



# Your First RecSys

**Даниил Потапов**

Руководитель группы персонализации и рекомендательных систем  
MTC BigData





## Зачем это все

Основная цель - собрать необходимый минимум знаний для построения рекомендательной системы в одном месте.

Поэтому *меньше теории* и *больше python кода*.

Но теорию изучать тоже надо, буду компенсировать ссылками :)



# Общий план лекций

- Введение в рекомендательные системы
- Методы валидации, метрики и бейзлайны
- Коллаборативная фильтрация и гибридные методы
  - Разбор библиотек implicit и LightFM
- Градиентный бустинг и задача реранжирования
  - CatBoost/XGBoost/LightGBM и learning to rank
- Как и куда “копать” дальше



# Введение в рекомендательные системы



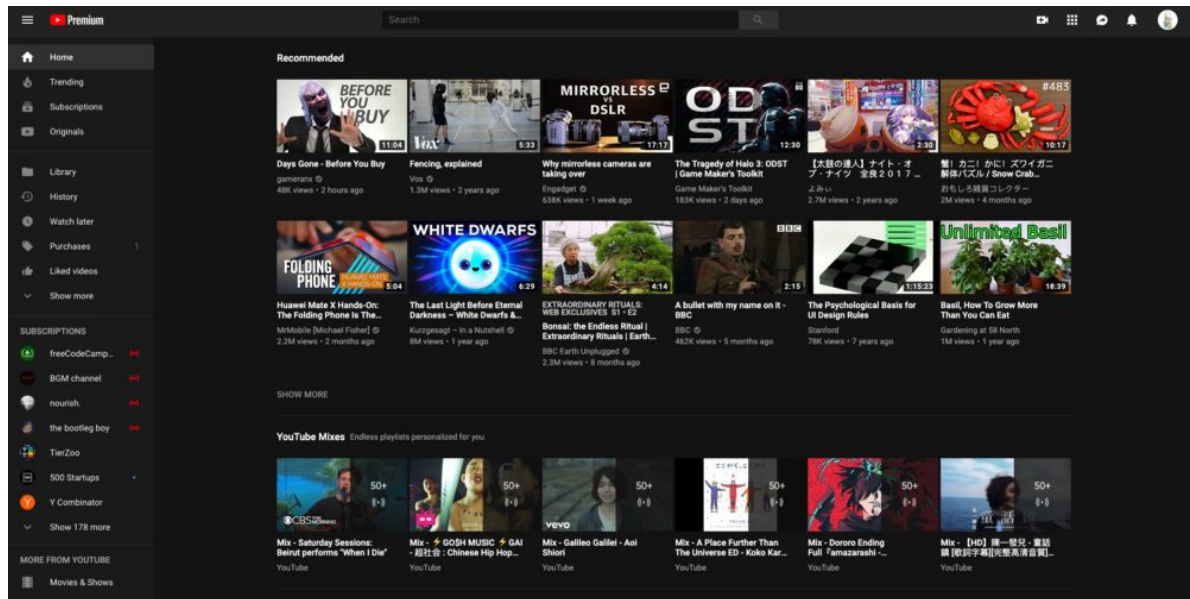
## План лекции

- Примеры рекомендательных систем
- Классика - Netflix prize
- Основная идея и постановка задачи
- Данные для рекомендательных систем и методы работы с ними
- Виды рекомендательных систем
- Полезные ссылки



# Примеры рекомендательных систем

Контентные системы, например Youtube



# Примеры рекомендательных систем

Интернет магазины, например Amazon

Customers Who Bought This Item Also Bought

Page 1 of 15



**Data Science from Scratch: First Principles with Python**  
 > Joel Grus  
 ★★★★★☆ 54  
**#1 Best Seller** in Data Mining  
 Paperback  
 \$33.99 ✓Prime



**Python for Data Analysis: Data Wrangling with Pandas, NumPy, and...**  
 > Wes McKinney  
 ★★★★★☆ 118  
 Paperback  
 \$27.68 ✓Prime



**Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking**  
 > Foster Provost  
 ★★★★★☆ 135  
 Paperback  
 \$37.99 ✓Prime



**Reproducible Research with R and RStudio**  
 Second Edition...  
 Christopher Gandrud  
 ★★★★★☆ 3  
 Paperback  
 \$51.97 ✓Prime



**An Introduction to Statistical Learning: with Applications in R**  
 > Gareth James  
 ★★★★★☆ 105  
 Hardcover  
 \$68.35 ✓Prime



