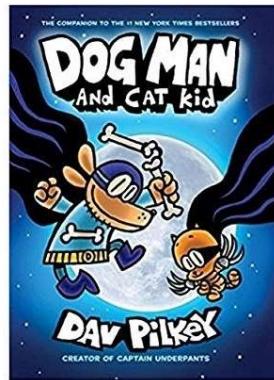
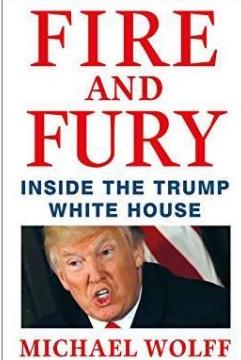
Books best sellers [See more](#)

JORDAN B.
PETERSON

12 RULES
FOR LIFE
AN ANTIDOTE TO CHAOS

"One of the most important thinkers to emerge on
the world stage for many years." —THE SPECTATOR

FOREWORD BY NORMAN DOIDGE



Netflix



Netflix

Profile Type	Score Image A	Score Image B
Comedy	5.7	6.3
Romance	7.2	6.5



Image A



Image B

Рекомендации контента

- Медийный бум приводит к взрывному росту объёмов информации в сети
 - Рекомендательные системы помогают ориентироваться
 - Для авторов — поиск целевой аудитории
-
- Youtube: 800M загруженных видео и 2600M активных пользователей

Цели с точки зрения продавца

Цели с точки зрения продавца

- Продать больше товаров
- Продать больше редких товаров
- Повысить лояльность пользователя
- Лучше понять покупателей

Цели с точки зрения покупателя

Цели с точки зрения покупателя

- Купить то, что нужно
- Понять, что покупать вместе с данным товаром
- Понять, что интересно (если нет задачи купить что-то конкретное)

Netflix Prize



Netflix Prize

Соревнование проводилось почти три года, с 2 октября 2006 по 26 июня 2009 года.

Основная цель - превзойти модель Netflix на 10% по RMSE на данных о пользовательских рейтингах фильмов

Призы - 1 миллион долларов за первое место.

Netflix Prize



Leaderboard

Showing Test Score. [Click here to show quiz score](#)

Display top leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace	0.8612	9.59	2009-07-24 17:18:43

Netflix Prize

Краткие итоги:

- “Бум” в исследовательской среде. Многие методы из этого соревнования до сих пор используются
- В “прод” ушла модель не с первого места. Классика kaggle vs prod
- Ошибка с метрикой. Модели в итоге использовались для ранжирования фильмов, а не для восстановления рейтингов как таковых.

Виды рекомендательных систем

Какие объекты мы хотим рекомендовать? Наиболее подходящие.

Но что такое наиболее подходящий?

Варианты:

- Популярное
 - глобально
 - по каким-то категориям (например по жанрам фильмов)
 - у каких-то групп (например по возрасту)
- Похожий объект на то, с чем взаимодействовал пользователь
- То, с чем взаимодействуют похожие люди

Виды рекомендательных систем

Верхнеуровнево можно выделить два подхода:

- Не персонализированные
- Персонализированные

Персонализированные подходы можно разделить на следующие:

- Content-based filtering
- Collaborative filtering
- Hybrid

Персонализированные рекомендательные системы

Основная идея - использовать или получить некое векторное представление пользователя и объекта.

Функция оценки - функция считающая оценку на основе векторных представлений пользователя и объекта. Типичные примеры:

- Скалярное произведение
- Косинусное сходство
- Евклидово расстояние

Процесс построения рекомендаций - процесс подбора объектов с наиболее высокой оценкой для данного пользователя.

Content-based filtering

Основная идея - использовать характеристики объекта для поиска похожих объектов.

Функция оценки - обычно скалярное произведение или косинусное расстояние

Процесс построения рекомендаций - ищем наиболее похожие на те объекты, с которыми пользователь уже взаимодействовал.



Collaborative filtering

Основная идея - использовать историю взаимодействий пользователей с объектами для получения векторных представлений.

Функция оценки - обычно скалярное произведение или косинусное расстояние

Два базовых подхода:

- Neighbour-based (Memory-based)
- Model-based

	1	2	3	4	5	6
a	+	?	-	-	?	-
b		-		+		+
c	+	+		-	-	-
d			+	+	-	
e	-		-		+	+

Neighbour-based collaborative filtering

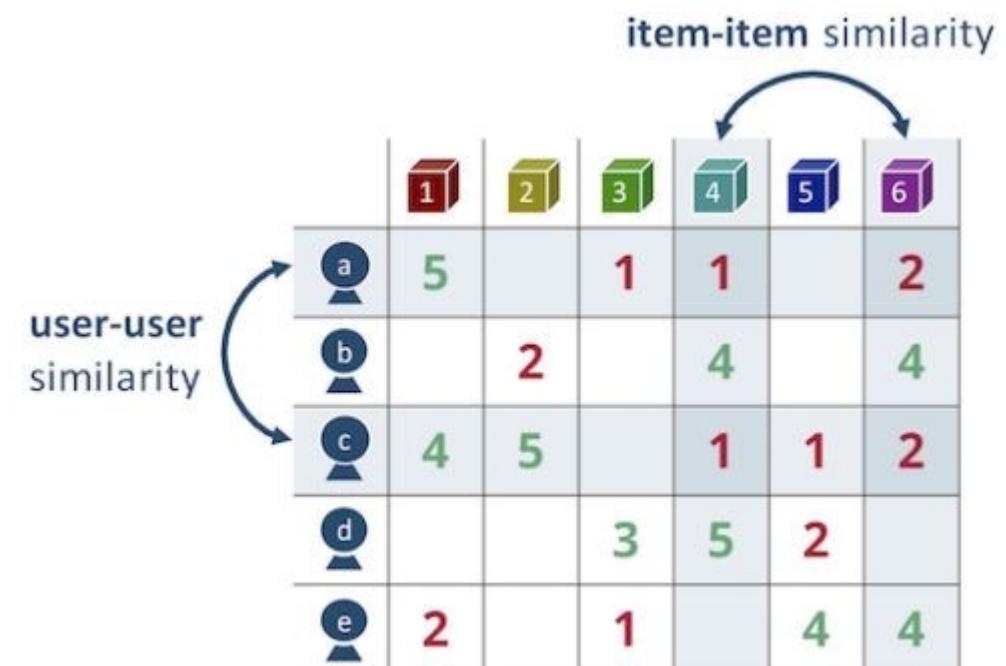
Основная идея - использовать строки или столбцы из матрицы оценок как векторное представление пользователя или объекта.

Два подхода:

- Item-item - матрица схожести объектов
- User-user - матрица схожести пользователей

Как использовать:

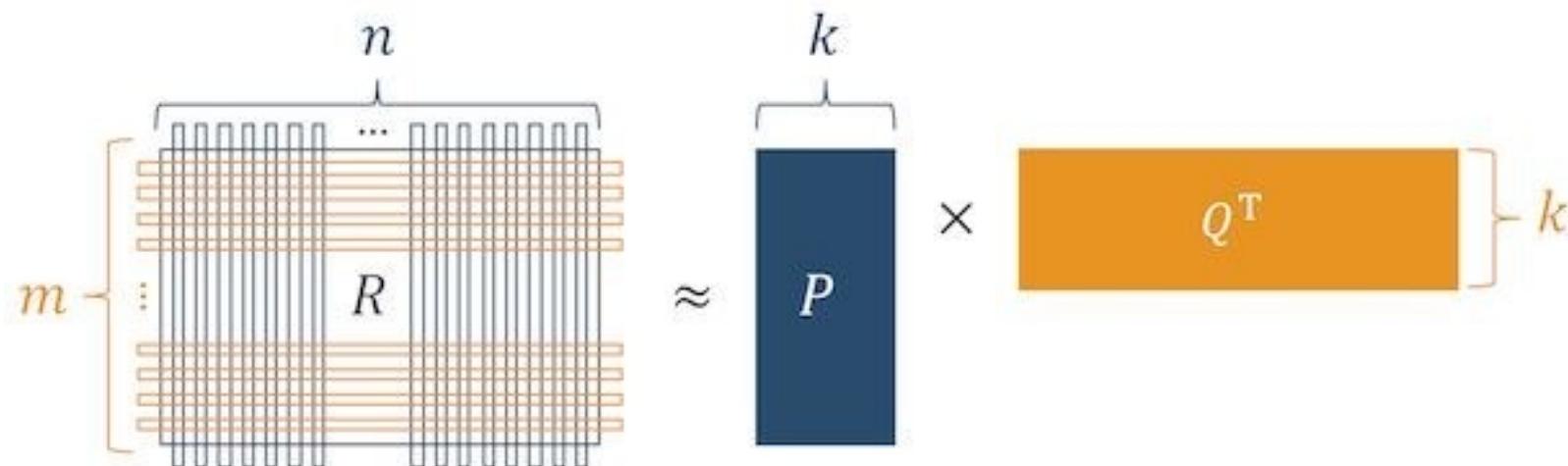
- Найти похожие объекты на то, с чем пользователь взаимодействовал
- Рекомендовать объекты из тех, с которыми взаимодействовали похожие пользователи



