# RecSys
## рекомендации фильмов

Авторы проекта - студенты МОВС:

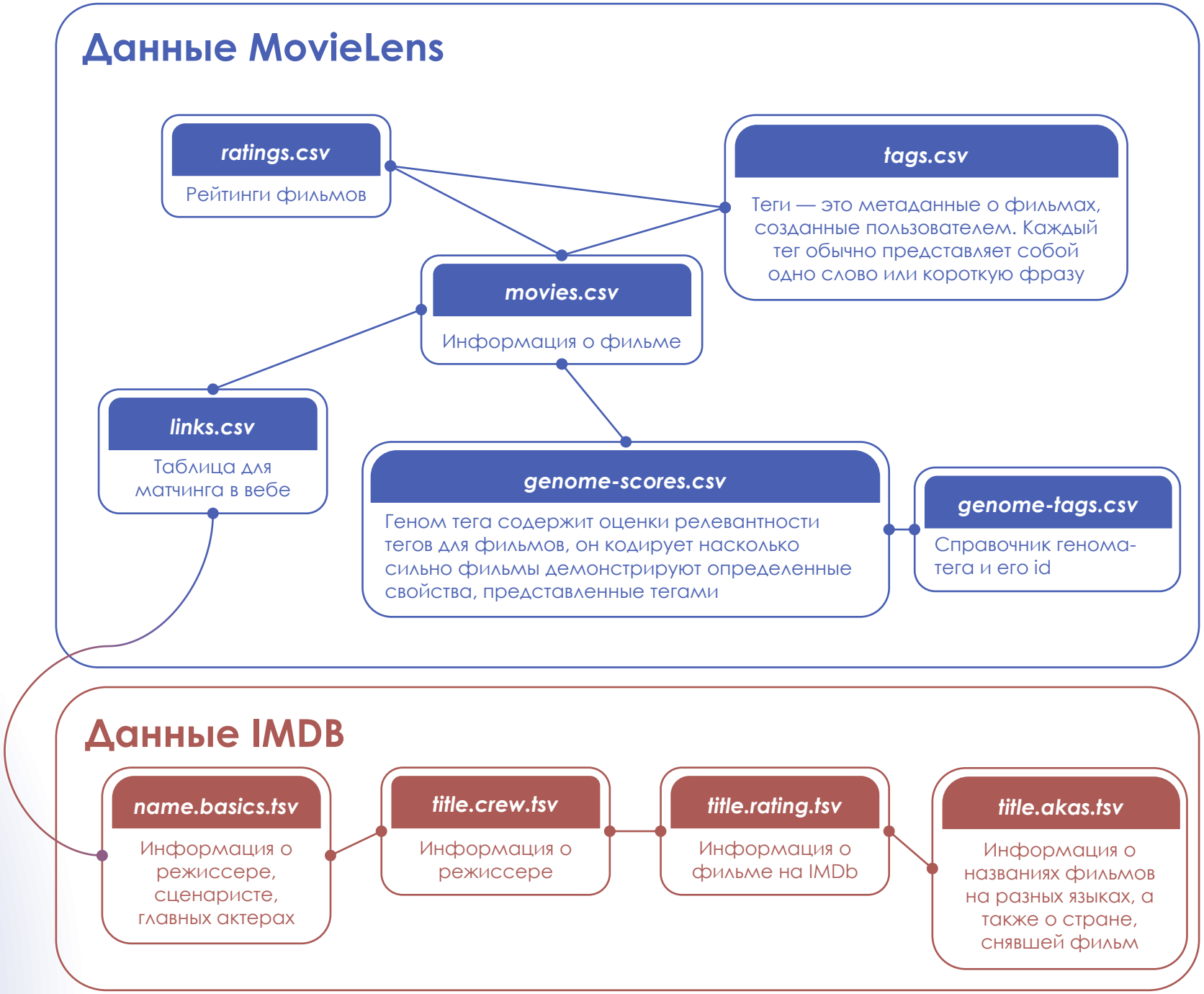
Владислав Панфиленко

Лилия Хорошенина

# Проделанная работа

- Выбор данных
- Проведение EDA
- Предобработка данных
- Анализ различных baseline-моделей (ML часть)
- Разработка микросервиса
- Применение DeepFM (DL часть)