**Data Smart: Using Data Science to Transform Information into Insight**  
 > John W. Foreman  
 ★★★★★☆ 99  
**#1 Best Seller** in Computer Simulation  
 Paperback  
 \$28.16 ✓Prime



**The Statistical Sleuth: A Course in Methods of Data Analysis**  
 Fred Ramsey  
 ★★★★★☆ 6  
 Hardcover  
 \$284.42 ✓Prime



# Примеры рекомендательных систем

У социальных сетей возникают следующие задачи:

- Персональные ленты
- Люди, с которыми вы можете быть знакомы

Но вообще ранжировать можно что угодно:

- Поисковую выдачу
- Адреса назначения в агрегаторах такси





# 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   | <a href="#">BellKor's Pragmatic Chaos</a>           | 0.8567          | 10.06         | 2009-07-26 18:18:28 |
| 2   | <a href="#">The Ensemble</a>                        | 0.8567          | 10.06         | 2009-07-26 18:38:22 |
| 3   | <a href="#">Grand Prize Team</a>                    | 0.8582          | 9.90          | 2009-07-10 21:24:40 |
| 4   | <a href="#">Opera Solutions and Vandelay United</a> | 0.8588          | 9.84          | 2009-07-10 01:12:31 |
| 5   | <a href="#">Vandelay Industries I</a>               | 0.8591          | 9.81          | 2009-07-10 00:32:20 |
| 6   | <a href="#">PragmaticTheory</a>                     | 0.8594          | 9.77          | 2009-06-24 12:06:56 |
| 7   | <a href="#">BellKor in BigChaos</a>                 | 0.8601          | 9.70          | 2009-05-13 08:14:09 |
| 8   | <a href="#">Dace</a>                                | 0.8612          | 9.59          | 2009-07-24 17:18:43 |

credit: <https://www.netflixprize.com/leaderboard.html>



# 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.



## Основная идея

Задача рекомендательной системы:

- отранжировать какой-то набор объектов согласно какому-то критерию
- предсказать оценку объекта

**Самый типовой случай** - персонализированное ранжирование.

2 действующих лица - пользователь и объект (статья, книга, фильм и тд),  
которые “взаимодействуют” друг с другом



## Постановка задачи

- **user** - пользователь
- **item** - объект
- $r_{ui}$  - оценка объекта **i** пользователем **u**

Задача - найти для пользователя **u** объект **i** с максимальной оценкой **r**














# Данные для рекомендательных систем

Оценки объектов пользователями - “*target*”  
в классических задачах машинного  
обучения.

Может быть численный - рейтинг товара,  
длительность просмотра фильма  
Может быть фактом - купил или не купил  
товар, like/dislike новости

**Базовый датасет** - матрица оценок  
объектов пользователями.

|   |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
|  a | 5   |   | 1   | 1   |   | 2   |
|  b |   | 2   |   | 4   |   | 4   |
|  c | 4   | 5   |   | 1   | 1   | 2   |
|  d |   |   | 3   | 5   | 2   |   |
|  e | 2   |   | 1   |   | 4   | 4   |



# Данные для рекомендательных систем

Implicit и explicit feedback:

**Explicit** - явный таргет, которому можно относительно доверять: оценки, покупки и тд. Такого обычно мало.

**Implicit** - неявный таргет. Это действия, которые не говорят явно о том, как пользователь оценил объект: клики, просмотры и тд. Таких данных гораздо больше в системе.

Обычно стараются использовать как можно больше данных, взвешивая разный тип таргета.



# Данные для рекомендательных систем

Что в итоге должен содержать датасет:

- идентификатор пользователя
- идентификатор объекта

Опционально:

- оценка
- время

Данные такого вида идеально ложатся в `pandas.DataFrame`

|   | user_id | item_id | rating | timestamp  |
|---|---------|---------|--------|------------|
| 0 | 126706  | 14433   | NaN    | 2018-01-01 |
| 1 | 127290  | 140952  | NaN    | 2018-01-01 |
| 2 | 66991   | 198453  | NaN    | 2018-01-01 |
| 3 | 46791   | 83486   | 5.0    | 2018-01-01 |
| 4 | 79313   | 188770  | 5.0    | 2018-01-01 |
| 5 | 63454   | 78434   | NaN    | 2018-01-01 |
| 6 | 127451  | 14876   | NaN    | 2018-01-01 |
| 7 | 42797   | 315927  | 5.0    | 2018-01-01 |
| 8 | 47287   | 258483  | NaN    | 2018-01-01 |
| 9 | 23439   | 9762    | 4.0    | 2018-01-01 |





# Данные для рекомендательных систем

При выгрузке таких данных в pandas может возникнуть проблемы с памятью.