Model-based collaborative filtering

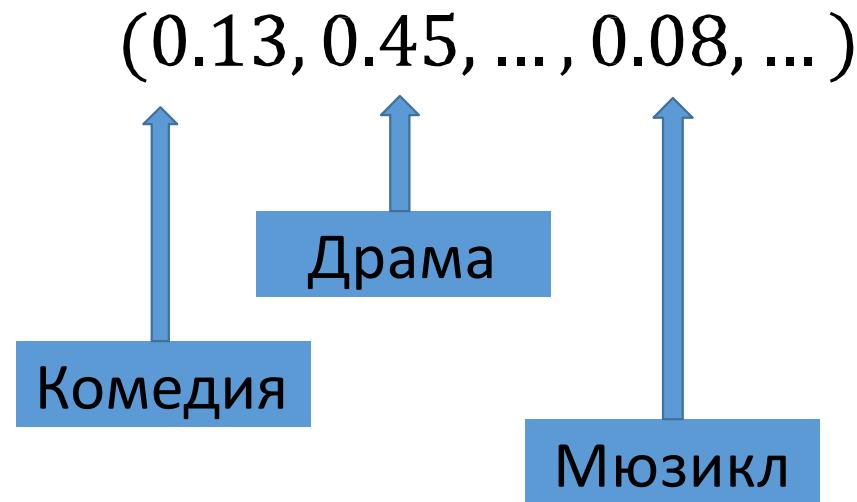
Основная идея - построить внутренние векторные представления для пользователей и объектов на основе матрицы оценок.

Основной подход - матричные разложения.



Векторы интересов

- Для пользователя — насколько он интересуется каждым жанром
- Для фильма — насколько он относится к каждому жанру



Рейтинг

- Предположение: заинтересованность определяется как скалярное произведение векторов пользователя и фильма

$$(0.1, 0.5, 0.01, 0.92) \times (0, 0, 0.1, 0.95) = 0.875$$

$$(0.1, 0.5, 0.01, 0.92) \times (0.9, 0, 0, 0.1) = 0.182$$

Пользователь

Фильм

Матрица рейтингов

	Пила	Улица Вязов	Ванильное небо	1+1
Маша	5	4	1	2
Юля	5	5	2	
Вова			3	5
Коля	3		4	5
Петя				4
Ваня		5	3	3

Матрица рейтингов

j

	Пила	Улица Вязов	Ванильное небо	1+1
Masha	5	4	1	2
Юля	5	5	2	
Вова			3	5
Kоля	3		4	5
Петя				4
Ваня		5	3	3

i

Матрица рейтингов

j

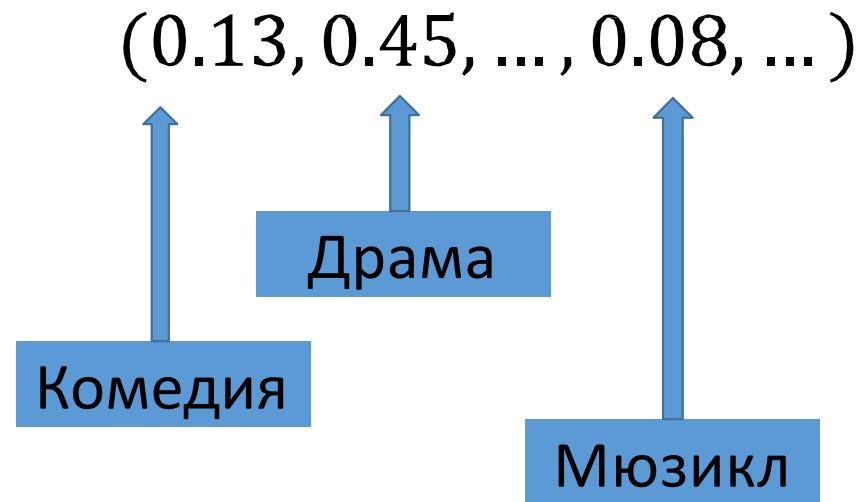
	Пила	Улица Вязов	Ванильное небо	1+1
Masha	5	4	1	2
Юля	5	5	2	
Вова			3	5
Kоля	3		4	5
Петя				4
Ваня		5	3	3

$$x_{ij} \approx \langle u_i, v_j \rangle$$

u_i - «интересы пользователей»
 v_j - «параметры фильмов»

Векторы интересов

- Для пользователя — насколько он интересуется каждым жанром
- Для фильма — насколько он относится к каждому жанру



Рейтинг

- Предположение: заинтересованность определяется как скалярное произведение векторов пользователя и фильма

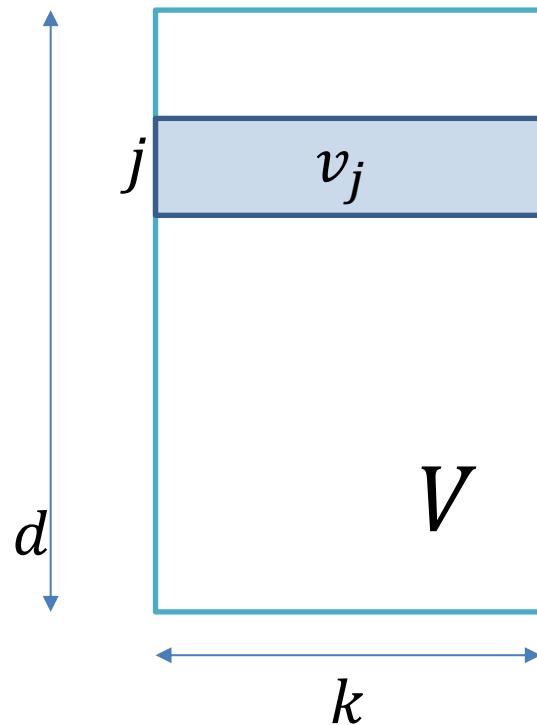
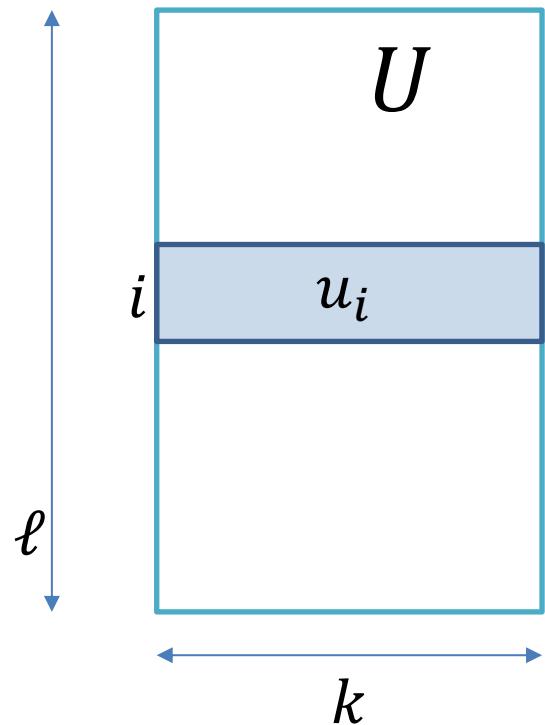
$$(0.1, 0.5, 0.01, 0.92) \times (0, 0, 0.1, 0.95) = 0.875$$

$$(0.1, 0.5, 0.01, 0.92) \times (0.9, 0, 0, 0.1) = 0.182$$

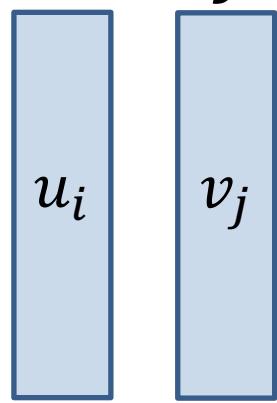
Пользователь

Фильм

Немного про обозначения



$$x_{ij} \approx \langle u_i, v_j \rangle$$



$$x_{ij} \approx {u_i}^T v_j$$

Оптимизационная задача

$$Q = \sum_{i,j} (\langle u_i, v_j \rangle - x_{ij})^2 \rightarrow \min_{u_i, v_j}$$

Градиентный спуск (GD)

$$Q = \sum_{i,j} (\langle u_i, v_j \rangle - x_{ij})^2 \rightarrow \min_{u_i, v_j}$$

$$\begin{aligned} \frac{\partial Q}{\partial u_i} &= \sum_{\tilde{i},j} \frac{\partial}{\partial u_i} (\langle u_i, v_j \rangle - x_{\tilde{i}j})^2 = \sum_j 2(\langle u_i, v_j \rangle - x_{ij}) \frac{\partial \langle u_i, v_j \rangle}{\partial u_i} = \\ &= \sum_j 2(\langle u_i, v_j \rangle - x_{ij}) v_j \quad \varepsilon_{ij} = (\langle u_i, v_j \rangle - x_{ij}) - \text{ошибка на } x_{ij} \end{aligned}$$

$$u_i^{(t+1)} = u_i^{(t)} - \gamma_t \sum_j \varepsilon_{ij} v_j$$

Стохастический градиентный спуск (SGD)