# Данные


Данные MovieLens
ratings.csv
Рейтинги фильмов
tags.csv
Теги — это метаданные о фильмах, созданные пользователем. Каждый тег обычно представляет собой одно слово или короткую фразу
movies.csv
Информация о фильме
links.csv
Таблица для матчинга в вебе
genome-scores.csv
Геном тега содержит оценки релевантности тегов для фильмов, он кодирует насколько сильно фильмы демонстрируют определенные свойства, представленные тегами
genome-tags.csv
Справочник генома-тега и его id
Данные IMDB
name.basics.tsv
Информация о режиссере, сценаристе, главных актерах
title.crew.tsv
Информация о режиссере
title.rating.tsv
Информация о фильме на IMDb
title.akas.tsv
Информация о названиях фильмов на разных языках, а также о стране, снявшей фильм

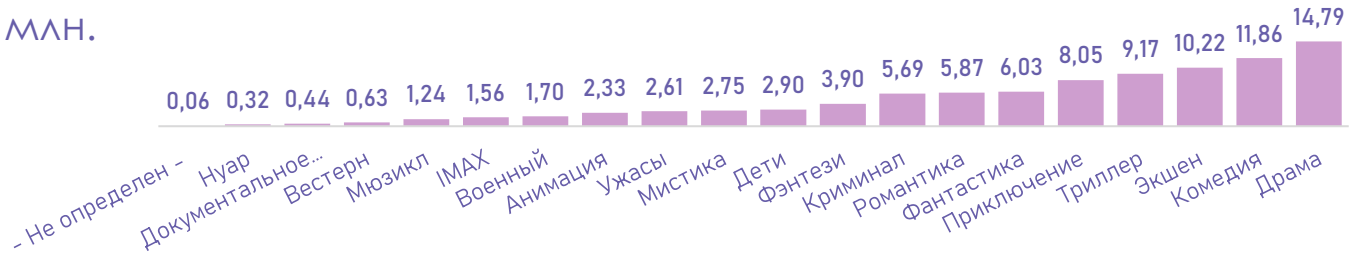
# EDA

**~34 млн.**
оценок

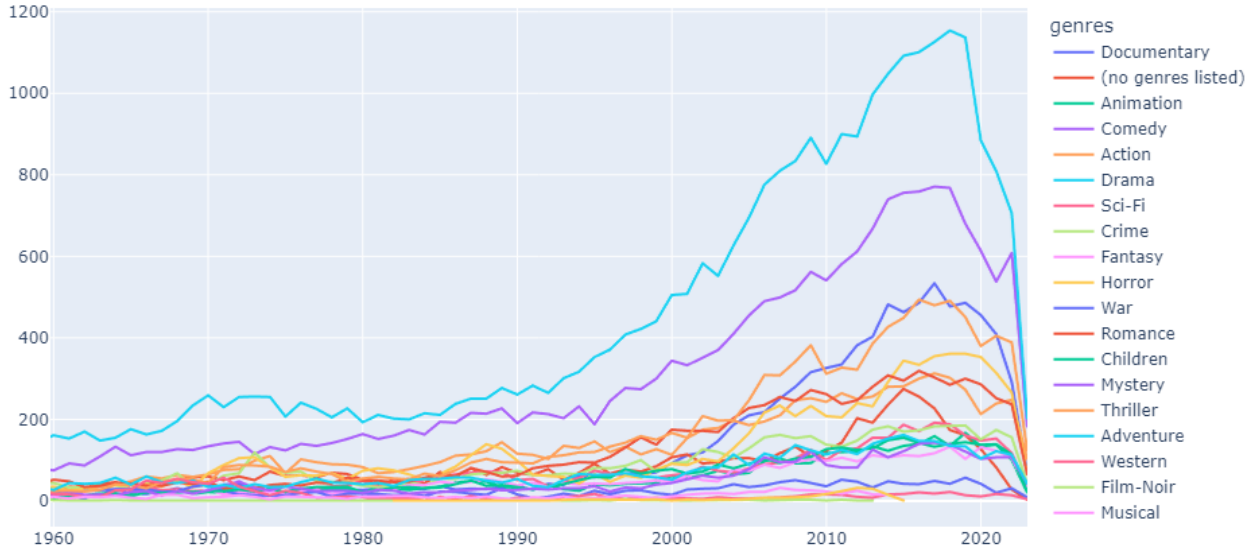
**86,5 тыс.**
уникальных фильмов с оценкой

**~331 тыс.**
уникальных пользователей поставили оценку

Количество оценок по жанрам, млн.

