# Introduction

## Data analysis

u.data, u.genre and u.user are the main 3 data files.
Firstly, let us look at the user data file. This file consists of 100000 rows and 4 columns: user_id, item_id, rating and timestamp (unix seconds). I found several important facts about this file:

- User data does not contain nan values.
- Average rating of the movie is equal to 3.53
- Standard deviation of the rating is equal to 1.13
- Minimal rating equals to 1.0
- Maximum rating equals to 5.0
  The second file is user item. This file contains information about movies: movie_id, movie_title, release_date, video_release_date, IMDB_URL and genres. Important moment is that each movie can be of several genres at once. u.genre file consists information about genres. User item:
- Has 1 nan value in release date, 1682 nan values in video release date and 3 nan values in imdb url column. I think that I will drop those columns.
- The most popular genres are Drama, Comedy and Action


  The third file is u.user. This file contains demographic information about the users (user_id, age, gender, occupation, zip code)
- File does not have non values
- Average age of the user equals to 34

- Standard deviation of the age 12.19
- Minimal age 7
- Maximal age 73
- Student, other, educator, administrator, engineer, programmer are the most frequent occupations

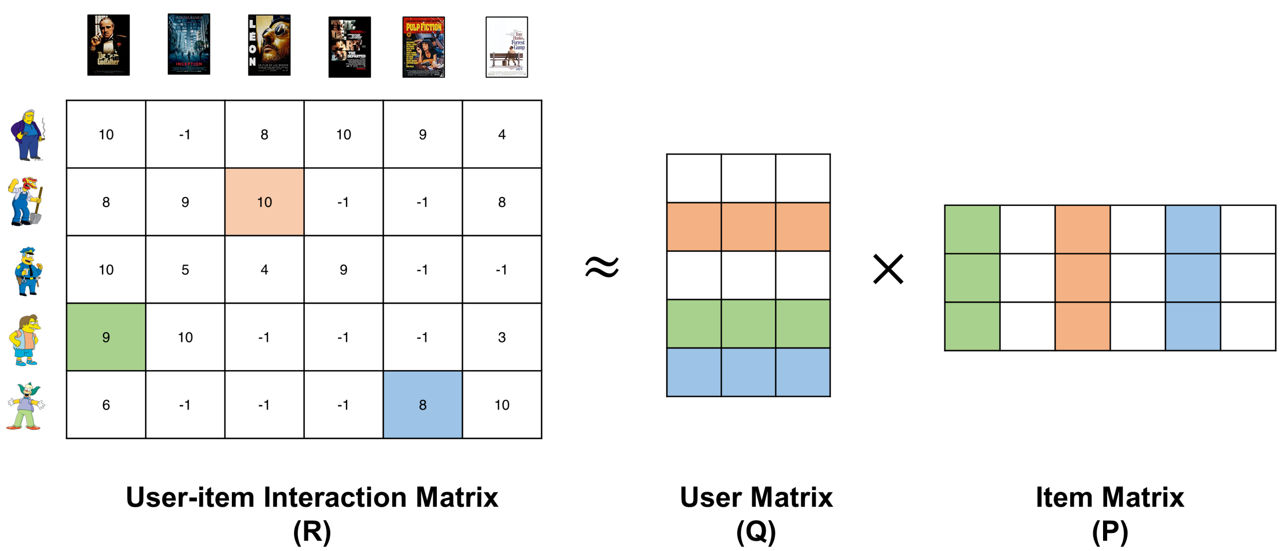
## Model implementation

My model was inspired by collaborative filtering. Collaborative filtering is a technique that finds item that a user might like based on the theory that similar users like the same items.
Types of collaborative filtering:

- Item CF
- User CF
- Matrix Factorization
  Item CF and User CF basically recommends just popular movies. ?????????????????
  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

### Matrix factorization

Matrix factorization uses linear algebra for recommendation.




User-item Interaction matrix can be decomposed into User Matrix and Item Matrix. The User Matrix contains learned features about users, Item Matrix contains features about items (in our case about movies). The algorithm defines the low-dimensional latent features in a way that captures correlated patterns from past user-item interactions. When accurately calculated, the matrices representing users and items can effectively mimic prior user-item interactions, enabling the prediction of unknown entries in the rating matrix. In the picture each row of matrix Q contains information about user, and each row of matrix P contains information about movie. Dot product of user matrix row and item matrix row is the predicted rating for the corresponding movie.

### Model optimization

I used root mean squared error as the loss function.

$$\mathrm{RMSE}(y, \hat{y}) = \sqrt{\frac{\sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2}{N}}$$

## Model advantages and disadvantages

Our model based on idea of collaborative filtering.

### Advantages

- **Personalization:**
  - Collaborative filtering considers the preferences and behaviors of similar users to make personalized recommendations. This leads to more accurate and relevant suggestions for individual users.
- **No Dependency on Item Metadata:**
  - Collaborative filtering relies on user-item interactions, making it less dependent on item metadata. This can be advantageous when dealing with a large number of items with limited metadata.
- **Scalability**
  - It is often scalable as the system grows, as the recommendations are based on user behavior rather than the entire item catalog.

### Disadvantages

- ***Cold start problem**
  - Collaborative filtering struggles when dealing with new items or new users with limited interaction history. It requires a sufficient amount of data to provide accurate recommendations.
- **Sparsity**
  - The user-item matrix is often sparse, meaning that users typically interact with only a small fraction of the available items. This sparsity can lead to challenges in finding meaningful patterns.

## Training process

- **Data preprocessing**
    - The final dataset consists of user IDs, movie IDs and corresponding ratings
- Loss function
    - Root Mean Squared Error is used as a loss function
- Optimizer
    - Adam optimizer with learning rate equals to 1e-4
- Training
    - The model had been trained for 20 epochs, however after 5-7 epochs model was overfitted. I decided to train model for 5 epochs

## Evaluation

- **Root mean squared error**
  *Root mean squared error provides a quantitative measure of how well the recommendation system's predicted ratings align with the actual ratings given by users.*
  It measures the average magnitude of the errors, giving you an indication of how accurate the system's predictions are across the entire dataset.
  * Root mean squared error penalizes large prediction errors more heavily than smaller errors. This is particularly relevant in recommendation systems, where accurately predicting highly-rated or disliked items is often more crucial than accurately predicting moderately-rated items.

$$\mathrm{RMSE}(y, \hat{y}) = \sqrt{\frac{\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2}{N}}$$

- **Precision@k**
  * Precision at k is the proportion of recommended items in the top-k set that are relevant. Precision is important when you want to ensure that the recommended items are highly relevant and minimize the number of irrelevant recommendations.

$$P@k = \frac{\text{Number of recommended items @k that are relevant}}{\text{Number of relevant items}}$$

- Recall@k
  * Recall at k is the proportion of relevant items found in the top-k recommendations. Recall is crucial when you want to make sure that the system captures as many relevant items as possible, even if it means including some irrelevant ones.

$$P@k = \frac{\text{Number of recommended items @k that are relevant}}{\text{Total number of relevant items}}$$

## Results

**Precision @ 10 (P@10):** 0.67
A precision of 0.67 means that, on average, 67% of the items recommended in the top 10 are relevant to the user.
**Recall @ 10 (R@10):** 0.53
A recall of 0.53 means that your model is capturing 53% of all relevant items in the top 10 recommendations.
**Root Mean Square Error (RMSE) :** 0.953
RMSE is often used in regression problems to measure the difference between predicted values and actual values. In the context of recommendation systems, it's common to use it to evaluate the accuracy of predicted ratings. A lower RMSE value (closer to 0) indicates better predictive performance.