Но есть трюки по оптимизации:

- Если идентификаторы пользователей или объектов это строки, то их лучше приводить к [CategoricalDType](#)
- Колонку с пропущенными значениями pandas сразу приводит ко float. Но если сами значения не float, а, например, целочисленные значения, то можно использовать [IntegerDType](#)

В среднем это позволяет экономить от 20 до 80 процентов потребляемой памяти.



# Данные для рекомендательных систем

А где, собственно, матрицы? По сути, такой лог (user, item, rating) в `pandas.DataFrame` и есть представление матрицы.

Но библиотеки на вход принимают именно разреженные матрицы.

Для работы с ними в python есть модуль [scipy.sparse](#)



# Данные для рекомендательных систем

Виды разреженных матриц:

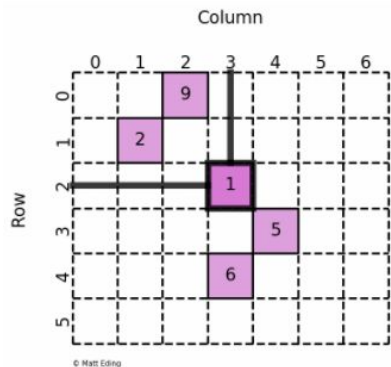
- [coo\\_matrix](#) - A sparse matrix in COOrdinate format.
- [csc\\_matrix](#) - Compressed Sparse Column matrix
- [csr\\_matrix](#) - Compressed Sparse Row matrix
- [bsr\\_matrix](#) - Block Sparse Row matrix
- [dia\\_matrix](#) - Sparse matrix with DIAGONAL storage
- [dok\\_matrix](#) - Dictionary Of Keys based sparse matrix.
- [lil\\_matrix](#) - Row-based list of lists sparse matrix

В основном используются первые три:

- `coo_matrix` - используется для создания разреженной матрицы
- `csr/csc_matrix` - используются для оптимизированных операций над матрицами

# Данные для рекомендательных систем

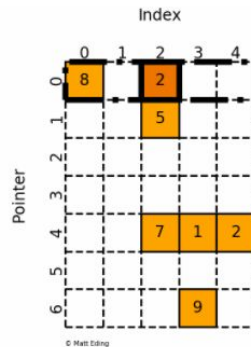
coo\_matrix



COO

|        |   |   |   |   |   |
|--------|---|---|---|---|---|
| Row    | 1 | 3 | 0 | 2 | 4 |
| Column | 1 | 4 | 2 | 3 | 3 |
| Data   | 2 | 5 | 9 | 1 | 6 |

csr\_matrix



CSR

|                |   |   |   |   |   |   |   |   |
|----------------|---|---|---|---|---|---|---|---|
| Index Pointers | 0 | 2 | 3 | 3 | 3 | 6 | 6 | 7 |
| Row: 0         |   |   |   |   |   |   |   |   |
| NNZ: 2         |   |   |   |   |   |   |   |   |
| Indices        | 0 | 2 | 2 | 2 | 3 | 4 | 3 |   |
| Data           | 8 | 2 | 5 | 7 | 1 | 2 | 9 |   |



# Данные для рекомендательных систем

Помимо таргета конечно могут и должны использоваться характеристики самих пользователей и объектов.

Сами идентификаторы пользователей или объектов также могут быть фичами, их называют индикаторными. Чаще всего это просто единичная матрица размером с кол-во пользователей или объектов.

# Данные для рекомендательных систем

|        | Indicator Features |   |   |   | Metadata Features  |     |     |   |
|--------|--------------------|---|---|---|--|-----|-----|---|
| User 1 | 1                  |   |   |   | 1  | -1  | 100 | $\begin{bmatrix} [n\_users \times n\_user\_features] \\ \text{or} \\ [n\_items \times n\_item\_features] \end{bmatrix}$ |
| User 2 |                    | 1 |   |   | 1  | 1   | 50  |   |
| User 3 |                    |   | 1 |   | 1  | .7  | 20  |   |
| User 4 |                    |   |   | 1 | 1  | -.5 | -20 |   |
| User 5 |                    |   |   |   | 1  | .2  | 10  |   |
| User 6 |                    |   |   |   | 1  |     | 100 |   |
|        |                    |   |   |   | $\underbrace{\hspace{10em}}_{\text{Encoded Labels/Tags/etc.}}$ |     |     |   |



# 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
- 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

Обсудим их поверхностно, конкретные реализации разберем в последующих лекциях.