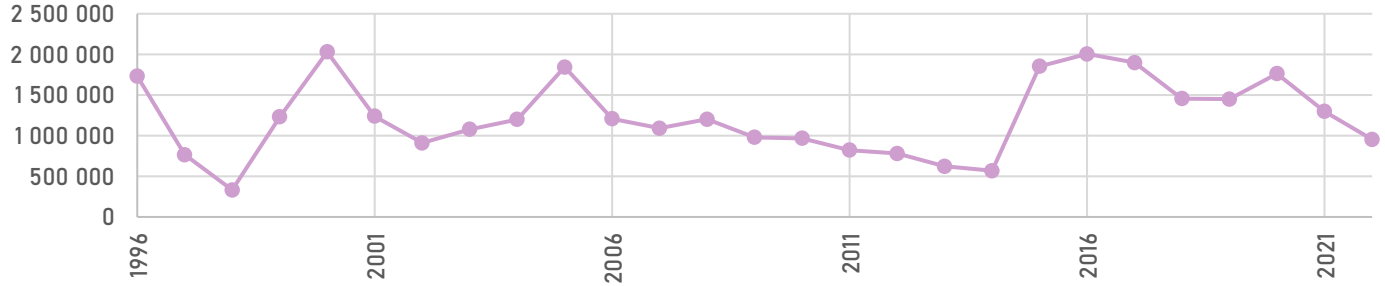



Количество выпущенных фильмов, шт.