GD:

$$u_i^{(t+1)} = u_i^{(t)} - \gamma_t \sum_j \varepsilon_{ij} v_j$$

$$v_j^{(t+1)} = v_j^{(t)} - \eta_t \sum_i \varepsilon_{ij} u_i$$

SGD:

$$u_i^{(t+1)} = u_i^{(t)} - \gamma_t \varepsilon_{ij} v_j$$

$$v_j^{(t+1)} = v_j^{(t)} - \eta_t \varepsilon_{ij} u_i$$



Для случайных i, j

Плюсы и минусы SGD

- + Простота реализации
- + Сходимость
- Медленно сходится
- Сложность выбора шага градиентного спуска (γ_t и η_t)
- При константном шаге сходится очень медленно

Идея ALS

$$Q \rightarrow \min_{u_i, v_j}$$

Повторяем до сходимости:

$$\frac{\partial Q}{\partial u_i} = 0 \quad \xrightarrow{\text{red arrow}} \quad u_i \qquad \qquad \frac{\partial Q}{\partial v_j} = 0 \quad \xrightarrow{\text{red arrow}} \quad v_j$$

Выписываем шаг в ALS

$$Q = \sum_{i,j} (\langle u_i, v_j \rangle - x_{ij})^2 \rightarrow \min_{u_i, v_j}$$

$$\frac{\partial Q}{\partial u_i} = \sum_j 2(\langle u_i, v_j \rangle - x_{ij})v_j = 0 \quad \sum_j v_j \langle v_j, u_i \rangle = \sum_j x_{ij} v_j$$

$$\sum_j v_j {v_j}^T u_i = \sum_j x_{ij} v_j$$

$$\left(\sum_j v_j {v_j}^T \right) u_i = \sum_j x_{ij} v_j$$

A

A

b

b

ALS: итоговый алгоритм

Повторяем по случайным i, j до сходимости:

$$\left(\sum_j v_j v_j^T \right) u_i = \sum_j x_{ij} v_j \quad \rightarrow \quad u_i$$

(решение системы
линейных уравнений)

$$\left(\sum_i u_i u_i^T \right) v_j = \sum_i x_{ij} u_i \quad \rightarrow \quad v_j$$

Регуляризация

$$Q = \sum_{i,j} (\langle u_i, v_j \rangle - x_{ij})^2 + \alpha \sum_i \|u_i\|^2 + \beta \sum_j \|v_j\|^2 \rightarrow \min_{u_i, v_j}$$

α и β - небольшие положительные числа (0.001, 0.01, 0.05)

Explicit и implicit

Explicit feedback

Есть положительные и отрицательные пример (например, низкие и высокие оценки фильмов, лайки и дислайки и т.д.)

Implicit feedback

Есть только положительные (покупки, просмотры, лайки) или только отрицательные примеры (дислайки)

Implicit matrix factorization

$$\sum_{i,j} w_{ij} (\langle u_i, v_j \rangle - x_{ij})^2 \rightarrow \min$$

Сумма по всем индексам (не только по известным элементам матрицы)

w_{ij} принимает большие значения для $x_{ij} \neq 0$
и значительно меньшие для $x_{ij} = 0$

Популярный метод: Implicit ALS

$$\sum_{i,j} w_{ij} (\langle u_i, v_j \rangle - x_{ij})^2 \rightarrow \min$$

$$w_{ij} = 1 + \alpha |x_{ij}| \quad \alpha = 10, 100, 1000$$

u_i, v_j оцениваем с помощью ALS

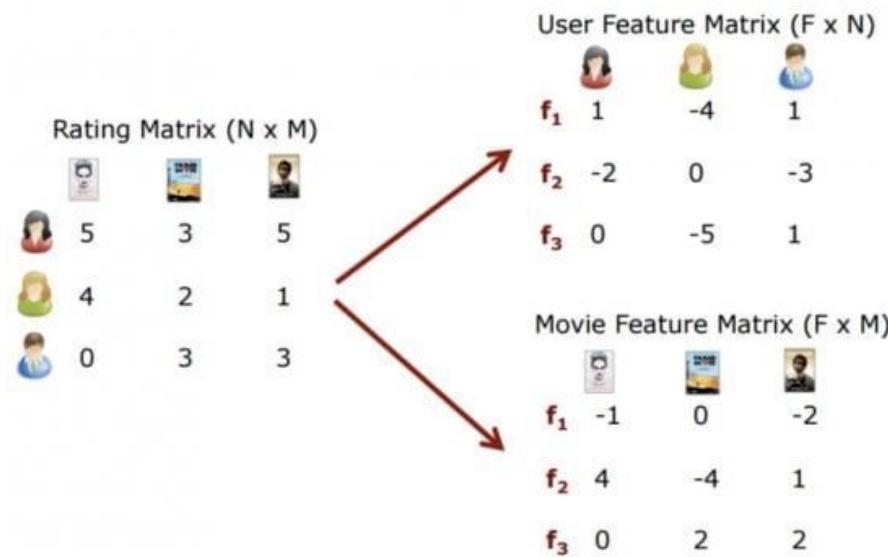
LightFM

Github - <https://github.com/lyst/lightfm>

Docs - <https://making.lyst.com/lightfm/docs/home.html>

Основная фишка - построение векторов для фичей пользователя и объекта.

В качестве векторных представлений самих пользователей и объектов берется сумма векторов их фичей



LightFM

Модель одна, с разными loss:

- Logistic
- Bayesian Personalized Ranking
- Weighted Approximate-Rank Pairwise
- k-os WARP

Обучается с помощью SGD

LightFM WARP

Основная идея - переход от проверки качества предсказания значения к предсказанию качества ранжирования.

Достигается это путем сэмплирования негативных примеров и построения функции ошибки на основе соотношения предсказанных значений.

Алгоритм обучения:

1. Берем пользователя u
2. Выбираем для него объект i , с которым он взаимодействовал
3. Выбираем случайный объект j , с которым он не взаимодействовал
4. Считаем скоры модели $pos=f(u,i)$ и $neg=f(u,j)$
5. Сравниваем pos и neg
 - a. Если $p > n$, то переходим на шаг 2 (или шаг 1)
 - b. Иначе считаем ошибку, правим веса и возвращаемся на шаг 3.

LightFM WARP

Neg - количество негативных примеров

\langle , \rangle - скалярное произведение

b - смещения (глобальное, по user, по item)

$$f(u, i) = b_u + b_i + \langle p_u, q_i \rangle$$

$$Loss(u, i) = \log \left(\left| \frac{Neg}{Sampled} \right| \right) \cdot \max(0, 1 - f(u, i) + f(u, j))$$

Sampled - сколько раз нам пришлось сэмплировать негативный пример, чтобы найти такой, для которого скор модели будет больше, чем скор модели для позитивного примера

Метрики качества рекомендаций

Метрики

Метрики в рекомендательных системах можно разделить на следующие группы

- Регрессионные
- Классификационные
- Ранжирующие

Метрики регрессии

Регрессионные метрики применяются для оценки качества предсказанных моделью значений

- Mean Absolute Error

$$MAE = \frac{1}{N} \sum_{i=1}^N |y_{true} - y_{pred}|$$

- Mean Squared Error

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_{true} - y_{pred})^2$$

- Rooted Mean Squared Error

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_{true} - y_{pred})^2}$$

Метрики классификации

Классификационные метрики оценивают качество топ- N рекомендаций с точки зрения бинарной классификации. Все считается на основе 4 базовых случаев:

- True positive (TP) - модель рекомендовала объект, с которым пользователь провзаимодействовал
- False positive (FP) - модель рекомендовала объект, с которым пользователь не провзаимодействовал
- True negative (TN) - модель не рекомендовала объект, с которым пользователь не провзаимодействовал
- False negative (FN) - модель не рекомендовала объект, с которым пользователь провзаимодействовал

Метрики классификации

	True condition			
Total population	Condition positive	Condition negative	Prevalence = $\frac{\sum \text{Condition positive}}{\sum \text{Total population}}$	Accuracy (ACC) = $\frac{\sum \text{True positive} + \sum \text{True negative}}{\sum \text{Total population}}$
Predicted condition	True positive	False positive, Type I error	Positive predictive value (PPV), Precision = $\frac{\sum \text{True positive}}{\sum \text{Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\sum \text{False positive}}{\sum \text{Predicted condition positive}}$
Predicted condition negative	False negative, Type II error	True negative	False omission rate (FOR) = $\frac{\sum \text{False negative}}{\sum \text{Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\sum \text{True negative}}{\sum \text{Predicted condition negative}}$
	True positive rate (TPR), Recall, Sensitivity, probability of detection, Power = $\frac{\sum \text{True positive}}{\sum \text{Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\sum \text{False positive}}{\sum \text{Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR}^+}{\text{LR}^-}$
	False negative rate (FNR), Miss rate = $\frac{\sum \text{False negative}}{\sum \text{Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\sum \text{True negative}}{\sum \text{Condition negative}}$	Negative likelihood ratio (LR-) = $\frac{\text{FNR}}{\text{TNR}}$	F_1 score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

Метрики классификации

Наиболее популярным метриками являются

- Precision@K
 - Формула: $TP / (TP + FP)$
 - Можно заметить, что под *positives* мы понимаем рекомендованные объекты, то есть наш топ-К, значит $TP + FP = K$
 - Итоговая формула: TP / K
 - Интерпретируется как доля релевантных рекомендаций
- Recall@K
 - Формула: $TP / (TP + FN)$
 - $TP + FN$ это количество известных релевантных объектов для пользователя
 - Интерпретируется как доля релевантных объектов, попавших в рекомендации

Метрики ранжирования

Классификационные метрики уже неплохо показывают качество наших топ-К рекомендаций, но они учитывают только попадания. А мы также хотим, чтобы наши релевантные рекомендации находились как можно выше.