# 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

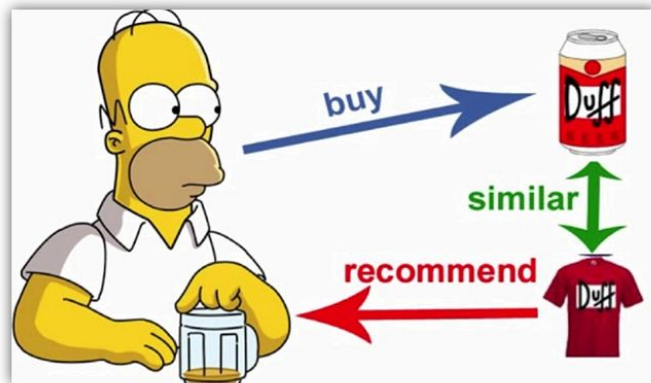
**Процесс построения рекомендаций** - процесс подбора объектов с наиболее высокой оценкой для данного пользователя.

# 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

|   |   |  |   |  |   |   |
|---|--|---|--|---|--|--|
|  |   |  |   |  |   |   |
|  |  |  |  |  |  |   |
|  |   |  |  |  |   |   |
|  |  |   |   |  |   |  |
|  |  |   |  |   |  |  |

# Neighbour-based collaborative filtering

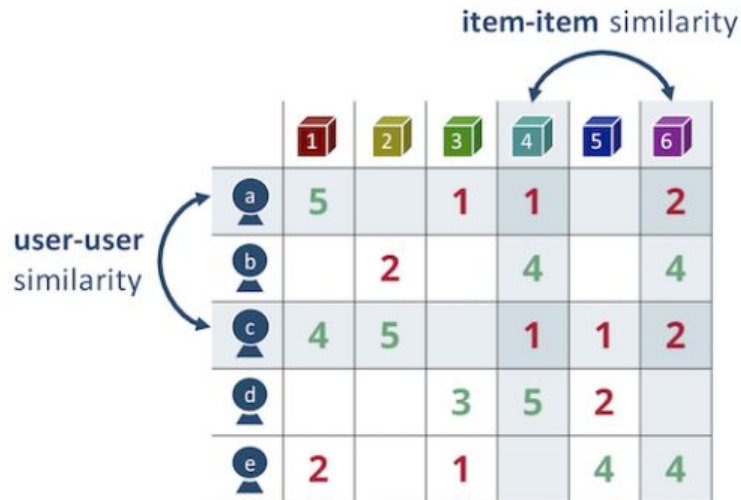
**Основная идея** - использовать строки или столбцы из матрицы оценок как векторное представление пользователя или объекта.

## Два подхода:

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

## Как использовать:

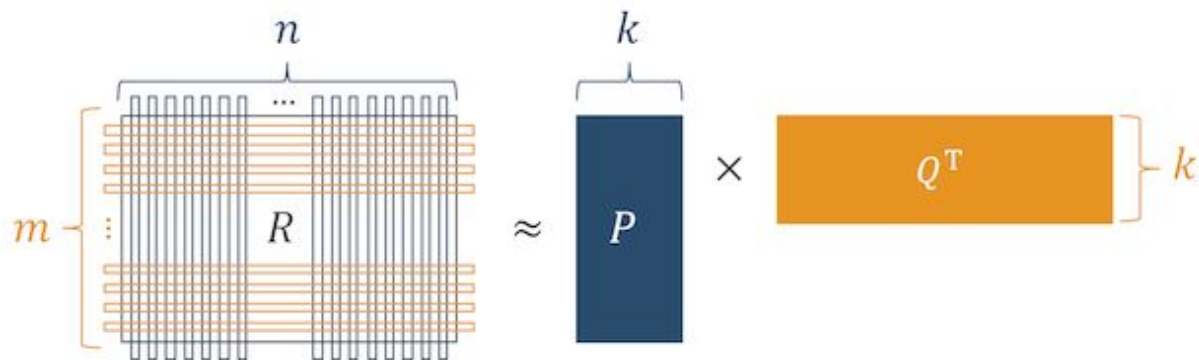
- Найти похожие объекты на то, с чем пользователь взаимодействовал
- Рекомендовать объекты из тех, с которыми взаимодействовали похожие пользователи