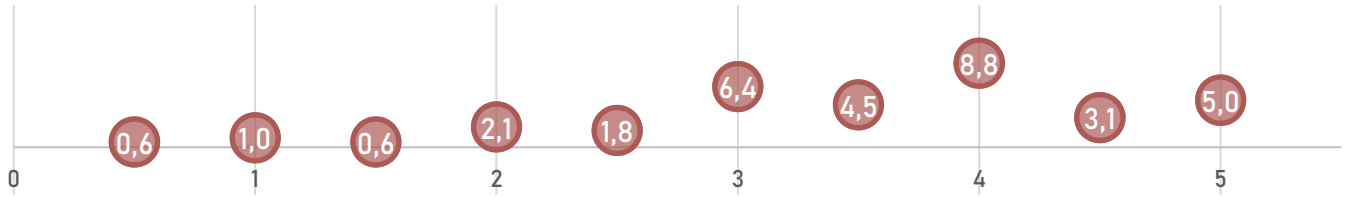


# EDA

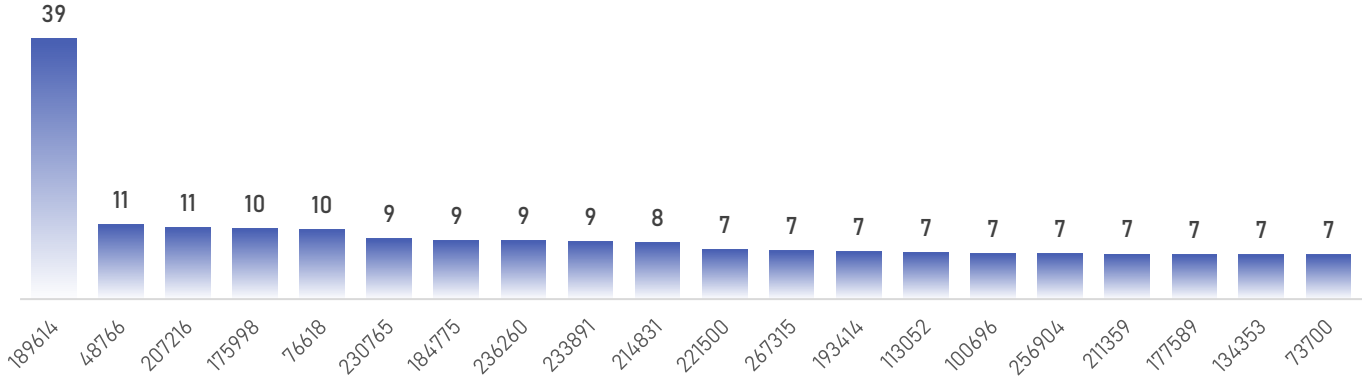
## Количество оценок по годам




## Динамика оценок пользователей по 5-бальной шкале, млн.




## Пользователи, у которых больше всего оценок, в %

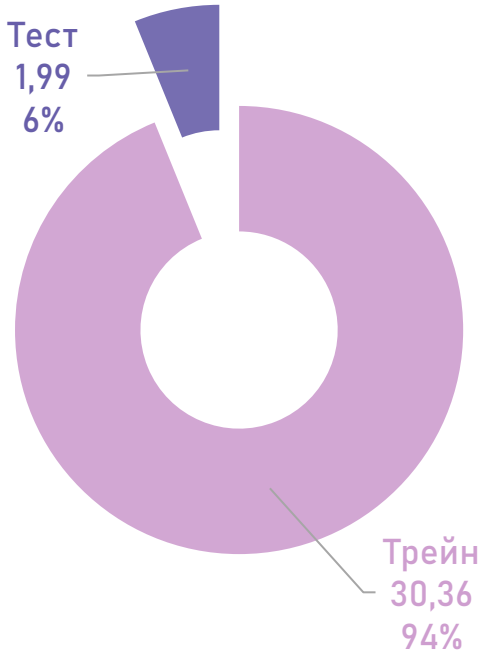


# Предобработка данных

## Сплит по данным

- Для модели брали **20 последних** оценок пользователя в тест, оставшиеся - в трейн;
- Среди пользователей отбирали только тех, у кого было **более 20 оценок** за все время, что соответствует 0,4 квантилю.

## Сплит по данным в млн. строк и % от всех данных



# Модель

При разработке модели мы учли, что все пользователи делятся на:

- «**холодных**» - по которым у нас нет информации;
- «**известных**» – по которым есть информация.

**IMDb-формула («холодные» пользователи):**

$$w = \frac{Rv + Cm}{v + m}$$

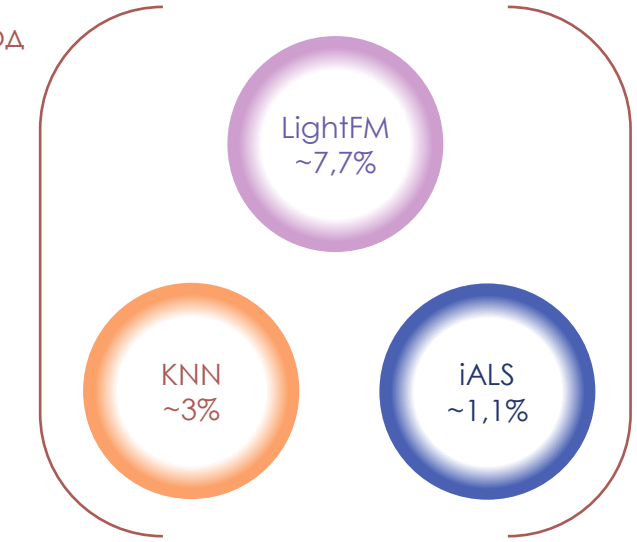
- w - взвешенный рейтинг по фильму
- v - количество оценок по фильму
- m - минимальное количество оценок, необходимое для попадания в число 250 лучших фильмов
- R - средняя оценка фильма
- C - среднее число оценок

**Применение ML-моделей (пользователи с данными):**

Нами были рассмотрены 3 основные модели классического ML, подходящие под задачу рекомендательных систем:

- KNN
- iALS
- LightFM

Качество по MAP@K=10



## KNN: rectools + implicit

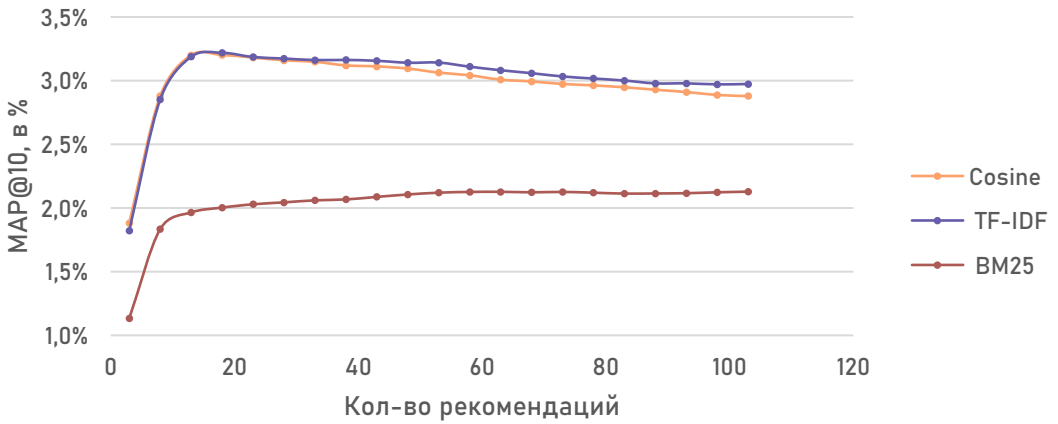
Для **KNN** были проанализированы 3 метода сходства между пользователями:

- Косинусное расстояние (Cosine);
- Мера TF-IDF;
- Мера Okapi BM25.

Наилучший результат на MAP@10 показал TF-IDF.

Более подробная информация:
recsyc_part1/KNN.ipynb

### Качество модели KNN: ~3




## iALS: implicit

Для **iALS** также был произведен подбор гиперпараметров, среди которых были:

- Кол-во факторов (factors);
- Кол-во итераций (iterations);
- Коэффициент регуляризации (regularization);
- Веса положительного класса (alpha).

Более подробная информация:
recsyc_part1/iALS.ipynb

### Качество модели iALS: ~1.1

Наилучшими гиперпараметрами были выбраны:

- factors = 3
- iterations=50
- regularization = 1.0
- alpha=4.0

## LigthFM (библиотека)

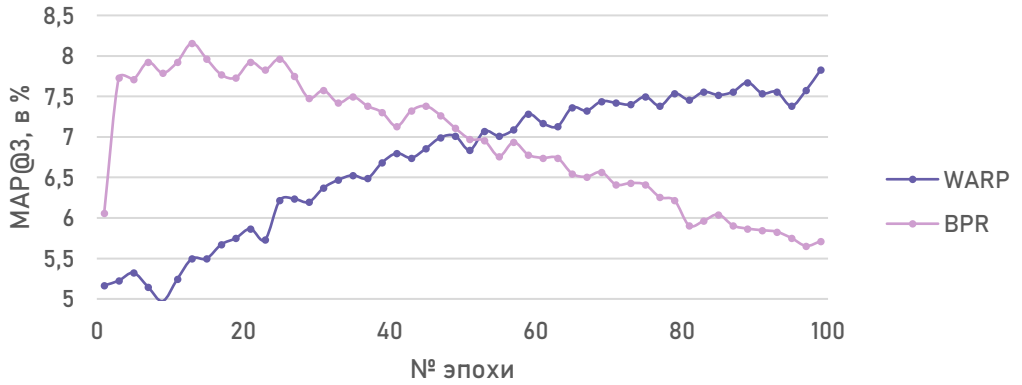
Данная модель показала **наилучшее качество**.

Для начала было проанализировано качество модели на двух **функциях потерь**:
- WARP - взвешенный приближенный ранг;
- BPR – байесовский ранг.