Здесь нам и помогут ранжирующие метрики, которые будут оценивать наши попадания, но с весами:

- Mean Reciprocal Rank
- Mean Average Precision
- Normalized Discounted Cumulative Gain

Метрики ранжирования

Mean Reciprocal Rank - средний обратный ранг

$$MRR = \frac{1}{N} \sum_{i=1}^N \frac{1}{rank_i}$$

N - кол-во пользователей, $rank_i$ - позиция первой релевантной рекомендации для пользователя i

Причем если для пользователя мы не рекомендовали ничего релевантного, то дробь $\frac{1}{rank_i}$ зануляется

Метрики ранжирования

Mean Average Precision - средняя точность по пользователям

$$MAP@K = \frac{1}{N} \sum_{i=1}^N AP@K(user_i)$$

$$AP@K(user) = \frac{1}{c_{user}} \sum_{i=1}^K Precision@i * rel_i$$

N - кол-во пользователей

c_{user} - кол-во релевантных объектов у пользователя

rel_i - релевантность i-ой рекомендации

Метрики ранжирования

Normalized Discounted Cumulative Gain - взвешенная точность по пользователям

$$CG@K = \sum_{i=1}^K rel_i \quad rel_i - \text{вес } i\text{-ой позиции, } 0 \text{ если не релевантна}$$

$$DCG@K = \sum_{i=1}^K \frac{rel_i}{log_2(i+1)} \quad IDCG@K = \sum_{i=1}^{\min(|Rel|, K)} \frac{rel_i}{log_2(i+1)} \quad |Rel| - \text{кол-во релевантных объектов для пользователя}$$

$$NDCG@K = \frac{DCG@K}{IDCG@K}$$

Beyond accuracy metrics

Чего не хватает этим рекомендациям?



Обозначения

i, j – индексы товаров

u, v – индексы пользователей

\mathcal{I}, \mathcal{U} – множества всех товаров и пользователей соответственно

\mathcal{I}_u – множество товаров, с которыми взаимодействовал пользователь u

\mathcal{U}_i – множество пользователей, которые взаимодействовали с товаром i

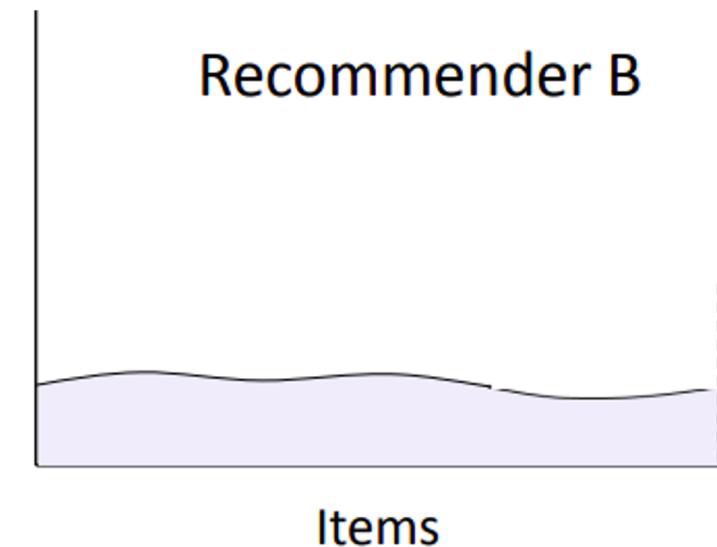
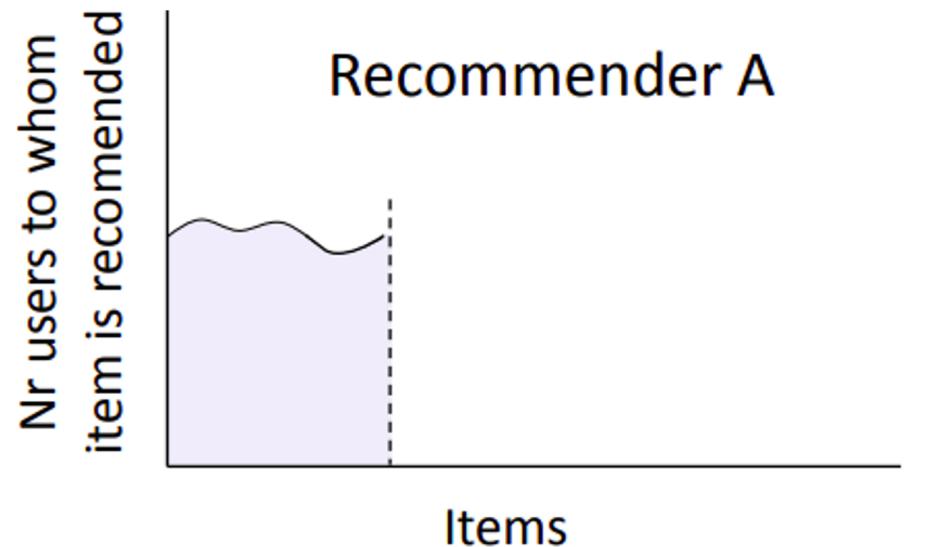
R_u – множество товаров, рекомендованных пользователю u

R – сокращение R_u в тех случаях, когда нет необходимости указывать индекс конкретного пользователя

Общее разнообразие

Aggregate diversity - количество товаров, в совокупности рекомендуемых алгоритмом всем пользователям (значения от 0 до $|\mathcal{I}|$)

$$\text{Aggdiv} = \left| \bigcup_{u \in \mathcal{U}} R_u \right|$$



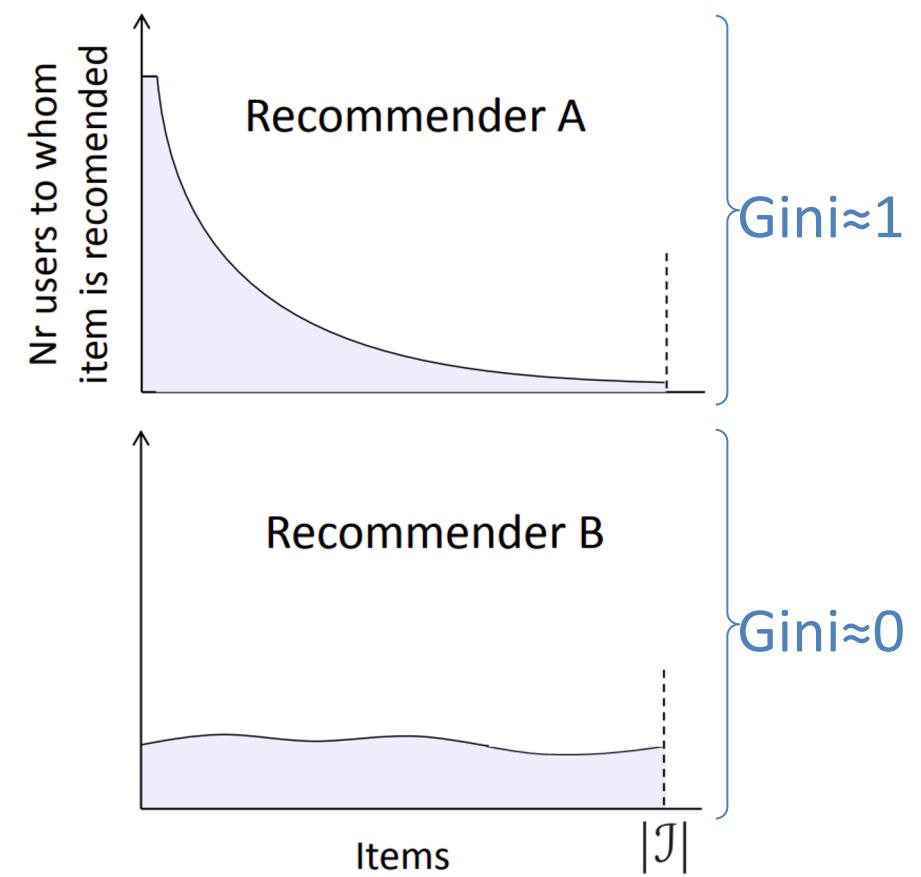
Коэффициент Gini

Gini измеряет равномерность распределения присутствия товаров в рекомендациях (значения от 0 до 1)

$$\text{Gini} = \frac{1}{|\mathcal{J}| - 1} \sum_{k=1}^{|\mathcal{J}|} (2k - |\mathcal{J}| - 1)p(i_k)$$

i_1, \dots, i_n - список товаров,
отсортированный по
вероятности вытягивания товара
из списка рекомендаций:

$$p(i) = \frac{|\{u \in \mathcal{U} \mid i \in R_u\}|}{\sum_{j \in \mathcal{J}} |\{u \in \mathcal{U} \mid j \in R_u\}|}$$