# Model-based collaborative filtering

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

**Основной подход** - матричные разложения.

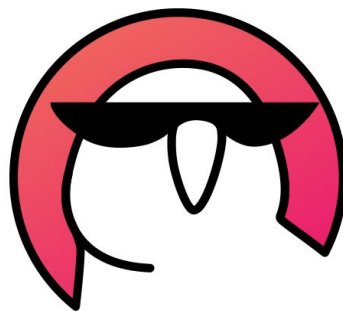




## Полезные ссылки

Каналы в ODS Slack:

- [#recommender\\_systems](#)
- [#theory\\_and\\_practice](#)



Open  
Data  
Science



## Полезные ссылки

### Литература

- Recommender Systems Handbook  
Francesco Ricci, Lior Rokach, Bracha Shapira, Paul B. Kantor
- Recommender Systems: The Textbook  
Charu C. Aggarwal
- Collaborative Recommendations: Algorithms, Practical Challenges and Applications Shlomo Berkovsky, Ivan Cantador, Domonkos Tikk

### Лекции

- [Лекция Евгения Фролова в МФТИ](#)
- [ФКН НИУ ВШЭ](#)
- [Технострим Mail.ru Group](#)



## Полезные ссылки

### Конференции

- ACM RecSys
- UMAP
- KDD
- ICML

### Соревнования

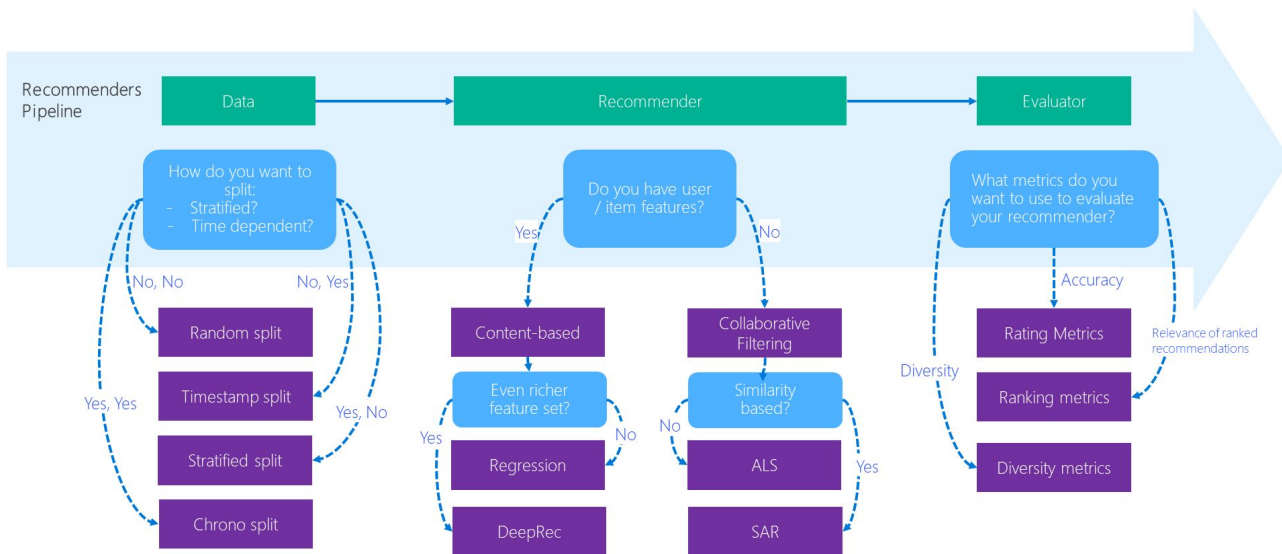
- [RecSys challenge](#)
- [Rekko challenge](#)
- [Retail Hero](#)
- [SNA Hackathon](#)



# Полезные ссылки

## Best Practices on Recommendation Systems

- <https://github.com/microsoft/recommenders>
- <https://microsoft-recommenders.readthedocs.io/en/latest/index.html>





## Полезные ссылки

Kaggle-датасет для практики

<https://www.kaggle.com/sharthz23/mts-library>

Обзор использования Pandas и SciPy.sparse -

<https://www.kaggle.com/sharthz23/pandas-sciPy-for-recsys>



## Контакты



Telegram чат по курсу

<https://t.me/joinchat/E52trBnd9b65VNxYZTerhQ>