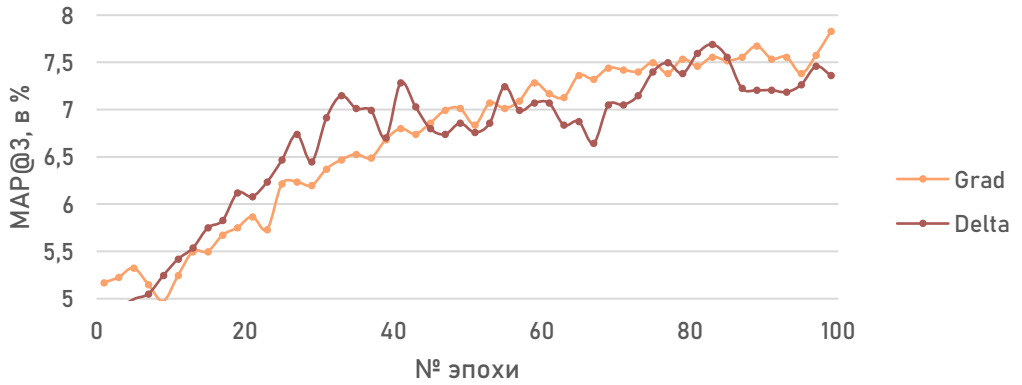
Затем были проанализированы различия в использовании оптимизаторов для обучения:
- Adagrad;
- Adadelta.