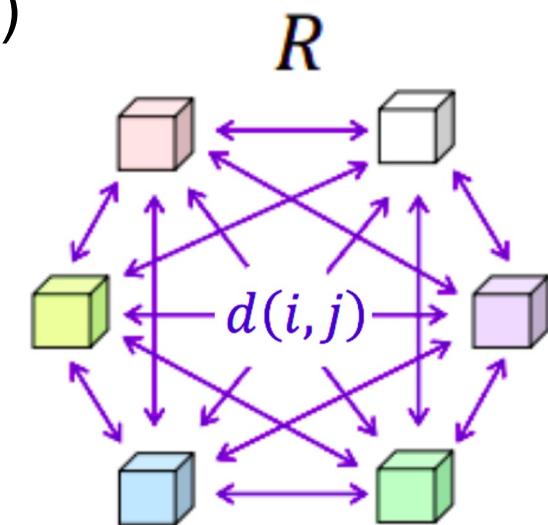
Разнообразие внутри полки

Intra-List Diversity определяется как среднее попарное расстояние между товарами в полке (значения от 0 до $+\infty$)

$$\text{ILD} = \frac{1}{|R|(|R| - 1)} \sum_{i \in R} \sum_{j \in R} d(i, j)$$

$d(i, j)$ определяются как расстояния между эмбеддингами или one-hot векторами признаков товаров:

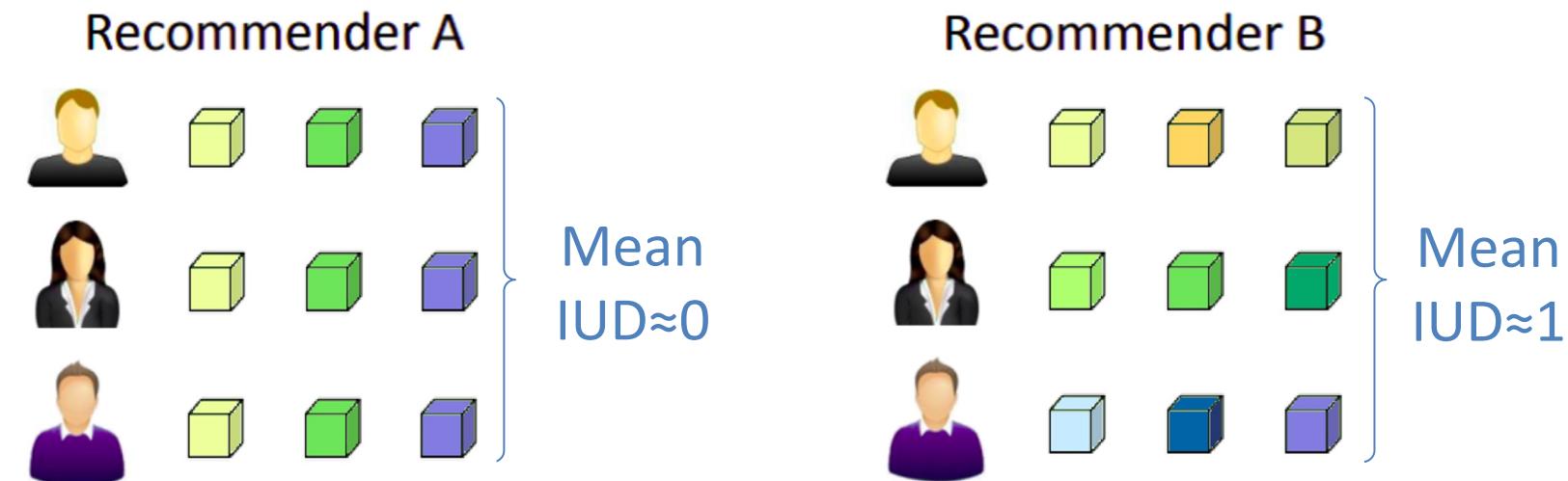
- евклидово
- косинусное
- Хэмминга и др.



“Персональность” рекомендаций

Inter-User Diversity - средняя доля пересечения рекомендаций для пользователя с рекомендациями для остальных пользователей (значения от 0 до 1)

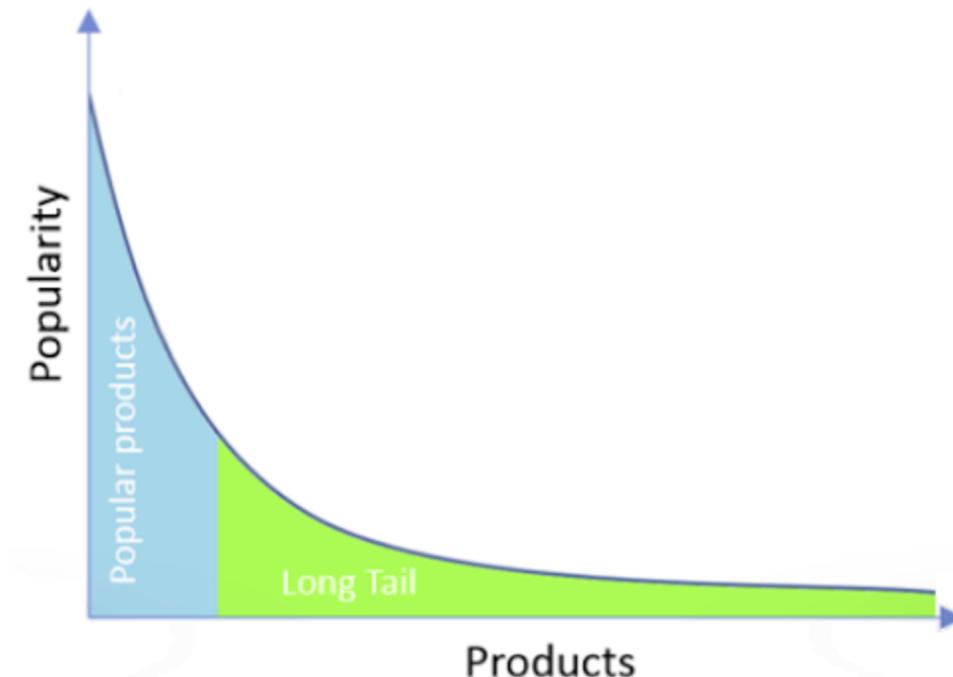
$$\text{IUD} = \frac{1}{|\mathcal{U}| - 1} \sum_{v \in \mathcal{U}} |R - R_v| / |R|$$



Новизна глобальная

Mean Inverse User Frequency - средняя новизна товаров в полке, где “новизна” товара обратно пропорциональна количеству пользователей, которые с ним взаимодействовали (значения от 0 до $+\infty$)

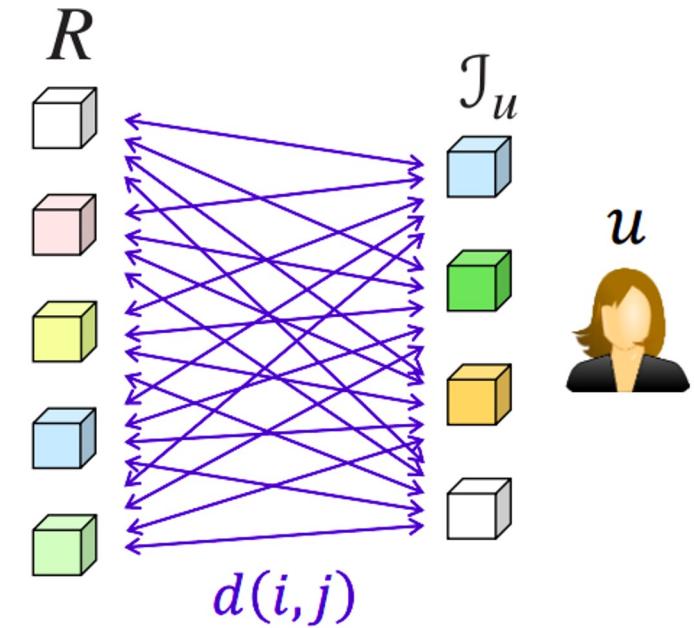
$$\text{MIUF} = -\frac{1}{|R|} \sum_{i \in R} \log_2 \underbrace{\frac{|\mathcal{U}_i|}{|\mathcal{U}|}}_{\text{популярность товара}}$$



Новизна персональная

Unexpectedness - отличие рекомендованных товаров от тех, с которыми пользователь уже взаимодействовал ранее (значения от 0 до $+\infty$)

$$\text{Unexp} = \frac{1}{|R||\mathcal{I}_u|} \sum_{i \in R} \sum_{j \in \mathcal{I}_u} d(i, j)$$



Продакшен

Мы построили модель

Послушали лекцию

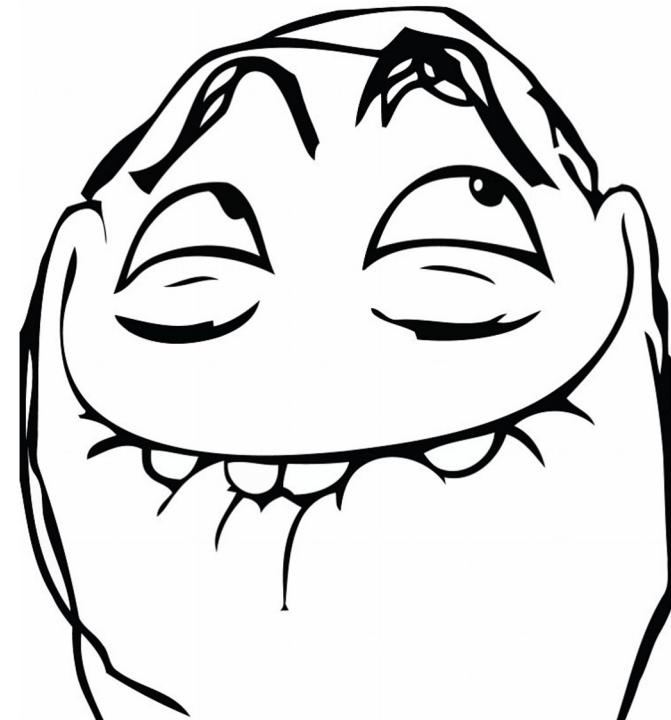
Построили модель

Посчитали реко в Jupyter

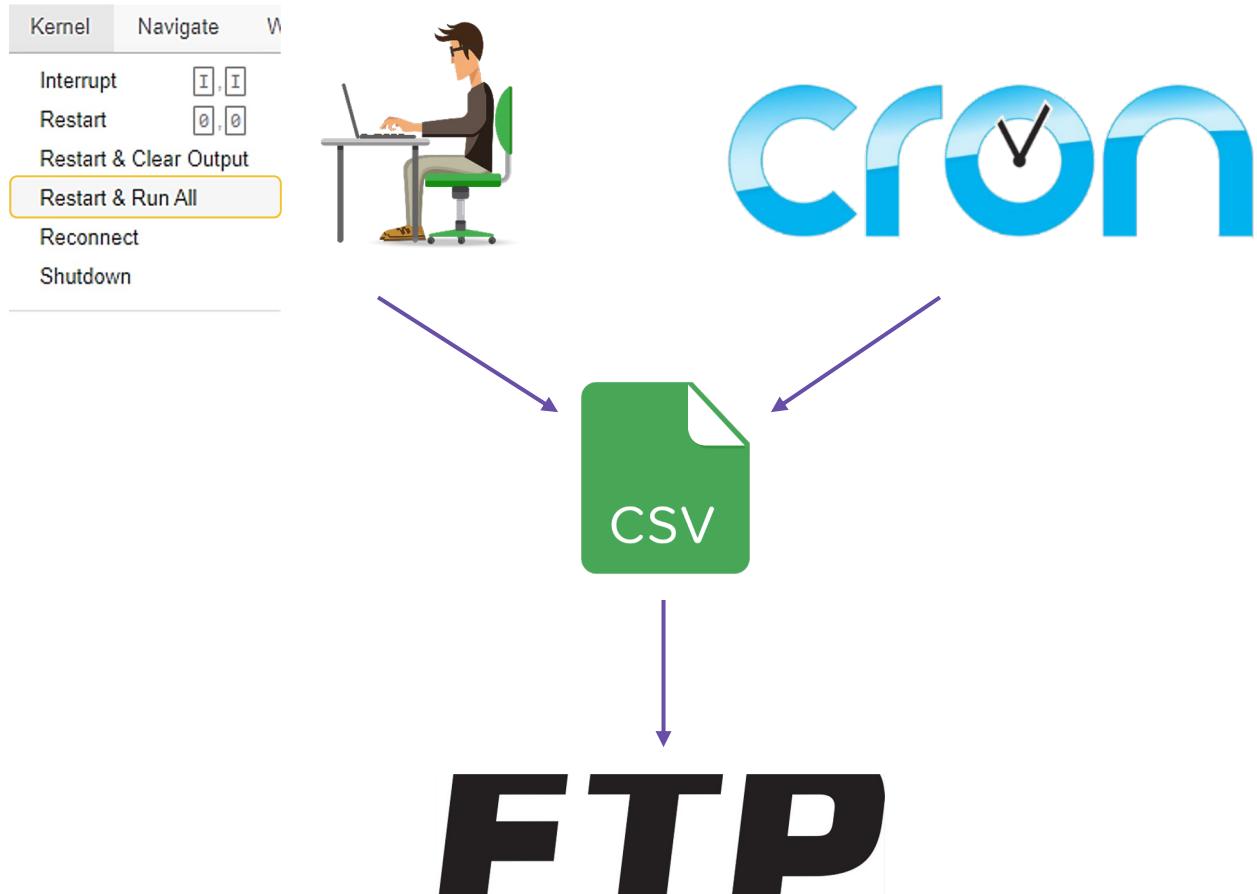
Провели АВ-тест

CTR +100500%, p-value << 0.05

Ну, теперь заживем ...



Как **не** нужно отдавать рекомендации заказчику

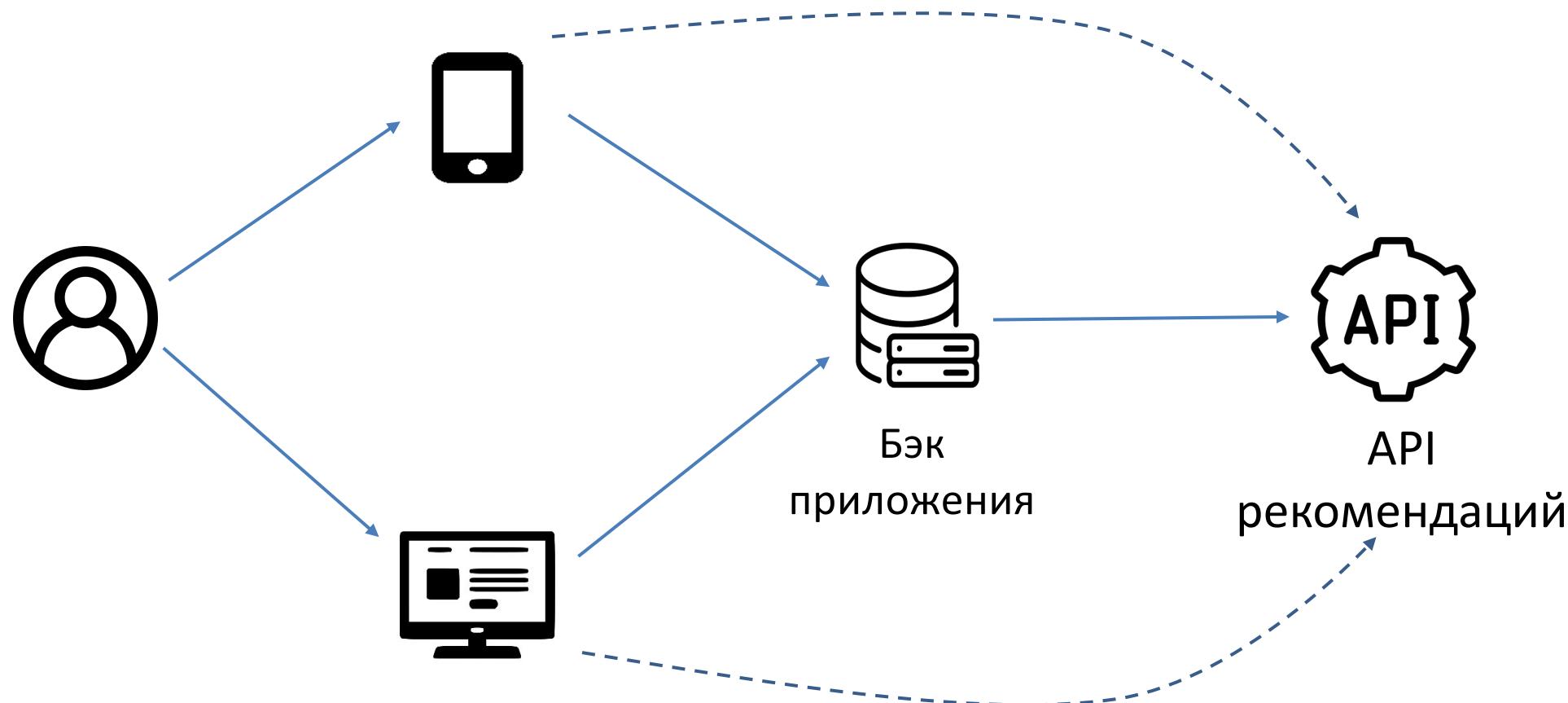


(Полу)ручная выгрузка

- Нужно тратить время
- Ненадежно
- Неудобно для заказчика
- Нет метрик, логов
- Не масштабируется