Более подробная информация:
recsyc_part1/LightFM.ipynb

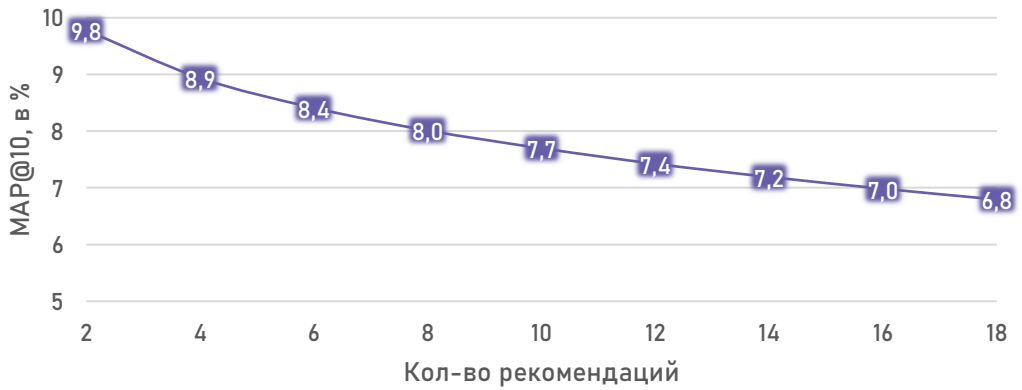
### Сравнение loss-функций WARP и BPR




### Сравнение оптимизаторов Adagrad и Adadelta




### Качество модели на MAP@10: ~7.7

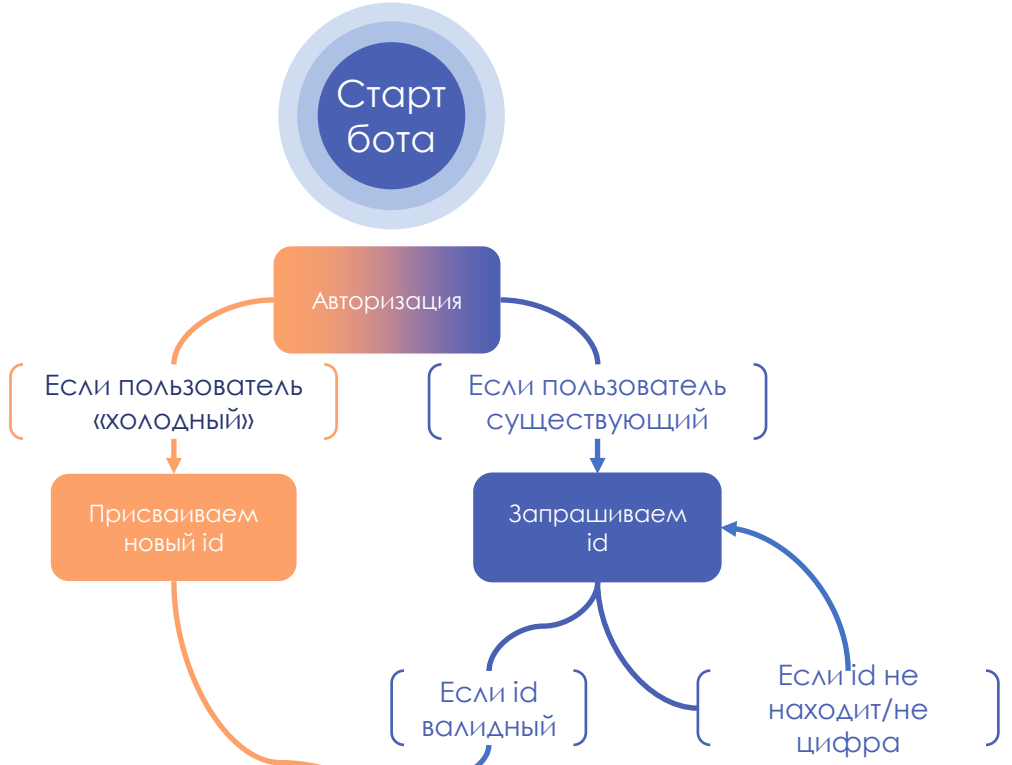


# Микросервис

## telegram - бот

Постарались учесть простой клиентский путь, для этого разработали систему последовательных сообщений, которые учитывают статус нахождения пользователя с помощью системы конечного автомата FSM.

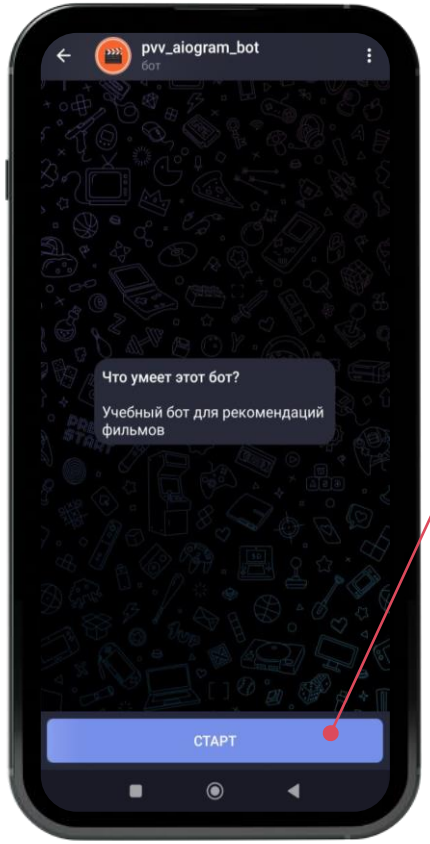
### I. Авторизация



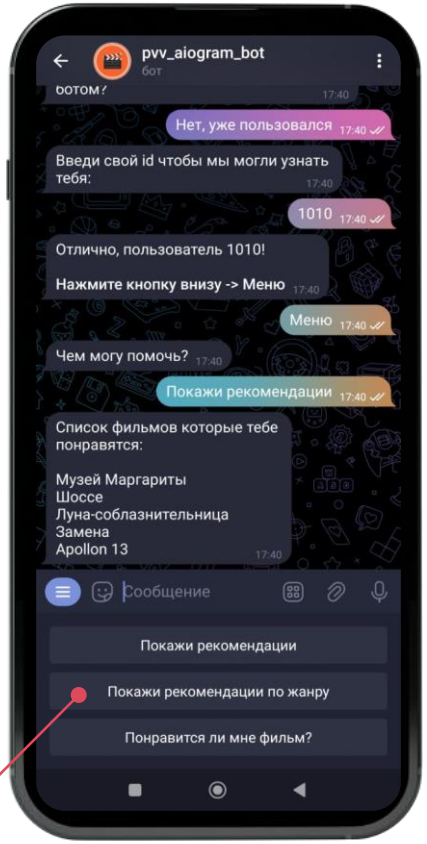

### II. Основной функционал

# Пример работа бота

Постарались сделать удобство пользования интуитивно понятным




Для начала работы бота необходимо нажать на «Старт»