Подходит только для этапа первичных испытаний модели

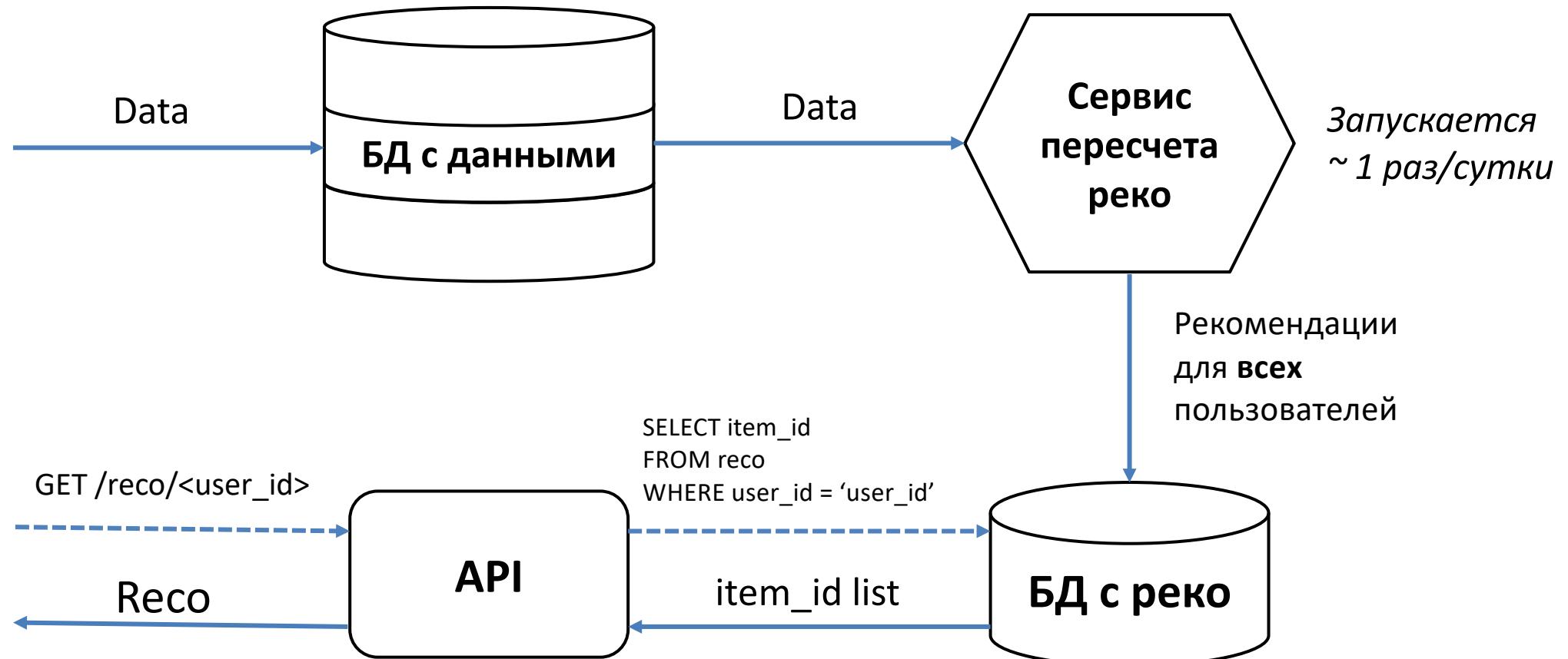
Как нужно отдавать рекомендации заказчику



Варианты продаж

- Offline (батчевые рекомендации)
- Online (realtime)

Offline

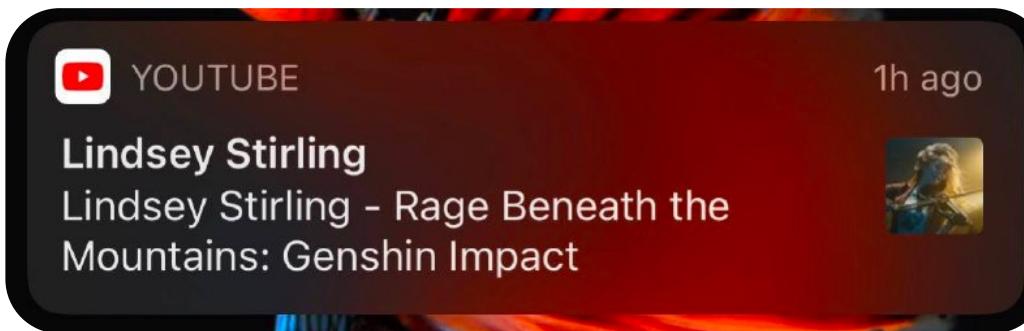


Offline рекомендации

где использовать



+
RecSys MVP
почти в любой области



Online рекомендации

где использовать



NETFLIX



Мониторинг качества - зачем?

- Защититься от косяков реализации
- Убедиться что на проде все так же, как и в ресерче
- Поймать момент, когда модель пора менять

Мониторинг качества - как?

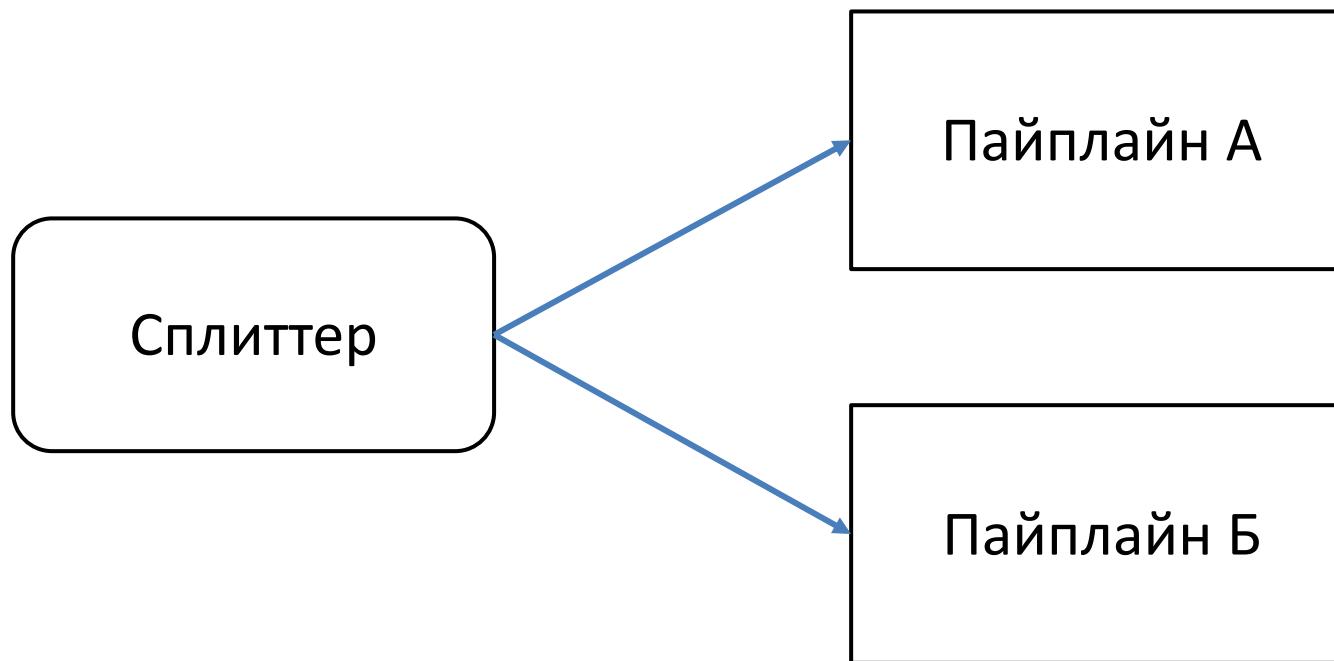
- Считать проверочные метрики (например, что в реко нет дублей)
- Каждый раз строить модель не только на текущую дату, но и на неделю/месяц назад и считать онлайн метрики
- Считать онлайн метрики

mlflow



EVIDENTLY AI

АВ-тесты



Offline. Выбор инструментов

БД с реко

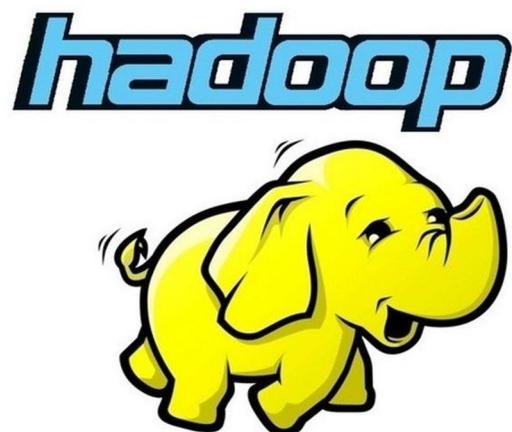


- Быстрая операция взятия по ключу
- Возможность быстрой фильтрации по дополнительным полям



Offline. Выбор инструментов

БД с данными



- Скорее всего уже есть
- В большинстве случаев подойдет любая
- Выбор зависит от структуры данных



Offline. Выбор инструментов

API

- Зависит от RPS и SLA по времени ответа
- Нагрузка на сервис минимальная - основное время на запрос к БД
- Скорее всего Python будет достаточно
- Пишите, на чем удобнее, но лучше сразу на асинхронном фреймворке



Golang



RPS

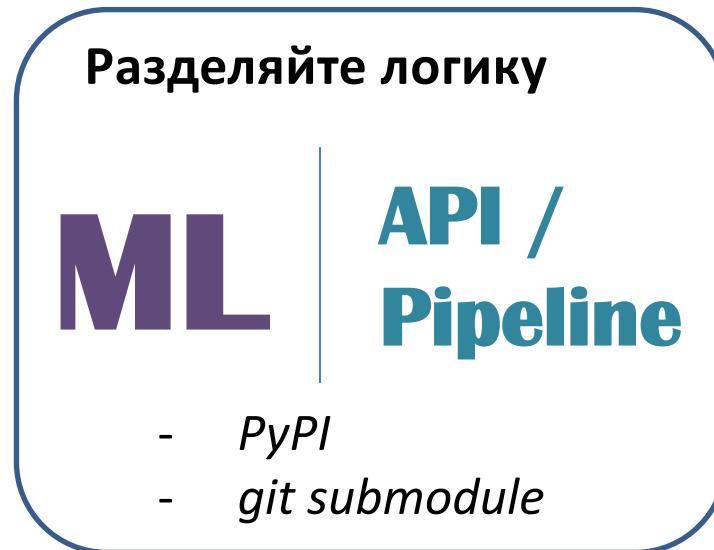
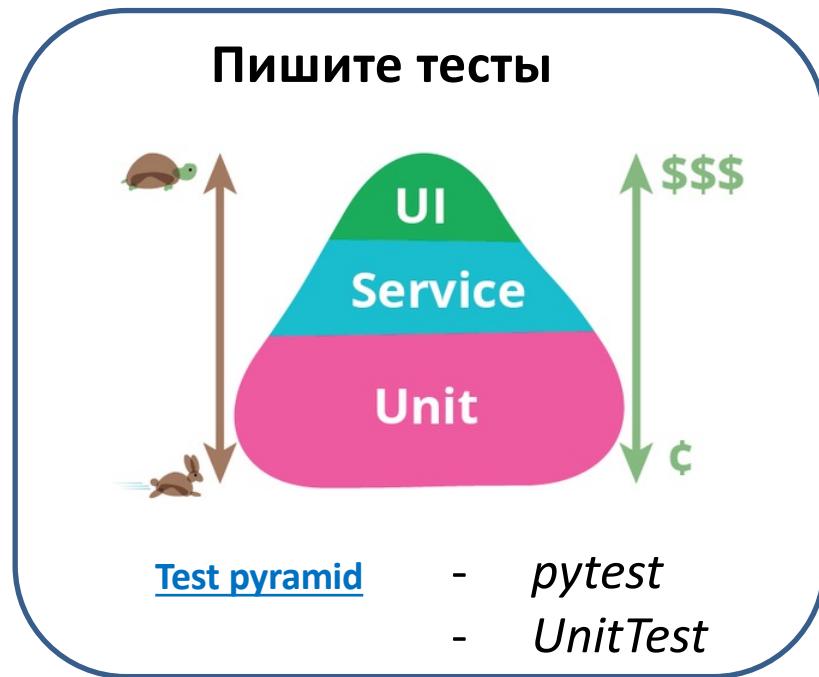
Offline. Выбор инструментов сервис пересчета реко



- Точно нужен *scheduller*, чтобы запускать по расписанию
- Очень полезен будет инструмент для построения DAG-ов задач
- Не помешает UI для контроля процессов



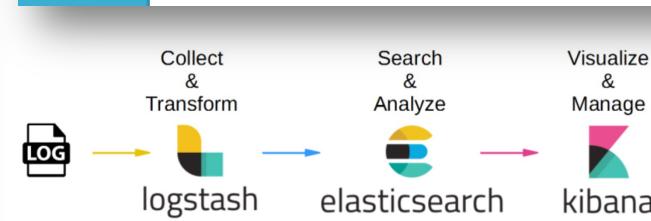
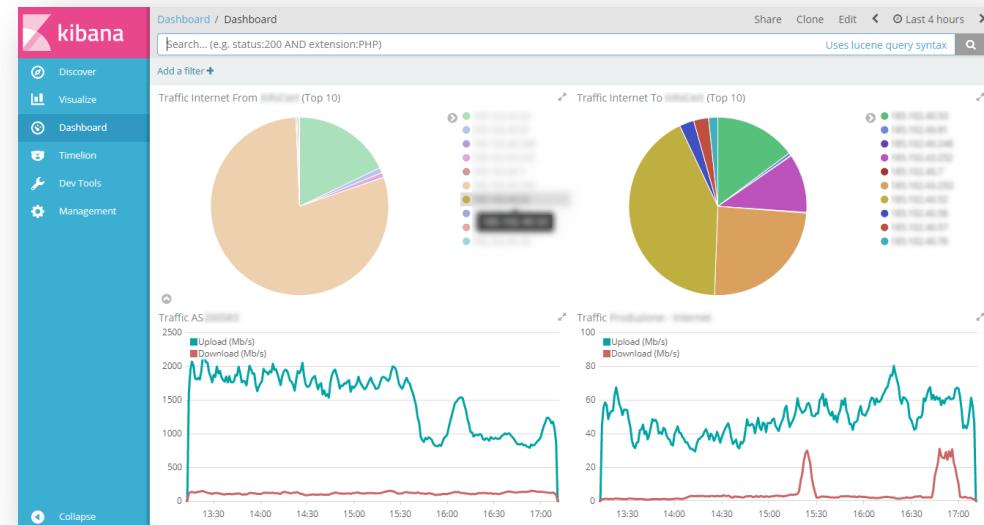
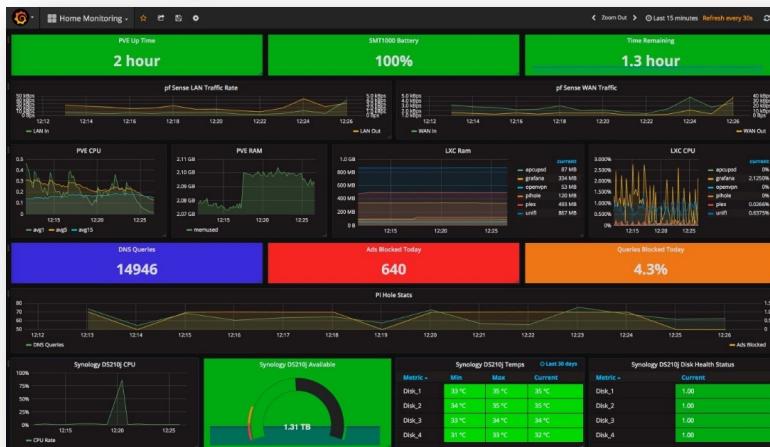
Как сделать хорошо



* справедливо для любых ML моделей

И СНОВА МОНИТОРИНГ

логи, метрики, алерты

A screenshot of a Sentry error log table. The columns are "TYPE", "CATEGORY", "DESCRIPTION", "LEVEL", and "TIME". The data includes: "User Action" (navigation, info, 15:51:22); "Navigation" (info, 15:51:24); "HTTP request" (info, 15:51:28); "Error" (error, 15:51:30); "POST /ap" (info, 15:51:31); "ui.click" (info, 15:51:34); "ui.click" (error, 15:51:42); "exception" (error, ReferenceError: getCardInfo is not defined, 15:51:43).

Резюме прод

- Если реко нужно считать регулярно, имеет смысл автоматизировать процесс
- Инструменты нужно выбирать в зависимости от задачи. В большинстве случаев для offline рекомендаций подойдет базовый набор: SQL DB + async Python + Airflow
- Контейнеры, тесты, мониторинг

Резюме

- Рекомендации — широкая задача с большим количеством коммерческих применений
- Модели: коллаборативная фильтрация, контентный подход
- Наиболее популярный подход: модели со скрытыми переменными
- Обилие метрик качества