Далее пользователь пользуется кнопками снизу

# DL часть – Deep FM

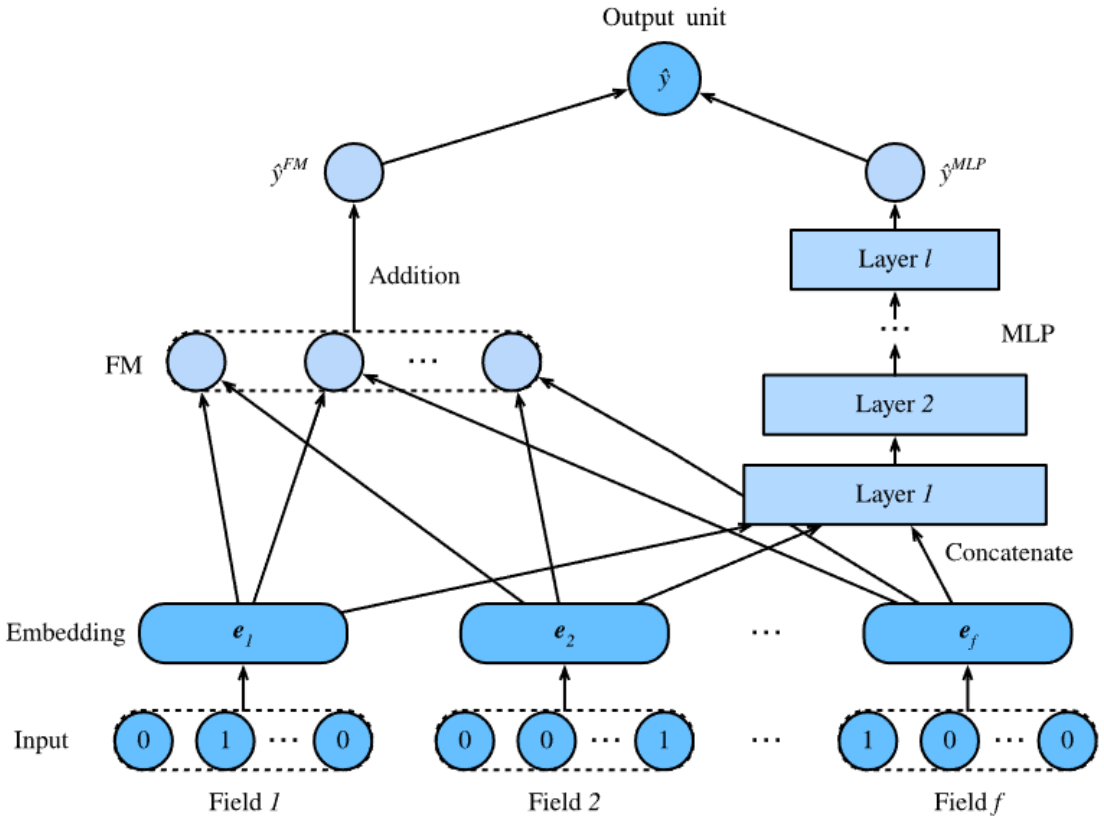
## описание архитектуры

**Архитектура DeepFM** (Deep Factorization Machine) объединяет преимущества двух различных подходов для прогноза кликабельности (CTR):

- факторизационных машин (FM);
- глубоких нейронных сетей (DNN).

Оба компонента (FM и DNN) получают один и тот же набор входных данных. Это позволяет им обучаться параллельно и использовать одни и те же признаки для своих расчетов

**Архитектура Deep FM**



# DL часть – Deep FM

## результаты

**Гиперпараметры:**

- Batch_size = 8224
- n_epochs = 5
- learning_rate = 1e-6
- embedding_size = 64
- hidden_size = [64, 64]
- num_classes = 2 (if rating >3, 1, else 0)
- dropuot = [0.5, 0.5]

- оптимизатор: ADAM
- критерий: binary_cross_entropy
- Метрика качества: map@k

Более подробная информация:
DeepFM.ipynb

Качество модели DeepFM
**test**: map@3 = 2.05%
map@5 = 3.42%
map@10 = 6.82%